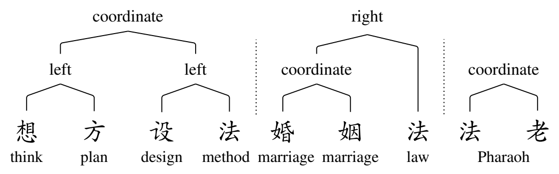
An In-depth Study on Internal Structure of Chinese Words

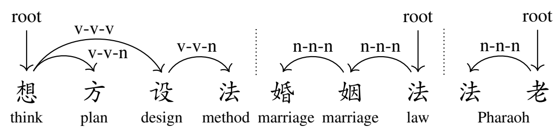
Chen Gong[1], **Saihao Huang**[1]*, **Houquan Zhou**[1], **Zhenghua Li**[1], **Min Zhang**[1], **Zhefeng Wang**[2], **Baoxing Huai**[2], **Nicholas Jing Yuan**[2]
[1]Institute of Artificial Intelligence, School of Computer Science and Technology, Soochow University, China; [2] Huawei Cloud, China
[1]{cgong,shhuang1999,hqzhou}@stu.suda.edu.cn
[1]{zhli13,minzhang}@suda.edu.cn
[2]{wangzhefeng, huaibaoxing, nicholas.yuan}@huawei.com

Abstract

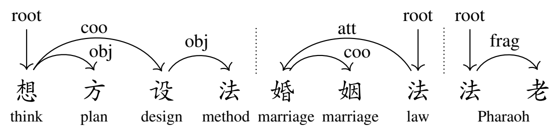
Unlike English letters, Chinese characters have rich and specific meanings. Usually, the meaning of a word can be derived from its constituent characters in some way. Several previous works on syntactic parsing propose to annotate shallow word-internal structures for better utilizing character-level information. This work proposes to model the deep internal structures of Chinese words as dependency trees with 11 labels for distinguishing syntactic relationships. First, based on newly compiled annotation guidelines, we manually annotate a word-internal structure treebank (WIST) consisting of over 30K multi-char words from Chinese Penn Treebank. To guarantee quality, each word is independently annotated by two annotators and inconsistencies are handled by a third senior annotator. Second, we present detailed and interesting analysis on WIST to reveal insights on Chinese word formation. Third, we propose word-internal structure parsing as a new task, and conduct benchmark experiments using a competitive dependency parser. Finally, we present two simple ways to encode word-internal structures, leading to promising gains on the sentence-level syntactic parsing task.



(a) Zhang et al. (2014): labels mark head positions.



(b) Li et al. (2018): labels correspond to POS tag triples.



(c) Ours: fine-grained structure with 11 labels.

Figure 1: Three example words with internal structure under different annotation paradigms. “想(think of) 方(plan) 设(design) 法(method)” is a verb and means “find ways or means to do”. “婚(marriage) 姻(marriage) 法(law)” is a noun. “法老” is phonetic transliteration of “Pharaoh”. The three words all contain the character “法” under different meanings.

1 Introduction

Unlike English, Chinese adopts a logographic writing system and contains tens of thousands of distinct characters. Many characters, especially frequently used ones, have rich and specific meanings.

However, words, instead of characters, are often considered as the basic unit in processing Chinese texts. We believe the reason may be two-fold. First, usually a character may have many meanings and usages. Word formation process greatly reduces such char-level ambiguity. Second, by definition, words are the minimal units that express a complete semantic concept or play a grammatical role independently (Xia, 2009; Yu et al., 2003).[1]

Roles played by characters in word formation can be divided into three types. **(1)** There is a stable and important set of *single-char words*, such as “你” (you)”, “的” (of), and most punctuation marks. **(2)** A character having no specific meaning acts as a *part of a single-morpheme word*, such as “仿

*Chen Gong and Saihao Huang make equal contributions to this work. Zhenghua is the corresponding author.

[1]There is still a dispute on the word granularity issue (Gong et al., 2017; Lai et al., 2021). Words are defined as a character sequence that is in tight and steady combination. However, the combination intensity is usually yet vaguely qualified according to co-occurrence frequency. We believe this work may also be potentially useful to this direction.

佛" (like) and "法(fǎ)老(lǎo)" (Pharaoh, transliteration of foreign words). **(3)** A character corresponds to a *morpheme*, the smallest meaningful unit in a language, and composes a polysyllabic word with other characters. This work targets multi-char words, and is particularly interested in the third type which most characters belong to.

Intuitively, modeling how multiple characters form a word, i.e., the word-formation process, allows us to more effectively represent the meaning of a word via composing the meanings of characters. This is especially helpful for handling rare words, considering that the vocabulary size of characters is much smaller than that of words. In fact, many NLP researchers have tried to utilize char-level word-internal structures for better Chinese understanding. Most related to ours, previous studies on syntactic parsing have proposed to annotate word-internal structures to alleviate the data sparseness problem (Zhang et al., 2014; Li et al., 2018). However, their annotations mainly consider flat and shallow word-internal structure, as shown in Figure 1-(a) and (b). Meanwhile, researchers try to make use of character information to learn better word embeddings (Chen et al., 2015; Xu et al., 2016). Without explicitly capturing word-internal structures, these studies have to treat a word as a bag of characters. See Section 2 for more discussion.

This paper presents an in-depth study on char-level internal structure of Chinese words. We endeavour to address three questions. **(1)** What are the word-formation patterns for Chinese words? **(2)** Can we train a model to predict deep word-internal structures? **(3)** Is modeling word-internal structures beneficial for word representation learning?

For the first question, we propose to use labeled dependency trees to represent word-internal structures, and employ 11 labels to distinguish syntactic roles in word formation. We compile annotation guidelines following the famous textbook of Zhu (1982) on Chinese syntax, and annotate a high-quality word-internal structure treebank (WIST), consisting of 30K words from Penn Chinese Treebank (CTB) (Xia, 2009). We conduct detailed analysis on WIST to gain insights on Chinese word-formation patterns.

For the second question, we propose word-internal structure parsing as a new task, and present benchmark experimental results using a competitive open-source dependency parser.

For the third question, we investigate two simple ways to encode word-internal structure, i.e., LabelCharLSTM and LabelGCN, and show that using the resulting word representation leads to promising gains on the dependency parsing task.

We release WIST at `https://github.com/SUDA-LA/ACL2021-wist`, and also provide a demo to parse the internal structure of any input word.

2 Related Work

Annotating word-internal structure. In the deep learning (DL) era, pretraining techniques are extremely powerful in handling large-scale unlabeled data, including Skip-Gram or CBOW models (Mikolov et al., 2013) for learning context-independent word embedding in the beginning, and the recent ELMo (Peters et al., 2018) or BERT (Devlin et al., 2019) for learning context-aware word representations. Conversely, in the pre-DL era, there exist few (if any) effective methods for utilizing unlabeled data, and statistical models rely on discrete one-hot features, leading to severe data sparseness for many NLP tasks. This directly motivates annotation of word-internal structure, especially for dealing with rare words.

Annotation of shallow internal structure of Chinese words was first mentioned in Zhao (2009), largely based on heuristic rules. Li (2011); Li and Zhou (2012) found that many multi-char words could be divided into two subwords, i.e., root and affix. They annotated structures of about 19K words (35% of 54,214) in CTB6. Their experiments showed that subword-level syntactic parsing is superior to word-level parsing. For the three words in Figure 1, their approach is only applicable to the second word, i.e., "婚姻/法". As an extension to Li and Zhou (2012), Zhang et al. (2013, 2014) proposed char-level syntactic parsing by further dividing subwords into chars. As shown in Figure 1-(a), for each word, they annotated a binary hierarchical tree, using constituent labels to mark which child constituent is more syntactically important, i.e., left, right, or coordinate. In such way, they could convert a word-level constituent/dependency tree into a char-level one. Similar to Li and Zhou (2012), Cheng et al. (2014) annotated internal structure of synthesis (multi-morpheme) words with four relations, i.e., branching, coordinate, beginning and other parts of a single-morpheme word.

In the DL era, three works have studied word-internal structure. Similarly to our work, Li et al. (2018) employed dependency trees to encode word-

Label	Meaning	Example	Annotation
root	word root	登场 (come on stage)	\$ $\xrightarrow{\text{root}}$ 登 (come) $\xrightarrow{\text{obj}}$ 场 (stage)
subj	subject	年轻 (young)	年 (age) $\xleftarrow{\text{subj}}$ 轻 (small)
obj	object	下雨 (rain)	下 (drop) $\xrightarrow{\text{obj}}$ 雨 (rain)
att	attribute modifier	大衣 (overcoat)	大 (large) $\xleftarrow{\text{att}}$ 衣 (coat)
adv	adverbial modifier	不同 (different)	不 (not) $\xleftarrow{\text{adv}}$ 同 (same)
cmp	complement modifier	放下 (put down)	放 (put) $\xrightarrow{\text{cmp}}$ 下 (down)
coo	coordination	上下文 (context)	上 (above) $\xrightarrow{\text{coo}}$ 下 (below)
pobj	preposition object	到期 (expire)	到 (reach) $\xrightarrow{\text{pobj}}$ 期 (deadline)
adjct	adjunct	走过 (pass by)	走 (walk) $\xrightarrow{\text{adjct}}$ 过 (by)
frag	fragment	沙发 (sofa)	沙 (sand) $\xrightarrow{\text{frag}}$ 发 (send)
repet	repetition	常常 (often)	常 (often) $\xrightarrow{\text{repet}}$ 常 (often)

Table 1: The 11 labels adopted in our guidelines for distinguishing syntactic roles in word formation.

internal structure. As shown in Figure 1-(b), for each multi-char word, they first annotate the part-of-speech (POS) tag of each character, and then determine an unlabeled dependency tree, and finally use a POS tag triple as arc label, corresponding to the POS tags of the modifier/head characters and the whole word. However, we argue POS tag triples are only loosely related with word-formation patterns, not to mention the severe difficulty of annotating char-level POS tags in each word.

Recently, Lin et al. (2020) extended Zhang et al. (2014) by using an extra label for marking single-morpheme words, and annotated hierarchical internal structure of 53K words from a Chinese-English machine translation (MT) dataset. Li et al. (2019a) annotated the internal structure of words with 4 dependency relations.

In summary, we can see that most previous studies adopted quite shallow hierarchical structure. In contrast, this work presents a more in-depth investigation on internal structure of Chinese words and employs 11 labels to distinguish different syntactic roles in word formation, as shown in Figure 1-(c).

Leveraging character information for better word representation. It has already become a standard way in many NLP tasks to obtain char-aware word representation by applying LSTM or CNN to the character sequence of a word, and concatenate it with word embedding as input, such as named entity recognition (Chiu and Nichols, 2016), dependency parsing (Zhang et al., 2020), and constituent parsing (Gaddy et al., 2018).

Another research direction is to leverage character information to obtain better word embeddings. Chen et al. (2015) extended the CBOW model and proposed to jointly learn character and word embeddings. Based on Chen et al. (2015), Yu et al. (2017) proposed to jointly learn embeddings of words, characters, and sub-characters.[2] However, both studies assume that characters contribute equally to the meaning of a word and directly average embeddings of all characters. To address this, Xu et al. (2016) extended Chen et al. (2015) and proposed a cross-lingual approach to distinguish contribution of characters for a word. The idea is to translate Chinese words and characters into English words, and use similarities between corresponding English word embeddings for contribution measurement. Instead of treating a word as a bag of characters, we experiment with two simple ways to obtain structure-aware word representations. Meanwhile, enhancing their approach with explicit word-internal structure could be also very interesting.

Utilizing word-internal structure. Word-internal structure have been explored in various NLP tasks. Several works propose to learn word-internal structure, word segmentation, POS tagging and parsing jointly (Zhang et al., 2013, 2014; Li et al., 2018), demonstrating the effectiveness of word-internal structure in helping downstream tasks. Cheng et al. (2015) attempt to convert words into fine-grained subwords according to the

[2]Following this direction, studies tried to explore more character information for better Chinese word representation, such as strokes (Cao et al., 2018) and ideographic shape (Sun et al., 2019).

internal structure of words for better dealing with unknown words during word segmentation. Lin et al. (2020) propose to integrate the representation of word-internal structure into the input of neural machine translation model, leading to improved translation performance.

3 Word-internal Structure Annotation

In this section, we describe in detail the annotation process of WIST. As shown in Figure 1-(c), we adopt dependency trees for representing word-internal structure. The reason is two-fold. First, word-formation process correlates with syntax in different ways depending on language type (Aikhenvald, 2007). Such correlation is especially close for Chinese due to its lack of morphological inflections. In particular, Zhu (1982) presented thorough investigation on Chinese word formation mainly from a syntactic view. Second, as a grammar formalism, dependency tree structure has been widely adopted for capturing sentence-level syntax due to its simplicity and flexibility in representing relations. Meanwhile, its computational modeling is also developed quite well.

Annotation guidelines. After several months' survey, we have compiled systematic and detailed guidelines for word-internal structure annotation. Our guidelines are mainly based on the famous textbook on Chinese grammar of Zhu (1982). We intensively studied all previous works on word-internal structure annotation, which are discussed in Section 2. We also find that it is quite beneficial to be familiar with guidelines developed by previous annotation projects for Chinese word segmentation (Xia, 2009; Yu et al., 2003).

Our guidelines contain 11 relations specifically designed to capture the internal dependency syntax for Chinese words, as shown in Table 1. We derive most of the dependency relations by referring to guidelines of three popular Chinese dependency treebanks, i.e., UD, Harbin Institute Technology Chinese Dependency Treebank (HIT-CDT) (Liu et al., 2006), and Chinese Open Dependency Treebank (CODT) (Li et al., 2019b). We give very detailed illustrations with examples in our 30-page guidelines to ensure annotation consistency and quality. Our guidelines are also gradually improved according to the feedback from the annotators.

Quality control. We employ 18 undergraduate students as part-time annotators who are familiar with Chinese syntax, and select 6 capable annotators with a lot of data annotation experience as expert annotators to handle inconsistent submissions. All the annotators (including expert annotators) were paid for their work . The salary is determined by both quantity and quality. Besides, we give extra bonus to the annotators with high accuracy. The average salary of the annotators is 30 RMB per hour. All annotators are trained for several hours to be familiar with our guidelines and the usage of annotation tool.

We apply strict double annotation in order to guarantee quality. Each word is randomly assigned to two annotators. Two identical submissions are directly used as the final answer. Otherwise, a third expert annotator is asked to decide the final answer after analyzing the two inconsistent annotations.

	Total #	1	2	3	≥4
word type	37,449	5.6	58.3	22.8	13.3
word token	508,764	48.0	44.1	6.0	1.9

Table 2: Word distr. regarding char number in CTB5.

Annotation tool. We build a browser-based annotation tool to support the annotation workflow and facilitate project management.

Given an annotation task, all its POS tags [3] of the focused word in CTB5 are presented to the annotator, in order to explore multiple internal structures for one word. In that case, the annotator can click a checkbox to inform us for further process. Please note that the manually annotated POS tags in CTB5 are converted into Universal Dependencies (UD) [4] POS tags based on predefined mapping rules, since the original CTB5 POS tags are too fine-grained (33 tags) and difficult for annotators to understand. The interface also presents several example sentences to improve annotation efficiency. We strongly encourage annotators to look up difficult words or characters in electronic dictionaries.[5]

[3]In CTB5, a word may be annotated with different POS tags under different contexts. For example, "发展 (development)" is annotated as NN (noun) in the context "促进经济发展 (boost the economic development)", whereas "发展 (develop)" is annotated as (VV) verb in the context "稳定地发展 (develop steadily)". Therefore, when annotating the word "发展 (develop/development)", we present both "noun" and "verb" to the annotators for reference."

[4]`universaldependencies.org/u/pos/`

[5]Eg., `hanyu.baidu.com`; `xh.5156edu.com/`

Data selection. Following previous works, we select multi-char words from CTB5 for annotation. Table 2 shows word distribution regarding character numbers. We can see that only 5.6% of words in the vocabulary contain one char, but they account for nearly half (48%) token occurrences in the text. The percent of words with two characters is high in both vocabulary (58.3) and text (44.1). We discard words containing special symbols such as English letters. Finally, we have annotated 32,954 multi-char words with their internal structure, containing 83,999 dependencies (2.5 characters per word).

4 Analysis on Annotated WIST

In this section, we analyze the annotated WIST from different aspects in order to gain more insights on Chinese word-formation patterns.

Inter-annotator consistency. As discussed earlier, each word is labeled by two annotators, and inconsistent submissions are handled by a third senior annotator for obtaining a final answer. The averaged inter-annotator consistency ratio is 83.0 dependency-wise, i.e., the percent of characters receiving the same head and label from two annotators, and 75.8 word-wise, i.e., the percent of words receiving the same whole trees. If we do not consider labels, the unlabeled consistency ratios increase to 87.5 dependency-wise and 85.1 word-wise. Although it may be a factor that most annotators are inexperienced in this new annotation task, such low consistency ratios indicate that annotating word-internal structure is quite challenging, especially when it comes to distinguishing syntactic roles. Meanwhile, this also demonstrates the importance of strict double annotation, considering that nearly a quarter of words are inconsistent and require handling by senior annotators.

Annotation accuracy. We calculate annotation accuracy by comparing all submissions (as denominator) from annotators against the final answers in WIST. Please note that each word is double annotated. The overall dependency-wise accuracy for all annotators is 90.9, and word-wise is 86.9. If not considering labels, the overall unlabeled accuracy increases to 93.4 and 92.1, dependency- and word-wise respectively.

The first major row in Table 3 shows the label-wise annotation accuracy. We divide characters in WIST into 11 groups according to their final-answer labels, and then calculate the percent of correct submissions for each group. The highest accuracy is obtained on "repet", since its pattern is quite regular. Determining the root character also seems relatively easy. The lowest accuracy is 62.0 on "subj" and 48.2 on "pobj".

Comparing unlabeled versus labeled accuracy, the gap is quite large. The extreme case is "pobj". Annotators usually can correctly decide the head (84.5%), but very unlikely choose its true label "pobj" (48.2%). Similarly, accuracy drops by 24.9 for "subj". We give more discussions on annotation difficulties below.

Label distribution. The third major row in Table 3 shows distribution of different labels in WIST. From the percentage of "root" (39.2%), we can infer that one word contains 2.5 characters on average. The overall percent for "att" is 29.1, almost half of the remaining labels, meaning that "att" appears once every 1.45 words. This reveals that attribute modification is the most dominated pattern in word formation. Coordination structure ("coo") takes the second place with 10.2%. The third most used pattern is fragment ("frag") with 5.7%. We give more discussion on "frag" below.

Besides the overall distribution, the third major row in Table 3 gives label distribution per POS tag. For clarity, we give the full name of each POS tag (UD, converted from the fine-grained CTB tags) in Table 3, and it means the POS tag of the focused word. If a word has multiple POS tags, then the same word-internal structure is used for each tag. For example, if a word "发 (expand) $\xrightarrow{coo}$ 展 (expand)" has two tags, i.e., Noun and Verb, then the number of "coo" is added by one for both Noun and Verb. Moreover, a label is repeatedly counted if it appears several times in the same word. Due to space limitation, we only present high-frequency POS tags, with percentage shown in parenthesis. Please note that we adopt a coarse-grained POS tag set for clarity.

We can see that nouns are mostly formed with "att" (33.8%) and "coo" (11.5%), whereas verbs are with "coo/obj/adv/cmp" in the descending order. Proper nouns are evenly dominated by "frag" (29.6%) and "att" (28.4%). It is also obvious that proper nouns tend to be longer, consisting of 2.7 characters according to its "root" percentage. Numerals are mainly composed via "att" (75.7%) and consist of 5.0 character on average.

	root	att	coo	frag	obj	adv	cmp	adjct	subj	repet	pobj
Annotation Accuracy	*93.9*	93.1	88.6	89.3	82.6	80.6	85.3	83.5	62.0	**96.0**	48.2
Unlabeled	93.8	94.2	92.3	93.3	92.7	88.1	*97.9*	92.2	86.9	**99.4**	84.5
Parsing Accuracy	*89.0*	**89.5**	75.8	80.6	77.4	68.0	84.0	76.8	64.2	81.1	58.1
Unlabeled	89.0	90.6	85.4	84.1	88.2	80.7	*93.5*	80.5	80.7	**97.3**	83.9
Overall Distribution	39.2	**29.1**	*10.2*	5.7	5.4	4.3	2.3	1.5	1.5	0.6	0.2
Noun (47.2%)	42.3	**33.8**	*11.5*	2.5	4.4	2.6	0.4	1.1	1.1	0.2	0.1
Verb (24.1%)	42.2	3.8	**17.9**	0.4	*12.7*	9.6	7.9	1.2	3.1	0.9	0.4
Proper Noun (13.1%)	36.6	*28.4*	2.3	**29.6**	0.8	0.6	0.1	0.9	0.6	0.3	0
Adjective (7.1%)	44.4	*16.5*	**17.7**	0.7	7.5	8.2	0.6	0.7	1.9	1.6	0.2
Adverb (3.9%)	45.5	**12.1**	10.3	0.6	6.4	*12.1*	1.8	5.3	1.0	2.8	2.3
Numeral (3.7%)	20.0	**75.7**	0.4	0.1	0.1	0.2	0	*3.6*	0	0.1	0
Others (0.9%)	47.6	**15.2**	*8.7*	2.1	1.4	7.7	4.8	8.2	0.3	3.9	0.1

Table 3: Label-wise accuracy and distribution. The first major row presents annotation accuracy of WIST and "unlabeled" means not considering labels. The second major row gives parsing accuracy on WIST-test, discussed in Section 5. The third major row gives distribution of different labels for words of different POS tags.

Multiple structures for one word? Many words have multiple meanings. Then the question is: how many words really have multiple internal structures? As illustrated in Section 3, we show all POS tags to annotators in order to obtain all internal structures of an ambiguous word. However, in annotated WIST, we find there are only 103 such words with multiple internal structures, accounting for about 0.3% of all annotated words, and 2.7% of those having multiple POS tags. As a typical example, "制服" have two structures. As a verb, it means "subdue" and has "制(control) $\xrightarrow{cmp}$ 服(tamely)". As a noun, it means "uniform" and has "制(regulated) $\xleftarrow{att}$ 服(cloth)". This low percentage reveals that most Chinese words actually have very steady internal structure. They have multiple POS tags, mainly because they are used for different syntactic functions without morphological inflections, such as "发展" as verb ("develop") or noun ("development").

More on "frag". The "frag" label is designed to handle all words that have no internal structure due to the lack of semantic composition. From Table 3, we can see that "frag" accounts for 5.7% of all labels. In order to gain more insights, we collect all 3,528 words containing "frag" in WIST, and randomly sample 100 words for investigation. Following the brief discussion in Section 1, we divide these words into three types, and find that 81 words are proper nouns (such as person name); 16 correspond to transliteration of foreign words; and 3 are single-morpheme words.

High-order structure distribution. To gain more insights on complex word-formation structure, we focus on all three-char words. We find that the root usually lies in the third character by 74.6%, and the percentage for the second and first characters is only 15.3 and 10.1 respectively. Looking more closely, we find the following four dominated structures.

$1 \leftarrow 2 \leftarrow 3$	34.7%	$(1 \rightarrow 2) \leftarrow 3$	34.2%
$1 \leftarrow 2 \rightarrow 3$	15.3%	$1 \rightarrow 2 \rightarrow 3$	7.0%

Difficulties in annotation. Since it is difficult to capture the patterns on unlabeled-dependency inconsistencies, we focus on confusion patterns in label annotation. Among all characters receiving the same head but different labels from two annotators, 20.1% correspond to "{att, adv}" confusion due to the ambiguity of the head character being a verb or a noun. The second confusion pattern is "{coo,frag}", with a proportion of 18.6, which are mainly from proper nouns. According to our guidelines, if the meaning of a proper noun is compounding, annotators have to annotate its real internal structures rather than using "frag". It is also very difficult to distinguish "obj" and "pobj", since the boundary between prepositions and verbs is vague in Chinese.

5 Word-internal Structure Parsing

With annotated WIST, we try to address the second question: can we train a model to predict word-internal structure? We adapt the Biaffine parser proposed by Dozat and Manning (2017), a widely

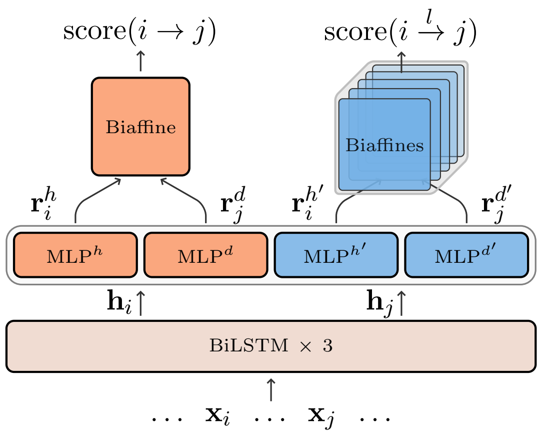


Figure 2: The basic architecture of Biaffine Parser.

used sentence-level dependency parser, for this purpose, and present results and analysis.

5.1 Biaffine Parser

We adopt the SuPar implementation released by Zhang et al. (2020).[6] As a graph-based parser, Biaffine parser casts a tree parsing task as searching for a maximum-scoring tree from a fully-connected graph, with nodes corresponding to characters in our case. As shown in Figure 2, it adopts standard encoder-decoder architecture, consisting of the following components.

Input layer. Given an input sequence, each item is represented as a dense vector $\mathbf{x}_i$. For word-internal structure parsing, an item corresponds to a character, and we use char embedding.

$$\mathbf{x}_i = \mathbf{emb}(c_i) \tag{1}$$

BiLSTM encoder. Then, a three-layer BiLSTM is applied to obtain context-aware representations. We denote the hidden vector of the top-layer BiLSTM for the i-th position as $\mathbf{h}_i$.

Biaffine scorer. Two separate MLPs are applied to each $\mathbf{h}_i$, resulting in two lower-dimensional vectors $\mathbf{r}_i^h$ (as head) and $\mathbf{r}_i^d$ (as dependent). Then the score of a dependency $i \rightarrow j$ is obtained via a biaffine attention over $\mathbf{r}_i^h$ and $\mathbf{r}_j^d$. Scoring of labeled dependencies such as $i \xrightarrow{l} j$ is analogous.

Decoder. With the scores of all dependencies, we adopt the first-order algorithm of Eisner (2000) to find the optimal unlabeled dependency tree, and then independently decide the highest-scoring label for each arc.

[6]https://github.com/yzhangcs/parser

	Dev		Test		
	UAS	LAS	UAS	LAS	CM
Random	81.18	76.15	80.63	75.58	65.13
Pretrained	82.42	77.30	81.64	76.98	67.09
	+1.24	+1.15	+1.01	+1.40	+1.96
BERT	88.27	85.18	88.33	84.98	77.72
	+5.85	+7.88	+6.69	+8.00	+10.63

Table 4: Results of word-internal structure parsing using different character representations.

Training loss. During training, the parser computes two independent cross-entropy losses for each position, i.e., maximizing the probability of its correct head and the correct label between them.

5.2 Settings

Data. We randomly split all words in WIST into three parts, 2,500/5,000 as development/test data and remaining as training data.

Hyperparameters. We set the dimension of char embeddings to 100. We obtain pre-trained character embeddings by training word2vec on Chinese Gigaword Third Edition. In order to see effect of contextualized character representations, we apply BERT (Devlin et al., 2019) [7] to each word as a char sequence. The output vectors of the top four layers are concatenated and reduced into a dimension of 100 via an MLP. For other hyper-parameters, we keep the default configuration in SuPar.

Evaluation metrics. We adopt the standard unlabeled and labeled attachment score (UAS/LAS), i.e., the percent of characters that receives the correct head (and label). The complete match (CM) is the percent of words having correct whole trees.

5.3 Results

Table 4 shows the main results under different char representations. It is obvious that using randomly initialized char embeddings, the parser can only reach about 76 in LAS. This shows that parsing word-internal structure is very challenging without using extra resources. When we pretrain char embeddings on large-scale labeled data, the performance can be consistently improved by over 1 point in both UAS/LAS, and nearly 2 points in CM. Finally, employing the contextualized character rep-

[7]BERT-base-Chinese: https://github.com/google-research/bert

resentations dramatically improves performance further by about 6/8/10 points in UAS/LAS/CM.

However, even with BERT, model performance still lags behind averaged human performance (90.9 in LAS) by large margin. Our experienced annotators can even reach more than 94. Our experience in manual annotation points out two possible directions to enhance the model: 1) making use of sentence-level contextual information; 2) leveraging the meanings in dictionaries, usually in the form of explanation or example sentences. We leave them for future exploration.

Analysis on label-wise accuracy. The second major row in Table 3 reports accuracy regarding different labels for the model with BERT. The model achieves the highest accuracy on "att" and "root", possibly because the two labels take very large proportion in the data for sufficient model training. By contrast, "pobj" and "subj" have the lowest accuracy, and are difficult for models as well as discussed in Section 3. This leads to another observation that model accuracy is roughly correlated with annotation accuracy, implying the difficulties for human and model are usually consistent.

6 Utilizing Word-internal Structure

This section presents a preliminary study on utilizing word-internal structure, aiming to address the third question: is modeling word-internal structures beneficial for word representation learning?

We use sentence-level dependency parsing as the focusing task (Kübler et al., 2009), mainly considering resemblance in tree structure representation and close relatedness between the two tasks. Given an input sentence $w_0 w_1 ... w_m$, the goal of dependency parsing is to find an optimal dependency tree for the sentence. Again, we adopt SuPar (Zhang et al., 2020) for implementation of Biaffine parser (Dozat and Manning, 2017) as our basic parser.

6.1 Methods

The basic parser applies a BiLSTM over character sequence to obtain word representation. In this part, we propose two simple alternative methods to encode internal structure shown in Figure 1-(c).

Basic CharLSTM method. For each word, the basic Biaffine parser uses the concatenation of word embeddings and CharLSTM outputs to represent each word in the input layer:

$$\begin{aligned} \mathbf{x}_i &= \mathbf{emb}(w_i) \oplus \text{CharLSTM}(w_i) \\ \text{CharLSTM}(w_i) &\leftarrow \text{BiLSTM}(..., \mathbf{z}_k, ...) \\ \mathbf{z}_k &= \mathbf{emb}(c_{i,k}) \end{aligned} \tag{2}$$

where $c_{i,k}$ is the k-th character of w_i. The final word representation from CharLSTM(w_i) is obtained by concatenating two last-timestamp hidden output vectors of a one-layer BiLSTM.

LabelCharLSTM Method. Considering that the word is usually very short and a bare label itself provides rich syntax information, we propose a straightforward extension to CharLSTM, named as LabelCharLSTM, via minor modification.

$$\mathbf{z}_k = \mathbf{emb}(c_{i,k}) \oplus \mathbf{emb}(l_{i,k}) \tag{3}$$

where $l_{i,k}$ represents the label between $c_{i,k}$ and its head in the word-internal structure.

LabelGCN method. Previous work show that GCN is very effective in encoding syntactic trees (Marcheggiani and Titov, 2017; Zhang et al., 2018). We follow the implementation of Zhang et al. (2018) and use a two-layer GCN as a more sophisticated way. In order to utilize labels, we extend vanilla GCN to have the same input with LabelCharLSTM, i.e., $\mathbf{z}_k$. We obtain the final word representation by performing average pooling over the output vectors of the top-layer GCN.

6.2 Experiments

Settings. Following Chen and Manning (2014), we conduct experiments on CTB5 with the same data split (16,091/803/1,910 sentences) and constituent-to-dependency conversion. Both char/label embeddings are randomly initialized and have the same dimension of 50. For the parsers using gold-standard POS tags, we randomly initialized the POS tagging embeddings and set the dimension to 50. For other hyperparameters, we adopt the default configuration of SuPar, including the pre-trained word embeddings.

For multi-char words without annotated internal structure, we use the automatic outputs from the trained parser with BERT in Section 5, so that every word corresponds to a single structure.

We use word-wise UAS/LAS/CM for evaluation, and punctuation is excluded in all metrics.

Main results. Table 5 shows the parsing performance. We can see that both LabelCharLSTM and LabelGCN substantially outperform the basic

	UAS	LAS	CM
Basic CharLSTM	88.31	85.96	32.04
LabelCharLSTM	88.78	86.51	**33.19**
LabelGCN	**89.02**	**86.76**	32.93
w/o label	88.66	86.28	32.20

Table 5: Parsing performance on CTB5-test.

	all	> 2	≤ 2	unk.
Basic CharLSTM	85.96	86.42	82.03	81.73
LabelGCN	86.76	87.10	83.79	84.30
	+0.80	+0.68	+1.76	+2.57

Table 6: Parsing LAS regarding to word frequency.

CharLSTM method. LabelGCN achieves the best performance on UAS and LAS, with a gain of 0.71 and 0.80 respectively.

The fourth row reports performance of LabelGCN without using label embedding, leading to consistent accuracy drop, demonstrating the usefulness of rich labels, which is a key contribution of this work, despite the extra annotation effort.

Analysis on rare words. To gain more insights on how word-internal structure helps word representation learning, we divide the words in CTB5-test into several groups according to their frequency in CTB5-train, and report fine-grained accuracy in Table 6. We can see that the overall performance gain is mostly contributed by improvement over rare words with low frequency or totally unknown. This verifies that word-internal structures can help the model to better represent rare words.

Results with gold-standard POS tags. As suggested by a reviewer, we train our parser with gold-standard POS tags by concatenating the original input (i.e., $\mathbf{x}_i$ in Equation 2) with gold-standard POS tag embeddings, in order to compare with previous works. Table 7 shows the results. Compared with the Basic CharLSTM results in Table 5, using gold-standard POS tags as extra features for the Basic CharLSTM leads to substantial improvements by 2.80 and 3.95 in UAS and LAS respectively, and outperforms the previous works as presented in Table 7, showing that the basic CharLSTM can be served as a strong baseline model.

Compared with the Basic CharLSTM, utilizing word-internal structure with LabelCharLSTM or LabelGCN achieves consistently better performance by 0.24 and 0.25 respectively in LAS in

	UAS	LAS
Ma and Hovy (2017)	89.05	87.74
Dozat and Manning (2017)	89.30	88.23
Ma et al. (2018)	**90.59**	**89.29**
Basic CharLSTM	91.11	89.91
LabelCharLSTM	**91.31**	90.15
LabelGCN	**91.31**	**90.16**

Table 7: Parsing performance with gold-standard POS tags on CTB5-test.

the scenario of using gold-standard POS tags. Besides the strong baseline, another reason that the improvement brings by the internal-word structure is slight when using gold-standard POS tags is that a part of linguistic information in the POS tags and the word-internal structures may be overlapping.

7 Conclusions

This paper presents a thorough study on internal structures of Chinese words. First, we annotate a high-quality word-internal structure treebank covering over 30K words in CTB5, named as WIST. Second, we perform analysis on WIST from different perspectives and draw many interesting findings on Chinese word-formation patterns. Third, we propose word-internal structure as a new task, and present benchmark results using a popular dependency parser. Finally, we conduct preliminary experiments with two simple methods, i.e., LabelCharLSTM and LabelGCN, to encode word-internal structure as extra word representation, and find promising performance gains on the sentence-level dependency parsing task. Analysis shows that the rich dependency labels adopted in WIST play a key role, and word-internal structure is most beneficial for rare word representation.

Acknowledgments

The authors would like to thank the anonymous reviewers for the helpful comments. We are very grateful to Guodong Zhou for the inspiring discussions and suggestions on Chinese word-internal structures. We thank Kaihua Lu for building the annotation system, and Mingyue Zhou, Haoping Yang, and Yahui Liu for their help in compiling annotation guidelines, and all the annotators for their hard work in data annotation. This work is supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002104.

References

Alexandra Y. Aikhenvald. 2007. *Typological distinctions in word-formation*, 2 edition, volume 3, page 1–65. Cambridge University Press.

Shaosheng Cao, Wei Lu, Jun Zhou, and Xiaolong Li. 2018. cw2vec: Learning Chinese word embeddings with stroke n-gram information. In *Proceedings of AAAI*, pages 5053–5061.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, pages 740–750.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huan-Bo Luan. 2015. Joint learning of character and word embeddings. In *Proceedings of IJCAI*, pages 1236–1242.

Fei Cheng, Kevin Duh, and Yuji Matsumoto. 2014. Parsing chinese synthetic words with a character-based dependency model. In *Proceedings of LREC*, pages 67–72.

Fei Cheng, Kevin Duh, and Yuji Matsumoto. 2015. Synthetic word parsing improves chinese word segmentation. In *Proceedings of IJCAI*, pages 262–267.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *TACL*, 4:357–370.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of ICLR*.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? an analysis. In *Proceedings of NAACL-HLT*, pages 999–1010.

Chen Gong, Zhenghua Li, Min Zhang, and Xinzhou Jiang. 2017. Multi-grained Chinese word segmentation. In *Proceedings of EMNLP*, pages 692–703.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Morgan and Claypool.

Yuxuan Lai, Yijia Liu, Yansong Feng, Songfang Huang, and ongyan Zhao. 2021. Lattice-bert: Leveraging multi-granularity representations in chinesepre-trained language models. In *Proceedings of NAACL*.

Haonan Li, Zhisong Zhang, Yuqi Ju, and Hai Zhao. 2018. Neural character-level dependency parsing for Chinese. In *Proceedings of AAAI*, pages 5205–5212.

Yixuan Li, Kim Gerdes, and Dong Chuanming. 2019a. Character-level annotation for chinese surface-syntactic universal dependencies. In *Proceedings of Depling*, pages 216–226.

Zhenghua Li, Xue Peng, Min Zhang, Rui Wang, and Luo Si. 2019b. Semi-supervised domain adaptation for dependency parsing. In *Proceedings of ACL*, pages 2386–2395.

Zhongguo Li. 2011. Parsing the internal structure of words: A new paradigm for Chinese word segmentation. In *Proceedings of ACL*, pages 1405–1414.

Zhongguo Li and Guodong Zhou. 2012. Unified dependency parsing of Chinese morphological and syntactic structures. In *Proceedings of EMNLP*, pages 1445–1454.

Qian Lin, Huating Wen, Jing Yang, Xin Liu, Huan Lin, Hongji Wang, and Jinsong Su. 2020. Establishment of corpus of internal hierarchical structure for chinese words (in Chinese). *Journal of Xiamen University (Natural Science)*, 59:83–88.

Ting Liu, Jinshan Ma, and Sheng Li. 2006. Building a dependency treebank for improving chinese parser. *Journal of Chinese Languige and Computing*, 16(4):207–224.

Xuezhe Ma and Eduard Hovy. 2017. Neural probabilistic model for non-projective MST parsing. In *Proceedings of IJCNLP*, pages 59–69.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard H. Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of ACL*, pages 1403–1414.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of EMNLP*, pages 1506–1515.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop on ICLR*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, page 2227–2237.

Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. VCWE: Visual character-enhanced word embeddings. In *Proceedings of NAACL*, pages 2710–2719.

Fei Xia. 2009. *The Segmentation Guidelines for the Penn Chinese Treebank (3.0)*. University of Pennsylvania.

Jian Xu, Jiawei Liu, Liangang Zhang, Zhengyu Li, and Huanhuan Chen. 2016. Improve Chinese word embeddings by exploiting internal structure. In *Proceedings of NAACL-HLT*, pages 1041–1050.

Jinxing Yu, Xun Jian, Hao Xin, and Yangqiu Song. 2017. Joint embeddings of Chinese words, characters, and fine-grained subcharacter components. In *Proceedings of EMNLP*, pages 286–291.

Shiwen Yu, Huiming Duan, Xuefeng Zhu, Bin Swen, and Baobao Chang. 2003. Specification for corpus processing at peking university: Word segmentation, pos tagging and phonetic notation (in Chinese). *Journal of Chinese Language and Computing*, 13(2):121–158.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2013. Chinese parsing exploiting characters. In *Proceedings of ACL*, pages 125–134.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Character-level Chinese dependency parsing. In *Proceedings of ACL*, pages 1326–1336.

Yu Zhang, Zhenghua Li, and Zhang Min. 2020. Efficient second-order TreeCRF for neural dependency parsing. In *Proceedings of ACL*, pages 3295–3305.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of EMNLP*, pages 2205–2215.

Hai Zhao. 2009. Character-level dependencies in Chinese: Usefulness and learning. In *Proceedings of EACL*, pages 879–887.

Dexi Zhu. 1982. *Lexical Notes on Chinese Grammar (in Chinese)*. The Commercial Press.