Journal of Artificial Intelligence Research 73 (2022) 461-509 Submitted 05/2021; published 01/2022

Neural Character-Level Syntactic Parsing for Chinese

Zuchao Li CHARLEE@SJTU.EDU.CN
Junru Zhou ZHOUJUNRU@SJTU.EDU.CN
Hai Zhao ZHAOHAI@CS.SJTU.EDU.CN
Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China

Zhisong Zhang ZHISONGZ@CS.CMU.EDU
Language Technologies Institute
Carnegie Mellon University, Pittsburg, USA

Haonan Li HAONANL5@STUDENT.UNIMELB.EDU.AU
School of Computing and Information Systems
the University of Melbourne, Melbourne, Australia

Yuqi Ju TONGKONG@SJTU.EDU.CN
Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China

Abstract

In this work, we explore character-level neural syntactic parsing for Chinese with two typical syntactic formalisms: the constituent formalism and a dependency formalism based on a newly released character-level dependency treebank. Prior works in Chinese parsing have struggled with whether to define words when modeling character interactions. We choose to integrate full character-level syntactic dependency relationships using neural representations from character embeddings and richer linguistic syntactic information from human-annotated character-level Parts-Of-Speech and dependency labels. This has the potential to better understand the deeper structure of Chinese sentences and provides a better structural formalism for avoiding unnecessary structural ambiguities. Specifically, we first compare two different character-level syntax annotation styles: constituency and dependency. Then, we discuss two key problems for character-level parsing: (1) how to combine constituent and dependency syntactic structure in full character-level trees and (2) how to convert from character-level to word-level for both constituent and dependency trees. In addition, we also explore several other key parsing aspects, including different character-level dependency annotations and joint learning of Parts-Of-Speech and syntactic parsing. Finally, we evaluate our models on the Chinese Penn Treebank (CTB) and our published Shanghai Jiao Tong University Chinese Character Dependency Treebank (SCDT). The results show the effectiveness of our model on both constituent and dependency parsing. We further provide empirical analysis and suggest several directions for future study.

1. Introduction

Chinese natural language processing suffers because of an obvious aspect of writing Chinese: there is no clear separator between Chinese words because Chinese is written in a consecutive character sequence. Typically, this is solved using necessary pre-processing before word-level language processing tasks like syntactic or semantic parsing; that is word segmentation. However, wordhood in Chinese is loosely defined, so there are multiple Chinese word segmentation conventions that all meet theoretical linguistics standards from different perspectives. Take the sentence "这动画片超暴力诶$_{this\ cartoon\ movie\ is\ super\ violent}$" as an example. Some word segmenters cut this sentence into "这$_{this}$ / 动画片$_{cartoon\ movie}$ / 超暴力$_{super\ violent}$ / 诶$_{eh}$", and some segmenters cut that into "这$_{this}$ / 动画$_{cartoon}$ / 片$_{movie}$ / 超$_{super}$ / 暴力$_{violent}$ / 诶$_{eh}$" due to different standards for wordhood. Since characters are segmented into words during pre-processing, word segmentation is the basis of all later language processes, so multiple or non-standardized word segmentations may lay a loose underlying foundation for all the later processing tasks.

Also, from the perspective of Chinese grammar theory, characters in Chinese may be more analogous to words in English: "*there is such a unit: between the phoneme and the sentence, it is a unit that ordinary people who don't understand linguistics are aware of, talk about, and use regular vocabulary to describe it; that unit is 'sociological word', in English it is 'word', but in Chinese, it is 'character'. (存在这样一个单位：大小在音素和句子之间，是不懂语言学的普通大众都意识到、谈论到、并用日常词汇描述的单位；这个单位是'sociological word'，在英语中是'词'，而在中文中是'字'。)*" (Yuan & Lv, 1979). Additionally, when considering the lexical integrity hypothesis, "words are analogous to atoms in that, from the point of view of syntax, words do not have any internal structure and are impenetrable by syntactic operations" (Ackema & Neeleman, 2002). It is obvious that in this hypothesis, we should make an analogy between English words and Chinese characters instead of Chinese words because Chinese characters are grammatically atomic, have no internal grammar structure, and are impenetrable by syntactic operations, unlike Chinese words. For example,

辶文 / 云力 / 一由凵 / 丄丿㇆ / 走召 / 日共水 / 丿㇆ / 言矣

is the substructural sequence of the Chinese characters "这$_{this}$ / 动$_{move}$ / 画$_{painting}$ / 片$_{movie}$ / 超$_{super}$ / 暴$_{violent}$ / 力$_{strength}$ / 诶$_{eh}$" and conforms to the lexical integrity hypothesis, while the sequence of words "这$_{this}$ / 动画片$_{cartoon\ movie}$ / 超暴力$_{super\ violent}$ / 诶$_{eh}$" obviously does not. This comparison and example may also show that the right counterpart for English characters (letters) is Chinese character components (as shown in the above sequence) rather than Chinese characters. Chinese words are thus not atomical like English words, and pre-processing Chinese characters into words potentially ignores Chinese words' salient internal structure information (i.e., syntactic information) and hampers language understanding.

Since our previous work (Zhao, 2009) pointed out that Chinese parsing also suffers because of the vague definition of words in Chinese, and we for the first time introduced character-level syntactic structure for a better formalism of Chinese sentence in Zhao, Kit, and Song (2009), a series of work (again including ours) have explored character-level parsing as an alternative in syntactic parsing (Li & Zhou, 2012; Zhang et al., 2014; Li et al., 2018). Chinese character-level parsing, which was proposed as an alternative to word-level parsing, has two benefits: (1) using character-level trees circumvents the issue that no universal standard exists for Chinese word segmentation, and (2) in-depth structure inside words offers additional information for deeper level processing and better understanding of the whole sentence. It is worth noting that, according to the discussion in Yan (2009), from the perspective of Chinese modern linguistic theory, Chinese characters should be first regarded as morphemes, but they are also different from morphemes as characters may be linguistically used as independent single-character words, while morphemes are generally regarded as units that cannot be independently used in linguistics. However, taking characters as the basic language processing units does not lose any generality, as characters are truly atomic writing units in Chinese. Besides, morphemes may be parts of larger units, meaning they may be units of larger constituents. The tasks of parsing a Chinese word's internal structure may be more aptly described as continuous string parsing and bi-character dependency parsing, but in keeping with previous literature, we will use the dependency and constituency terminology for describing the word-internal syntactic structure as in word-level parsing.

The need for the first benefit has been demonstrated repeatedly, as linguistic views about Chinese word standard have constantly diverged. Since the first SIGHAN Bakeoff shared task for Chinese word segmentation (Sproat & Emerson, 2003), many different Chinese word segmentation standards have been proposed. Until Bakeoff-4 (Jin & Chen, 2008), there were seven kinds of word segmentation conventions. The discrepancies between these standards arose because there are no obvious answers to whether a Chinese character sequence should be divided and how granular segmentation should be.

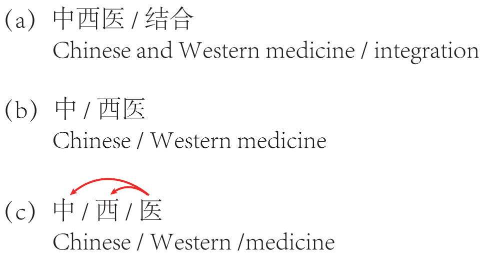
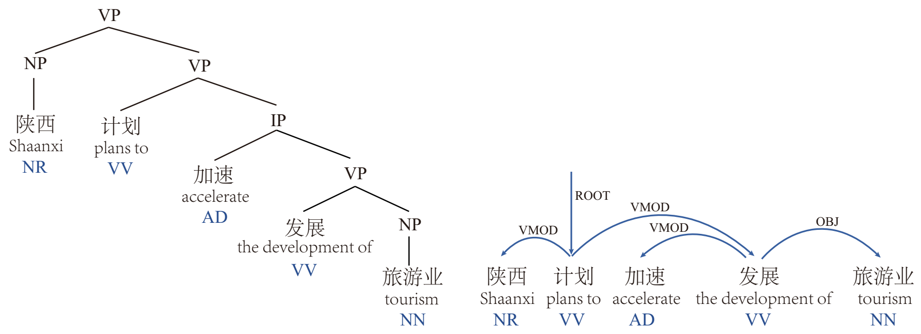

(a) 中西医 / 结合
Chinese and Western medicine / integration
(b) 中 / 西医
Chinese / Western medicine
(c) 中 / 西 / 医
Chinese / Western /medicine

Figure 1: Example of different segmentation choices from Li et al. (2018).

Figure 1 illustrates a word segmentation case. Figure 1(a) shows that the three-character segment 中西医 is a noun phrase in a sentence. Figure 1(b) gives an intuitive but insufficient example for the last two characters, which indeed compose a true word, *Western medicine*. Figure 1(c) gives another unsatisfactory and problematic segmentation in which each character is a single-character word. Neither of the above segmentations is semantically proper, as the character sequence 中西医 actually means *Chinese medicine and Western medicine* and both the first character (meaning: *Chinese*) and the second (meaning: *Western*) are modifiers of the third (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 later impede syntactic parsing, as word segmentation is a key and early aspect of all language processing pipelines. Though treating Chinese word segmentation as a sequence learning task has been successful in recent years, as seen in our previous work (Zhao et al., 2010; Zhao, 2011; Cai & Zhao, 2016; Cai et al., 2017), the linguistic difficulties still persist and hamper further progress.

The second benefit, that of the additional information about the internal structure of Chinese words, is also particularly relevant. It has been noticed for a long time that a



(a) Constituency structure

(b) Dependency structure

Figure 2: Constituent and dependency structures.

lot of Chinese words have internal structures as noted in Zhao (2009) and Zhang et al. (2013). A Chinese word contains one or more characters that subtly affect each other, both syntactically and semantically. For example, 窗$_{\text{window}}$ and 帘$_{\text{screen}}$ together create 窗帘$_{\text{curtain}}$, which indeed is a literal translation of *the screen on the window*. Additionally, as demonstrated by the empirical results in Zhang et al. (2014), including character-based information brings better performance for syntactic parsing.

In this paper, we are concerned with the combination of constituent and dependency syntactic structure in character-level parsing. Shown in Figure 2, the constituency and dependency formalisms are two typical syntactic structure representations that have been well studied from both linguistic and computational perspectives (Chomsky, 1981; Bresnan et al., 2015). Previous works on Chinese character-level parsing mainly focused on the dependency structure (Li et al., 2018; Zhang et al., 2014), while works on the constituent structure were often related to Chinese word segmentation. To the best of our knowledge, only Zhang et al. (2013) proposed character-level parsing for the constituent structure using a traditional method. By integrating constituent information in character-level parsing, syntactic trees can be effectively converted from the character-level to the word-level without relying on character-level POS tags (Li et al., 2018).

In addition, we also include joint learning of POS tags in our model, which means our model can be regarded as a truly end-to-end framework , as it does not have an additional POS tagger or word segmenter. In fact, nearly all parsing models are still based on the traditional pipeline-styled workflow; even other character-level parsing models also rely on extra POS taggers, as seen in Li et al. (2018), in which each step takes as input the output of the previous step, resulting in an inevitable substantial error accumulation. Figure 3 compares Chinese sentence processing pipelines between traditional parsing models and our joint learning character-level parsing models.

Since pre-training models have been successful with a wide range of NLP tasks, particularly improving parsing performance (Kitaev & Klein, 2018; Zhou & Zhao, 2019), we also experimented with character-level syntactic parsing using Chinese pre-trained language models (BERT, Devlin et al., 2018; RoBERTa, Liu et al., 2019; XLNet, Yang et al., 2019; ALBERT, Lan et al., 2019; and ELECTRA, Clark et al., 2019) to evaluate pre-trained models' broad performance impact on character-level parsing.

In summary, the main contributions of this work[1] are:

(1) This work makes the first attempt to combine constituent and dependency structure in Chinese character-level parsing using *joint span* structure (Zhou & Zhao, 2019) which is related to the head-driven phrase structure grammar (HPSG) (Pollard & Sag, 1994) that can incorporate head and phrase information of dependency and constituent syntax representations. Based on the *joint span* structure, we propose an effective way to infer syntactic trees from the character-level to the word-level without relying on character-level POS tags.

(2) For Chinese, character-level dependencies have been previously explored by Zhang et al. (2014), who used ZDep[2] annotations, and Li et al. (2018), who used SCDT (Zhao, 2009)[3]. We make a comparison between these two annotations, which differ in character-level POS tags and dependency labels, and we then show the advantages of our SCDT annotations.

(3) We provide a truly end-to-end character-level parsing model by using character-level annotations and joint learning of Char-POS tags and syntactic parsing. By finally providing a full structure decomposition for Chinese sentences rather than enhancing word segmentation or POS tagging to improve parsing (as in Zhao, 2009; Li and Zhou, 2012; Zhang et al., 2013, 2014), our model can avoid the error propagation often present in pipelines.

(4) Our experimental results show that our *joint span* character-based parser brings better prediction on both constituent and dependency tree structures. In addition, the empirical results show that coupled with pre-trained models, our parser reaches new state-of-the-art in character-level evaluation and competitive results in word-level evaluation for both parsing tasks.

2. Character-Level Structure

In this section, we focus on character-level structure for both the constituent and dependency styles. In Chinese syntactic parsing, words are usually regarded as the atomic processing unit, and the leaves in constituent trees and the nodes in dependency trees represent the whole words. However, there still exists a smaller unit in Chinese: characters; therefore, another possible level of analysis which goes deeper to the most basic composition units for Chinese: character-level. At the character-level, by taking characters instead of words as the basic units, the same syntactic analysis performed at the word level is possible at the character-level. Just as syntax trees can be built oriented around words, character-oriented constituent or dependency trees can also be established.

[1] Part of the annotations of the SCDT dataset in this work was published in our AAAI conference paper (Li et al., 2018). In this journal version, the difference from the AAAI version is that we 1) supplemented a more complete linguistic motivation, 2) carried out a more complete introduction for the annotation details and comparison between related works, 3) proposed new parsing models, 4) conducted a detailed study on the performance of SCDT on downstream tasks and conducted a more in-depth discussion and analysis.

[2] https://github.com/zhangmeishan/ACL2014-CharDep/

[3] SJTU (Shanghai Jiao Tong University) Chinese Character Dependency Treebank

[5] Word tagging information is copied to all internal characters to help build a full character-level dependency tree.

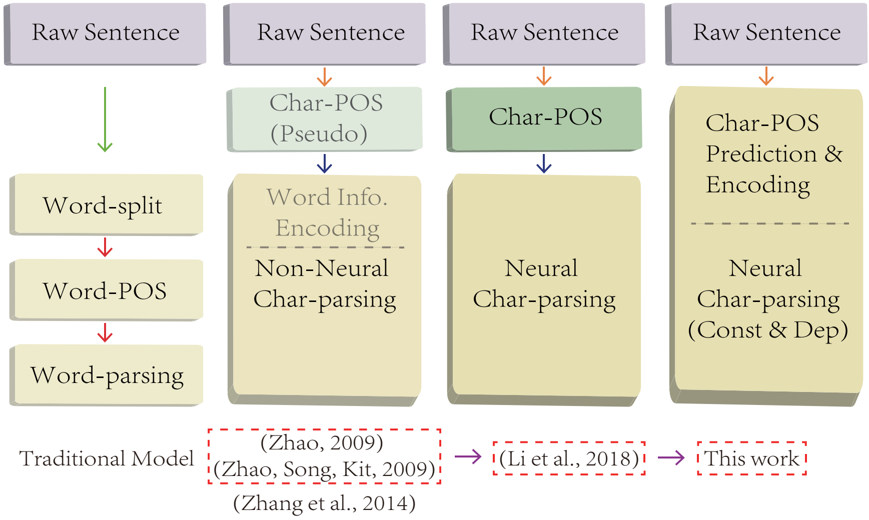


Figure 3: A traditional parsing pipeline, two previous parsing pipelines, and our model's pipeline[5]. The red dashed box shows our previous and current work.

We manually annotated such character-level dependencies for all the words in the Chinese Penn Treebank (CTB-7.0). We name the resulting corpus as SJTU Chinese Character Dependency Treebank (SCDT)[6]. We will first describe the annotation guidelines of SCDT.

For the character-level structure of constituents, we construct the constituent representation of SCDT based on the Head Feature Principle (HFP) (Pollard & Sag, 1994) in which there is only one head word for each phrase. There is another character-level treebank (Zhang et al., 2013), which we refer to as ZCTB. It contains constituent structures for words and can be converted to dependency structures following Zhang et al. (2013). We will compare between SCDT and ZCTB, looking at both constituent and dependency structures and demonstrate the advantages of our annotations.

2.1 Annotations of SCDT

Our SCDT was manually labeled and transformed in steps according to linguistic hierarchies. First, we manually labeled the POS tags of Chinese characters inside words. Then, we defined the head rules according to the dependency grammar (Robinson, 1970; Mel'cuk et al., 1988) to get the dependency structure inside words. We then transformed the dependency structure into an intra-word constituent structure according to the HFP. These head rules we use here are consistent with the head rules used to transform word-level constituent trees into dependency trees in terms of linguistic theory. The only differences are the tag set and transfer rules. Therefore, this transformation has a solid linguistic foundation.

[6] http://bcmi.sjtu.edu.cn/~zebraform/

2.1.1 Annotation Goal

The general annotation strategy we used for character-level dependencies is proposed by Zhao (2009). We assume that the annotators have basic knowledge about the Chinese language for both writing and speaking. Most of the annotated words are extracted from Chinese Penn Treebank (CTB-7.0), and consequently during the annotating, annotators have to retrieve the original text of CTB to fully understand the meaning of some words. The goal of our annotation is to annotate all character-level dependencies with both character-level POS tags and dependency labels for all words.

2.1.2 Annotation Format

In this subsection, we will briefly describe the annotations presented later in this paper. For an example, take the three-character Chinese word **天安门**$_{\text{Tian'anmen Square}}$. Each character in the word is indexed from **1** to **3**, so we have indices for "**1:天, 2:安, 3:门**". Next, structural annotation will be done by finding the head for each character. As **天**'s head is **安**, **安**'s head is **门** , and **门** is the root of this word. We denote this structure with "**1:天 2, 2:安 3, 3:门 0**". Here, index **0** signifies the root character. For annotations where dependency labels are required, we add the labels for the dependencies: "**1:天 2 *n-n-v***", "**2:安 3 *nn***", "**3:门 0 *root***". For the above words, there are three types of dependency labels: ***n-n-v***, ***nn***, and ***root***. For simplicity, we will write the annotation results using the following description: "**天安门; 2 3 0; *n-n-v nn root***".

2.1.3 Special Word Types

We distinguish different word types for the annotation. Some special word types are described as follows:

1. Single-character words: since there will be only one character, no annotation for this type will be required.

2. Numbers and letters: words that only consist of numbers or letters, for example:

   一百万; 2 3 0; *g g root*
   100,000; 2 3 4 5 6 7 0; *g g g g g g root*
   NATO; 2 3 4 0; *l l l root*

   For these words, annotation for each character should be done in an incremental order: each character takes its right neighbor as its head and the last character as the root. We use dependency label ***g*** for number words, and ***l*** for letter words.

3. Named entities: words referring to people, locations, organizations, or other translated names, such as:

   胡锦涛; 2 3 0; *r r root*
   上海; 2 0; *r root*
   纽约; 2 0; *r root*
   奥巴马; 2 3 0; *r r root*

Like numbers and letters, annotation is also done for each character in an incremental order; each character takes its right neighbor character as its head and the last character as the root, and ***r*** is the dependency label of the named entity.

4. Mixed names: a mixed name typically includes a non-Chinese name part and other Chinese characters as follows:

   纽约城; 2 3 0; *r nn root*

   Here, the word consists of two parts: **纽约** (New York), which is a named entity and **城**, which is the Chinese character denoting city.

2.1.4 Generic Annotation

In this subsection, we will describe how to annotate the basic dependencies between characters to represent their syntactic and semantic relationships. We adopt an iterative annotation strategy. For example, considering a four-character word: ***abcd***, where ***abc*** is first identified as an adjective constituent, and then ***d*** is identified as a noun constituent. We could mark ***abc*** as the modifier and ***d*** as its head. Then, inside ***abc***, ***bc*** is identified as a verb constituent and ***a*** as an adverbial constituent, so ***bc*** will be selected as the head for ***abc***. Finally, in the last multiple character part, ***bc***, ***c*** is taken as the head. Thus, after this iterative annotation, we get the final annotation for the word, "***abcd*; 2 3 4 0**". There are no absolute rules for determining which character should be the head, but we propose basic rules to direct the annotation. With this understanding of the basic strategy of our annotating procedure, we now describe in detail the basic components in our annotations.

Word-Level POS Tags First, we need POS tags for the annotated words. Since the annotation will be done without considering the context of the word, we will give a simplified POS set for words. We define a group of "absolute POS tags" for words as shown in Table 1. Note that Chinese is a language whose words may hold quite different POS tags compared to how they are really used in the text, and nearly no Chinese word only fits one possible POS tag. For this reason, when we say an absolute POS tag for a word, it means to choose a possible tag from a priority list for the word. The following are the rules for making the right choice in selecting the absolute POS tag for a given word:

Table 1: Word-level absolute POS tags.

Annotation	**POS tag**	**Example**
p	Pronoun	这个(this one), 他们(they)
n	Noun	大门(main door), 汽车(automobile)
v	Verb	书写(write), 玩耍(play), 举行(held)
a	Adjective	热闹(alive), 邪恶(evil), 平坦(plain)
d	Adverb	最近(recently), 很难(difficultly), 大概(about)

1. The most popular case for multiple POS involves the words that can either be nouns or verbs. The rule for this is if a word can act as both a verb and as a noun, then its absolute POS is verb.

2. For a word that can be either an adjective or a noun, its absolute POS is set to adjective. To distinguish those words with nebulous POS, we set a semantic criterion to make the decision: nouns refer to real entities, and adjectives give property descriptions.

3. For a word that can be either an adjective or an adverb, the absolute POS for it is adverb.

The absolute POS tag determines the most essential syntactic categories of words and is then used for syntactic structure derivation. This derivation based on absolute POS tags can be done accurately because a segmenter-created word having different context-dependent meanings or internal syntactic structures is a rare linguistic phenomenon. In fact, in Chinese language processing practice, when a "character string" does have such context-dependent syntactic or semantic ambiguities, it may have been already segmented differently in the first place by the word segmenter. Traditional Chinese word segmentation needs to solve two kinds of ambiguities: combinational ambiguity and intersection ambiguity. Chinese word segmentation thus immediately makes decisions to remove a large portion of syntactic ambiguity at the very beginning of language processing. Taking Chinese sentence "鲁滨逊的仆人星期五下午把棚子搭起来了" (Robinson's Servant Friday set up the shed in the afternoon) and sentence "星期五下午把棚子搭起来了" (Set up the shed on Friday afternoon) as examples, the former sentence should be cut into

"鲁滨逊/的/仆人/星期五/下午/把/棚子/搭/起来/了",

while the latter should be cut into

"星期五下午/把/棚子/搭/起来/了".

This explains why each Chinese word rarely (or almost never) has different contextual internal structures and meanings. Therefore, we propose to use character-level POS tags in conjunction with word-level absolute POS tags as the subsequent head rule transformation basis.

Character-Level POS Tags Next, we will delve into finer annotations for the characters inside the words. Character-level POS tagging will be dynamic or contextual for these annotations, just like typical word-level POS tagging, which is quite different from the above absolute POS assignment without any context for the annotated word.

The adopted character-level POS tags are shown in Table 2. Note that the goal of character-level POS tagging is to resolve the POS ambiguity that exists at the word-level. For example, 书 is mostly considered as a noun character in 读书$_{\text{read book}}$, but for 书写$_{\text{write, writing}}$ 书 is actually a verb character. This is because in ancient Chinese, 书 was typically a verb. Thus, the character-level POS annotation is context-sensitive, meaning all the characters in the whole word should be taken into consideration.

Table 2: Character-level POS tags.

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	Adverb	很(very), 最(most)
f	Functional character	的(of), 们(-es), 在(at)

Head Rules Next, we will describe the core part of the annotation procedure; that is, the annotation of the dependencies themselves: the head rules. Assuming that the absolute POS tag for the given word and all character-level POS tags have been determined, we finish the annotation by utilizing the following head rules, which observe the general principles defined by traditional dependency grammar. For the dependency labels presented in examples, we will describe them in the next part.

1. For a noun character and its modifier characters in a noun, the noun character should be the head. For example:

   蓝天; 2 0; *ad root*
   一本书; 2 3 0; *ad ad root*

2. For a given word, the kernel verb character or the verb part (if there is only one) should be the root for the word. For example:

   吃掉; 0 1; *root cd*
   快跑; 2 0; *vv root*

3. For a coordinate structure, the right character will be always be the head. For example:

   玩耍; 2 0; *vc root*
   神仙; 2 0; *nc root*

4. If a functional character is involved in a constituent, then it should be always the head. For example:

   好的; 2 0; *af root*
   我们; 2 0; *pf root*

Dependency Label Set Finally, we will also provide labels for the dependencies annotated in the previous step. For a dependency relation, its dependency label is generally

Algorithm 1 Constituent-Construction(u)

```
$u$ is a node of character-level dependency tree.
if $u$ is a leaf of the character-level dependency tree then
    constructing constituent leaf node leaf-node using $u$
    return leaf-node
end if
children-list = []
for each child $v$ on the left of $u$ do
    left-node = Constituent-Construction($v$)
    putting left-node into children-list
end for
constructing constituent leaf node leaf-node using $u$
putting leaf-node into children-list
for each child $v$ on the right of $u$ do
    right-node = Constituent-Construction($v$)
    putting right-node into children-list
end for
constructing constituent node constituent-node using children-list
return constituent-node
```

determined by POS tags of three constituents; i.e., those of its modifier, its head, and their concatenation.

For a dependency relation where ***a*** is the modifier and ***b*** is the head, the corresponding character subsequence should follow ***a...b***, assuming there are some other characters between ***a*** and ***b***. If $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, and ***p*** are the POS tags of ***a***, ***b***, and ***a...b***, respectively, then the dependency label should be $\boldsymbol{p}$-$\boldsymbol{p}_1$-$\boldsymbol{p}_2$. For the case that ***a*** is included in an inseparable constituent such as a named entity created from transliteration, $\boldsymbol{p}_1$ should be the POS tag of the constituent, and ***a*** should be its head.

For convenience, we will define a group of abbreviations for the frequent dependency labels, specifically: ***ad*** for ***n-a-n***, ***af*** for ***a-a-f***, ***pf*** for ***p-p-f***, ***vd*** for ***v-d-v***, ***cd*** for ***v-v-d***, ***nn*** for ***n-n-n***, ***vv*** for ***v-v-v***. For other labels that appear less frequently, the original format, such as ***d-d-f***, will be used for the annotations.

For a coordinate relation of noun, verb, adjective and adverbial, we correspondingly define four types of labels, ***nc***, ***vc***, ***ac***, and ***dc***. The following are two examples:

凤凰; 2 0; *nc root*,
书写; 2 0; *vc root*.

Thus, ***nc*** and ***vc*** can be regarded as ***n-n-n*** and ***v-v-v***, respectively; however, these are different from ***nn*** and ***vv***, which aim to represent modification relations between two noun or verb characters. And for different word types, including numbers, letters, and inseparable named entities, we use three labels to identify them, respectively: ***g*** for numbers, ***l*** for letters, and ***r*** for named entities.

2.2 Constituent Representations of SCDT

Following the method of our previous work (Li et al., 2018), which focused on Chinese character-level dependency parsing, we further explore Chinese character-level constituent parsing and the relationship between character-level constituent parsing and dependency

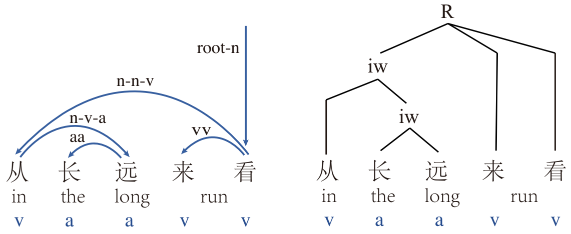


Figure 4: Constituent and Dependency Representation of SCDT

parsing. Since SCDT annotations only include character-level dependencies for the words of CTB without constituent representations, we construct the constituent representation of the words based on the HFP that there is only one head word for each phrase.

Algorithm 1 shows the pseudo-code for constructing a constituent representation of the words from a character-level dependency tree using a recursive method. We first traverse the character-level dependency tree from the root. For each internal node u of the dependency tree, we construct a constituent node that includes all the converted children of u and the constituent leaf node of u, and then, we return this constituent node.

We assign the special category ***iw*** to each node of the word-level constituent tree besides the root. Because the representations of two-character words are almost the same as their corresponding word-level representations, in order to distinguish from the word-level representations, we use the special token ***R*** as the category of root in character-level constituent trees. The corresponding constituent and dependency word-level structure is shown in Figure 4. In word-internal syntax structures, the constituent representation shows the composition level of the word, while the dependency representation indicates which characters dominate others.

Analyzing and assigning categories to characters is a complicated topic in linguistics because the "syntactic distributions" of Chinese characters are rather unclear and may not provide sufficient categorization evidence. Yang (2006) manually annotated 39,459 words from the Modern Chinese Grammar Information Dictionary. Their motivation is similar to that of our work on constituent representation. However, our annotation is much more comprehensive than theirs. Considering that part-of-speech should be essentially defined as the most basic syntactic category, we perform the complete syntactic structure annotation by decomposing complex annotation problems into multiple hierarchies. Due to the lack of evidence, annotating categories directly has the risk of uneven quality. We thus annotated gradually and according to linguistic hierarchies. We first annotated the POS tags of characters, and then we used the dependency head rules and HFP to provide categories.

2.3 Comparisons

For Chinese, character-level constituents and dependencies structures have been previously explored by Zhang et al. (2013) and Zhang et al. (2014), respectively, whose annotations

Table 3: Statistics for the two dependency treebanks. “Intersections” denotes the number of intersecting words in the two datasets, while “Identical-Structures” denotes the number of identically-structured words in the two datasets, and the last column shows the percentage of identical structures.

Word Length	**ZCTB**	**SCDT**	**Intersections**	**Identical-Structures**	**Identical Percentage**
2	40567	50639	39897	35641	89.33%
3	15972	20970	14709	8655	58.84%
4	6114	11232	5060	2468	48.77%
5	1922	1035	872	549	62.95%
>5	2610	923	788	205	26.01%
total	67185	84799	61326	47518	77.48%

have also been publicly available[7]. We denote these annotations as ZConst for constituents and ZDep for dependencies in ZCTB[8]; however, there are quite a few differences between their annotations and ours, some of which are rooted in the basic annotation schemes. We will compare our SCDT annotation with ZDep both quantitatively and qualitatively and will also compare our constructed character-level constituents with ZConst, showing that our annotation potentially provides more character-level information.

Statistics Firstly, the statistics for the two treebanks are collected based on the distributions of lengths of the words, as shown in Table 3. Comparing the character-level annotation sets, we see that SCDT includes more words than ZDep, especially for words less than five characters. In fact, most Chinese words are quite short and some of the longer words can be further cut into short words, indicating that SCDT could have better coverage of Chinese words.

Structures Since the two treebanks adopt different annotation schemes (which we will describe later), the unlabeled version of the dependencies, which reveals only the structures, will be examined first. The statistics for intersecting words and identically-structured words are also shown in Table 3. From the statistics, we can see that for shorter words, the rates of structural agreement between the two datasets are much more higher than those for longer words. This is natural since shorter-sized words are easier to annotate.

Table 4 lists some examples of the divergences between the two datasets. Comparing the versions of unlabeled constituent trees, ZCTB treebank defines its constituency trees as binary trees in order to indicate the head direction for generating corresponding dependency trees, while our SCDT constituent treebank is converted from dependency structures and thus is not restricted to being a binary tree. We also observe that some of the disagreements occur over words like named entities and are due to the specified annotating schemes for those special words.

[7] https://github.com/zhangmeishan/ACL2013-CharParsing, https://github.com/zhangmeishan/ACL2014-CharDep

[8] ZDep can be converted from ZConst following Zhang et al. (2013)

Table 4: Examples of the structural divergence between the two treebanks (the annotations for the dependency structures are described in Section 2.1.2).

Word	ZCTB Head & Constituent	SCDT Head & Constituent
竞拍 (auction)	0 1 *(z (b 竞) (i 拍))*	2 0 *(R (v 竞) (v 拍))*
发愁 (worry)	0 1 *(z (b 发) (i 愁))*	2 0 *(R (v 发) (f 愁))*
停车场 (parking-lot)	3 1 0 *(y (z (b 停) (i 车)) (i 场))*	2 3 0 *(R (iw (v 停) (n 车)) (n 场))*
超暴力 (super-violent)	3 3 0 *(y (b 超) (y (i 暴) (i 力)))*	2 3 0 *(R (iw (v 超) (a 暴)) (n 力))*
有利可图 (profitable)	3 1 0 3 *(y (z (b 有) (i 利)) (x (i 可) (i 图)))*	2 4 4 0 *(R (iw (v 有) (n 利)) (v 可) (v 图))*
从长远来看 (in the long run)	4 1 2 0 4 *(y (z (b 从) (x (i 长) (i 远))) (x (i 来) (i 看)))*	5 3 1 4 0 *(R (iw (v 从) (iw (a 长) (a 远))) (iw (v 来)) (v 看))*
反其道而行之 (act in contravention)	5 3 1 1 0 5 *(y (z (z (b 反) (y (i 其) (i 道))) (i 而)) (z (i 行) (i 之)))*	3 3 4 6 6 0 *(R (iw (iw (v 反) (n 其) (n 道)) (f 而)) (v 行) (f 之))*

Table 5: Illustrations of different tagging schemes of the two datasets.

Word	Zdep		SCDT	
	POS	Dep. Label	POS	Dep. Label
开心(happy)	*b i*	*out in*	*v n*	*a-v-n root-a*
拥有者(owner)	*b i i*	*in in out*	*v v n*	*vc n-v-n root-n*
不起眼(inconspicuous)	*b i i*	*in out in*	*d v n*	*vd n-v-n root-n*
出乎意外(unexpected)	*b i i i*	*out in in in*	*v p n n*	*v-v-p n-v-n nn root-n*
民不聊生(destitute)	*b i i i*	*in in in out*	*n d v v*	*n-n-d n-n-v vv root-n*

Annotations The major differences between the two character-level trees are the annotations for labels: character-level POS tags, dependency labels for characters, and constituent labels for constituent tree nodes.

For character-level POS tags, we employ a more fine-grained tag-set that explores the functional utilizations of individual characters; that is, we annotate characters with real character-level POS tags like we do for words, as described in Section 2.1.4, while Zhang et al. (2014) utilizes a simple scheme that only annotates the word-level POS and the intra-word positions for the characters: ***b*** for the starting character of the word and ***i*** for the continuing ones in the rest of the word. Obviously, our annotation scheme provides far more information by indicating the fine-grained functionalities of the characters.

For dependency labels, we provide labels basing on the POS tags of the characters in the dependency edge, which is still more informative than the scheme of Zhang et al. (2014) that only discriminates between root and non-root characters (annotated as ***out*** and ***in***, respectively).

For constituent labels, ZCTB treebank uses ***x***, ***y*** and ***z*** labels that indicate the *left*, *right*, and *coordination* head directions, respectively (Zhang et al., 2013), while our SCDT converted constituent structure only contains ***R*** and ***iw*** labels, which aim to distinguish

Table 6: Examples of full annotation.

Index	Char	POS	Head	Dep. Label	Constituent Tree
Word: 鲟鱼(sturgeon)					
1	鲟	*n*	2	*nn*	*(R (n 鲟) (n 鱼))*
2	鱼	*n*	0	*root-n*	
Word: 天安门(Tian'anmen)					
1	天	*n*	2	*n-n-v*	*(R (iw (n 天) (v 安)) (n 门))*
2	安	*v*	3	*nn*	
3	门	*n*	0	*root-n*	
Word: 普天同庆(the whole world is celebrating)					
1	普	*a*	2	*ad*	*(R (iw (a 普) (n 天)) (d 同) (v 庆))*
2	天	*n*	4	*n-n-v*	
3	同	*d*	4	*vd*	
4	庆	*v*	0	*root-n*	
Word: 50余(more than fifty)					
1	5	*i*	2	*g*	*(R (iw (i 5) (i 0)) (a 余))*
2	0	*i*	3	*a-g-a*	
3	余	*a*	0	*root-a*	
Word: 拉美裔(Hispanic)					
1	拉	*n*	2	*r*	*(R (iw (n 拉) (n 美)) (n 裔))*
2	美	*n*	3	*nn*	
3	裔	*n*	0	*root-n*	
Word: 空对地(air to ground)					
1	空	*n*	3	*nn*	*(R (v 对) (iw (n 空) (n 地)))*
2	对	*v*	0	*root-n*	
3	地	*n*	2	*v-v-n*	
Word: 好人好事(good man and good deed)					
1	好	*a*	2	*ad*	*(R (iw (a 好) (n 人)) (a 好) (n 事))*
2	人	*n*	4	*nc*	
3	好	*a*	4	*ad*	
4	事	*n*	0	*root-n*	

the root nodes from internal nodes. Table 4 and 5 provides some examples to illustrate these different schemes.

2.4 Examples of Full Annotation

Several examples for full annotations for our character-level dependency and converted constituent structures are shown in Table 6. The annotations include character-level POS tags, head indices, dependency labels, and constituent labels. Note that we require the absolute POS tag when assembling character-level and word-level constituent dependency or dependency trees. We format these in the form of ***root-xx*** where ***xx*** denotes word-level POS tags, which will be described in the next section.

3. Complete Structure

In this section, we introduce the complete structure representation and two key problems to solve: (1) how to assemble character-level and word-level treebanks, and (2) how to combine the dependency and constituent representations in the fully annotated character-level treebank.

Firstly, we discuss how to apply dependency and constituent structures inside words (hereinafter, we refer to these as intra-word representations) to construct the full character-level treebanks for both constituency and dependency trees. Secondly, we combine the dependency and constituent representations in the fully annotated character-level treebanks to create a formal structure called *joint span* following Zhou and Zhao (2019). The *joint span* structure is able to cover both constituent and head information of syntactic trees based on HPSG, which is a highly lexicalized, constraint-based grammar.

3.1 Assembling Character-Level and Word-Level Treebanks

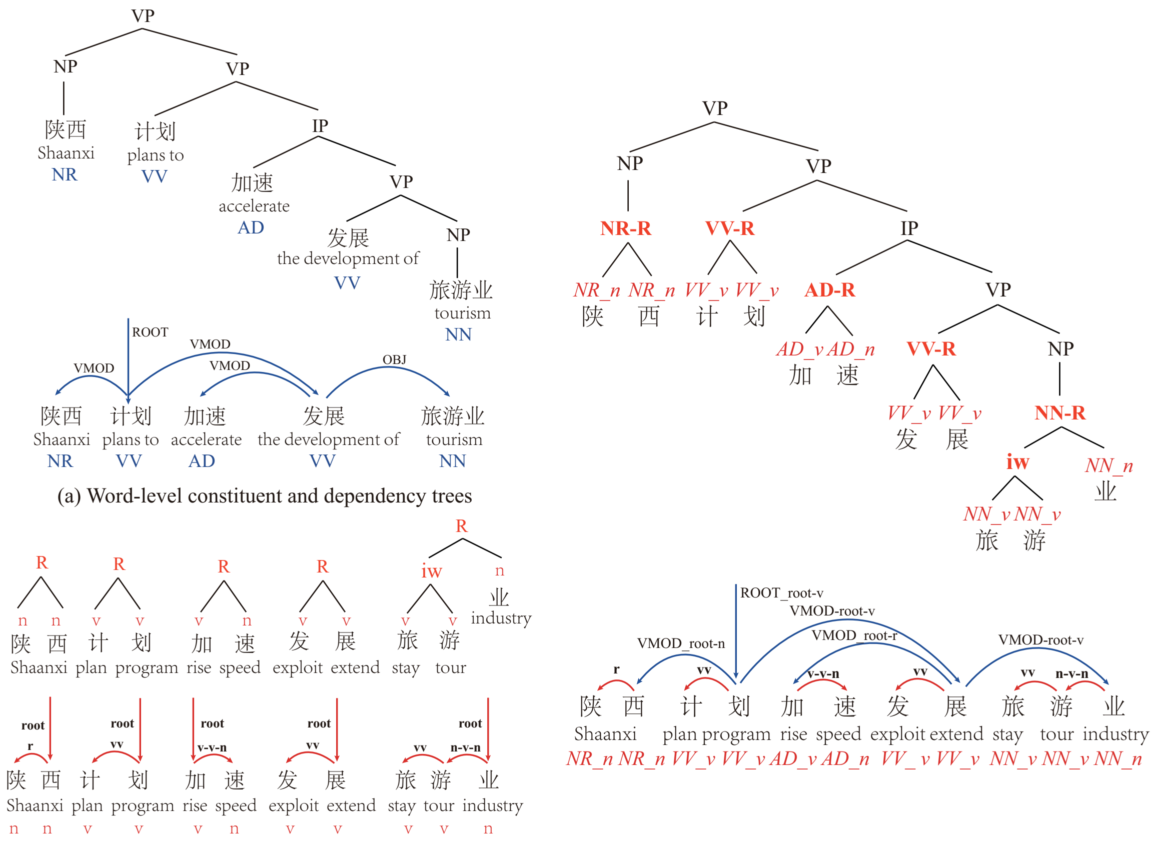
At the word-level, as in traditional methods, we convert dependency treebanks from the Chinese Penn Treebank (CTB), which uses constituent structure, as shown in Figure 5(a). At the character-level, the dependencies will be from SCDT, which includes character-level dependencies for all words of CTB. As has been already described in detail in Section 2.2, SCDT directly describes the intra-word dependencies, and we apply Algorithm 1 to generate intra-word constituent structure from SCDT. Each word is annotated with a dependency tree and a constituent tree, both of which are built on its characters, as shown in Figure 5(b).

With the word-level and intra-word structures of both constituency and dependency trees, the next task is to assemble them to construct the respective fully annotated character-level constituent and dependency treebanks.

Assembling Unlabeled Constituents and Dependencies For the character dependencies inside words, each word has its own root character, which is used to stand for the whole word in the original word-level treebank. This allows character dependencies and word dependencies to be connected naturally and without any ambiguity. Similarly, since each word is represented as a leaf in the word-level constituent structure, we can replace each leaf with a character-level constituent tree representing the word to construct fully annotated character-level constituent trees naturally.

As shown in Figure 5, the final character-level trees will be obtained by combining the following trees: its word-level constituent and dependency trees shown in 5(a) and all related intra-word (character) constituent and dependency trees for all words as shown in 5(b). Figure 5(c) shows the resulting fully annotated character-level constituent and dependency trees for the whole sentence.

Assembling Character POS Tags, Constituent Categories, and Dependency Labels Characters that belong to a word may inherit the corresponding word-level POS and dependency label from the word-level treebank, though each character already has its own



(a) Word-level constituent and dependency trees

(b) Word-internal constituent and dependency trees

(c) Character-level constituent and dependency trees

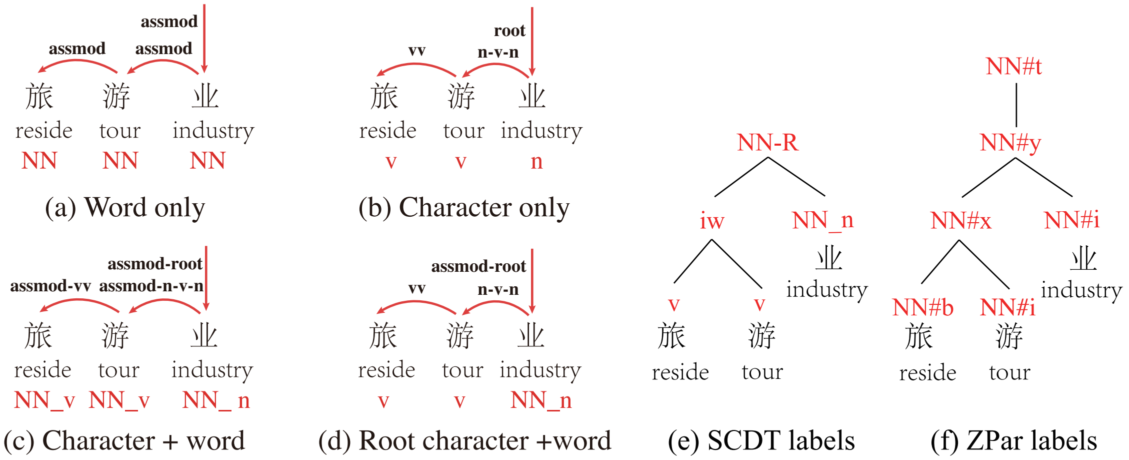
Figure 5: Character-level constituent and dependency trees (c) and the word-level and intra-word trees used to build them (a and b, respectively).

character-level tag or label from character-level treebank. In a fully annotated character dependency tree, we have to decide which detailed POS tag and dependency label, whether from the character-level tree, the word-level tree, or both should be assigned to each character. In a fully annotated character constituent tree, we need to combine word-level POS tags with the constituent categories of intra-word constituent trees.

For assembling POS tags, Zhang et al. (2014) explored a character POS tagging strategy, in which they trivially assigned tags according to character position and word POS tags from CTB[9], but this under-utilized characters tags' and labels' ability to represent the true syntactical roles of the characters within words.

While SCDT includes manual annotations of both character POS tags and dependency labels, Li et al. (2018) considered four intuitive strategies for POS and dependency labels based on SCDT as shown in Figure 6. They are specified as follows:

[9] For example, if a Chinese word has POS tag ***NN***, then the first character is labeled as ***NN-b***, and remaining characters are labeled as ***NN-i***, where ***-b*** indicates the beginning character position and ***-i*** indicates other positions.



(a) Word only

(b) Character only

(c) Character + word

(d) Root character +word

(e) SCDT labels

(f) ZPar labels

Figure 6: Different strategies for tagging character POS tags, constituent labels, and dependency labels.

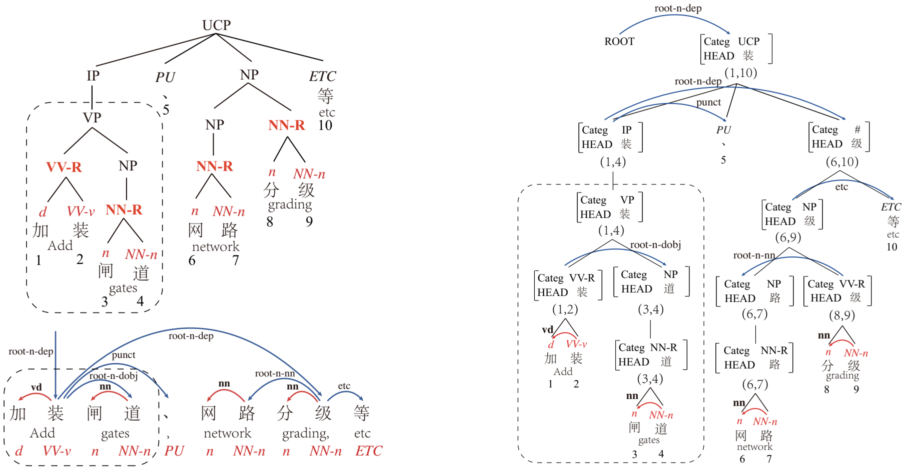
- Using only word-level tags and labels. (Figure 6(a))
- Using only character-level tags and labels. (Figure 6(b))
- Combining word-level and character-level tags and labels for all characters. (Figure 6(c))
- Combining word-level and character-level tags and labels for only root characters. (Figure 6(d))

Li et al. (2018) showed that combining word-level and character-level tags and labels for only root characters (Figure 6(d)) achieves the best performance, and we apply this strategy for our POS tags and dependency labels.

To assemble constituent categories, Zhang et al. (2013) assigned a word-level POS to each character constituent category and constructed a new root for each character constituent tree to represent the word segmentation, assigning the new root label ***t*** as show in 6(f). In our fully annotated constituent trees based on SCDT, we combine word-level POS tags with only the root node and combine word-level and character-level POS tags for only root characters in the character-level constituent trees, as shown in 6(e).

With these enhanced character tagging strategies, the final character constituent and dependency trees express both lexical and syntactic information. As for word-level constituent and dependency parsing, the learning pipeline includes identification of words and prediction of syntactic tree. As to character analysis, we only need to train the prediction of syntactic tree without word segmentation. The error propagation in different processing hierarchies will be greatly reduced which is supposed to lead to a better learning effect.

Since the word is represented as a constituent in character-level constituent tree, we can easily convert to word-level trees without POS information by constituent categories which is different from Li et al. (2018) based on POS.



(a) Constituent and dependency.

(b) Joint span structure.

Figure 7: Constituent, dependency, and *joint span* structures (Zhou & Zhao, 2019). The *joint span* structure is indexed from 1 to 10, and each node is assigned an interval range. The dotted box represents the corresponding portions. The special category # is assigned to divide phrases with multiple heads. The *joint span* structure contains both constituents and dependency arcs. The *Categ* in each node represents the set of constituent categories, which includes a collapsed entry for each unary chain of each constituent. *HEAD* indicates the head word.

3.2 *Joint Span* Structure

Since constituent and dependency representations have strong inherent linguistic relations and can benefit each other, we combine our full character-level constituent and dependency trees to create a single structure representation. We call this structure *joint span* following our previous work (Zhou & Zhao, 2019). This structure covers both constituent and head information of syntactic trees. The *joint span* structure, which is related to the HFP of HPSG, consists of all its children phrases in the constituent tree and all dependency arcs between the heads and their children in the dependency tree. When we naturally re-define Chinese sentence structure on a basis of character-level syntactic formalism, it is very natural and straightforward to use our previously developed word-level modeling approach to study the character-level parsing and special language phenomenon in Chinese.

For example, in the constituent tree of Figure 7(a), **加装闸道** (Add gates) is a phrase $(1, 4)$ assigned with category ***VP*** and in the dependency tree, **装** is parent of **加** and **道**, thus in our *joint span* structure, the head of phrase $(1, 4)$ is **装**. The node $\mathrm{S}_H(1,\ 10)$ in Figure 7(b) is represented as a joint span by:
$\mathrm{S}_H(1,\ 10) = \{\ \mathrm{S}_H(1,\ 4)\ ,\ \mathrm{S}_H(5,\ 5)\ ,\ \mathrm{S}_H(6,\ 10),\ l(1,\ 10,\ (\mathrm{UCP}))\ ,\ d(\text{、},\ \text{装})\ ,\ d(\text{级},\ \text{装})\ \}$,
where $l(i,\ j,\ (\mathrm{UCP}))$ denotes span $(i,\ j)$ with category UCP, and $d(r,\ h)$ indicates the dependency between the word r and its parent h. The final entire syntactic tree T can be represented in the *joint span* structure as:

$$S_H(T) = \{S_H(1,\ 10),\ d(\text{装},\ root)\}.$$

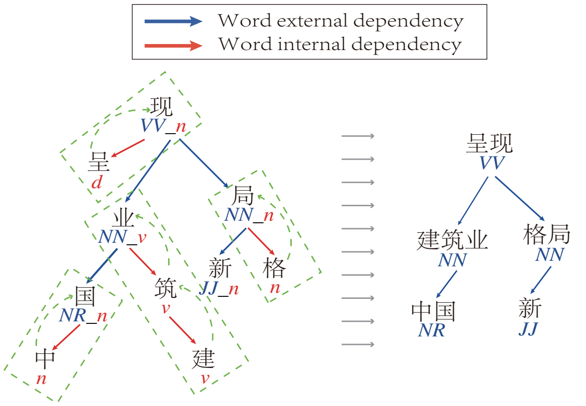


Figure 8: Word-level dependencies restored from character-level dependencies using POS tags.

As shown in Figure 7(a), the phrase (1,10) with a category UCP contains 2 head words, 装 and 级 in its dependency tree. To deal with phrases containing two or more head words, we follow Zhou and Zhao (2019) to use a special category # to divide phrases with multiple heads to ensure that there is only one head word for each phrase. We define the dependency head of tokens in constituents which violate the HFP as "error". This error still exists in the transformed dataset because the original word-level dependency treebank transformed by head rules such as Stanford (de Marneffe & Manning, 2008) does not completely follow this principle. Therefore, after the character-level conversion, there are 364 heads in SCDT and 340 heads in ZCTB fully annotated character-level trees that are errors.

Moreover, to simplify the syntactic parsing algorithm, we add a special empty category Ø to spans to binarize the n-ary nodes and apply a unary atomic category to deal with the nodes of the unary chain. This is a popular process adopted in constituent syntactic parsing (Stern, Fried, & Klein, 2017b; Gaddy, Stern, & Klein, 2018).

3.3 Word-Level Restoration

As the parsing granularity has been shifted from word to character, directly comparing with the results of word-based parsers will not be meaningful. Thus, it is necessary to restore word-level trees from character-level parsing predictions, namely using word segmentation for restoration. Li et al. (2018) shows a method to restore words based on the POS tag of each character, which we use for our dependency trees, while we propose an intuitive way to restore words based on the constituent categories of our joint span structure.

As shown in Figure 8, Li et al. (2018) adopts a heuristic method to transform parsing predictions from character-level to word-level[10]. First, find all characters that are rooted by a character whose character-level POS tag is ***root***. Then, check each root character node to see if its tag includes a word-level POS part. For example, if a character POS tag is

[10] As character-level dependency trees contain word-level information, extracting or restoring the word-level dependencies is always possible, although our character-level model does not rely on the concept of words to convey information, even if it is derived from words.

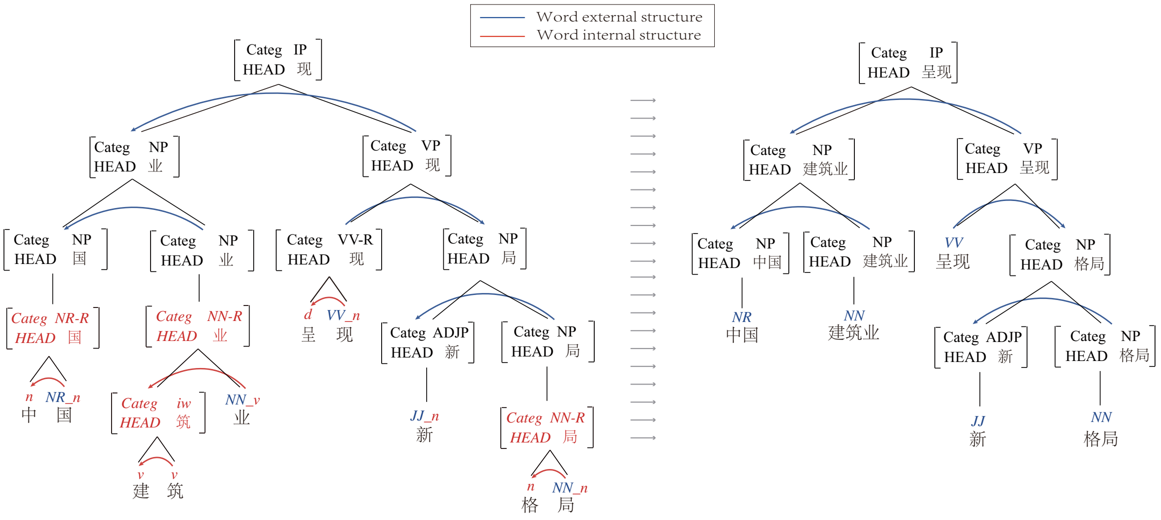


Figure 9: Word-level *joint span* structure restored from character-level *joint span* structures using constituent categories.

NN_n, then the corresponding character must be a root character because ***NN*** is known as a word-level POS tag. This word-level POS tag will be given as the tag of soon-to-be-found word. Once the root character is identified, all its descendant nodes (those with only character-level POS tags) can be collected and combined with the root character to compose a word. After all these character-level edge subtrees collapse into word-level subtrees with corresponding POS tags, a word-level dependency tree can be finally built.

Li et al. (2018)'s word-level restoration is based on character-level POS tags and performed on dependency tree, so its word segmentation results may conflict with those of constituent trees. Thus, we propose a word-level restoration method based on the categories of our *joint span* structure that can transform both constituent and dependency trees from the character-level to the word-level.

As shown in Figure 9, if the category in the node of our joint span structure tree indicates ROOT for the word (has ***-R*** at the end of its category), all its leaf nodes can be collected to compose a word. Then, we replace the subtree of this node with a leaf node containing a word and word-level POS tag. Also, we change the HEADs from characters to words for all nodes and replace the leaf nodes' POS tags with word-level POS tags. Note that when we transform our character-level *joint span* structure trees to the word-level, because these trees contain both word-level constituent and dependency information, we then obtain consistent constituent and dependency trees based on the same word segmentation.

4. Character-Level Parsing Model

In this section, we introduce the proposed character level parsing model. To parse our character-level *joint span* structure, following Zhou and Zhao (2019), we apply an encoder-decoder backbone with a self-attention encoder (Vaswani et al., 2017) with partitioned

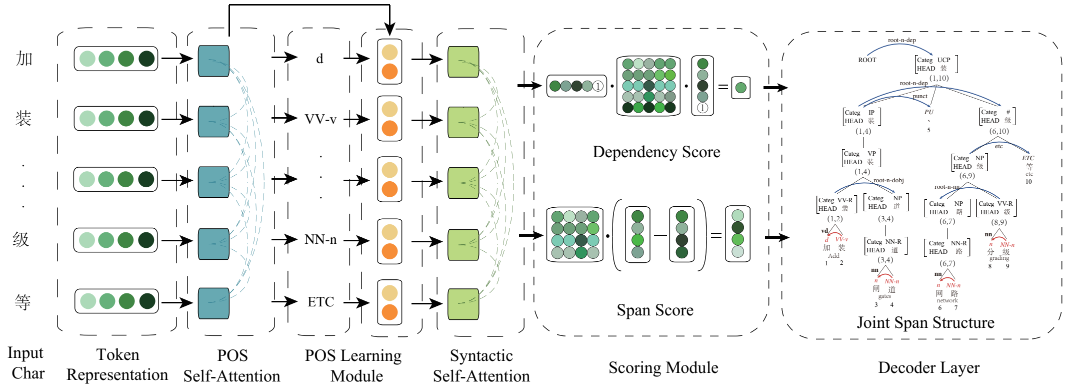


Figure 10: The framework of character-level parsing model.

position information (Kitaev & Klein, 2018). Recently, deep NLP modes have shown benefits from predicting many increasingly complex tasks each at a successively deep layer. Since the language is the construction of hierarchy: phonology, morphology, lexical, syntax, semantics and pragmatics (Allen, 1995), we jointly learn POS tags by considering linguistic hierarchies inspired by (Collobert et al., 2011; Hashimoto et al., 2017). Their work showed that using linguistic hierarchies and predicting different linguistic parsing tasks at different depth of NN layers jointly is more effective than handling different tasks in the same layer or in entirely separate networks. As shown in Figure 10, our model includes five modules: a token representation module, self-attention encoder, a POS learning module, a scoring module, and a CKY-style decoder. We insert the POS learning module into the self-attention encoder, which is divided into two encoders: a POS self-attention encoder and a syntactic self-attention encoder. We take a multi-task learning (MTL) (Caruana, 1993) approach and share the parameters of the token representation and the self-attention encoder. After the decoding process, we convert our *joint span* structure to character-level constituent and dependency trees and perform word-level restoration for evaluation and comparison.

4.1 Token Representation Module

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

Since characters instead of words are treated as the basic tokens, token representation x_i is only composed of character representations. We concatenate randomly initialized embeddings x_{rand} and pre-trained embeddings x_{pre} as our token representation $x_i = [x_{\text{rand}}; x_{\text{pre}}]$. In addition, if we apply pre-trained language models such as BERT, RoBERTa or XLNet, we replace character embeddings x_i with the last layer outputs of the model to create our token representation x_i.

4.2 Self-Attention Encoder

The encoder in our model is adapted from Vaswani et al. (2017). And the encoding is factored into content information encoding process and position information encoding process explicitly. The input matrix $X = [x_1, x_2, \ldots, x_n]$, in which each x_i is concatenated with a position embedding, is transformed by a self-attention encoder. We factor both the model's self-attention sub-layers and feed-forward layers into explicit content information and position information portions, following the settings of Kitaev and Klein (2018).

4.3 POS Learning Module

To account for linguistic hierarchies, we predict POS early in the process and use predicted POS to aid in the subsequent parsing task. We insert our POS learning module into the self-attention encoder and divide it into a POS self-attention encoder and a syntactic self-attention encoder, in which the POS self-attention encoder only takes token information for POS prediction, while the syntactic self-attention encoder takes both token and POS information for later parsing tasks. The output of POS self-attention encoder X_P is fed into a one-layer feedforward network:

$$Y_P = W_2^P g(LN(W_1^P X_P + b_1^P)) + b_2^P,$$

where LN denotes Layer Normalization, and g is the Rectified Linear Unit (ReLU) nonlinearity function.

We minimize the negative log-likelihood of the gold POS tag and implement this as a cross-entropy loss:

$$J_1(\theta) = -logP_\theta(g_p|y_p),$$

where $P_\theta(g_p|y_p)$ is the probability of correct POS tag g_p for y_p.

Then, we use the predicted POS to get POS embedding e_{POS} and sum this with the output of token self-attention encoder X_P to get our POS representation: $x_{\text{POS}} = x_p + e_{\text{POS}}$, which is then used by the syntactic self-attention encoder for later parsing tasks. Since we factor explicit content and position information in the self-attention process, we only add POS embedding e_{POS} to the content information of x_p.

4.4 Scoring Module

Since the *joint span* structure is composed of constituents and dependencies, we need two types of scorers: constituent and dependency head scorers[11].

Constituent Scorer We follow constituent syntactic parsing methods (Zhou & Zhao, 2019; Kitaev & Klein, 2018; Gaddy et al., 2018) and train a constituent scorer. Specifically, span vector s_{ij} is a concatenation of vectors as:

$$s_{ij} = [\overrightarrow{y}_j - \overrightarrow{y}_{i-1}; \overleftarrow{y}_{j+1} - \overleftarrow{y}_i],$$

in which $\overrightarrow{y}_j$ is constructed by splitting in half the outputs from the syntactic self-attention encoder. Notably, these over arrows do not indicate the direction information from encoder,

[11] For the dependency label of each word, we train a separate multiclass classifier simultaneously with the parser by optimizing the sum of their objectives.

but to illustrate whether the first half or the second half of the split representation is employed. We apply one-layer feedforward networks to generate span scores vectors, taking span vector s_{ij} as input:

$$S(i,j) = W_2^S g(LN(W_1^S s_{ij} + b_1^S)) + b_2^S.$$

The individual score of a span with category ℓ is denoted by:

$$S_{categ}(i,j,\ell) = [S(i,j)]_\ell,$$

where $[]_\ell$ indicates the value corresponding to the ℓ-th element of the score vector. The score $s(T)$ of the constituent parse tree T is obtained by adding the scores of all its spans, where i and j denote a span's fencepost (boundary) positions and ℓ denotes a span's category:

$$s(T) = \sum_{(i,j,\ell)\in T} S_{categ}(i,j,\ell).$$

The goal is then to find the tree with the highest score:

$$\hat{T} = \arg\max_T s(T).$$

We use a CKY-style algorithm (Gaddy et al., 2018) to obtain the tree $\hat{T}$ in $O(n^3)$ time complexity. This structured prediction problem is tasked with satisfying the margin constraint:

$$s(T^*) \geq s(T) + \Delta(T, T^*),$$

where T^* denotes the correct parse tree, and Δ is the Hamming loss on the spans with categories. The dynamic program portion is conducted with a slight modification as described in Gaddy et al. (2018). The objective function is the hinge loss:

$$J_2(\theta) = \max(0, \max_T[s(T) + \Delta(T, T^*)] - s(T^*)).$$

Dependency Head Scorer We predict a distribution over the possible heads for each word and use the biaffine attention mechanism (Dozat & Manning, 2017) to calculate the score as follows:

$$\alpha_{ij} = h_i^T W g_j + U^T h_i + V^T g_j + b,$$

where α_{ij} indicates the child-parent score, W denotes the weight matrix of the bi-linear term, U and V are the weight vectors of the linear term, b is the bias item, and h_i and g_i are calculated by a distinct one-layer perceptron network.

We minimize the negative log-likelihood of the gold dependency tree Y, which is implemented as a cross-entropy loss:

$$J_3(\theta) = -\left(logP_\theta(h_i|x_i) + logP_\theta(l_i|x_i, h_i)\right),$$

where $P_\theta(h_i|x_i)$ is the probability of correct parent node h_i for x_i, and $P_\theta(l_i|x_i, h_i)$ is the probability of the correct dependency label l_i for the child-parent pair (x_i, h_i).

Finally, we train our scorer for simply minimizing the overall loss:

$$J_{overall}(\theta) = J_1(\theta) + J_2(\theta) + J_3(\theta).$$

Algorithm 2 Joint Span Syntactic Parsing Algorithm

```
Require: sentence length $n$, span and dependency score $s(i, j, \ell)$, $d(r, h)$, $1 \leq i \leq j \leq n, \forall r, h, \ell$
Ensure: maximum value $S_H(T)$ of tree $T$
  Initialization:
  $s_c[i][j][h] = s_i[i][j][h] = 0, \forall i, j, h$
  for $len = 1$ to $n$ do
    for $i = 1$ to $n - len + 1$ do
      $j = i + len - 1$
      if $len = 1$ then
        $s_c[i][j][i] = s_i[i][j][i] = \max_\ell s(i, j, \ell)$
      else
        for $h = i$ to $j$ do
          $split_l = \max_{i \leq r < h} \{ \max_{r \leq k < h} \{ s_c[i][k][r] + s_i[k+1][j][h] \} + d(r, h) \}$
          $split_r = \max_{h < r \leq j} \{ \max_{h \leq k < r} \{ s_i[i][k][h] + s_c[k+1][j][r] \} + d(r, h) \}$
          $s_c[i][j][h] = \max \{ split_l, split_r \} + \max_{\ell \neq \varnothing} s(i, j, \ell)$
          $s_i[i][j][h] = \max \{ split_l, split_r \} + \max_{\ell} s(i, j, \ell)$
        end for
      end if
    end for
  end for
  $S_H(T) = \max_{1 \leq h \leq n} \{ s_c[1][n][h] + d(h, root) \}$
```

4.5 Decoder Module

As the joint span is defined recursively, scoring the root joint span requires scoring all spans and dependencies in a syntactic tree. During testing, we apply the joint span CKY-style algorithm (Zhou & Zhao, 2019), as shown in Algorithm 2, to explicitly find the globally highest score $S_H(T)$ of our joint span syntactic tree T[12] .

Also, to control the effect of combining span and dependency scores, we apply a weight[13] λ_H:

$$s(i, j, \ell) = \lambda_H S_{categ}(i, j, \ell), d(i, j) = (1 - \lambda_H)\alpha_{ij},$$

where λ_H is in the range of 0 to 1. With this weight, we can also trivially generate constituent or dependency syntactic parsing trees by setting λ_H to 1 or 0, respectively.

5. Experiments of Empirical Evaluation

Next, we conduct an empirical evaluation of the character level parsing model we proposed[14].

[12] For further details, see Zhou and Zhao (2019), which discusses the differences in the constituent syntactic parsing CKY-style algorithm, how to binarize the joint span tree, and the time and space complexity.

[13] We also try to incorporate the head information in the constituent syntactic training process, namely max-margin loss for both two scores, but it makes the training process become more complex and unstable. Thus, we employ a parameter to balance two different scores in the joint decoder, an easily implemented addition that produces better performance.

[14] The code is available at `https://github.com/bcmi220/ccharpar`.

5.1 Settings

Data We use Chinese Penn Treebank 5.1 (CTB5) for both constituent and dependency evaluation with articles 001-270 and 440-1151 for training, articles 301-325 as the development set and articles 271-300 for the test set in constituent parsing evaluation following the standard split seen in Liu and Zhang (2017b); with articles 001-815 and 1001-1136 for training, articles 886-931, 1148-1151 as the development set, and articles 816-885 and 1137-1147 for the test set in dependency parsing evaluation following the standard split seen in Zhang and Clark (2008). Due to the difference in constituent and dependency evaluation dataset splits, in order to be comparable with the existing parsers' results, we trained a model using the constituent split and one using the dependency split (though both were jointly trained to handle both types of syntax trees). We then evaluated them on their corresponding test splits. The placeholders with the **-NONE-** tag are stripped from the CTB. To prepare the character-level treebank, the word-level dependencies are obtained using Stanford Parser[15] (De Marneffe, MacCartney, Manning, et al., 2006).

For the purpose of comparing our joint learning model with models that use traditional pipelines, BaseSeg and BasePoS (Zhao, Huang, & Li, 2006) are used for predicting word segmentation and POS tags, respectively, in word-level experiments. The predicted character-level POS tags are learned and annotated with a Conditional Random Field (CRF) tagger using the same features and settings as Chen, Zhang, and Sun (2008)[16]. If we use the gold or predicted POS setting, we remove the POS learning module and concatenate gold or predicted POS embeddings with character embeddings to create our token representation x_i, as noted in 4.1.

Then, we use two character treebanks, SCDT and ZCTB, as noted in Section 2.3. We apply the assembling strategy from Section 3 for SCDT character treebank and follow Zhang et al. (2013, 2014) to obtain ZConst (ZDep), respectively denoting final character-level constituent and dependency trees.

Hyperparameters In our experiments, the dimension size of character-embedding is 100, we use pre-trained structured-skipgram (Ling et al., 2015) embeddings to initialize our character embeddings. For the self-attention encoder, we apply 4 layers for the POS self-attention encoder and 8 layers for the syntactic self-attention encoder, keeping other hyperparameters settings are the same as Kitaev and Klein (2018). In the Pipeline settings of POS tagging and syntactic parsing, the POS tag model uses the same model architecture as in the main model, that is, 4-layers are used for the self attention encoder.

For span scores, we apply feed-forward networks with a hidden layer size of 250. For the dependency biaffine scorer, we employ two 1024-dimensional MLP layers with ReLU as the activation function and a 1024-dimensional parameter matrix for biaffine attention.

[15] http://nlp.stanford.edu/software/lex-parser.html

[16] In fact, Chen et al. (2008) used a maximum entropy Markov model as sequence labeling tool for word-level POS tagging instead of CRF; however, maximum entropy models and CRF share similar feature representations.

For pre-trained language models used in Chinese character-level syntactic parsing, we use the Chinese versions of BERT-base[17], RoBERTa-large[18], XLNet-mid[19], ALBERT-xxlarge[20], and ELECTRA-180g-large[21]. In addition, when augmenting our model with the Chinese pre-trained language models, we only use 2 layers for the POS self-attention encoder and 2 layers for the syntactic self-attention encoder.

Training Details We use dropout of 0.33 for biaffine attention and MLP layers. Adam optimizer with initial learning rate 5e-3 and warmup steps 160 is employed for optimization. We apply the same training settings as Kitaev and Klein (2018) and Kitaev et al. (2019) if using BERT, RoBERTa or XLNet. It is worth noting that the pre-trained language model (if used) in our proposed character-level parser will be further finetuned during the training of parsing. Parsers that does not explicitly mark what kind of pre-trained language model is used do not actually use pre-trained language model. All parse models are trained for up to 150 epochs on a single NVIDIA GeForce GTX TITAN X GPU with Intel i7-7800X CPU. In the text classification experiment, the initial learning rate is set to 2e-5, and a maximum of 10 epochs are trained. The early stopping mechanism is used on the development set to prevent overfitting. In the machine translation experiment, we use the open source Fairseq as the basic implementation, the initial learning rate of the RNN models is set to 0.01, and is set to 5e-4 for the Transformer models, and max update step is 200,000. In the pre-trained language model experiments, we use WoBERT[22] checkpoint for initialization in LIMIT-BERT-*word*, while in LIMIT-BERT-*char*, we use the original BERT-base checkpoint for initialization, 500,000 parsed Wikipedia sentences are used to continue training 100,000 steps.

Evaluation At present, there is no general standard to evaluate full character-level parsing. Thus, we use two evaluation strategies: character-level evaluation and word-level evaluation, similar to Li et al. (2018). Since we need to find the best model on the development set, we sum the F1 score of constituent parsing and the Labeled Attachment Score (LAS) of dependency parsing (labeled F1 if performing word-level evaluation) for the model selection criterion.

For constituent parsing, we use the standard evalb[23] tool to evaluate the F1 score. For dependency parsing, following (Dozat & Manning, 2017; Kuncoro et al., 2016; Ma et al., 2018), we report the results without punctuation for both treebanks.

5.2 Character-Level Evaluation

For character-level evaluation, we still follow the practice of word-level parsing and take F1 scores for constituent parsing and Unlabeled Attachment Score (UAS) / Labeled Attachment Score (LAS) on character-level tokens for dependency parsing as the metrics. In the presentation of experimental results, the proposed model using constituent and dependency

[17] https://github.com/google-research/bert
[18] https://github.com/brightmart/roberta_zh
[19] https://github.com/ymcui/Chinese-XLNet
[20] https://github.com/brightmart/albert_zh
[21] https://github.com/ymcui/Chinese-ELECTRA
[22] https://github.com/ZhuiyiTechnology/WoBERT
[23] http://nlp.cs.nyu.edu/evalb/

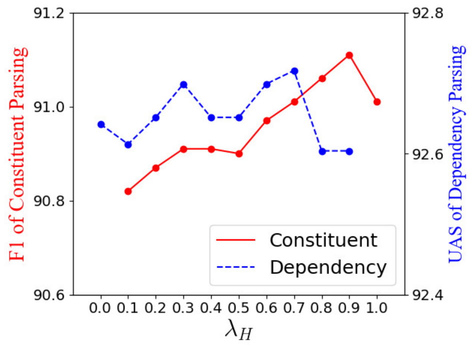


Figure 11: Syntactic parsing performance of our Joint Span Parser with different λ_H on the SCDT development set.

Joint Span Structure is called **Joint Span Parser**. In addition, in order to demonstrate the effects of joint syntactic structure learning, we also compared the two structures trained separately instead of using the constituent and dependency joint structure and named it as **Disjoint Span Parser**.

Moderating Constituent and Dependency This subsection examines the joint span decoder from Section 4.5 with parameter λ_H on the development set of SCDT. The weight parameter λ_H plays an important role: balancing the span and dependency scores. When λ_H is set to 0 or 1, the joint span parser works as either the dependency-only parser or constituent-only parser, respectively. Setting λ_H between 0 to 1 indicates the general setup for joint span syntactic parsing, which provides both constituent and dependency structure prediction. We set the λ_H parameter to 0 and increase by 0.1 until it reaches 1 as shown in Figure 11. The best results occur when λ_H is set to 0.7, which we gauge by examining the sum of the F1 constituent and UAS dependency scores. Thus, the λ_H in further experiments is set to 0.7.

Model Comparison on Different Datasets Character-level parsing results on the test sets of SCDT and ZCTB are given in Table 7 and 8, in which we see that the character-level parsing seriously depends on the character-level POS tagging. The parser with gold character POS greatly outperforms other POS settings. Overall, when comparing methods using traditionally predicted character POS (i.e., Pipeline), our method of jointly learning character POS tags achieves better results on POS, constituent, and dependency parsing. Specifically, the joint POS tag and parsing method outperforms the pipeline method by 0.2-0.3 in POS accuracy, 0.2-0.3 in constituent F1, and 0.3-0.5 in dependency LAS for both Disjoint Span Parser and Joint Span Parser. This demonstrates our end-to-end model can effectively avoid error propagation that would arise from pipeline inputs.

In Table 7, the constituent F1 and dependency UAS differences between Disjoint Span Parser and Joint Span Parser are only 0.15 and 0.05, respectively, which seems to be marginal. In fact, the results of Disjoint Span Parser are evaluated from two separately trained models, which originally benefited from being larger in total model parameters and focusing on the parsing of single syntax form respectively. As a result, the improvement of

Table 7: Character-level evaluation on the SCDT test set. The "Joint*" in POS Setting column indicates that POS tags were jointly learned with constituent or dependency parsing (rather than joint span parsing) in Disjoint Span Parser. Our Joint Span Parser† refers to the result of replacing the character POS tag in the SCDT dataset with the boundary indicator tags (such as b, i) in the ZCTB dataset.

	POS Setting	POS Accuracy	Constituent			Dependency	
			LR	LP	F1	UAS	LAS
(Li et al., 2018)	Gold	100.00	–	–	–	90.00	87.91
	W/O	–	–	–	–	80.09	–
	Predicted	90.12	–	–	–	82.53	80.47
Our Disjoint Span Parser	Gold	100.00	94.90	94.45	94.67	95.62	93.89
	W/O	–	88.97	89.75	89.35	92.63	88.48
	Pipeline	94.15	89.23	88.94	89.08	92.40	88.16
	Joint*	94.37 / 94.42	89.26	89.65	89.45	92.71	88.67
Our Joint Span Parser	Gold	100.00	94.75	94.88	94.82	95.73	94.01
	W/O	–	89.13	89.81	89.47	92.92	88.82
	Pipeline	94.15	88.74	89.95	89.34	92.55	88.40
	Joint	94.50	89.25	89.96	89.60	92.76	88.70
Our Joint Span Parser†	Gold	100.00	94.32	94.55	94.43	95.38	93.70
	Pipeline	94.96	88.94	89.15	89.04	92.15	88.02
	Joint	95.32	89.72	88.89	89.30	92.43	88.36

our joint span structure on such baseline may be not treated marginal considering that the current parsing accuracy is very high and the error rate decrease cannot been ignored. Furthermore, because the evaluation scores of our main experiment is the average of multiple runs, these improvements are also stable.

In addition, we analyzed the joint learning and joint parsing (decoding) algorithm by comparing the results of Joint Span Parser and Disjoint Span Parser on the SCDT and ZCTB datasets. Thanks to the sharing of network parameters, the joint optimization in the training stage, and the consideration of both constituent and dependency structures in the decoding of inference phase, Joint Span Parser improved in various settings compared to Disjoint Span Parser. This indicates that not only can word-level constituent structure and dependency structure assist each other in linguistic structure learning (refer to our previous work Zhou and Zhao, 2019), but their character-level counterparts can also mutually benefit each other in linguistic structure learning. In addition, this performance advantage also verifies the effectiveness of our proposed Joint Span Structure in modeling syntactical joint structures and in accommodating both structures.

For reference, we also list previous character-level parsing models. Zhang et al. (2013, 2014) used a transition parser on the ZConst and ZDep dataset, which does not provide character-level POS or dependency label annotation. Li et al. (2018) compared a traditional model (Char MaltParser) and a neural model (Char LSTM Parser) on the SCDT dataset

Table 8: Character-level evaluation on the ZCTB test set. The "Joint*" in POS Setting column indicates that POS tags were jointly learned with constituent or dependency parsing (rather than joint span parsing) in Disjoint Span Parser.

	POS Setting	POS Accuracy	Constituent			Dependency	
			LR	LP	F1	UAS	LAS
(Zhang et al., 2014)	W/O	–	–	–	–	82.07	–
Our Disjoint Span Parser	Gold	100.00	96.35	96.42	96.38	93.05	93.00
	W/O	–	92.31	90.74	91.51	92.00	89.04
	Pipeline	93.97	91.75	90.87	91.30	91.83	88.75
	Joint*	94.20 / 94.23	92.06	91.94	92.00	92.17	89.20
Our Joint Span Parser	Gold	100.00	96.40	96.54	96.47	94.07	93.12
	W/O	–	91.63	92.14	91.88	92.07	89.18
	Pipeline	93.97	91.73	91.85	91.79	91.96	88.92
	Joint	94.28	92.24	92.55	92.39	92.22	89.59

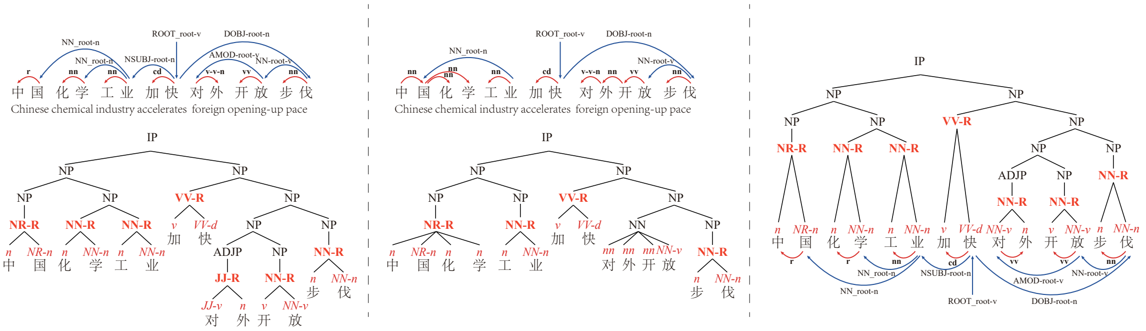
Table 9: Comparison of the results of our proposed Parser and Multi-task Learned Parser.

	Constituent			Dependency	
	LR	LP	F1	UAS	LAS
Our Disjoint Span Parser	89.23	88.94	89.08	92.40	88.16
Our Joint Span Parser	88.74	89.95	89.34	92.55	88.40
Multi-task Learning	88.69	89.17	88.93	92.25	88.03

with different POS sources. Our character-level Joint Span Parser achieved new state-of-the-art results, surpassing all previous character-level parsing models on both datasets.

The distinction between SCDT and ZCTB is the annotations of character-level POS tag and syntactic structure. Since we cannot directly compare character-level syntactic structure, we can replace the manual annotated POS tag of SCDT with trivial tags used in ZCTB like *b* and *i*. The corresponding experimental results are listed in Table 7 as "Our Joint Span Parser†". When compared to the Joint Span Parser utilizing the original SCDT data set, the results of constituent and dependency parses have all dropped after switching to the trivial tag adopted in ZCTB. Although the POS result of Joint Span Parser† is shown to be higher than that of Joint Span Parser, it is actually incomparable because the former is based on the trivial POS tag in ZCTB, while the latter is based on the character-level POS tag in SCDT. This indicates that the word-internal POS tag information is also valuable for overall parsing, as the information contained in the manually annotated tags extends beyond the word boundary information given in the trivial tags.

Joint Span Structure vs. Multi-task Learning As Multi-task Learning can also accomplish the joint learning of dependency syntax and constituent syntax in a single model, we additionally trained a multi-task learning baseline. Specifically, we share the encoder in



(a) gold tree (b) predicted tree from Disjoint Span Parser (c) predicted tree from Joint Span Parser

Figure 12: Case study on character-level parsing. The blue arc in the dependency tree is the dependency relationship at the original word-level, and the red arc is the dependency relationship newly introduced at the character-level.

a single model, learn the constituent and dependency syntax structure through two separate neural scorer components. The overall loss in the training phase is obtained by sum of the constituent and dependency parsing loss. And during the decoding phase, we adopt the MST and CKY algorithms to obtain two parse trees separately.

We compare the results of the Joint Span Parser and Multi-task Learned Parser and show them in Table 9. The results show that the performance of multi-task learned parser is not even better than that of our individually trained Disjoint Span Parser. This could be due to the fact that multi-task learning is not effective in all scenarios, though it enjoy the convenience with a single model, the performance suffers as a result. Instead, in our Joint Span parser, the training and decoding processes modeling the interactions between the two tasks, thus it is better suited than multi-task learning for such two tasks with explicit relationships or constraints.

Case Study on Character-Level Parsing To illustrate the role of Joint Span Structure in character-level parsing more intuitively, we extracted an example from the test set for case study, as shown in Figure 12. In the Disjoint Span Parser baseline, both "中国化学" and "对外开放" produced dependency arcs within words, which means that the parser treats these two spans as one word, so it will be wrong when restoring to the word-level dependency tree. The parsing of the constituent tree is also similar. Due to the wrong span division, the predicted constituent span is also with error. Thus the label based on the wrong dependency arc or division span is basically wrong. While in the prediction of our Joint Span Parser, since the dependency structure and the constituent structure are taken into consideration simultaneously, which means more constraints are performed to the decision-making, the overall structure is more likely to be correct. Although there are also label prediction errors (For example, the label "JJ-R" of "对外" is predicted to be "NN-R" in constituent tree, dependency relationship "v-v-n" of "对" → "外" is predicted to be "vv"), our Joint Span Parser can generate overall better prediction due to the structure is more correct.

In fact, since the neural scoring of span or dependency arc may produce errors, and the tree decoded with the highest scores of single structures only may be misled by these errors. Our joint span parsing takes into account both the span of the constituent syntax and the arc of the dependency syntax, which alleviates the errors of neural scorers and thus leads to better results. In other words, this shows that in the case when the dependency information and the constituency information are complementary, they can be effectively disambiguate each other.

5.3 Word-Level Evaluation

Because character-level syntax parsing lacks metrics that make it directly comparable to the current word-level parsers, in this subsection, we convert our syntactic parse trees from the character level to the word level to provide a comparison, relying on character-level constituent categories for trees' restoring. According to the character-level evaluation results, we chose our best-performing system, **Joint Span Parser**, and convert the resultant parse tree to word-level then compared it with the other word-level parsers. We use **Char Joint Span Parser** to represent our system that is converted from character-level. For word-level evaluation, we use F1 scores for both constituent and dependency parsing if there is no gold word segmentation, i.e., UAS-F1/LAS-F1 replaces UAS/LAS scores for dependency parsing following Hatori et al. (2012a), Zhang et al. (2014), Zhang et al. (2015), Kurita et al. (2017) and Yan et al. (2020)[24].

UAS-F1/LAS-F1 are used to evaluate the parsing performance in scenario of joint word segmentation and dependency parsing, since the widely used UAS/LAS are not enough to measure the performance in our scenario, as errors arise from two aspects: improper word segmentation and incorrect predictions for head words. A dependent-head pair is correct only when both the dependent and head words are accurately segmented and the dependent word correctly finds its head word. The precision of unlabeled dependency parsing (denoted as UAS-P) is calculated as the correct dependent-head pair versus the total number of dependent-head pairs (namely the number of segmented words). The recall of unlabeled dependency parsing (denoted as UAS-R) is computed by the correct dependent-head pair divided by the total number of gold dependent-head pairs (namely, the number of gold words). The calculation of UAS-F1 is similar to the normal F1. UAS-F1's only difference with LAS-F1 is that in addition to matching between the head and dependent words, the pairs must have the same labels as the gold dependent-head pairs. The precision and recall are calculated correspondingly. Notably, for word-level parsers, UAS-F1/LAS-F1 are equivalent to UAS/LAS.

Word Restoration Comparison Firstly, we compare the different word restoration strategies discussed in Section 3.3 on the development set of our Char Joint Span Parser. As shown in Table 10, the Constituent Categ strategy can not only handle both constituent and dependency word restoration, but also achieve better performance compared to the POS Tags strategy (Li et al., 2018). Thus, we apply the Constituent Categ strategy in the following word-level experiments.

[24] In case a word is wrongly segmented, all later related parts like word POS tag and unlabeled/labeled dependencies will be considered wrong, that means a complete error over the word.

Table 10: Word restoration comparison on the SCDT development set.

	Word		Parsing		
	SEG	POS	Phrase-F1	UAS-F1	LAS-F1
Golden Restoration	–	–	–	90.26	86.38
POS Tags	95.82	92.04	–	84.69	81.84
Constituent Categ	97.13	93.97	88.82	86.20	83.27

Table 11: Analysis of pipeline methods on the SCDT development set.

	Input SEG	Input POS	Constituent			Dependency	
			LR	LP	F1	UAS-F1	LAS-F1
Our Word-level Parser	Gold	Gold	92.98	93.32	93.15	93.23	91.64
	Predicted	Predicted	86.43	88.52	87.46	84.64	82.17
Our Character-level Parser	–	Gold	93.35	93.85	93.60	93.85	92.14
	–	Joint	88.52	89.13	88.82	86.20	83.27

Pipeline analysis Following Li et al. (2018), we also perform pipeline analysis at the word level on our Joint Span Parser to demonstrate that our end-to-end model is resistant from error propagation. The baseline model for comparison is the word-level HPSG model we previously proposed for word-level syntactic parsing. There are two pipeline inputs: word segmentation and POS tags[25]. In Table 11, using the SCDT development set, we compare the performance of the Pipeline parsers using gold and predicted word segmentation and POS, which of course have different accuracies. This comparison demonstrates how the errors in precursor steps (word segmentation, POS tagging), can affect later performance.

According to the comparison at the word level, first, the gold POS tags show a very significant improvement, which is consistent with the phenomenon in the character-level evaluation. Second, when using gold POS tags to restore the word-level syntactic parse trees, the character-level Joint Span Structure model obtains even better results than our word-level baseline model that also uses gold POS tags. This shows that the syntactic structure information inside words is beneficial to the syntactic structure parsing of the whole sentence. Third, our proposed joint learning of POS tags and syntactic parse tree restoration (accommodated by the Constituent Categ strategy) also outperforms the pipeline processing of word-level parser when gold word segmentation and POS tag information are not used.

Word-Level Comparison To evaluate word-level performance, we perform two types of comparison. One looks at previous character-level works which restore word-level parsing trees, the other compares previous word-level parsing models based on gold word segmentation and POS. Note that we are the first to report the accuracy of restored word-level constituent parsing. Using predicted word segmentation and character POS, we compare the accuracies of the restored trees in previous work including (Li, 2011; Zhang et al., 2014)

[25] POS tagging type corresponds to word input or character input, namely if using character input character input then POS tagging is character-level POS.

Table 12: Word-level evaluation on the test set. Our Char Joint Span Parser is trained on the SCDT treebank. Since gold POS tags are usually not used in the evaluation of word-level constituent parsing models (denoted as *), the comparison with them is only for a rough reference. † indicates that the model uses the BERT pre-trained language model for enhancement.

	Word		Constituent			Dependency	
	SEG	POS	LR	LP	F1	UAS-F1	LAS-F1
Predicted SEG & POS							
(Zhang et al., 2013)	97.84	94.80	84.43	84.43	84.43	–	–
(Zhang et al., 2014)	97.84	94.62	–	–	–	82.14	–
(Li et al., 2018)	96.64	92.88	–	–	–	79.44	77.35
Char Joint Span Parser	96.90	93.27	85.76	86.93	86.34	85.26	81.56
+ BERT	97.35	95.12	91.06	91.22	91.14	90.75	88.03
+ RoBERTa	97.68	95.52	91.44	91.51	91.48	91.20	88.50
+ XLNet	98.06	95.62	91.44	91.21	91.33	90.80	88.14
+ ALBERT	97.66	95.30	91.56	90.95	91.25	90.77	87.96
+ ELECTRA	98.12	95.73	92.03	91.24	91.63	91.53	88.71
Golden SEG & POS							
(Teng & Zhang, 2018)*	–	–	87.10	87.50	87.30	–	–
(Kitaev et al., 2019)*†	–	–	91.55	91.96	91.75	–	–
(Dozat & Manning, 2017)	–	–	–	–	–	89.30	88.23
(Li et al., 2018)	–	–	–	–	–	88.78	86.23
(Ma et al., 2018)	–	–	–	–	–	90.59	89.29
(Zhou & Zhao, 2019)	–	–	92.03	92.33	92.18	93.24	91.95
(Zhou & Zhao, 2019)*	–	–	89.09	89.70	89.40	–	–
+ RoBERTa*	–	–	92.50	92.61	92.55	–	–
Char Joint Span Parser	–	–	91.67	92.03	91.85	92.74	90.89
+ BERT	–	–	92.65	91.90	92.27	93.54	91.60
+ RoBERTa	–	–	92.99	92.73	92.86	94.28	92.19
+ XLNet	–	–	92.51	92.18	92.35	93.63	91.65
+ ALBERT	–	–	91.97	92.38	92.17	93.60	91.75
+ ELECTRA	–	–	93.14	92.83	92.98	94.46	92.39

and (Li et al., 2018) in the upper part of Table 12. The final results of restoring parse trees from character-level to word-level reflect the overall performance of the separate parts: the word segmenter, the POS tagger, and the syntactic parser. With the help of the new deep neural network Transformer and the joint training and decoding of the Joint Span Structure we proposed, our model achieved better results than other character-level parsers at the restored word-level, even though in some cases, the word segmentation accuracy of our joint model was lower than that of the pipeline mode. Furthermore, with the help of pre-trained language models, our performance has been further improved. Our proposed model achieved state-of-the-art results for the word level using trees restored from the character level without relying on any gold word segmentation or POS tag information. In addition, the restored word-level parse trees, when compared to the parse trees of purely word-level models with gold segmentation, (e.g., Kitaev et al., 2018), shows that our model gives competitive performance on word-level dependency parsing, while our parser is required to produce much more informative intra-word syntactic structures compared to any previous work.

Moreover, as shown in the bottom part of Table 12, we compared our character-level parser with other word-level parsing models using gold word segmentation and POS tag information. Since gold word segmentation and POS inputs are allowed to be used, our Char Joint Span Parser applies gold character POS inputs and restores with gold word segmentation instead of with the Constituent Categ for fair comparison. Without any help from pre-trained language models and when using gold word segmentation information for restoring, our Char Joint Span Parser does seem to be slightly weaker than word-level parsers (namely, Zhou and Zhao, 2019) because our character-level syntax parser needs to deal with more complex syntax structures. After augmenting our parser with pre-trained language model embeddings, both constituent and dependency parsing make obvious improvement gains and achieve new state-of-the-art.

6. The Value of Character-Level Syntactic Parsing

The discussion of whether to use character-based or word-based Chinese NLP has always been the focus of the Chinese NLP community. With the support of deep networks and stronger computing power, the mainstream approaches for many Chinese NLP tasks have gradually changed from word-based to character-based. Character-level syntactic parsing is one character-based method of understanding sentence structure, though its value as a fundamental NLP task has not been well assessed. In addition, typical character-level syntax tree annotations include transforming from the word level using BIO/BMSE tagging strategies, ZCTB proposed by Zhang et al. (2013, 2014), and our manually annotated SCDT. Due to the use of different models for different annotation strategies, seeing the advantages and disadvantages of different character-level parse trees is difficult. In order to bridge the gap between theoretical research and practicality for character-level syntactic parsing, we chose two typical tasks for experiments: Text Classification (TC) and Neural Machine Translation (NMT).

6.1 Text Classification

For text classification, three currently popular and public available benchmarks were adopted: TouTiao Text Classification for News Titles (TNEWS), IFLYTEK (CO, 2019), and THUCNews (Sun et al., 2016). TNEW, IFLYTEK and THUCNWES are short text, long text and document text classification tasks, respectively. In our experiments, the text classification task is used to evaluate the enhancement of the word-level and character-level dependency parse trees. We considered three settings: 1) word-level RNN-based encoders, 2) word-level Transformer-based encoders, and 3) character-level Transformer-based encoders.

In RNN-based models, we adopted the Bi-Tree-LSTM introduced in Chen et al. (2017), which is a variant of an RNN-based tree encoder described in Tai et al. (2015). In Transformer-based models, we use the Dependency Tree (Graph) Attention (Tree-Attn) mechanism introduced in Jin et al. (2020) as the encoder for the dependency semantic graph. This Graph Attention mechanism was first proposed by Veličković et al. (2018) for undirected graph encoding, and Jin et al. (2020) adapted it to directed graphs by aggregating neighbor relation triples rather than directly aggregating neighbor nodes. In this experiment, the dependency syntax trees are provided by typical parsers. MaltParser is a classic dependency parser proposed by Nivre et al. (2007), HPSG Parser is a word-level joint syntactic parser we proposed that achieved advanced parsing performance, ZCTB and SCDT refer to models trained using our proposed *Joint Span Structure* on the ZCTB and SCDT datasets, respectively.

The experimental results are shown in Table 13. First, comparing the RNN-based and Transformer-based baselines in the word-based setting, the Transformer-based model outperforms the RNN-based model. Second, in the Transformer-based baseline, comparing word-based and character-based approaches, the character-based approach generally has better performance. The findings are consistent with Li et al. (2019), suggesting that with deep neural networks (like Transformer), character-based approaches are better for language encoding. Third, in the word-based approaches, the models that used syntactic tree encoding achieved better performance, and MaltParser, which has relatively weaker parsing performance, correspondingly had the least enhancement effect. Although in the previous parsing evaluation, the restored trees from character-level is comparable to the original word-level parse trees, the restored from character-level trees leads to less downstream performance gains compared to trees originally parsed at the word-level due to the loss in restoring. Forth, in the character-based setting, our results show that the true character-level syntactic parsing in SCDT and ZCTB produces results significantly stronger than those obtained from word-level conversions. This shows that the internal syntactic structure of a word is very useful, and only using character-level syntactic trees obtained by word boundary-based conversions is not enough. In addition, comparing the character-level parser trained on the two datasets, ZCTB and SCDT, in the Transformer-based model, although both ZCTB and SCDT parsers use the same model structure, according to the scores on the downstream task, the parse tree generated from the model trained on SCDT obtains a better enhancement effect than the parser trained on ZCTB does.

Through these experimental analyses, we showed that our manually annotated dataset is better than ZCTB, which was derived with heuristic rules, further illustrating the completeness of our SCDT dataset for character-level syntactic parsing. In addition, we also demonstrate through experiments that character-level syntactic parsing and modeling have advantages over their word-level counterparts under the current Transformer network structure. This also reflects the current trend of transitioning from word-based to character-based modeling and reaffirms whether to use word segmentation, which has been a topic of discussion in the Chinese NLP community.

6.2 Neural Machine Translation

A Chinese-to-English machine translation task is used to evaluate the role of word-level and character-level constituent parse trees. Similar to the experimental setting in the text classification evaluation, it is divided into word-based and character-based approaches according to the granularity of Chinese inputs and divided into RNN-based and Transformer-based according to the types of neural networks. In the RNN-based setting, we choose the baseline model and syntax enhanced model in Chen et al. (2017) as our baselines. In the Transformer-based setting, Transformer (base) (Vaswani et al., 2017) serves as our baseline. Since the integration is based on constituent parse trees, we adopt the Tree-Attn

Table 13: Results on the development and the test set for text classification task. * indicates that the word-level dependency trees used are restored from character-level parse trees. † means that the character-level dependency trees used are transformed from word-level trees through intra-word right dependent rules.

#	Model	Parser	TNEWS		IFLYTEK		THUCNews	
			dev	*test*	*dev*	*test*	*dev*	*test*
RNN-based + Word-based								
1	BiLSTM	no	51.3	50.9	48.3	48.4	91.6	91.2
4	+Bi-Tree-LSTM	MaltParser	51.6	51.4	49.2	48.8	91.7	91.4
5	+Bi-Tree-LSTM	HPSG	53.0	**52.7**	49.5	**49.4**	92.4	**92.2**
6	+Bi-Tree-LSTM	ZCTB*	52.6	52.5	49.4	49.3	92.3	92.1
7	+Bi-Tree-LSTM	SCDT*	52.7	52.5	49.2	49.3	92.4	92.0
Transformer-based + Word-based								
8	Transformer	no	51.9	51.7	48.7	48.6	92.5	92.0
9	+Tree-Attn	MaltParser	52.4	51.9	49.2	48.8	92.5	92.1
10	+Tree-Attn	HPSG	53.5	**52.9**	50.0	**49.7**	93.2	**92.9**
11	+Tree-Attn	ZCTB*	53.2	52.8	49.7	49.5	93.3	92.8
12	+Tree-Attn	SCDT*	53.2	**52.9**	49.7	49.3	93.1	92.6
Transformer-based + Char-based								
13	Transformer	no	52.2	52.0	48.9	48.8	93.2	92.8
14	+Tree-Attn	MaltParser†	52.4	52.0	48.7	48.8	93.4	93.1
15	+Tree-Attn	HPSG†	52.8	52.4	50.0	49.8	94.0	93.6
16	+Tree-Attn	ZCTB	52.7	52.9	50.2	50.2	94.2	93.8
17	+Tree-Attn	SCDT	53.4	**53.1**	50.8	**50.6**	94.3	**94.0**

proposed by Nguyen et al. (2019) instead of the Tree-Attn presented in the text classification experiment as the parse tree encoder. All training details and hyper-parameters are consistent with their implementations. We conduct the evaluation on the NIST Chinese-English translation benchmark. The training set consists of 1.25M sentence pairs extracted from the LDC corpora (LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06). The development set is from NIST 2002 and the models are evaluated on NIST 2003, 2004, 2005, and 2006. In terms of the syntactic parser, we chose the classical Berkeley Parser (Petrov & Klein, 2007) as the source of the constituent word-level syntax trees. Other sources are the same as those in our text classification experiment.

We present the results for Chinese-to-English translation in Table 14. Comparing the results of the baseline models when not using syntax trees, Transformer and character-based approach are important factors for translation performance. The best results of word-based models using word-level parse trees lag behind the results of character-based models using character-level parse trees, which shows that character-based inputs is a better option for Chinese machine translation source input. In character-based modeling, the character-level parse trees generated by the model trained on our manual annotation are better than those produced using rule conversions from the word level. We see that the models with SCDT

Table 14: Results of different models on the Chinese-to-English machine translation task. * indicates that the word-level constituent trees used are restored from character-level parse trees. † means that the character-level constituent trees used are transformed from word-level trees through rules.

#	Model	Parser	MT02	MT03	MT04	MT05	MT06	Average
RNN-based + Word-based SRC&TGT								
1	BiLSTM	no	33.76	31.88	33.15	30.55	27.47	31.36
4	+Bi-Tree-LSTM	Berkeley	35.52	33.91	35.51	33.34	29.91	33.64
5	+Bi-Tree-LSTM	HPSG	37.55	35.74	38.21	35.08	34.70	**36.26**
6	+Bi-Tree-LSTM	ZCTB*	37.52	35.68	38.02	35.05	34.53	36.16
7	+Bi-Tree-LSTM	SCDT*	37.50	35.65	38.11	35.02	34.49	36.15
Transformer-based + Word-based SRC + Word-based TGT								
8	Transformer	no	43.16	42.15	43.74	42.71	40.58	42.47
Transformer-based + Word-based SRC + Subword-based TGT								
9	Transformer	no	43.65	42.71	43.88	41.81	41.36	42.68
10	+Tree-Attn	Berkeley	44.23	43.35	43.99	42.46	41.87	43.18
11	+Tree-Attn	HPSG	44.85	43.98	44.52	43.18	42.28	**43.76**
12	+Tree-Attn	ZCTB*	44.78	43.80	44.19	43.00	42.06	43.57
13	+Tree-Attn	SCDT*	44.70	43.83	44.22	43.06	42.10	43.58
Transformer-based + Char-based SRC + Subword-based TGT								
14	Transformer	no	45.79	44.37	44.40	43.39	42.14	44.02
15	+Tree-Attn	Berkeley†	45.95	44.52	44.65	43.62	42.20	44.19
16	+Tree-Attn	HPSG†	46.44	44.86	44.90	43.89	42.55	44.53
17	+Tree-Attn	ZCTB	46.86	45.10	45.21	44.20	43.09	44.89
18	+Tree-Attn	SCDT	46.97	45.53	45.49	44.25	43.48	**45.14**

and ZCTB parse trees outperform other word-level transformed parse trees, which suggests that Chinese words internal structures do benefit downstream performance.

Additionally, the character-based models trained on SCDT produce parse trees that lead to more competitive performance on both text classification and machine translation compared to ZCTB. The reason for this might be that the character-level POS and intra-word dependency relations in ZCTB only encode word boundaries extracted from the segmentation annotation, while SCDT provides human-annotated rich intra-word syntactic information.

6.3 Discussion for Enhancement

The improvements of character-level syntactic parsing are dependent on the characteristics of the specific downstream task and datasets when it comes to aid the downstream tasks. When some downstream tasks or datasets rely on these character-level syntactic features, the improvement is bigger, while the existing word-level information suffices, the enhancement of character-level features is underestimated. The enhancement on some tasks are not significant enough when viewed individually, but more detailed character-level information is generally more useful from a global view on all of these results. Specifically, character-level

Table 15: The effects of character-level syntax for language model pre-training.

Model	**CMRC18** (F1/EM)		**ChID** (acc)		**C^3** (acc)	
	dev	*test*	*dev*	*test*	*dev*	*test*
BERT	85.48 / 64.77	71.60 / 88.10	82.20	82.04	65.70	64.50
BERT-*wwm-ext*	86.68 / 66.96	89.62 / 73.95	83.36	82.90	67.80	68.50
LIMIT-BERT-*word*	86.95 / 67.05	89.78 / 73.97	83.50	83.05	68.26	68.65
LIMIT-BERT-*char*	87.23 / 67.39	90.05 / 74.22	83.77	83.17	68.60	68.70

parsing provides more fine-grained syntactic structure information, which is more helpful for tasks that rely on fine-grained structure of tokens in sentences, and less helpful for tasks that rely less on word structures.

7. Character-Level Syntax for Language Model Pre-training

Recently, pre-trained language models have been shown to be greatly effective across a range of linguistics-inspired NLP tasks such as syntactic parsing, semantic parsing, and machine reading comprehension. In our previous work (Zhou et al., 2020), we presented Linguistics Informed Multi-Task BERT (LIMIT-BERT) for learning language representations across multiple linguistics tasks by using multi-task learning in English. Five key linguistics tasks (Part-Of-Speech (POS) tags, constituent and dependency syntactic parsing, and both span and dependency semantic role labeling (SRL)) are integrated in the pre-training to make our LIMIT-BERT a fully linguistically-motivated approach. Benefiting from a regularization effect and the linguistics information in the semi-supervised learning, LIMIT-BERT achieved better results in comparison to the baseline.

Inspired by our work (Zhou et al., 2020), in order to better and more conveniently make use of syntax (i.e., avoid adding additional tree encoders, etc.), we propose the Chinese LIMIT-BERT-*word* and LIMIT-BERT-*char*. In Chinese LIMIT-BERT-*word* and LIMIT-BERT-*char*, to focus on the role of syntactic parsing, we remove the semantic parsing task. The suffixes *word* and *char* in LIMIT-BERT-*word* and LIMIT-BERT-*char* indicate whether the parse trees used in multi-task training are word-level or character-level. The word-level parse trees are generated from the model trained on CTB, while the character-level parse trees are from the model trained on SCDT. We use the BERT model with fine-tuned BERT-*wwm-ext* as the multi-task training baseline[26] and CMRC18 (Cui et al., 2019), ChID (Zheng et al., 2019), and C^3 (Sun et al., 2019) as the evaluation tasks. To fairly compare the baselines BERT and BERT-*wwm-ext*, we only used the plain text in the Wikipedia corpus for parse trees' generation and multi-task learning.

As shown in Table 15, BERT-*wwm-ext* significantly surpasses BERT, showing the effectiveness of using more data during pre-training and whole-word-masking (wwm). LIMIT-BERT surpasses our strong baseline BERT-*wwm-ext*, which illustrates that the integration of syntactic information into the pre-training learning process can bring additional benefits. The LIMIT-BERT-*char* further outperforms LIMIT-BERT-*word*, confirming our claim that the internal syntactic structure of Chinese words is also useful; i.e., more fine-grained

[26] https://github.com/ymcui/Chinese-BERT-wwm

syntactic information matters. Additionally, the results of whole-word-masking on Chinese pre-trained language models show that using the word segmentation outputs as the learning goal of the pre-training language model is very helpful. Chinese word segmentation can be regarded as a rough syntactic parsing based on character sequence, as it essentially provides a kind of coarse-grained syntactic information in the form of word boundaries. As our character-level syntactic parsing provides more informative and fine-grained syntax, better results are to be expected.

8. Related Work

8.1 Constituent and Dependency Parsing

Constituent and dependency parsing are core problems in NLP where the goal is to obtain representations for the syntactic structure of sentences. Constituent parsing aims to predict a phrase structure tree, while dependency parsing predicts a dependency tree for the input sentence (Nivre & McDonald, 2008). In parsing, the sentence is divided into segments that correspond to leaves in constituent tree or nodes in dependency trees, respectively.

The two formalisms carry distinct information and each has its strength: the constituent structure is better at describing phrasal continuity, while the dependency structure is better at indicating the dependency relations between words. Since constituent and dependency trees share many commonalities in their grammatical constructions and their machine learning solutions, studying the relationship between them, and the joint learning of their parsing is also beneficial (Collins, 1997; Charniak, 2000; Charniak & Johnson, 2005; Farkas et al., 2011; Green & Žabokrtský, 2012; Ren et al., 2013; Yoshikawa et al., 2017; Zhou & Zhao, 2019). Traditional parsing models usually use linear models and sparse features, while recent models for constituency and dependency parsing have been well developed with neural networks. These models attain state-of-the-art results for dependency parsing (Chen & Manning, 2014; Dozat & Manning, 2017; Ma et al., 2018) and constituent parsing (Dyer, Kuncoro, Ballesteros, & Smith, 2016; Cross & Huang, 2016; Kitaev & Klein, 2018).

Among these parsing models, graph-based (Zhang, Zhao, & Qin, 2016) and transition-based parsing models (Chen & Manning, 2014; Weiss, Alberti, Collins, & Petrov, 2015) are the most two popular styles. Transition-based methods , which are categorized by their three different parsing directions, top-down (Dyer et al., 2016; Liu & Zhang, 2017b), bottom-up (Zhu, Zhang, Chen, Zhang, & Zhu, 2013; Wang, Mi, & Xue, 2015; Cross & Huang, 2016), and recently, in-order (Liu & Zhang, 2017a) generally suffer from compounding errors caused by exposure to ground-truth decisions in model training phase, which creates a bias and has catastrophic effects on test performance. Thus, Stern et al. (2017b) proposed an effective inference method for generative parsing, which enables direct decoding in those models. Other efforts researched neural graph-based (namely chart-based) parsing (Stern, Andreas, & Klein, 2017a; Kitaev & Klein, 2018), which ensures structural consistency and offers exact inference using the CKY (Cocke, 1969; Younger, 1967; Kasami, Tadao, 1965) algorithm at the cost of high time complexity in $O(n^3)$ compared with transition-based methods.

8.2 Chinese Character-Level Parsing

In the Chinese NLP community, whether to use word-based or character-based approaches has been a constant debate. Simultaneously, there has also been a long-term argument in theoretical Chinese linguistics as to which one is more essential, words or characters (i.e., "词本位$_{\textit{word-based}}$ " or "字本位$_{\textit{character-based}}$ ") (Huang & Zhao, 2007; Zhao, Cai, Huang, & Kit, 2019). Especially when applying the outcome of such a core NLP task to downstream tasks, this dilemma is even more serious. Downstream NLP tasks introduce even more uncertainty about word segmentation (though theoretical linguistics does have several conflicting word segmentation conventions), domain-specific factors also influence which segmentation standard is best in a given situation. Such scenario-based specificities make Chinese word segmentation unlikely to have a universal convention any time soon. As an alternative, however, a character-level uniform formalism over a sentence would conveniently remove the need for word segmentation and a corresponding standard, thus providing a unified, universal solution for all downstream tasks.

When using segmented sequences of words, Chinese word-level syntactic parsing is exactly the same in form as in languages like English. Conventionally, traditional Chinese parsing takes words as the input nodes, which require word segmentation beforehand. Existing character-based work on Chinese word-level dependency and constituent parsing is mostly concerned with transforming word-level syntactic parsing and Chinese word segmentation into a unified task so that they can be jointly processed. A unified model in this way would solve the syntactic parsing and Chinese word segmentation issues once and for all.

For Chinese parsing, there are generally two solutions for tackling the word definition ambiguity and the many hierarchies existing over a sentence. The first is to perform a joint learning task for all levels of processing: word segmentation, POS tagging, and parsing (Luo, 2003; Hatori, Matsuzaki, Miyao, & Tsujii, 2012b; Qian & Liu, 2012; Yan et al., 2020). We regard this as computationally-motivated solution. Although these works claimed to use character-level parsing, their "character-level" parsing is very different from our character-level parsing, as we create trees where nodes are characters rather than words (unlike these previous works), and their models simply just process inputs at the character level before generating word-level trees. In these works, BMSE/BIO tagging strategies were usually adopted to re-encode word boundary information and create a character-level formalism, in which the true grammatical functions of Chinese characters and the levels of word components were not actually annotated, as only predictions based on rules were used. The second solution is quite different and seeks to exploit the linguistic roots of the Chinese language. It does not need this re-encoding step and formulates a unified framework starting from characters.

We adopt the second solution in this work and present a fully character-level syntactic parsing solution. This type of work was pioneered by Zhao (2009). Character-level parsing for sentences is a linguistically-motivated rewriting scheme for hierarchical Chinese processing. It removes the ambiguity around what constitutes a word and consequently proposes a more natural and simpler computational framework. Zhao (2009) proposed that dependencies inside words could be helpful for Chinese word segmentation, and he also suggested integrating character-level dependencies into word-level dependency trees to create a unified parsing scheme. The suggestion was soon implemented in Zhao et al. (2009), which was the first work to produce a full character dependency parsing that gives better Complete Match (CM) rate compared to the traditional word segmentation starting pipelines. This research was further studied in two branches. One is the works of Li and Zhou (2012) and Zhang et al. (2013) that considered using a constituent style of internal structure for words and focused on constituent parsing. The other branch came from Zhang et al., Li et al. (2014, 2018), who again considered unified character dependency parsing as Zhao (2009) suggested and how Zhao et al. (2009) implemented using a different character treebank. Zhang et al. (2014) also gave important insight and empirical results to show that a linguistically-motivated character-level model essentially outperformed the computationally-motivated joint learning models (Hatori et al., 2012b; Qian & Liu, 2012) with traditional workflows that included word segmentation and word-level parsing.

Unlike Zhang et al. (2014), in our annotation, we fully and manually annotated a treebank using solid linguistic theory. Specifically, our treebank uses a complete character-level grammar (including character-level POS tags and dependency head rules) and is manually annotated by experienced human annotators instead of simply converting the only bracketed word structures and head annotations by heuristic rules. The motivation in creating our treebank is to provide a fully and grammatically strict Chinese character-level treebank for character-level parsing. Because syntactic parsing as a fundamental or core NLP task is used to aid broad downstream tasks, the syntactic structures automatically derived from only heuristic rules or lexicon structures cannot meet the diverse requirements of various downstream tasks due to their limited accuracy and adaptiveness, while our manual annotated syntax structures are much more comprehensive and hence have many advantages.

9. Conclusion

This paper firstly combines constituent and dependency parsing at a character level and compares the different character level structure of two character-level treebanks. Our empirical comparison shows that using a truly character-based parsing model born of both computation and linguistic motivations can lead to improved performance and more informative structures for Chinese sentence analysis. Jointly learning parsing with POS tagging, which can avoid error propagation in the pipeline, is also shown to be more effective than the methods of previous traditional workflow parsers for both constituency parsing and dependency parsing when evaluating using their respective main metrics. Our comparisons demonstrate that the proposed model gives promising performance aiming at a better understanding of Chinese sentences.

10. Acknowledgments

This work was supported by Key Projects of National Natural Science Foundation of China (U1836222 and 61733011) and Funding from Chinese National Key Laboratory of Science and Technology on Information System Security. We thank Kevin Parnow (parnow@sjtu.edu.cn), from the Department of Computer Science and Engineering Shanghai Jiao Tong University for his kind help in proofreading when we were working on this paper. We also appreciate the help from the editors and anonymous reviewers. Corresponding author: Hai Zhao.

References

Ackema, P., & Neeleman, A. (2002). Syntactic atomicity. *The Journal of Comparative Germanic Linguistics*, *6*(2), 93.

Allen, J. (1995). *Natural language understanding.* Pearson.

Bresnan, J., Asudeh, A., Toivonen, I., & Wechsler, S. (2015). *Lexical-functional syntax*, Vol. 16. John Wiley & Sons.

Cai, D., & Zhao, H. (2016). Neural word segmentation learning for chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 409–420, Berlin, Germany.

Cai, D., Zhao, H., Zhang, Z., Xin, Y., Wu, Y., & 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)*, pp. 608–615, Vancouver, Canada.

Caruana, R. A. (1993). Multitask Learning: A Knowledge-Based Source of Inductive Bias. *Machine Learning Proceedings*, *10*(1), 41–48.

Charniak, E. (2000). A Maximum-Entropy-Inspired Parser. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL).*

Charniak, E., & Johnson, M. (2005). Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 173–180.

Chen, A., Zhang, Y., & Sun, G. (2008). A two-stage approach to Chinese part-of-speech tagging. In *the Sixth SIGHAN Workshop on Chinese Language Processing (SIGHAN)*, pp. 82–85, Hyderabad, India.

Chen, D., & 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)*, pp. 740–750.

Chen, H., Huang, S., Chiang, D., & Chen, J. (2017). Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1936–1945, Vancouver, Canada. Association for Computational Linguistics.

Chomsky, N. (1981). *Lectures on Government and Binding.* Mouton de Gruyter.

Clark, K., Luong, M.-T., Le, Q. V., & Manning, C. D. (2019). Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations.*

CO, L. I. (2019). Iflytek: a multiple categories chinese text classifier. competition official website..

Cocke, J. (1969). *Programming Languages and Their Compilers: Preliminary Notes.* New York University.

Collins, M. (1997). Three Generative, Lexicalised Models for Statistical Parsing. In *35th Annual Meeting of the Association for Computational Linguistics (ACL).*

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research*, *12*(1), 2493–2537.

Cross, J., & Huang, L. (2016). Span-Based Constituency Parsing with a Structure-Label System and Provably Optimal Dynamic Oracles . In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1–11.

Cui, Y., Liu, T., Che, W., Xiao, L., Chen, Z., Ma, W., Wang, S., & Hu, G. (2019). A span-extraction dataset for Chinese machine reading comprehension. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5883–5889, Hong Kong, China. Association for Computational Linguistics.

De Marneffe, M.-C., MacCartney, B., Manning, C. D., et al. (2006). Generating Typed Dependency Parses from Phrase Structure Parses. In *Lrec*, Vol. 6, pp. 449–454.

de Marneffe, M.-C., & Manning, C. D. (2008). The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pp. 1–8, Manchester, UK. Coling 2008 Organizing Committee.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, *abs/1810.04805*.

Dozat, T., & Manning, C. D. (2017). Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Dyer, C., Kuncoro, A., Ballesteros, M., & Smith, N. A. (2016). Recurrent Neural Network Grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pp. 199–209.

Farkas, R., Bohnet, B., & Schmid, H. (2011). Features for Phrase-Structure Reranking from Dependency Parses. In *Proceedings of the 12th International Conference on Parsing Technologies*, pp. 209–214.

Gaddy, D., Stern, M., & Klein, D. (2018). What's Going On in Neural Constituency Parsers? An Analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL: HLT)*, pp. 999–1010.

Green, N., & Žabokrtský, Z. (2012). Hybrid Combination of Constituency and Dependency Trees into an Ensemble Dependency Parser. In *Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*, pp. 19–26.

Hashimoto, K., Xiong, C., Tsuruoka, Y., & Socher, R. (2017). A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1923–1933.

Hatori, J., Matsuzaki, T., Miyao, Y., & Tsujii, J. (2012a). Incremental joint approach to word segmentation, POS tagging, and dependency parsing in Chinese. In *Proceedings* *of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1045–1053, Jeju Island, Korea. Association for Computational Linguistics.

Hatori, J., Matsuzaki, T., Miyao, Y., & Tsujii, J. (2012b). 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)*, pp. 1045–1053, Jeju Island, Korea.

Huang, C., & Zhao, H. (2007). Chinese word segmentation: A decade review. *Journal of Chinese Information Processing*, *21*(3), 8–20.

Jin, G., & 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)*, pp. 69–81, Hyderabad, India.

Jin, H., Wang, T., & Wan, X. (2020). Semsum: Semantic dependency guided neural abstractive summarization.. In *AAAI*, pp. 8026–8033.

Kasami, Tadao (1965). An Efficient Recognition and Syntax-Analysis Algorithm for Context-Free Languages. *Technical Report Air Force Cambridge Research Lab.*

Kitaev, N., Cao, S., & Klein, D. (2019). Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3499–3505, Florence, Italy. Association for Computational Linguistics.

Kitaev, N., & Klein, D. (2018). Constituency Parsing with a Self-Attentive Encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2676–2686.

Kuncoro, A., Ballesteros, M., Kong, L., Dyer, C., & Smith, N. A. (2016). Distilling an Ensemble of Greedy Dependency Parsers into One MST Parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1744–1753.

Kurita, S., Kawahara, D., & Kurohashi, S. (2017). Neural joint model for transition-based Chinese syntactic analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1204–1214, Vancouver, Canada. Association for Computational Linguistics.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations.*

Li, H., Zhang, Z., Ju, Y., & Zhao, H. (2018). Neural Character-level Dependency Parsing for Chinese. In *AAAI Conference on Artificial Intelligence.*

Li, X., Meng, Y., Sun, X., Han, Q., Yuan, A., & Li, J. (2019). Is word segmentation necessary for deep learning of Chinese representations?. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3242–3252, Florence, Italy. Association for Computational Linguistics.

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)*, pp. 1405–1414, Portland, Oregon.

Li, Z., & 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)*, pp. 1445–1454, Jeju Island, Korea.

Li, Z., Cai, J., He, S., & Zhao, H. (2018). Seq2seq Dependency Parsing. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pp. 3203–3214.

Ling, W., Dyer, C., Black, A. W., & Trancoso, I. (2015). Two/Too Simple Adaptations of Word2Vec for Syntax Problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL: HLT)*, pp. 1299–1304.

Liu, J., & Zhang, Y. (2017a). In-Order Transition-based Constituent Parsing. *Transactions of the Association for Computational Linguistics (TACL)*, *5*, 413–424.

Liu, J., & Zhang, Y. (2017b). Shift-Reduce Constituent Parsing with Neural Lookahead Features. *Transactions of the Association for Computational Linguistics (TACL)*, *5*, 45–58.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, *abs/1907.11692*.

Luo, X. (2003). A maximum entropy Chinese character-based parser. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pp. 192–199.

Ma, X., Hu, Z., Liu, J., Peng, N., Neubig, G., & Hovy, E. (2018). Stack-Pointer Networks for Dependency Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1403–1414.

Mel'cuk, I. A., et al. (1988). *Dependency syntax: theory and practice*. SUNY press.

Nguyen, X.-P., Joty, S., Hoi, S., & Socher, R. (2019). Tree-structured attention with hierarchical accumulation. In *International Conference on Learning Representations*.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., & Marsi, E. (2007). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, *13*(2), 95–135.

Nivre, J., & McDonald, R. (2008). Integrating graph-based and transition-based dependency parsers. In *Proceedings of the Meeting of the Association for Computational Linguistics (ACL)*, pp. 950–958, Columbus, Ohio.

Petrov, S., & Klein, D. (2007). Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pp. 404–411, Rochester, New York. Association for Computational Linguistics.

Pollard, C., & Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar.* University of Chicago Press.

Qian, X., & 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)*, pp. 501–511, Jeju Island, Korea.

Ren, X., Chen, X., & Kit, C. (2013). Combine Constituent and Dependency Parsing via Reranking. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2155–2161.

Robinson, J. J. (1970). Dependency structures and transformational rules. *Language*, 259–285.

Sproat, R., & Emerson, T. (2003). The first international Chinese word segmentation bakeoff. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing (SIGHAN)*, pp. 133–143, Sapporo, Japan.

Stern, M., Andreas, J., & Klein, D. (2017a). A Minimal Span-Based Neural Constituency Parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 818–827.

Stern, M., Fried, D., & Klein, D. (2017b). Effective Inference for Generative Neural Parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1695–1700.

Sun, K., Yu, D., Yu, D., & Cardie, C. (2019). Probing prior knowledge needed in challenging chinese machine reading comprehension. *CoRR*, *abs/1904.09679*.

Sun, M., Li, J., Guo, Z., Yu, Z., Zheng, Y., Si, X., & Liu, Z. (2016). Thuctc: an efficient chinese text classifier. *GitHub Repository*.

Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1556–1566, Beijing, China. Association for Computational Linguistics.

Teng, Z., & Zhang, Y. (2018). Two Local Models for Neural Constituent Parsing. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pp. 119–132.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., & Polosukhin, I. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 5998–6008.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.

Wang, Z., Mi, H., & Xue, N. (2015). Feature Optimization for Constituent Parsing via Neural Networks. 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)*, pp. 1138–1147.

Weiss, D., Alberti, C., Collins, M., & 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)*, pp. 323–333, Beijing, China.

Yan, C. (2009). The "sound, form, meaning, and law" of Chinese character as the basic structural unit. *Journal of PLA University of Foreign Languages*, pp. 1–7.

Yan, H., Qiu, X., & Huang, X. (2020). A graph-based model for joint chinese word segmentation and dependency parsing. *Transactions of the Association for Computational Linguistics*, *8*, 78–92.

Yang, M. (2006). *Research on Word Formation of Modern Chinese Compound Words*. Ph.D. thesis, Nanjing Normal University.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding.. *abs/1906.08237*.

Yoshikawa, M., Noji, H., & Matsumoto, Y. (2017). A* CCG Parsing with a Supertag and Dependency Factored Model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 277–287.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n3. *Information and control*, *10*(2), 189–208.

Yuan, R., & Lv, S. (1979). *Spoken Chinese Grammar*. Beijing Commercial Press.

Zhang, M., Zhang, Y., Che, W., & Liu, T. (2013). Chinese parsing exploiting characters. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 125–134, Sofia, Bulgaria.

Zhang, M., Zhang, Y., Che, W., & Liu, T. (2014). Character-level Chinese dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1326–1336, Baltimore, Maryland.

Zhang, Y., Li, C., Barzilay, R., & Darwish, K. (2015). Randomized greedy inference for joint segmentation, POS tagging and dependency parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 42–52, Denver, Colorado. Association for Computational Linguistics.

Zhang, Y., & 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)*, pp. 562–571, Honolulu, Hawaii.

Zhang, Z., Zhao, H., & 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)*, pp. 1382–1392, Berlin, Germany.

Zhao, H. (2011). Integrating unsupervised and supervised word segmentation: The role of goodness measures. *Information Sciences*, *181*(1), 163–183.

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)*, pp. 879–887, Athens, Greece.

Zhao, H., Cai, D., Huang, C., & Kit, C. (2019). Chinese word segmentation: Another decade review (2007-2017). *CoRR*, *abs/1901.06079*.

Zhao, H., Huang, C. N., & Li, M. (2006). An improved Chinese word segmentation system with conditional random field. In *the Fifth SIGHAN Workshop on Chinese Language Processing (SIGHAN)*, pp. 162–165, Sydney, Australia.

Zhao, H., Huang, C. N., Li, M., & 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., Kit, C., & 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)*, pp. 82–88, Yantai, China.

Zheng, C., Huang, M., & Sun, A. (2019). ChID: A large-scale Chinese IDiom dataset for cloze test. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 778–787, Florence, Italy. Association for Computational Linguistics.

Zhou, J., Zhang, Z., Zhao, H., & Zhang, S. (2020). LIMIT-BERT : Linguistics informed multi-task BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4450–4461, Online. Association for Computational Linguistics.

Zhou, J., & Zhao, H. (2019). Head-Driven Phrase Structure Grammar Parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2396–2408, Florence, Italy. Association for Computational Linguistics.

Zhu, M., Zhang, Y., Chen, W., Zhang, M., & Zhu, J. (2013). Fast and Accurate Shift-Reduce Constituent Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 434–443.