# Character-Level Chinese Dependency Parsing

**Meishan Zhang**[†], **Yue Zhang**[‡] , **Wanxiang Che**[†], **Ting Liu**[†]*
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{mszhang, car, tliu}@ir.hit.edu.cn
[‡]Singapore University of Technology and Design
yue_zhang@sutd.edu.sg

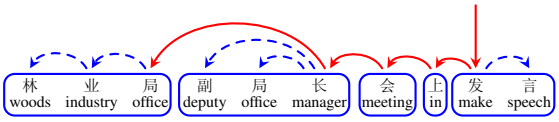
## Abstract

Recent work on Chinese analysis has led to large-scale annotations of the internal structures of words, enabling character-level analysis of Chinese syntactic structures. In this paper, we investigate the problem of character-level Chinese dependency parsing, building dependency trees over characters. Character-level information can benefit downstream applications by offering flexible granularities for word segmentation while improving word-level dependency parsing accuracies. We present novel adaptations of two major shift-reduce dependency parsing algorithms to character-level parsing. Experimental results on the Chinese Treebank demonstrate improved performances over word-based parsing methods.

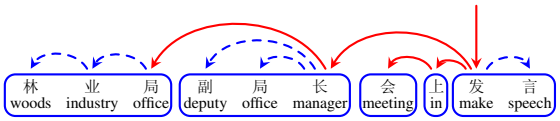
## 1 Introduction

As a light-weight formalism offering syntactic information to downstream applications such as SMT, the dependency grammar has received increasing interest in the syntax parsing community (McDonald et al., 2005; Nivre and Nilsson, 2005; Carreras et al., 2006; Duan et al., 2007; Koo and Collins, 2010; Zhang and Clark, 2008; Nivre, 2008; Bohnet, 2010; Zhang and Nivre, 2011; Choi and McCallum, 2013). Chinese dependency trees were conventionally defined over *words* (Chang et al., 2009; Li et al., 2012), requiring word segmentation and POS-tagging as pre-processing steps. Recent work on Chinese analysis has embarked on investigating the syntactic roles of characters, leading to large-scale annotations of word internal structures (Li, 2011; Zhang et al., 2013). Such annotations enable dependency parsing on the character level, building dependency trees over Chinese *characters*. Figure 1(c) shows an example of a character-level dependency tree, where the leaf nodes are Chinese characters.


(a) a word-based dependency tree




(b) a character-level dependency tree by Zhao (2009) with real intra-word and pseudo inter-word dependencies




(c) a character-level dependency tree investigated in this paper with both real intra- and inter-word dependencies

Figure 1: An example character-level dependency tree. "林业局副局长在大会上发言 (The deputy director of forestry administration make a speech in the meeting)".

Character-level dependency parsing is interesting in at least two aspects. First, character-level trees circumvent the issue that no universal standard exists for Chinese word segmentation. In the well-known Chinese word segmentation bakeoff tasks, for example, different segmentation standards have been used by different data sets (Emerson, 2005). On the other hand, most disagreement on segmentation standards boils down to disagreement on segmentation granularity. As demonstrated by Zhao (2009), one can extract both fine-grained and coarse-grained words from character-level dependency trees, and hence can adapt to flexible segmentation standards using this formalism. In Figure 1(c), for example, "副局长 (deputy

*Corresponding author.



*Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1326–1336,
Baltimore, Maryland, USA, June 23-25 2014. 

director)" can be segmented as both "副 (deputy) | 局长 (director)" and "副局长 (deputy director)", but not "副 (deputy) 局 (office) | 长 (manager)", by dependency coherence. Chinese language processing tasks, such as machine translation, can benefit from flexible segmentation standards (Zhang et al., 2008; Chang et al., 2008).

Second, word internal structures can also be useful for syntactic parsing. Zhang et al. (2013) have shown the usefulness of word structures in Chinese constituent parsing. Their results on the Chinese Treebank (CTB) showed that character-level constituent parsing can bring increased performances even with the pseudo word structures. They further showed that better performances can be achieved when manually annotated word structures are used instead of pseudo structures.

In this paper, we make an investigation of character-level Chinese dependency parsing using Zhang et al. (2013)'s annotations and based on a transition-based parsing framework (Zhang and Clark, 2011). There are two dominant transition-based dependency parsing systems, namely the arc-standard and the arc-eager parsers (Nivre, 2008). We study both algorithms for character-level dependency parsing in order to make a comprehensive investigation. For direct comparison with word-based parsers, we incorporate the traditional word segmentation, POS-tagging and dependency parsing stages in our joint parsing models. We make changes to the original transition systems, and arrive at two novel transition-based character-level parsers.

We conduct experiments on three data sets, including CTB 5.0, CTB 6.0 and CTB 7.0. Experimental results show that the character-level dependency parsing models outperform the word-based methods on all the data sets. Moreover, manually annotated intra-word dependencies can give improved word-level dependency accuracies than pseudo intra-word dependencies. These results confirm the usefulness of character-level syntax for Chinese analysis. The source codes are freely available at `http://sourceforge.net/projects/zpar/, version 0.7`.

## 2 Character-Level Dependency Tree

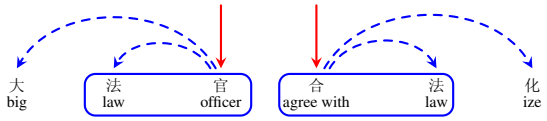
Character-level dependencies were first proposed by Zhao (2009). They show that by annotating character dependencies within words, one can adapt to different segmentation standards. The dependencies they study are restricted to intra-word characters, as illustrated in Figure 1(b). For inter-word dependencies, they use a pseudo right-headed representation.

In this study, we integrate inter-word syntactic dependencies and intra-word dependencies using large-scale annotations of word internal structures by Zhang et al. (2013), and study their interactions. We extract unlabeled dependencies from bracketed word structures according to Zhang et al.'s head annotations. In Figure 1(c), the dependencies shown by dashed arcs are *intra-word* dependencies, which reflect the internal word structures, while the dependencies with solid arcs are *inter-word* dependencies, which reflect the syntactic structures between words.

In this formulation, a character-level dependency tree satisfies the same constraints as the traditional word-based dependency tree for Chinese, including projectivity. We differentiate intra-word dependencies and inter-word dependencies by the arc type, so that our work can be compared with conventional word segmentation, POS-tagging and dependency parsing pipelines under a canonical segmentation standard.

The character-level dependency trees hold to a specific word segmentation standard, but are not limited to it. We can extract finer-grained words of different granulities from a coarse-grained word by taking projective subtrees of different sizes. For example, taking all the intra-word modifier nodes of "长 (manager)" in Figure 1(c) results in the word "副局长 (deputy director)", while taking the first modifier node of "长 (manager)" results in the word "局长 (director)". Note that "副局 (deputy office)" cannot be a word because it does not form a projective span without "长 (manager)".

Inner-word dependencies can also bring benefits to parsing word-level dependencies. The head character can be a less sparse feature compared to a word. As intra-word dependencies lead to fine-grained subwords, we can also use these subwords for better parsing. In this work, we use the innermost left/right subwords as atomic features. To extract the subwords, we find the innermost left/right modifiers of the head character, respectively, and then conjoin them with all their descendant characters to form the smallest left/right subwords. Figure 2 shows an example, where the smallest left subword of "大法官 (chief lawyer)" is "法官 (lawyer)", and the smallest right subword




(a) smallest left subword (b) smallest right subword

Figure 2: An example to illustrate the innermost left/right subwords.

of "合法化 (legalize)" is "合法 (legal)".

## 3 Character-Level Dependency Parsing

A transition-based framework with global learning and beam search decoding (Zhang and Clark, 2011) has been applied to a number of natural language processing tasks, including word segmentation, POS-tagging and syntactic parsing (Zhang and Clark, 2010; Huang and Sagae, 2010; Bohnet and Nivre, 2012; Zhang et al., 2013). It models a task incrementally from a start state to an end state, where each intermediate state during decoding can be regarded as a partial output. A number of actions are defined so that the state advances step by step. To learn the model parameters, it usually uses the online perceptron algorithm with early-update under the inexact decoding condition (Collins, 2002; Collins and Roark, 2004). Transition-based dependency parsing can be modeled under this framework, where the state consists of a stack and a queue, and the set of actions can be either the arc-eager (Zhang and Clark, 2008) or the arc-standard (Huang et al., 2009) transition systems.

When the internal structures of words are annotated, character-level dependency parsing can be treated as a special case of word-level dependency parsing, with "words" being "characters". A big weakness of this approach is that full words and POS-tags cannot be used for feature engineering. Both are crucial to well-established features for word segmentation, POS-tagging and syntactic parsing. In this section, we introduce novel extensions to the arc-standard and the arc-eager transition systems, so that word-based and character-based features can be used simultaneously for character-level dependency parsing.

### 3.1 The Arc-Standard Model

The arc-standard model has been applied to joint segmentation, POS-tagging and dependency parsing (Hatori et al., 2012), but with pseudo word structures. For unified processing of annotated word structures and fair comparison between character-level arc-eager and arc-standard systems, we define a different arc-standard transition system, consistent with our character-level arc-eager system.

In the word-based arc-standard model, the transition state includes a stack and a queue, where the stack contains a sequence of partially-parsed dependency trees, and the queue consists of unprocessed input words. Four actions are defined for state transition, including *arc-left* (AL, which creates a left arc between the top element $s_0$ and the second top element $s_1$ on the stack), *arc-right* (AR, which creates a right arc between $s_0$ and $s_1$), *pop-root* (PR, which defines the root node of a dependency tree when there is only one element on the stack and no element in the queue), and the last *shift* (SH, which shifts the first element $q_0$ of the queue onto the stack).

For character-level dependency parsing, there are two types of dependencies: inter-word dependencies and intra-word dependencies. To parse them with both character and word features, we extend the original transition actions into two categories, for inter-word dependencies and intra-word dependencies, respectively. The actions for inter-word dependencies include *inter-word arc-left* ($\text{AL}_\text{w}$), *inter-word arc-right* ($\text{AR}_\text{w}$), *pop-root* (PR) and *inter-word shift* ($\text{SH}_\text{w}$). Their definitions are the same as the word-based model, with one exception that the *inter-word shift* operation has a parameter denoting the POS-tag of the incoming word, so that POS disambiguation is performed by the $\text{SH}_\text{w}$ action.

The actions for intra-word dependencies include *intra-word arc-left* ($\text{AL}_\text{c}$), *intra-word arc-right* ($\text{AR}_\text{c}$), *pop-word* (PW) and *inter-word shift* ($\text{SH}_\text{c}$). The definitions of $\text{AL}_\text{c}$, $\text{AR}_\text{c}$ and $\text{SH}_\text{c}$ are the same as the word-based arc-standard model, while PW changes the top element on the stack into a full-word node, which can only take inter-word dependencies. One thing to note is that, due to variable word sizes in character-level parsing, the number of actions can vary between different sequences of actions corresponding to different analyses. We use the padding method (Zhu et al., 2013), adding an *IDLE* action to finished transition action sequences, for better alignments between states in the beam.

In the character-level arc-standard transition

| step | action | stack | queue | dependencies |
|---|---|---|---|---|
| 0 | - | $\phi$ | 林 业 $\cdots$ | $\phi$ |
| 1 | $SH_w$(NR) | 林/NR | 业 局 $\cdots$ | $\phi$ |
| 2 | $SH_c$ | 林/NR 业/NR | 局 副 $\cdots$ | $\phi$ |
| 3 | $AL_c$ | 业/NR | 局 副 $\cdots$ | $A_1 = \{林^\frown业\}$ |
| 4 | $SH_c$ | 业/NR 局/NR | 副 局 $\cdots$ | $A_1$ |
| 5 | $AL_c$ | 局/NR | 副 局 $\cdots$ | $A_2 = A_1 \bigcup \{业^\frown局\}$ |
| 6 | PW | 林业局/NR | 副 局 $\cdots$ | $A_2$ |
| 7 | $SH_w$(NN) | 林业局/NR 副/NN | 局 长 $\cdots$ | $A_2$ |
| $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ |
| 12 | PW | 林业局/NR 副局长/NN | 会 上 $\cdots$ | $A_i$ |
| 13 | $AL_w$ | 副局长/NN | 会 上 $\cdots$ | $A_{i+1} = A_i \bigcup \{林业局/NR^\frown副局长/NN\}$ |
| $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ |

(a) character-level dependency parsing using the arc-standard algorithm

| step | action | stack | deque | queue | dependencies |
|---|---|---|---|---|---|
| 0 | - | $\phi$ | | 林 业 $\cdots$ | |
| 1 | $SH_c$(NR) | $\phi$ | 林/NR | 业 局 $\cdots$ | $\phi$ |
| 2 | $AL_c$ | $\phi$ | $\phi$ | 业/NR 局 $\cdots$ | $A_1 = \{林^\frown业\}$ |
| 3 | $SH_c$ | $\phi$ | 业/NR | 局 副 $\cdots$ | $A_1$ |
| 4 | $AL_c$ | $\phi$ | $\phi$ | 局/NR 副 $\cdots$ | $A_2 = A_1 \bigcup \{业^\frown局\}$ |
| 5 | $SH_c$ | $\phi$ | 局/NR | 副 局 $\cdots$ | $A_2$ |
| 6 | PW | $\phi$ | 林业局/NR | 副 局 $\cdots$ | $A_2$ |
| 7 | $SH_w$ | 林业局/NR | $\phi$ | 副 局 $\cdots$ | $A_2$ |
| $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ |
| 13 | PW | 林业局/NR | 副局长/NN | 会 上 $\cdots$ | $A_i$ |
| 14 | $AL_w$ | $\phi$ | 副局长/NN | 会 上 $\cdots$ | $A_{i+1} = A_i \bigcup \{林业局/NR^\frown副局长/NN\}$ |
| $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ |

(b) character-level dependency parsing using the arc-eager algorithm, $t = 1$

Figure 3: Character-level dependency parsing of the sentence in Figure 1(c).

system, each word is initialized by the action $SH_w$ with a POS tag, before being incrementally modified by a sequence of intra-word actions, and finally being completed by the action PW. The inter-word actions can be applied when all the elements on the stack are full-word nodes, while the intra-word actions can be applied when at least the top element on the stack is a partial-word node. For the actions $AL_c$ and $AR_c$ to be valid, the top two elements on the stack are both partial-word nodes. For the action PW to be valid, only the top element on the stack is a partial-word node. Figure 3(a) gives an example action sequence.

There are three types of features. The first two types are traditionally established features for the dependency parsing and joint word segmentation and POS-tagging tasks. We use the features proposed by Hatori et al. (2012). The word-level dependency parsing features are added when the inter-word actions are applied, and the features for joint word segmentation and POS-tagging are added when the actions PW, $SH_w$ and $SH_c$ are applied. Following the work of Hatori et al. (2012), we have a parameter $\alpha$ to adjust the weights for joint word segmentation and POS-tagging features. We apply word-based dependency parsing features to intra-word dependency parsing as well, by using subwords (the conjunction of characters spanning the head node) to replace words in word features. The third type of features is word-structure features. We extract the head character and the smallest subwords containing the head character from the intra-word dependencies (Section 2). Table 1 summarizes the features.

### 3.2 The Arc-Eager Model

Similar to the arc-standard case, the state of a word-based arc-eager model consists of a stack and a queue, where the stack contains a sequence of partial dependency trees, and the queue consists of unprocessed input words. Unlike the arc-standard model, which builds dependencies on the top two elements on the stack, the arc-eager model builds dependencies between the top element of the stack and the first element of the queue. Five actions are defined for state transformation: *arc-left* (AL, which creates a left arc between the top element of the stack $s_0$ and the first element in the queue $q_0$, while popping $s_0$ off the stack), *arc-right* (AR, which creates a right arc between

| Feature templates |
|---|
| $L\underline{c}$, $L\underline{c}\underline{t}$, $R\underline{c}$, $R\underline{c}\underline{t}$, $L_{lc1}\underline{c}$, $L_{rc1}\underline{c}$, $R_{lc1}\underline{c}$, |
| $L\underline{c} \cdot R\underline{c}$, $L_{lc1}\underline{c}\underline{t}$, $L_{rc1}\underline{c}\underline{t}$, $R_{lc1}\underline{c}\underline{t}$, |
| $L\underline{c} \cdot R\underline{w}$, $L\underline{w} \cdot R\underline{c}$, $L\underline{c}\underline{t} \cdot R\underline{w}$, |
| $L\underline{w}\underline{t} \cdot R\underline{c}$, $L\underline{w} \cdot R\underline{c}\underline{t}$, $L\underline{c} \cdot R\underline{w}\underline{t}$, |
| $L\underline{c} \cdot R\underline{c} \cdot L_{lc1}\underline{c}$, $L\underline{c} \cdot R\underline{c} \cdot L_{rc1}\underline{c}$, |
| $L\underline{c} \cdot R\underline{c} \cdot L_{lc2}\underline{c}$, $L\underline{c} \cdot R\underline{c} \cdot L_{rc2}\underline{c}$, |
| $L\underline{c} \cdot R\underline{c} \cdot R_{lc1}\underline{c}$, $L\underline{c} \cdot R\underline{c} \cdot R_{lc2}\underline{c}$, |
| $Ll\underline{sw}$, $Lr\underline{sw}$, $Rl\underline{sw}$, $Rr\underline{sw}$, $Ll\underline{swt}$, |
| $Lr\underline{swt}$, $Rl\underline{swt}$, $Rr\underline{swt}$, $Ll\underline{sw} \cdot R\underline{w}$, |
| $Lr\underline{sw} \cdot R\underline{w}$, $L\underline{w} \cdot Rl\underline{sw}$, $L\underline{w} \cdot Rr\underline{sw}$ |

Table 1: Feature templates encoding intra-word dependencies. $L$ and $R$ denote the two elements over which the dependencies are built; the subscripts $lc1$ and $rc1$ denote the left-most and right-most children, respectively; the subscripts $lc2$ and $rc2$ denote the second left-most and second right-most children, respectively; $\underline{w}$ denotes the word; $\underline{t}$ denotes the POS tag; $\underline{c}$ denotes the head character; $\underline{lsw}$ and $\underline{rsw}$ denote the smallest left and right subwords respectively, as shown in Figure 2.

$s_0$ and $q_0$, while shifting $q_0$ from the queue onto the stack), *pop-root* (PR, which defines the ROOT node of the dependency tree when there is only one element on the stack and no element in the queue), *reduce* (RD, which pops $s_0$ off the stack), and *shift* (SH, which shifts $q_0$ onto the stack).

There is no previous work that exploits the arc-eager algorithm for jointly performing POS-tagging and dependency parsing. Since the first element of the queue can be shifted onto the stack by either SH or AR, it is more difficult to assign a POS tag to each word by using a single action. In this work, we make a change to the configuration state, adding a deque between the stack and the queue to save partial words with intra-word dependencies. We divide the transition actions into two categories, one for inter-word dependencies ($AR_w$, $AL_w$, $SH_w$, $RD_w$ and PR) and the other for intra-word dependencies ($AR_c$, $AL_c$, $SH_c$, $RD_c$ and PW), requiring that the intra-word actions be operated between the deque and the queue, while the inter-word actions be operated between the stack and the deque.

For character-level arc-eager dependency parsing, the inter-word actions are the same as the word-based methods. The actions $AL_c$ and $AR_c$ are the same as $AL_w$ and $AR_w$, except that they operate on characters, but the $SH_c$ operation has a parameter to denote the POS tag of a word. The PW action recognizes a full-word. We also have an *IDLE* action, for the same reason as the arc-standard model.

In the character-level arc-eager transition system, a word is formed in a similar way with that of character-level arc-standard algorithm. Each word is initialized by the action $SH_c$ with a POS tag, and then incrementally changed a sequence of intra-word actions, before being finalized by the action PW. All these actions operate between the queue and deque. For the action PW, only the first element in the deque (close to the queue) is a partial-word node. For the actions $AR_c$ and $AL_c$ to be valid, the first element in the deque must be a partial-word node. The action $SH_c$ have a POS tag when shifting the first character of a word,but does not have such a parameter when shifting the next characters of a word. For the action $SH_c$ with a POS tag to be valid, the first element in the deque must be a full-word node. Different from the arc-standard model, at any stage we can choose either the action $SH_c$ with a POS tag to initialize a new word on the deque, or the inter-word actions on the stack. In order to eliminate the ambiguity, we define a new parameter $t$ to limit the max size of the deque. If the deque is full with $t$ words, inter-word actions are performed; otherwise intra-word actions are performed. All the inter-word actions must be applied on full-word nodes between the stack an the deque. Figure 3(b) gives an example action sequence.

Similar to the arc-standard case, there are three types of features, with the first two types being traditionally established features for dependency parsing and joint word segmentation and POS-tagging. The dependency parsing features are taken from the work of Zhang and Nivre (2011), and the features for joint word segmentation and POS-tagging are taken from Zhang and Clark (2010)[1]. The word-level dependency parsing features are triggered when the inter-word actions are applied, while the features of joint word segmentation and POS-tagging are added when the actions $SH_c$, $AR_c$ and PW are applied. Again we use a parameter $\alpha$ to adjust the weights for joint word segmentation and POS-tagging features. The word-level features for dependency parsing are applied to intra-word dependency parsing as well, by using subwords to replace words. The third type of features is word-structure features, which are the

[1]Since Hatori et al. (2012) also use Zhang and Clark (2010)'s features, the arc-standard and arc-eager character-level dependency parsing models have the same features for joint word segmentation and POS-tagging.

| | | CTB50 | CTB60 | CTB70 |
|---|---|---|---|---|
| Training | #sent | 18k | 23k | 31k |
| | #word | 494k | 641k | 718k |
| Development | #sent | 350 | 2.1k | 10k |
| | #word | 6.8k | 60k | 237k |
| | #oov | 553 | 3.3k | 13k |
| Test | #sent | 348 | 2.8k | 10k |
| | #word | 8.0k | 82k | 245k |
| | #oov | 278 | 4.6k | 13k |

Table 2: Statistics of datasets.

same as those of the character-level arc-standard model, shown in Table 1.

## 4 Experiments

### 4.1 Experimental Settings

We use the Chinese Penn Treebank 5.0, 6.0 and 7.0 to conduct the experiments, splitting the corpora into training, development and test sets according to previous work. Three different splitting methods are used, namely CTB50 by Zhang and Clark (2010), CTB60 by the official documentation of CTB 6.0, and CTB70 by Wang et al. (2011). The dataset statistics are shown in Table 2. We use the head rules of Zhang and Clark (2008) to convert phrase structures into dependency structures. The intra-word dependencies are extracted from the annotations of Zhang et al. (2013)[2].

The standard measures of word-level precision, recall and F1 score are used to evaluate word segmentation, POS-tagging and dependency parsing, following Hatori et al. (2012). In addition, we use the same measures to evaluate intra-word dependencies, which indicate the performance of predicting word structures. A word's structure is correct only if all the intra-word dependencies are all correctly recognized.

### 4.2 Baseline and Proposed Models

For the baseline, we have two different pipeline models. The first consists of a joint segmentation and POS-tagging model (Zhang and Clark, 2010) and a word-based dependency parsing model using the arc-standard algorithm (Huang et al., 2009). We name this model STD (pipe). The second consists of the same joint segmentation and POS-tagging model and a word-based dependency parsing model using the arc-eager algorithm (Zhang and Nivre, 2011). We name this model EAG (pipe). For the pipeline models, we use a beam of size 16 for joint segmentation and POS-tagging, and a beam of size 64 for dependency parsing, according to previous work.

We study the following character-level dependency parsing models:

- STD (real, pseudo): the arc-standard model with annotated intra-word dependencies and pseudo inter-word dependencies;
- STD (pseudo, real): the arc-standard model with pseudo intra-word dependencies and real inter-word dependencies;
- STD (real, real): the arc-standard model with annotated intra-word dependencies and real inter-word dependencies;
- EAG (real, pseudo): the arc-eager model with annotated intra-word dependencies and pseudo inter-word dependencies;
- EAG (pseudo, real): the arc-eager model with pseudo intra-word dependencies and real inter-word dependencies;
- EAG (real, real): the arc-eager model with annotated intra-word dependencies and real inter-word dependencies.

The annotated intra-word dependencies refer to the dependencies extracted from annotated word structures, while the pseudo intra-word dependencies used in the above models are similar to those of Hatori et al. (2012). For a given word $w = c_1 c_2 \cdots c_m$, the intra-word dependency structure is $c_1 \curvearrowleft c_2 \curvearrowleft \cdots \curvearrowleft c_m$[3]. The real inter-word dependencies refer to the syntactic word-level dependencies by head-finding rules from CTB, while the pseudo inter-word dependencies refer to the word-level dependencies used by Zhao (2009) ($w_1 \curvearrowleft w_2 \curvearrowleft \cdots \curvearrowleft w_n$). The character-level models with annotated intra-word dependencies and pseudo inter-word dependencies are compared with the pipelines on word segmentation and POS-tagging accuracies, and are compared with the character-level models with annotated intra-word dependencies and real inter-word dependencies on word segmentation, POS-tagging and word-structure predicating accuracies. All the proposed

[2] https://github.com/zhangmeishan/wordstructures; their annotation was conducted on CTB 5.0, while we made annotations of the remainder of the CTB 7.0 words. We also make the annotations publicly available at the same site.

[3] We also tried similar structures with right arcs, which gave lower accuracies.

| STD (real, real) | SEG | POS | DEP | WS |
|---|---|---|---|---|
| $\alpha = 1$ | 95.85 | 91.60 | 76.96 | 95.14 |
| $\alpha = 2$ | 96.09 | 91.89 | 77.28 | **95.29** |
| $\alpha = 3$ | 96.02 | 91.84 | 77.22 | 95.23 |
| $\alpha = \mathbf{4}$ | **96.10** | **91.96** | **77.49** | **95.29** |
| $\alpha = 5$ | 96.07 | 91.90 | 77.31 | 95.21 |

Table 3: Development test results of the character-level arc-standard model on CTB60.

| EAG (real, real) | | SEG | POS | DEP | WS |
|---|---|---|---|---|---|
| $\alpha = 1$ | $t = 1$ | **96.00** | 91.66 | 74.63 | **95.49** |
| | $t = 2$ | 95.93 | 91.75 | 76.60 | 95.37 |
| | $\mathbf{t = 3}$ | 95.93 | **91.74** | **76.94** | 95.36 |
| | $t = 4$ | 95.91 | 91.71 | 76.82 | 95.33 |
| | $t = 5$ | 95.95 | 91.73 | 76.84 | 95.40 |
| $t = 3$ | $\alpha = 1$ | 95.93 | 91.74 | 76.94 | 95.36 |
| | $\alpha = 2$ | 96.11 | 91.99 | 77.17 | 95.56 |
| | $\alpha = \mathbf{3}$ | **96.16** | **92.01** | **77.48** | **95.62** |
| | $\alpha = 4$ | 96.11 | 91.93 | 77.40 | 95.53 |
| | $\alpha = 5$ | 96.00 | 91.84 | 77.10 | 95.43 |

Table 4: Development test results of the character-level arc-eager model on CTB60.

models use a beam of size 64 after considering both speeds and accuracies.

### 4.3 Development Results

Our development tests are designed for two purposes: adjusting the parameters for the two proposed character-level models and testing the effectiveness of the novel word-structure features. Tuning is conducted by maximizing word-level dependency accuracies. All the tests are conducted on the CTB60 data set.

#### 4.3.1 Parameter Tuning

For the arc-standard model, there is only one parameter $\alpha$ that needs tuning. It adjusts the weights of segmentation and POS-tagging features, because the number of feature templates is much less for the two tasks than for parsing. We set the value of $\alpha$ to $1 \cdots 5$, respectively. Table 3 shows the accuracies on the CTB60 development set. According to the results, we use $\alpha = 4$ for our final character-level arc-standard model.

For the arc-eager model, there are two parameters $t$ and $\alpha$. $t$ denotes the deque size of the arc-eager model, while $\alpha$ shares the same meaning as the arc-standard model. We take two steps for parameter tuning, first adjusting the more crucial parameter $t$ and then adjusting $\alpha$ on the best $t$. Both parameters are assigned the values of 1 to 5. Table 4 shows the results. According to results, we set $t = 3$ and $\alpha = 3$ for the final character-level arc-eager model, respectively.

| | SEG | POS | DEP | WS |
|---|---|---|---|---|
| STD (real, real) | 96.10 | 91.96 | 77.49 | 95.29 |
| STD (real, real)/wo | 95.99 | 91.79 | 77.19 | 95.35 |
| $\Delta$ | -0.11 | -0.17 | -0.30 | +0.06 |
| EAG (real, real) | 96.16 | 92.01 | 77.48 | 95.62 |
| EAG (real, real)/wo | 96.09 | 91.82 | 77.12 | 95.56 |
| $\Delta$ | -0.07 | -0.19 | -0.36 | -0.06 |

Table 5: Feature ablation tests for the novel word-structure features, where "/wo" denotes the corresponding models without the novel intra-word dependency features.

#### 4.3.2 Effectiveness of Word-Structure Features

To test the effectiveness of our novel word-structure features, we conduct feature ablation experiments on the CTB60 development data set for the proposed arc-standard and arc-eager models, respectively. Table 5 shows the results. We can see that both the two models achieve better accuracies on word-level dependencies with the novel word-structure features, while the features do not affect word-structure predication significantly.

### 4.4 Final Results

Table 6 shows the final results on the CTB50, CTB60 and CTB70 data sets, respectively. The results demonstrate that the character-level dependency parsing models are significantly better than the corresponding word-based pipeline models, for both the arc-standard and arc-eager systems. Similar to the findings of Zhang et al. (2013), we find that the annotated word structures can give better accuracies than pseudo word structures. Another interesting finding is that, although the arc-eager algorithm achieves lower accuracies in the word-based pipeline models, it obtains comparative accuracies in the character-level models.

We also compare our results to those of Hatori et al. (2012), which is comparable to STD (pseudo, real) since similar arc-standard algorithms and features are used. The major difference is the set of transition actions. We rerun their system on the three datasets[4]. As shown in Table 6, our arc-standard system with pseudo word structures

[4] `http://triplet.cc/`. We use a different constituent-to-dependency conversion scheme in comparison with Hatori et al. (2012)'s work.

| Model | CTB50 | | | | CTB60 | | | | CTB70 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SEG | POS | DEP | WS | SEG | POS | DEP | WS | SEG | POS | DEP | WS |
| The arc-standard models | | | | | | | | | | | | |
| STD (pipe) | 97.53 | 93.28 | 79.72 | – | 95.32 | 90.65 | 75.35 | – | 95.23 | 89.92 | 73.93 | – |
| STD (real, pseudo) | 97.78 | 93.74 | – | **97.40** | **95.77**‡ | 91.24‡ | – | **95.08** | **95.59**‡ | 90.49‡ | – | **94.97** |
| STD (pseudo, real) | 97.67 | 94.28‡ | 81.63‡ | – | 95.63‡ | **91.40**‡ | 76.75‡ | – | 95.53‡ | 90.75‡ | 75.63‡ | – |
| STD (real, real) | **97.84** | **94.62**‡ | **82.14**‡ | 97.30 | 95.56‡ | 91.39‡ | **77.09**‡ | 94.80 | 95.51‡ | **90.76**‡ | **75.70**‡ | 94.78 |
| Hatori+ '12 | 97.75 | 94.33 | 81.56 | – | 95.26 | 91.06 | 75.93 | – | 95.27 | 90.53 | 74.73 | – |
| The arc-eager models | | | | | | | | | | | | |
| EAG (pipe) | 97.53 | 93.28 | 79.59 | – | 95.32 | 90.65 | 74.98 | – | 95.23 | 89.92 | 73.46 | – |
| EAG (real, pseudo) | 97.75 | 93.88 | – | 97.45 | 95.63‡ | 91.07‡ | – | 95.06 | **95.50**‡ | 90.36‡ | – | **95.00** |
| EAG (pseudo, real) | 97.76 | **94.36**‡ | 81.70‡ | – | 95.63‡ | 91.34‡ | 76.87‡ | – | 95.39‡ | 90.56‡ | 75.56‡ | – |
| EAG (real, real) | **97.84** | **94.36**‡ | **82.07**‡ | **97.49** | **95.71**‡ | **91.51**‡ | **76.99**‡ | **95.16** | 95.47‡ | **90.72**‡ | **75.76**‡ | 94.94 |

Table 6: Main results, where the results marked with ‡ denote that the p-value is less than 0.001 compared with the pipeline word-based models using pairwise t-test.

brings consistent better accuracies than their work on all the three data sets.

Both the pipelines and character-level models with pseudo inter-word dependencies perform word segmentation and POS-tagging jointly, without using real word-level syntactic information. A comparison between them (STD/EAG (pipe) vs. STD/EAG (real, pseudo)) reflects the effectiveness of annotated intra-word dependencies on segmentation and POS-tagging. We can see that both the arc-standard and arc-eager models with annotated intra-word dependencies can improve the segmentation accuracies by 0.3% and the POS-tagging accuracies by 0.5% on average on the three datasets. Similarly, a comparison between the character-level models with pseudo inter-word dependencies and the character-level models with real inter-word dependencies (STD/EAG (real, pseudo) vs. STD/EAG (real, real)) can reflect the effectiveness of annotated inter-word structures on morphology analysis. We can see that improved POS-tagging accuracies are achieved using the real inter-word dependencies when jointly performing inner- and inter-word dependencies. However, we find that the inter-word dependencies do not help the word-structure accuracies.

## 4.5 Analysis

To better understand the character-level parsing models, we conduct error analysis in this section. All the experiments are conducted on the CTB60 test data sets. The new advantage of the character-level models is that one can parse the internal word structures of intra-word dependencies. Thus we are interested in their capabilities of predicting word structures. We study the word-structure accuracies in two aspects, including OOV, word length, POS tags and the parsing model.

### 4.5.1 OOV

The word-structure accuracy of OOV words reflects a model's ability of handling unknown words. The overall recalls of OOV word structures are 67.98% by STD (real, real) and 69.01% by EAG (real, real), respectively. We find that most errors are caused by failures of word segmentation. We further investigate the accuracies when words are correctly segmented, where the accuracies of OOV word structures are 87.64% by STD (real, real) and 89.07% by EAG (real, real). The results demonstrate that the structures of Chinese words are not difficult to predict, and confirm the fact that Chinese word structures have some common syntactic patterns.

### 4.5.2 Parsing Model

From the above analysis in terms of OOV, word lengths and POS tags, we can see that the EAG (real, real) model and the STD (real, real) models behave similarly on word-structure accuracies. Here we study the two models more carefully, comparing their word accuracies sentence by sentence. Figure 4 shows the results, where each point denotes a sentential comparison between STD (real, real) and EAG (real, real), the x-axis denotes the sentential word-structure accuracy of STD (real, real), and the y-axis denotes that of EAG (real, real). The points at the diagonal show the same accuracies by the two models, while others show that the two models perform differently on the corresponding sentences. We can see that most points are beyond the diagonal line, indicat-

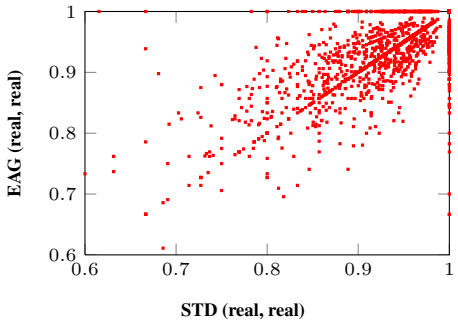



Figure 4: Sentential word-structure accuracies of STD (real, real) and EAG (real, real).

ing that the two parsing models can be complementary in parsing intra-word dependencies.

## 5 Related Work

Zhao (2009) was the first to study character-level dependencies; they argue that since no consistent word boundaries exist over Chinese word segmentation, dependency-based representations of word structures serve as a good alternative for Chinese word segmentation. Thus their main concern is to parse intra-word dependencies. In this work, we extend their formulation, making use of large-scale annotations of Zhang et al. (2013), so that the syntactic word-level dependencies can be parsed together with intra-word dependencies.

Hatori et al. (2012) proposed a joint model for Chinese word segmentation, POS-tagging and dependency parsing, studying the influence of joint model and character features for parsing, Their model is extended from the arc-standard transition-based model, and can be regarded as an alternative to the arc-standard model of our work when pseudo intra-word dependencies are used. Similar work is done by Li and Zhou (2012). Our proposed arc-standard model is more concise while obtaining better performance than Hatori et al. (2012)'s work. With respect to word structures, real intra-word dependencies are often more complicated, while pseudo word structures cannot be used to correctly guide segmentation.

Zhao (2009), Hatori et al. (2012) and our work all study character-level dependency parsing. While Zhao (2009) focus on word internal structures using pseudo inter-word dependencies, Hatori et al. (2012) investigate a joint model using pseudo intra-word dependencies. We use manual dependencies for both inner- and inter-word structures, studying their influences on each other.

Zhang et al. (2013) was the first to perform Chinese syntactic parsing over characters. They extended word-level constituent trees by annotated word structures, and proposed a transition-based approach to parse intra-word structures and word-level constituent structures jointly. For Hebrew, Tsarfaty and Goldberg (2008) investigated joint segmentation and parsing over characters using a graph-based method. Our work is similar in exploiting character-level syntax. We study the dependency grammar, another popular syntactic representation, and propose two novel transition systems for character-level dependency parsing.

Nivre (2008) gave a systematic description of the arc-standard and arc-eager algorithms, currently two popular transition-based parsing methods for word-level dependency parsing. We extend both algorithms to character-level joint word segmentation, POS-tagging and dependency parsing. To our knowledge, we are the first to apply the arc-eager system to joint models and achieve comparative performances to the arc-standard model.

## 6 Conclusions

We studied the character-level Chinese dependency parsing, by making novel extensions to two commonly-used transition-based dependency parsing algorithms for word-based dependency parsing. With both pseudo and annotated word structures, our character-level models obtained better accuracies than previous work on segmentation, POS-tagging and word-level dependency parsing. We further analyzed some important factors for intra-word dependencies, and found that two proposed character-level parsing models are complementary in parsing intra-word dependencies. We make the source code publicly available at `http://sourceforge.net/projects/zpar/, version 0.7`.

## Acknowledgments

We thank the anonymous reviewers for their constructive comments, and gratefully acknowledge the support of the National Basic Research Program (973 Program) of China via Grant 2014CB340503, the National Natural Science Foundation of China (NSFC) via Grant 61133012 and 61370164, the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301 and SRG ISTD 2012 038 from Singapore University of Technology and Design.

## References


Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the EMNLP-CONLL*, pages 1455–1465, Jeju Island, Korea, July.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd COLING*, number August, pages 89–97.

Xavier Carreras, Mihai Surdeanu, and Lluís Màrquez. 2006. Projective dependency parsing with perceptron. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 181–185, New York City, June.

Pi-Chuan Chang, Michel Galley, and Chris Manning. 2008. Optimizing chinese word segmentation for machine translation performance. In *ACL Workshop on Statistical Machine Translation*.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, , and Christopher D. Manning. 2009. Discriminative reordering with chinese grammatical relations features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *Proceedings of ACL*, pages 1052–1062, August.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, July.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 7th EMNLP*.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic models for action-based chinese dependency parsing. In *Proceedings of ECML/ECPPKDD*, volume 4701 of *Lecture Notes in Computer Science*, pages 559–566.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, pages 123–133.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *Proceedings of the 50th ACL*, pages 1045–1053, Jeju Island, Korea, July.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th ACL*, pages 1077–1086, Uppsala, Sweden, July.

Liang Huang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1222–1231. Association for Computational Linguistics.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the ACL*, pages 1–11.

Zhongguo Li and Guodong Zhou. 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*, pages 1445–1454, Jeju Island, Korea, July.

Zhenghua Li, Ting Liu, and Wanxiang Che. 2012. Exploiting multiple treebanks for parsing with quasi-synchronous grammars. In *Proceedings of the 50th ACL*, pages 675–684, Jeju Island, Korea, July.

Zhongguo Li. 2011. Parsing the internal structure of words: A new paradigm for chinese word segmentation. In *Proceedings of the 49th ACL*, pages 1405–1414, Portland, Oregon, USA, June.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, number June, pages 91–98, Morristown, NJ, USA.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of ACL*.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Reut Tsarfaty and Yoav Goldberg. 2008. Word-based or morpheme-based? annotation strategies for modern hebrew clitics. In *LREC*. European Language Resources Association.

Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving chinese word segmentation and pos tagging with semi-supervised methods using large auto-analyzed data. In *Proceedings of 5th IJCNLP*, pages 309–317, Chiang Mai, Thailand, November.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of EMNLP*, pages 562–571, Honolulu, Hawaii, October.

Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In *Proceedings of the EMNLP*, pages 843–852, Cambridge, MA, October.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th ACL*, pages 188–193, Portland, Oregon, USA, June.

Ruiqiang Zhang, Keiji Yasuda, and Eiichiro Sumita. 2008. Chinese word segmentation and statistical machine translation. *IEEE Transactions on Signal Processing*, 5(2).

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2013. Chinese parsing exploiting characters. In *Proceedings of the 51st ACL*, pages 125–134, Sofia, Bulgaria, August.

Hai Zhao. 2009. Character-level dependencies in chinese: Usefulness and learning. In *Proceedings of the EACL*, pages 879–887, Athens, Greece, March.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st ACL*, pages 434–443, Sofia, Bulgaria, August.