Character-Level Dependencies in Chinese: Usefulness and Learning

Hai Zhao
Department of Chinese, Translation and Linguistics
City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong, China
haizhao@cityu.edu.hk

Abstract

We investigate the possibility of exploiting character-based dependency for Chinese information processing. As Chinese text is made up of character sequences rather than word sequences, word in Chinese is not so natural a concept as in English, nor is word easy to be defined without argument for such a language. Therefore we propose a character-level dependency scheme to represent primary linguistic relationships within a Chinese sentence. The usefulness of character dependencies are verified through two specialized dependency parsing tasks. The first is to handle trivial character dependencies that are equally transformed from traditional word boundaries. The second furthermore considers the case that annotated internal character dependencies inside a word are involved. Both of these results from character-level dependency parsing are positive. This study provides an alternative way to formularize basic character- and word-level representation for Chinese.

1 Introduction

In many human languages, word can be naturally identified from writing. However, this is not the case for Chinese, for Chinese is born to be written in character[1] sequence rather than word sequence, namely, no natural separators such as blanks exist between words. As word does not appear in a natural way as most European languages[2], it brings the argument about how to determine the word-hood in Chinese. Linguists' views about what is a Chinese word diverge so greatly that multiple word segmentation standards have been proposed for computational linguistics tasks since the first Bakeoff (Bakeoff-1, or Bakeoff-2003)[3] (Sproat and Emerson, 2003).

Up to Bakeoff-4, *seven* word segmentation standards have been proposed. However, this does not effectively solve the open problem what a Chinese word should exactly be but raises another issue: what a segmentation standard should be selected for the successive application. As word often plays a basic role for the further language processing, if it cannot be determined in a unified way, then all successive tasks will be affected more or less.

Motivated by dependency representation for syntactic parsing since (Collins, 1999) that has been drawn more and more interests in recent years, we suggest that character-level dependencies can be adopted to alleviate this difficulty in Chinese processing. If we regard traditional word boundary as a linear representation for neighbored characters, then character-level dependencies can provide a way to represent non-linear relations between non-neighbored characters. To show that character dependencies can be useful, we develop a parsing scheme for the related learning task and demonstrate its effectiveness.

The rest of the paper is organized as follows. The next section shows the drawbacks of the current word boundary representation through some language examples. Section 3 describes a character-level dependency parsing scheme for traditional word segmentation task and reports its evaluation results. Section 4 verifies the usefulness of annotated character dependencies inside a word. Section 5 looks into a few issues concern-

[1]Character here stands for various tokens occurring in a naturally written Chinese text, including Chinese character(hanzi), punctuation, and foreign letters. However, Chinese characters often cover the most part.

[2]Even in European languages, a naive but necessary method to properly define word is to list them all by hand. Thank the first anonymous reviewer who points this fact.

[3]First International Chinese Word Segmentation Bakeoff, available at http://www.sighan.org/bakeoff2003.

ing the role of character dependencies. Section 6 concludes the paper.

2 To Segment or Not: That Is the Question

Though most words can be unambiguously defined in Chinese text, some word boundaries are not so easily determined. We show such three examples as the following.

The first example is from the MSRA segmented corpus of Bakeoff-2 (Bakeoff-2005) (Emerson, 2005):

> 一 / 张 / “ / 北京市京剧 O K 联谊会 / 会友 / 入场 / 证 / ”
>
> a / piece of / “ / Beijing City Beijing Opera OK Sodality / member / entrance / ticket / ”

As the guideline of MSRA standard requires any organization's full name as a word, many long words in this form are frequently encountered. Though this type of 'words' may be regarded as an effective unit to some extent, some smaller meaningful constituents can be still identified inside them. Some researchers argue that these should be seen as phrases rather than words. In fact, e.g., a machine translation system will have to segment this type of words into some smaller units for a proper translation.

The second example is from the PKU corpus of Bakeoff-2,

> 中国 / 驻 / 南非 / 大使馆
>
> China / in / South Africa / embassy
>
> (the Chinese embassy in South Africa)

This example demonstrates how researchers can also feel inconvenient if an organization name is segmented into pieces. Though the word '大使馆'(embassy) is right after '南非'(South Africa) in the above phrase, the embassy does not belong to South Africa but China, and it is only located in South Africa.

The third example is an abbreviation that makes use of the characteristics of Chinese characters.

> 星期 / 一 / 三 / 五
>
> Week / one / three / five
>
> (Monday, Wednesday and Friday)

This example shows that there will be in a dilemma to perform segmentation over these characters. If a segmentation position locates before '三'(three) or '五'(five), then this will make them meaningless or losing its original meaning at least because either of these two characters should logically follow the substring '星期' (week) to construct the expected word '星期三'(Wednesday) or '星期五' (Friday). Otherwise, to make all the above five characters as a word will have to ignore all these logical dependent relations among these characters and segment it later for a proper tackling as the above first example.

All these examples suggest that dependencies exist between discontinuous characters, and word boundary representation is insufficient to handle these cases. This motivates us to introduce character dependencies.

3 Character-Level Dependency Parsing

Character dependency is proposed as an alternative to word boundary. The idea itself is extremely simple, character dependencies inside sequence are annotated or formally defined in the similar way that syntactic dependencies over words are usually annotated.

We will initially develop a character-level dependency parsing scheme in this section. Especially, we show character dependencies, even those trivial ones that are equally transformed from pre-defined word boundaries, can be effectively captured in a parsing way.

3.1 Formularization

Using a character-level dependency representation, we first show how a word segmentation task can be transformed into a dependency parsing problem. Since word segmentation is traditionally formularized as an unlabeled character chunking task since (Xue, 2003), only unlabeled dependencies are concerned in the transformation. There are many ways to transform chunks in a sequence into dependency representation. However, for the sake of simplicity, only well-formed and projective output sequences are considered for our processing.

Borrowing the notation from (Nivre and Nilsson, 2005), an unlabeled dependency graph is formally defined as follows:

> An unlabeled dependency graph for a string of cliques (i.e., words and characters) $W =$

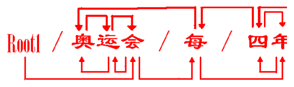


Figure 1: Two character dependency schemes

$w_1 ... w_n$ is an unlabeled directed graph $D = (W, A)$, where

(a) W is the set of ordered nodes, i.e. clique tokens in the input string, ordered by a linear precedence relation $<$,

(b) A is a set of unlabeled arcs (w_i, w_j), where $w_i, w_j \in W$,

If $(w_i, w_j) \in A$, w_i is called the head of w_j and w_j a dependent of w_i. Traditionally, the notation $w_i \rightarrow w_j$ means $(w_i, w_j) \in A$; $w_i \rightarrow^* w_j$ denotes the reflexive and transitive closure of the (unlabeled) arc relation. We assume that the designed dependency structure satisfies the following common constraints in existing literature (Nivre, 2006).

(1) D is weakly connected, that is, the corresponding undirected graph is connected. (CONNECTEDNESS)

(2) The graph D is acyclic, i.e., if $w_i \rightarrow w_j$ then not $w_j \rightarrow^* w_i$. (ACYCLICITY)

(3) There is at most one arc $(w_i, w_j) \in A, \forall w_j \in W$. (SINGLE-HEAD)

(4) An arc $w_i \rightarrow w_k$ is projective iff, for every word w_j occurring between w_i and w_k in the string ($w_i < w_j < w_k$ or $w_i > w_j > w_k$), $w_i \rightarrow^* w_j$. (PROJECTIVITY)

We say that D is well-formed iff it is acyclic and connected, and D is projective iff every arcs in A are projective. Note that the above four conditions entail that the graph D is a single-rooted tree. For an arc $w_i \rightarrow w_j$, if $w_i < w_j$, then it is called right-arc, otherwise left-arc.

Following the above four constraints and considering segmentation characteristics, we may have two character dependency representation schemes as shown in Figure 1 by using a series of trivial dependencies inside or outside a word. Note that we use arc direction to distinguish connected and segmented relation among characters. The scheme with the assistant root node before the sequence in Figure 1 is called Scheme B, and the other Scheme E.

3.2 Shift-reduce Parsing

According to (McDonald and Nivre, 2007), all data-driven models for dependency parsing that have been proposed in recent years can be described as either graph-based or transition-based. Since both dependency schemes that we construct for parsing are well-formed and projective, the latter is chosen as the parsing framework for the sake of efficiency. In detail, a shift-reduce method is adopted as in (Nivre, 2003).

The method is step-wise and a classifier is used to make a parsing decision step by step. In each step, the classifier checks a clique pair [4], namely, *TOP*, the top of a stack that consists of the processed cliques, and, *INPUT*, the first clique in the unprocessed sequence, to determine if a dependent relation should be established between them. Besides two arc-building actions, a shift action and a reduce action are also defined, as follows,

Left-arc: Add an arc from *INPUT* to *TOP* and pop the stack.

Right-arc: Add an arc from *TOP* to *INPUT* and push *INPUT* onto the stack.

Reduce: Pop *TOP* from the stack.

Shift: Push *INPUT* onto the stack.

In this work, we adopt a left-to-right arc-eager parsing model, that means that the parser scans the input sequence from left to right and right dependents are attached to their heads as soon as possible (Hall et al., 2007). In the implementation, as for Scheme E, all four actions are required to pass through an input sequence. However, only three actions, i.e., reduce action will never be used, are needed for Scheme B.

3.3 Learning Model and Features

While memory-based and margin-based learning approaches such as support vector machines are popularly applied to shift-reduce parsing, we apply maximum entropy model as the learning model for efficient training and producing some comparable results. Our implementation of maximum entropy adopts L-BFGS algorithm for parameter optimization as usual. No additional feature selection techniques are used.

With notations defined in Table 1, a feature set as shown in Table 2 is adopted. Here, we explain some terms in Tables 1 and 2.

[4]Here, clique means character or word in a sequence, which depends on what constructs the sequence.

Table 1: Feature Notations

Notation	Meaning
s	The character in the top of stack
s_{-1},...	The first character below the top of stack, etc.
i, i_{+1},...	The first (second) character in the unprocessed sequence, etc.
dprel	Dependent label
h	Head
lm	Leftmost child
rm	Rightmost child
rn	Right nearest child
char	Character form
.	's, i.e., '*s.dprel*' means dependent label of character in the top of stack
+	Feature combination, i.e., '*s.char*+*i.char*' means both *s.char* and *i.char* work as a feature function.

Table 2: Features for Parsing

Basic	Extension
x.*char*	itself, its previous two and next two characters, and all bigrams within the five-character window. (x is s or i.)
–	*s.h.char*
–	*s.dprel*
–	*s.rm.dprel*
–	s_{-1}.*cnseq*
–	s_{-1}.*cnseq*+s.*char*
–	s_{-1}.*curroot.lm.cnseq*
–	s_{-1}.*curroot.lm.cnseq*+*s.char*
–	s_{-1}.*curroot.lm.cnseq*+*i.char*
–	s_{-1}.*curroot.lm.cnseq*+s_{-1}.*cnseq*
–	s_{-1}.*curroot.lm.cnseq*+*s.char*+s_{-1}.*cnseq*
–	s_{-1}.*curroot.lm.cnseq*+*i.char*+s_{-1}.*cnseq*
–	$s.av_n$+$i.av_n$, $n = 1, 2, 3, 4, 5$
–	*preact*$_{-1}$
–	*preact*$_{-2}$
–	*preact*$_{-2}$+*preact*$_{-1}$

Since we only considered unlabeled dependency parsing, *dprel* means the arc direction from the head, either left or right. The feature *curroot* returns the root of a partial parsing tree that includes a specified node. The feature *cnseq* returns a substring started from a given character. It checks the direction of the arc that passes the given character and collects all characters with the same arc direction to yield an output substring until the arc direction is changed. Note that all combinational features concerned with this one can be regarded as word-level features.

The feature *av* is derived from unsupervised segmentation as in (Zhao and Kit, 2008a), and the *accessor variety* (AV) (Feng et al., 2004) is adopted as the unsupervised segmentation criterion. The AV value of a substring s is defined as

$$AV(s) = \min\{L_{av}(s), R_{av}(s)\},$$

where the left and right AV values $L_{av}(s)$ and $R_{av}(s)$ are defined, respectively, as the numbers of its distinct predecessor and successor characters. In this work, AV values for substrings are derived from unlabeled training and test corpora by substring counting. Multiple features are used to represent substrings of various lengths identified by the AV criterion. Formally put, the feature function for a n-character substring s with a score $AV(s)$ is defined as

$$av_n = t, \text{ if } 2^t \leq AV(s) < 2^{t+1}, \qquad (1)$$

where t is an integer to logarithmize the score and taken as the feature value. For an overlap character of several substrings, we only choose the one with the greatest AV score to activate the above feature function for that character.

The feature *preact*$_n$ returns the previous parsing action type, and the subscript n stands for the action order before the current action.

3.4 Decoding

Without Markovian feature like *preact*$_{-1}$, a shift-reduce parser can scan through an input sequence in linear time. That is, the decoding of a parsing method for word segmentation will be extremely fast. The time complexity of decoding will be $2L$ for Scheme E, and L for Scheme B, where L is the length of the input sequence.

However, it is somewhat complicated as Markovian features are involved. Following the work of (Duan et al., 2007), the decoding in this case is to search a parsing action sequence with the maximal probability.

$$S_{d_i} = \text{argmax} \prod_i p(d_i | d_{i-1} d_{i-2} ...),$$

where S_{d_i} is the object parsing action sequence, $p(d_i | d_{i-1} ...)$ is the conditional probability, and d_i is i-th parsing action. We use a beam search algorithm as in (Ratnaparkhi, 1996) to find the object parsing action sequence. The time complexity of this beam search algorithm will be $4BL$ for Scheme E and $3BL$ for Scheme B, where B is the beam width.

3.5 Related Methods

Among character-based learning techniques for word segmentation, we may identify two main types, classification (GOH et al., 2004) and tagging (Low et al., 2005). Both character classification and tagging need to define the position of character inside a word. Traditionally, the four tags, b, m, e, and s stand, respectively, for the *b*eginning, *m*idle, *e*nd of a word, and a *s*ingle-character as word since (Xue, 2003). The following n-gram features from (Xue, 2003; Low et al., 2005) are used as basic features,

(a) $C_n(n = -2, -1, 0, 1, 2)$,

(b) $C_nC_{n+1}(n = -2, -1, 0, 1)$,

(c) $C_{-1}C_1$,

where C stands for a character and the subscripts for the relative order to the current character C_0. In addition, the feature *av* that is defined in equation (1) is also taken as an option. av_n (n=1,...,5) is applied as feature for the current character.

While word segmentation is conducted as a classification task, each individual character will be simply assigned a tag with the maximal probability given by the classifier. In this case, we restore word boundary only according to two tags b and s. However, the output tag sequence given by character classification may include illegal tag transition (e.g., m is after e.). In (Low et al., 2005), a dynamic programming algorithm is adopted to find a tag sequence with the maximal joint probability from all legal tag sequences. If such a dynamic programming decoding is adopted, then this method for word segmentation is regarded as character tagging [5].

The time complexity of character-based classification method for decoding is L, which is the best result in decoding velocity. As dynamic programming is applied, the time complexity will be $16L$ with four tags.

Recently, conditional random fields (CRFs) becomes popular for word segmentation since it provides slightly better performance than maximum entropy method does (Peng et al., 2004). However, CRFs is a structural learning tool rather than a simple classification framework. As shift-reduce parsing is a typical step-wise method that checks each character one by one, it is reasonable to compare it to a classification method over characters.

[5]Someone may argue that maximum entropy Markov model (MEMM) is truly a tagging tool. Yes, this method was initialized by (Xue, 2003). However, our empirical results show that MEMM never outperforms maximum entropy plus dynamic programming decoding as (Low et al., 2005) in Chinese word segmentation. We also know that the latter reports the best results in Bakeoff-2. This is why MEMM method is excluded from our comparison.

3.6 Evaluation Results

Table 3: Corpus size of Bakeoff-2 in number of words

	AS	CityU	MSRA	PKU
Training(M)	5.45	1.46	2.37	1.1
Test(K)	122	41	107	104

The experiments in this section are performed in all four corpora from Bakeoff-2. Corpus size information is in Table 3.

Traditionally, word segmentation performance is measured by F-score ($F = 2RP/(R + P)$), where the recall (R) and precision (P) are the proportions of the correctly segmented words to all words in, respectively, the gold-standard segmentation and a segmenter's output. To compute the word F-score, all parsing results will be restored to word boundaries according to the direction of output arcs.

Table 4: The results of parsing and classification/tagging approaches using different feature combinations

S.[a]	Feature	AS	CityU	MSRA	PKU
B	Basic[b]	.935	.922	.950	.917
	+AV[c]	.941	.933	.956	.927
	+Prev[d]	.937	.923	.951	.918
	+AV+Prev	.942	.935	.958	.929
E	Basic	.940	.932	.957	.926
	+AV	.948	.947	.964	.942
	+Prev	.944	.940	.962	.931
	+AV+Prev	.949	.951	.967	.943
C[f]	n-gram/c[e]	.933	.923	.948	.923
	+AV/c	.942	.936	.957	.933
	n-gram/d[g]	.945	.938	.956	.936
	+AV/d	.950	.949	.966	.945

[a]Scheme
[b]Features in top two blocks of Table 2.
[c]Five *av* features are added on the above basic features.
[d]Three Markovian features in Table 2 are added on the above basic features.
[e]/c: Classification
[f]Character classification or tagging using maximum entropy
[g]/d: Only search in legal tag sequences.

Our comparison with existing work will be conducted in closed test of Bakeoff. The rule for the closed test is that no additional information beyond training corpus is allowed, while open test of Bakeoff is without such restrict.

The results with different dependency schemes are in Table 4. As the feature $preact$ is involved, a beam search algorithm with width 5 is used to decode, otherwise, a simple shift-reduce decoding is used. We see that the performance given by Scheme E is much better than that by Scheme B. The results of character-based classification and tagging methods are at the bottom of Table 4[6]. It is observed that the parsing method outperforms classification and tagging method without Markovian features or decoding throughout the whole sequence. As full features are used, the former and the latter provide the similar performance.

Due to using a global model like CRFs, our previous work in (Zhao et al., 2006; Zhao and Kit, 2008c) reported the best results over the evaluated corpora of Bakeoff-2 until now[7]. Though those results are slightly better than the results here, we still see that the results of character-level dependency parsing approach (Scheme E) are comparable to those state-of-the-art ones on each evaluated corpus.

4 Character Dependencies inside a Word

We further consider exploiting annotated character dependencies inside a word (internal dependencies). A parsing task for these internal dependencies incorporated with trivial external dependencies [8] that are transformed from common word boundaries are correspondingly proposed using the same parsing way as the previous section.

4.1 Annotation of Internal Dependencies

In Subsection 3.1, we assign trivial character dependencies inside a word for the parsing task of word segmentation, i.e., each character as the head of its predecessor or successor. These trivial formally defined dependencies may be against the syntactic or semantic senses of those characters, as we have discussed in Section 2. Now we will consider human annotated character dependencies inside a word.

As such an corpus with annotated internal dependencies has not been available until now, we launched an annotation job based on UPUC segmented corpus of Bakeoff-3(Bakeoff-2006)(Levow, 2006). The training corpus is with 880K characters and test corpus 270K. However, the essential of the annotation job is actually conducted in a lexicon.

After a lexicon is extracted from CTB segmented corpus, we use a top-down strategy to annotate internal dependencies inside these words from the lexicon. A long word is first split into some smaller constituents, and dependencies among these constituents are determined, character dependencies inside each constituents are then annotated. Some simple rules are adopted to determine dependency relation, e.g., modifiers are kept marking as dependants and the only rest constituent will be marked as head at last. Some words are hard to determine internal dependency relation, such as foreign names, e.g., '葡萄牙'(Portugal) and '马拉多纳'(Maradona), and uninterrupted words (连绵词), e.g., '蚂蚁'(ant) and '苜蓿'(clover). In this case, we simply adopt a series of linear dependencies with the last character as head to mark these words.

In the previous section, we have shown that Scheme E is a better dependency representation for encoding word boundaries. Thus annotated internal dependencies are used to replace those trivial internal dependencies in Scheme E to obtain the corpus that we require. Note that now we cannot distinguish internal and external dependencies only according to the arc direction any more, as both left- and right-arc can appear for internal character dependency representation. Thus two labeled left arcs, external and internal, are used for the annotation disambiguation. As internal dependencies are introduced, we find that some words (about 10%) are constructed by two or more parallel constituent parts according to our annotations, this not only lets two labeled arcs insufficiently distinguish internal- and external dependencies, but also makes parsing extremely difficult, namely, a great amount of non-projective dependencies will appear if we directly introduce these internal dependencies. Again, we adopt a series of linear dependencies with the last character as head to represent internal dependencies for these words by ignoring their parallel constituents. To handle the remained non-projectivities, a strengthened pseudo-projectivization technique as in (Zhao and Kit,

[6]Only the results of open track are reported in (Low et al., 2005), while we give a comparison following closed track rules, so, our results here are not comparable to those of (Low et al., 2005).

[7]As n-gram features are used, F-scores in (Zhao et al., 2006) are, AS:0.953, CityU:0.948, MSRA:0.974,PKU:0.952.

[8]We correspondingly call dependencies that mark word boundary external dependencies that correspond to internal dependencies.

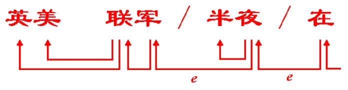


Figure 2: Annotated internal dependencies (Arc label e notes trivial external dependencies.)

Table 5: Features for internal dependency parsing

Basic	Extension
s.char	itself, its next two characters, and all bigrams within the three-character window.
i.char	its previous one and next three characters, and all bigrams within the four-character window.
–	*s.char+i.char*
–	*s.h.char*
–	*s.rm.dprel*
–	*s.curtree*
–	*s.curtree+s.char*
–	s_{-1}*.curtree+s.char*
–	*s.curroot.lm.curtree*
–	s_{-1}*.curroot.lm.curtree*
–	*s.curroot.lm.curtree+s.char*
–	s_{-1}*.curroot.lm.curtree+s.char*
–	*s.curtree+s.curroot.lm.curtree*
–	s_{-1}*.curtree+*s_{-1}*.curroot.lm.curtree*
–	*s.curtree+s.curroot.lm.curtree+s.char*
–	s_{-1}*.curtree+*s_{-1}*.curroot.lm.curtree+s.char*
–	s_{-1}*.curtree+*s_{-1}*.curroot.lm.curtree+i.char*
–	$x.av_n$, $n = 1, ..., 5$ (x is s or i.)
–	$s.av_n$+$i.av_n$, $n = 1, ..., 5$
–	$preact_{-1}$
–	$preact_{-2}$
–	$preact_{-2}$+$preact_{-1}$

2008b) is used during parsing. An annotated example is illustrated in Figure 2.

4.2 Learning of Internal Dependencies

To demonstrate internal character dependencies are helpful for further processing. A series of similar word segmentation experiments as in Subsection 3.6 are performed. Note that this task is slightly different from the previous one, as it is a five-class parsing action classification task as left arc has two labels to differ internal and external dependencies. Thus a different feature set has to be used. However, all input sequences are still projective.

Features listed in Table 5 are adopted for the parsing task that annotated character dependencies exist inside words. The feature *curtree* in Table 5 is similar to *cnseq* of Table 2. It first greedily searches all connected character started from the given one until an arc with external label is found over some character. Then it collects all characters that has been reached to yield an output substring as feature value.

A comparison of classification/tagging and parsing methods is given in Table 6. To evaluate the results with word F-score, all external dependencies in outputs are restored as word boundaries. There are three models are evaluated in Table 6. It is shown that there is a significant performance enhancement as annotated internal character dependency is introduced. This positive result shows that annotated internal character dependencies are meaningful.

Table 6: Comparison of different methods

Approach [a]	basic	+AV	+Prev[b]	+AV+Prev
Class/Tag[c]	.918	.935	.928	.941
Parsing/wo[d]	.921	.937	.924	.942
Parsing/w [e]	.925	.940	.929	.945

[a]The highest F-score in Bakeoff-3 is 0.933.

[b]As for the tagging method, this means dynamic programming decoding; As for the parsing method, this means three Markovian features.

[c]Character-based classification or tagging method

[d]Using trivial internal dependencies in Scheme E.

[e]Using annotated internal character dependencies.

5 Is Word Still Necessary?

Note that this work is not about joint learning of word boundaries and syntactic dependencies such as (Luo, 2003), where a character-based tagging method is used for syntactic constituent parsing from unsegmented Chinese text. Instead, this work is to explore an alternative way to represent "word-hood" in Chinese, which is based on character-level dependencies instead of traditional word boundaries definition.

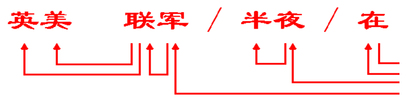
Though considering dependencies among words is not novel (Gao and Suzuki, 2004), we recognize that this study is the first work concerned with character dependency. This study originally intends to lead us to consider an alternative way that can play the similar role as word boundary annotations.

In Chinese, not word but character is the actual minimal unit for either writing or speaking. Word-hood has been carefully defined by many means, and this effort results in multi-standard segmented corpora provided by a series of Bakeoff evaluations. However, from the view of linguistics, Bakeoff does not solve the problem but technically skirts round it. As one asks what a Chinese word is, Bakeoff just answers that we have many definitions and each one is fine. Instead, motivated from the results of the previous two sections, we suggest that character dependency representation could present a natural and unified way to alleviate the drawbacks of word boundary representation that is only able to represent the relation of neighbored characters.

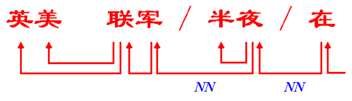
Table 7: What we have done for character dependency

Internal	External	Our work
trivial	trivial	Section 3
annotated	trivial	Section 4
annotated		?

If we regard that our current work is stepping into more and more annotated character dependencies as shown in Table 7, then it is natural to extend annotated internal character dependencies to the whole sequence without those unnatural word boundary constraints. In this sense, internal and external character dependency will not need be differed any more. A full character-level dependency tree is illustrated as shown in Figure 3(a)[9] With the help of such a tree, we may define word or even phrase according to what part of subtree is picked up. Word-hood, if we still need this concept, can be freely determined later as further processing purpose requires.



(a)



(b)

Figure 3: Extended character dependencies

Basically we only consider unlabeled dependencies in this work, and dependant labels can be emptied to do something else, e.g., Figure 3(b) shows how to extend internal character dependencies of Figure 2 to accommodate part-of-speech tags. This extension can also be transplanted to a full character dependency tree of Figure 3(a), then this may leads to a character-based labeled syntactic dependency tree. In brief, we see that character dependencies provide a more general and natural way to reflect character relations within a sequence than word boundary annotations do.

[9]We may easily build such a corpus by embedding annotated internal dependencies into a word-level dependency tree bank. As UPUC corpus of Bakeoff-3 just follows the word segmentation convention of Chinese tree bank, we have built such a full character-level dependency tree corpus.

6 Conclusion and Future Work

In this study, we initially investigate the possibility of exploiting character dependencies for Chinese. To show that character-level dependency can be a good alternative to word boundary representation for Chinese, we carry out a series of parsing experiments. The techniques are developed step by step. Firstly, we show that word segmentation task can be effectively re-formularized character-level dependency parsing. The results of a character-level dependency parser can be comparable with traditional methods. Secondly, we consider annotated character dependencies inside a word. We show that a parser can still effectively capture both these annotated internal character dependencies and trivial external dependencies that are transformed from word boundaries. The experimental results show that annotated internal dependencies even bring performance enhancement and indirectly verify the usefulness of them. Finally, we suggest that a full annotated character dependency tree can be constructed over all possible character pairs within a given sequence, though its usefulness needs to be explored in the future.

Acknowledgements

This work is beneficial from many sources, including three anonymous reviewers. Especially, the authors are grateful to two colleagues, one reviewer from EMNLP-2008 who gave some very insightful comments to help us extend this work, and Mr. SONG Yan who annotated internal dependencies of top frequent 22K words extracted from UPUC segmentation corpus. Of course, it is the duty of the first author if there still exists anything wrong in this work.

References

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic parsing action models for multi-lingual dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 940–946, Prague, Czech, June 28-30.

Thomas Emerson. 2005. The second international Chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 123–133, Jeju Island, Korea, October 14-15.

Haodi Feng, Kang Chen, Xiaotie Deng, and Weimin Zheng. 2004. Accessor variety criteria for Chinese word extraction. *Computational Linguistics*, 30(1):75–93.

Jianfeng Gao and Hisami Suzuki. 2004. Capturing long distance dependency in language modeling: An empirical study. In K.-Y. Su, J. Tsujii, J. H. Lee, and O. Y. Kwong, editors, *Natural Language Processing - IJCNLP 2004*, volume 3248 of *Lecture Notes in Computer Science*, pages 396–405, Sanya, Hainan Island, China, March 22-24.

Chooi-Ling GOH, Masayuki Asahara, and Yuji Matsumoto. 2004. Chinese word segmentation by classification of characters. In *ACL SIGHAN Workshop 2004*, pages 57–64, Barcelona, Spain, July. Association for Computational Linguistics.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryiğit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? a study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 933–939, Prague, Czech, June.

Gina-Anne Levow. 2006. The third international Chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117, Sydney, Australia, July 22-23.

Jin Kiat Low, Hwee Tou Ng, and Wenyuan Guo. 2005. A maximum entropy approach to Chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 161–164, Jeju Island, Korea, October 14-15.

Xiaoqiang Luo. 2003. A maximum entropy chinese character-based parser. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP 2003)*, pages 192 – 199, Sapporo, Japan, July 11-12.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 122–131, Prague, Czech, June 28-30.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL-2005)*, pages 99–106, Ann Arbor, Michigan, USA, June 25-30.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 149–160, Nancy, France, April 23-25.

Joakim Nivre. 2006. Constraints on non-projective dependency parsing. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, pages 73–80, Trento, Italy, April 3-7.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *COLING 2004*, pages 562–568, Geneva, Switzerland, August 23-27.

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Method in Natural Language Processing Conference*, pages 133–142, University of Pennsylvania.

Richard Sproat and Thomas Emerson. 2003. The first international Chinese word segmentation bakeoff. In *The Second SIGHAN Workshop on Chinese Language Processing*, pages 133–143, Sapporo, Japan.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Hai Zhao and Chunyu Kit. 2008a. Exploiting unlabeled text with different unsupervised segmentation criteria for chinese word segmentation. In *Research in Computing Science*, volume 33, pages 93–104.

Hai Zhao and Chunyu Kit. 2008b. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *Twelfth Conference on Computational Natural Language Learning (CoNLL-2008)*, pages 203–207, Manchester, UK, August 16-17.

Hai Zhao and Chunyu Kit. 2008c. Unsupervised segmentation helps supervised learning of character tagging for word segmentation and named entity recognition. In *The Sixth SIGHAN Workshop on Chinese Language Processing*, pages 106–111, Hyderabad, India, January 11-12.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2006. Effective tag set selection in Chinese word segmentation via conditional random field modeling. In *Proceedings of the 20th Asian Pacific Conference on Language, Information and Computation*, pages 87–94, Wuhan, China, November 1-3.