Chinese Word Segmentation and Unknown Word Extraction by Mining Maximized Substring

Mo Shen[†], Daisuke Kawahara[†] and Sadao Kurohashi[†]

Chinese word segmentation is an initial and important step in Chinese language processing. Recent advances in machine learning techniques have boosted the performance of Chinese word segmentation systems, yet the identification of out-of-vocabulary words is still a major problem in this field of study. Recent research has attempted to address this problem by exploiting characteristics of frequent substrings in unlabeled data. We propose a simple yet effective approach for extracting a specific type of frequent substrings, called maximized substrings, which provide good estimations of unknown word boundaries. In the task of Chinese word segmentation, we use these substrings which are extracted from large scale unlabeled data to improve the segmentation accuracy. The effectiveness of this approach is demonstrated through experiments using various data sets from different domains. In the task of unknown word extraction, we apply post-processing techniques that effectively reduce the noise in the extracted substrings. We demonstrate the effectiveness and efficiency of our approach by comparing the results with a widely applied Chinese word recognition method in a previous study.

Key Words: *Chinese Word Segmentation, Unknown Word Extraction, Word Recognition, Morphology*

1 Introduction

Chinese sentences are written without explicit word boundaries, which makes Chinese word segmentation (CWS) an initial and important step in Chinese language processing. Recent advances in machine learning techniques have boosted the performance of CWS systems. However, previous studies have shown that with a comprehensive lexicon, even a simple maximum matching segmentation algorithm can yield an F-score as high as 0.99 (Sproat and Emerson 2003). This suggests that the lack of knowledge of vocabulary presents the biggest challenge in Chinese word segmentation. It is impossible, however, to collect a complete list of Chinese words. The Chinese language is continually and rapidly evolving, particularly with the rapid growth of the internet. Therefore, it is necessary to develop techniques that automatically develop vocabulary lists from large-scale web texts instead of relying on supervised CWS systems.

Recent research has attempted to exploit characteristics of frequent substrings in unlabeled

[†] Graduate School of Informatics, Kyoto University

data. Statistical criteria that measure the likelihood of a substring being a word have been proposed in previous studies of unsupervised segmentation, such as “accessor variety” (Feng, Chen, Deng, and Zheng 2004), a criterion measuring the likelihood of a substring being a word by counting distinct surrounding characters. In (Jin and Tanaka-Ishii 2006) the researchers proposed “branching entropy”, a similar criterion based on the assumption that the uncertainty of surrounding characters of a substring peaks at the word boundaries. In (Ye, Wu, Li, Chow, Hui, and Yiu 2013), the likelihood of an character n-gram being a meaningful Chinese word is measured by “overlap variety”, a criteria which considers the goodness of the candidate together with the goodness of the strings overlapping the candidate. The authors have also adopted a word refinement module to filter out known words during the post-processing.

The authors of (Zhao and Kit 2007) incorporated accessor variety and another type of criteria, called “co-occurrence sub-sequence”, with a supervised segmentation system and conducted comprehensive experiments to investigate their impacts. There are several restrictions in using the co-occurrence sub-sequence criteria: it requires post-processing to remove overlapping instances; sub-sequences are retrievable only from different sentences; and the retrieval is performed only on training and testing data. In (Sun and Xu 2011), the authors proposed a semi-supervised segmentation system enhanced with multiple statistical criteria; Large-scale unlabeled data were used in their experiments. Jiang, Liu, Sung, and Hsu (2010) used term contributed boundary information extracted from unlabeled text as features in semi-supervised learning to improve character-based Chinese word segmentation.

Li and Sun presented a model to learn the features of word delimiters from punctuation marks in (Li and Sun 2009). Wang et al. proposed a semi-supervised word segmentation method that took advantages from auto-analyzed data (Wang, Kazama, Tsuruoka, Tsuruoka, Chen, Zhang, and Torisawa 2011).

Nakagawa showed the advantage of the hybrid model combining both character-level information and word-level information in Chinese and Japanese word segmentation (Nakagawa 2004). In (Nakagawa and Uchimoto 2007) and (Kruengkrai, Uchimoto, Kazama, Wang, Torisawa, and Isahara 2009a; Kruengkrai, Uchimoto, Kazama, Yiu, Torisawa, and Isahara 2009b) the researchers presented word-character hybrid models for joint word segmentation and POS tagging, and achieved the state-of-the-art accuracy on Chinese and Japanese datasets.

Another related research area is unknown Chinese word extraction. Previous studies have explored statistical criteria for measuring the likelihood of a substring being a word (Feng et al. 2004), methods that combine morphological rules with statistical information (Chen and Ma 2002; Ma and Chen 2003), character-based role tagging (Zhang, Liu, Zhang, and Cheng 2002), and

generating words from an initial segmentation (GuoDong 2005; Ling, Asahara, and Matsumoto 2003; Goh, Asahara, and Matsumoto 2005). Recent advances have also verified the ability to process large-scale web texts (Zhang, Wang, Xue, and Sun 2011).

In this paper, we propose a novel method that extracts substrings as reliable word boundary indicators. This method takes linear time in the average case, which is usually a requirement for processing raw text of very large size, such as web pages on the internet.

To illustrate the idea, we first consider the following example taken from a scientific article:

“使一致认定界限数的期望值近似于一致正确界限数的期望值，求得一致认定界限的期望值/认定界限数的值。”

Without any knowledge of the Chinese language one may still notice that some substrings like “一致” and “的期望值”, occur multiple times in the sentence and are likely to be valid words or chains of words. Consider a particular type of frequent substrings (the formal definition can be found in Section 2.1), in the sense that all its occurrences have surrounding characters which are distinct from each other from both left and right-hand sides (Table 1). We can observe that the boundaries of such substrings can be used as cues for word segmentation. We then segment the sentence by simply matching the substrings in Table 1 in the sentence; each time there is an occurrence of a substring, we split the sentence at the boundaries of the detected occurrence. The resulted segmentation is as follows:

“使 | 一致 | 认定 | 界限 | 数 | 的 | 期望 | 值 | 近似于 | 一致 | 正确 | 界限数 | 的期望 | 值 |，求得 | 一致 | 认定界限 | 的期望 | 值 | / | 认定界限数的 | 值 |。”

Compared with the gold-standard segmentation, this segmentation has a precision of 100% and a recall of 73% with regard to boundary estimation. This is high when we consider that the method does not use a trained segmenter or annotated data. While we have obtained this result on a selected instance, it still suggests that unlabeled data has the potential to enhance

Table 1 A particular type of substrings with multiple occurrences in the Chinese sentence: “使一致认定界限数的期望值近似于一致正确界限数的期望值，求得一致认定界限的期望值/认定界限数的值。”

Substring	Frequency
一致	3
界限数的期望值	2
一致认定界限	2
的期望值	3
认定界限数的	2
值	4

the performance of supervised segmentation systems by tracking consistency among substrings.

There are some key properties of the substrings listed in Table 1: The substrings are frequent, in the sense that they appear multiple times in a relatively small window of text; Also, any substring in the list does not have a superstring that has the same frequency. These properties are studied in previous research. For example, (Nagao and Mori 1994) described these kind of substrings as “longest compound word” and successfully extracted words and phrases based on their properties from large scale Japanese text data. (Lin and Yu 2001) described these substrings as “reduced N-gram” and proposed a two-stage algorithm for extracting frequent substrings from a corpus and removing internal substrings of the same frequency; the second step is called statistical substring reduction (SSR), which can be done in linear time by recent advance in its studies (Lü, Zhang, and Hu 2004).

We exploit these substrings in two ways. First, similar to (Jiang et al. 2010), we incorporate the substrings as features in a semi-supervised CWS system. In this case the unlabeled data can be either test data only, or a large-scale external corpus. Second, we also refine the extracted substrings into a list of unknown Chinese words as a language resource, which can be easily used by other language processing applications. We will formally define this particular type of substring, referred to as a “maximized substring”, in a later section.

The remainder of this paper is organized as follows. Section 2 defines maximized substrings, and proposes an algorithm for extracting these substrings from unlabeled data. Section 3 describes our baseline segmentation system and introduces the maximized substring features. Section 4 describes our post-processing techniques for reducing the redundancy and noise in unknown word extraction. Section 5 presents the experimental results. Section 6 discusses the relation and difference between our proposed method and previous work. The last section summarizes this paper.

We summarize the main contributions of this work as follows.

- The frequent substring mining algorithm that we propose is the first algorithm that is able to extract maximized substrings in one scan. These substrings can be reliably used to estimate word boundaries. The algorithm is efficient for processing very large scale text, and its underlying data structure makes it suitable for incorporating with caching methods to further increase the quality of the output.
- We incorporate the extracted list of maximized substrings with a baseline Chinese word segmentation system and demonstrate its performance in relation to previous studies. We show that our method outperforms other state-of-the-art systems.
- We apply post-processing strategies on the extracted list of maximized substrings to ex-

tract unknown words. Our evaluation shows that the accuracy of our method outperforms a previous study (Feng et al. 2004).

2 Maximized Substring Extraction

2.1 Maximized Substring: Definition

Frequent substrings in unlabeled data can be used as clues for identifying word boundaries, as we have illustrated in Section 1. Nevertheless, some substrings, although frequent, are not useful to the system. In the example in Section 1, the substring “致認定界” occurs the same amount of times as the substring “一致認定界限”. However, only the latter is a valid identifier for word delimiters. The former substring is “internal”: it is overlapped by other substrings and can be further extended by its surrounding characters, without reducing its frequency. What we are interested in are non-overlapping substrings, meaning that in each side of these substrings, the surrounding characters are all different from each other. We use the term maximized substring to describe these substrings.

Formally, we define a maximized substring as follows:

Definition 2.1 (Maximised substring). Given a document D that is a collection of sentences, denote a length n substring which starts with character c_t by $s_t = [c_t c_{t+1} \dots c_{t+n-1}]$, where c_t is the character at t -th position in D . s_t is called a maximized substring if:

- (1) It has a set of distinct occurrences, M , with at least two elements¹:

$$M = \{s_{t_1}, s_{t_2}, \dots, s_{t_k}\}, k > 1, t_1, t_2, \dots, t_k \text{ are distinct from each other s.t. } s_{t_1} = s_{t_2} = \dots = s_{t_k}; \text{ and}$$

- (2) $c_{t_i-1} \neq c_{t_j-1}$ and $c_{t_i+n} \neq c_{t_j+n}, \forall i, j = 1, 2, \dots, k, i \neq j$.

The substrings listed in Table 1 are therefore maximized substrings, given that D is the example sentence. Note that these are not all maximized substrings extractable from the example sentence, but are the result of the extraction algorithm that we will describe in the next section.

2.2 Maximized Substring Extraction: Algorithm and Data Structure

The problem of mining frequent substrings in a document has been extensively researched. Data structures and substring mining strategies, such as suffix trees (Nelson 1996), suffix arrays (Yamamoto and Church 2001; Fischer, Heun, and Kramer 2005), and Apriori property (Agrawal and Srikant 1994), have been used in existing algorithms. The Apriori property states that a

¹ It should be noted that, in order to extract a substring, the size of M is not necessarily identical to its total count in the document.

string of length $k + 1$ is frequent only if its substring of length k is frequent, which can reduce the size of enumerable substring candidates. In this section, we propose a novel algorithm for fast maximized substring mining with a simple hash table implementation.

The data structure we used in this algorithm is illustrated in Figure 1. It supports fast prefix searching for storing and extracting maximized substrings, with each entry associated to a list of occurrences that refer to the original positions in the document. Fast prefix matching is a particular advantage of a trie, which is a type of prefix tree; however we choose a simple hash table structure in our implementation, which is mainly due to the frequent deletion operations required in the so-called “short-term store” technique, which we will describe in next section. A hash table data structure is efficient for the deletion operation in the sense that the average case complexity is $O(1)$, and it also has less computational overhead compared to a trie, since less data structure manipulation is needed after a deletion is made.

The hash table stores the detected maximized substrings, where each maximized substring s is a key that is associated with a list of its occurrences. The occurrence list also stores the original positions of all the occurrences of a maximized substring in the document with the surrounding context being visible, so that new (longer) maximized substrings can be found by extension.

We sketch the process of maximized substring extraction in Algorithm 1. The data structure described above is corresponding to H in the algorithm. From the beginning of the document D , we scan each position and register maximized substrings into the data structure H . If an incoming

Fig. 1 Data structure for maximized substring mining. Hash is a hash table which stores the detected maximized substrings, where each maximized substring s is a key that is associated with a list of its occurrences $occurList(s)$. $occurList(s)$ also stores the original positions of all the occurrences of s in the document. (a) shows a certain state of the data structure, and (b) the state after a maximized substring “ABCCFA” is inserted with the context being “ABCCFAT...” in the document.

Algorithm 1 Maximized Substring Extraction

```

1: function EXTRACTMAXSUB( $D$ )
2:    $i \leftarrow 0$ 
3:    $H \leftarrow \emptyset$   $\triangleright H$  is a hash table of maximized substrings
 $\triangleright$  each key in  $H$  is associated with a list of its occurrences
4:   repeat
5:      $s \leftarrow$  longest substring from position  $i$  which can be found in  $H$ 
6:     if  $|s| = 0$  then  $\triangleright$  empty string
7:        $s_{new} \leftarrow [c_i]$   $\triangleright$  single-character string
8:        $occurList_{s_{new}} \leftarrow \{i\}$   $\triangleright$  starting position of new occurrence
9:        $H.Add(\langle s_{new}, occurList_{s_{new}} \rangle)$   $\triangleright$  associate string  $s_{new}$  with its occurrence list and add to  $H$ 
10:       $i \leftarrow i + 1$ 
11:     else
12:        $(H, i) \leftarrow Maximize(H, i, s)$ 
13:   until  $i$  reaches the end of document  $D$ 
14:   return  $H$ 
15:
16: function MAXIMIZE( $H, i, s$ )
17:    $t \leftarrow 0$ 
18:   for each  $u$  in  $H.Find(s)$  do  $\triangleright H.Find(s)$  returns the  $occurList$  associated with  $s$ 
19:     if  $c_{u+|s|} = c_{i+|s|}$  then
20:       while  $c_{u+|s|+t} = c_{i+|s|+t}$  do
21:          $t \leftarrow t + 1$ 
22:        $s_{new} \leftarrow [c_i c_{i+1} \dots c_{i+|s|+t-1}]$ 
23:        $occurList_{s_{new}} \leftarrow \{i, u\}$ 
24:        $H.Add(\langle s_{new}, occurList_{s_{new}} \rangle)$ 
25:        $i \leftarrow i + |s| + t$ 
26:       return  $(H, i)$ 
27:    $s.occurList.Add(i)$ 
28:    $i \leftarrow i + |s|$ 
29:   return  $(H, i)$ 

```

substring already exists in H , we look up its occurrence list to check if its succeeding characters can extend the substring. As the current occurrence list is a set of maximized substrings, there will be only two possible outcomes. Either exactly one element in the occurrence list is found to have a longer common prefix with the incoming substring, in which case we create a new occurrence list consisting of the two longer substrings. Alternatively, the prefix remains the same and we add the incoming substring to the occurrence list. In both cases, the scanning skips the rest of the positions inside the newly added occurrence (line 25 and 28). This ensures that the algorithm will not extract any “internal” substrings.

The average case complexity of Algorithm 1 is $O(n)$, assuming the number of occurrences of each maximized substring is less than a constant C , such that $C \ll n$. The worst case complexity is $O(n^2)$, which happens when the document consists of a sequence of extractable occurrences of the same maximized substring s , e.g., $D = [x_1 s x_2 s \dots x_N s]$, where $x_1, x_2, \dots, x_N$ are distinct from each other. This is because of the block starts from line 18 in Algorithm 1 which requires a comparison operation for each occurrence, and in this example the number of occurrences of s is $n/2$. The “while loop” starts from line 20 however does not increase the complexity, since the searching always skip those characters that have already been checked in this loop and starts at the next unchecked position (line 25). However, the worst case can rarely happen when processing real world text, since the largest number of occurrences of any maximized substring is much smaller than $n/2$. The cost for computing the frequencies for all substrings is at least $O(n^2)$ for a naïve algorithm. In Yamamoto and Church (2001) an efficient method is proposed for computing the term frequency and document frequency based on suffix array, which cost $O(n \log n)$ time to construct. The efficiency of suffix array construction has been further improved to $O(n)$ (Kärkkäinen and Sanders 2003; Kärkkäinen, Sanders, and Burkhardt 2006; Ko and Aluru 2003; Kim, Sim, Park, and Park 2003; Nong, Zhang, and Chan 2011), which facilitated later studies which exploit substring frequencies in information retrieval (Wu, Ng Hong, and Ruibin 2003; Grossi and Vitter 2005). In our algorithm, though it is possible to use a suffix array-based implementation for achieving better space efficiency, we choose to use a hash table structure, which can be easily extended with a technique to reduce the noises in the extracted set of frequent substrings. We will describe this technique in detail in Section 2.3.

It can be demonstrated that a substring with characters extracted by this algorithm with more than one occurrence is a maximized substring:

Proof.

- (1) s has more than one occurrence, which form a set M :

$$M = \{s_{t_1}, s_{t_2}, \dots, s_{t_k}\}, k > 1, t_1, t_2, \dots, t_k \text{ are distinct from each other s.t. } s_{t_1} = s_{t_2} = \dots = s_{t_k}.$$

- (2) Assume there exist two elements, s_a and s_b ($a < b$), s.t. $c_{a+n} = c_{b+n}$ or $c_{a-1} = c_{b-1}$.

- (a) If $c_{a+n} = c_{b+n}$, in line 20 of Algorithm 1 the character c_{b+n} will be appended to s_b , s.t. a string $s_{b'} = [c_b c_{b+1} \dots c_{b+n+t}]$ ($t \geq 0$) will be extracted and added into the hash table, in which case s_b will not be extracted since it is a substring of $s_{b'}$ starting from the same position in the document.

- (b) If $c_{a-1} = c_{b-1}$, let u be the largest positive integer s.t. $c_{a-i} = c_{b-i} \forall i = 1, 2, \dots, u$. Since $c_{a-u-1} \neq c_{b-u-1}$, Algorithm 1 visited position $b - u$ and string $s_{b''} =$

$[c_{b-u} c_{b-u+1} \dots c_{b+n+t-1}]$ will be extracted in line 22, in which case s_b will not be extracted since it is a substring of s_b .

By contradiction, there exists no pair of elements s_a and s_b ($a < b$) s.t. $c_{a+n} = c_{b+n}$ and $c_{a-1} = c_{b-1}$, therefore $c_{t_i-1} \neq c_{t_j-1}$ and $c_{t_i+n} \neq c_{t_j+n}, \forall i, j = 1, 2, \dots, k, i \neq j$.

By *Definition 2.1*, s is a maximized substring

However, the algorithm does not generally guarantee to extract all maximized substrings from unlabeled data. Namely, in the example sentence we have shown in Section 1 (“使一致认定界限数的期望值近似于一致正确界限数的期望值，求得一致认定界限的期望值/认定界限数的值。”), “界限” is a maximized substring according to the definition (with the occurrences being “正确界限数” and “认定界限的”), but it is not extractable by Algorithm 1. This happens because after the algorithm extracted “一致认定界限”, the new position for the next look-up becomes the end of this substring, and the second valid occurrence of “界限” is automatically omitted. We consider this as a necessary compromise in order to keep the efficiency of one-time scanning. To further investigate this issue, we have extracted all maximized substrings from the dev set of Chinese Treebank 7 (CTB7) which has 10,136 sentences. We used the following method in order to extract all maximized strings from a doc: first, we modified Algorithm 1 to force it to look-up from all possible positions regardless of whether a maximized string is found, which can be done by simply replacing line 28 (“ $i \leftarrow i + |s|$ ”) with “ $i \leftarrow i + 1$ ”. Then, from the occurrence list of an extracted substring, we remove all the sets of occurrences if they can be extended from the left-hand side from their original positions in the document while still being equal to each other. Finally, we remove all the extracted substrings if their occurrence list has either zero or one instance left.

We compared the result with the maximized substrings extracted by Algorithm 1 from the same dataset. With Algorithm 1, we are able to extract 99.62% of the maximized substrings. This small difference suggests that the disadvantage of Algorithm 1 of incomplete extraction can be easily overcome by adding more raw sentences.

In addition, we have observed in preliminary experiments that extracting all maximized substrings is not only unnecessary, but can introduce harmful noise. In the next section, we will discuss our solution to this problem.

2.3 Short-Term Store

Maximized substrings can provide good estimations of word boundaries, but noise can be introduced during the extraction process in Algorithm 1.

To address this problem, we take advantage of a linguistic phenomenon. It has been observed

that a word occurring in the recent past has a much higher probability to occur again soon, when compared with its overall frequency (Kuhn and De Mori 1990). It follows that, for speech recognition, we can then use a window of recent history to adjust the static overall language mode.

This observation is applicable to the task of maximized substring extraction in the following way. Suppose a substring is registered into the data structure. If the substring is in fact a word, it is much more likely to reoccur in the next 50 to 100 sentences than in the remainder of the corpus (especially when it is a technical term or a named entity). Otherwise, the substring represents noise, and it should have a more sparse distribution in the corpus.

This motivated us to introduce a functionality into the process of maximized substring extraction, called “short-term store” (STS). The STS is an analogy to the cache component in speech recognition as well as the human phonological working memory in language acquisition. It restricts the length of the visible context when extracting the next occurrence of a registered substring, making it proportional to the current number of occurrences of the substring. For a registered substring, the extraction algorithm scans a certain number of sentences after the latest occurrence of the substring, where the number of sentences $D(s)$ is determined as follows:

$$D(s) = \begin{cases} \lambda \cdot \text{count}(s), & \text{if } \text{count}(s) < \theta \\ \infty, & \text{otherwise} \end{cases}$$

where $\text{count}(s)$ is the current number of occurrences of s in the data structure. The parameter λ contributes a fixed-length distance to the visible context. The parameter θ works as a threshold of reliability. If we can observe s at least θ times in a short period, we can treat s as a word, or a sequence of words, with a high level of confidence, in which case $D(s) = \infty$ means that s is no longer subject to periodical decaying and will stay in the data structure statically.

During the scanning of the $D(s)$ sentences, if a new occurrence of s is found, it is added into the data structure and $D(s)$ is recalculated immediately, starting a new scanning period. If no new occurrences are found, we remove the earliest occurrence of s from the data structure and then re-calculate $D(s)$. Note that although we have described the short-term store functionality as if each substring in the data structure is scanned separately, in practice we only need to make minor changes in Algorithm 1: each time an occurrence of a maximized substring is found, we associate a “time stamp” which is the order of the current sentence in the document with it. Then we perform the following two simple steps when the algorithm attempts to add this occurrence into the data structure:

- *Step 1.* If the occurrence list is empty, add the new occurrence into the list; otherwise the difference between time stamps of it and the most recent occurrence is calculated.
- *Step 2.* If the difference is smaller than $D(s)$, add the new occurrence into the list; otherwise the occurrence lists pop out the most recent occurrence and go back to Step 1.

By making this modification to Algorithm 1, we can obtain the effects of STS without increasing the complexity. Introducing STS into the extraction process results in a substantial improvement to the quality of extracted substrings. It is also important that STS greatly improves the processing efficiency for large scale unlabeled data by keeping the size of the data structure relatively small. This is because a substring entry will decay from the data structure if it has not been refreshed in a short period.

3 Chinese Word Segmentation

3.1 Baseline Segmentation System

We have used a word-character hybrid model as our baseline Chinese word segmentation system (Nakagawa and Uchimoto 2007; Kruengkrai et al. 2009a, 2009b). This hybrid model constructs a lattice that consists of word-level and character-level nodes from a given input sentence. Word-level nodes correspond to words found in the system’s lexicon, which has been compiled from training data. Character-level nodes have special tags called position-of-character (POC) that indicate the word-internal position (Asahara 2003; Nakagawa 2004). We have adopted the 6-tag tagset, which (Zhao, Huang, Li, and Lu 2006) reported to be optimal. This tagset is illustrated in Table 2.

Previous studies have shown that jointly processing word segmentation and part-of-speech tagging is preferable to separate processing, which can propagate errors (Nakagawa and Uchimoto 2007; Kruengkrai et al. 2009a). If the training data was annotated by part-of-speech tags, we have combined them with both word-level and character-level nodes.

Figure 2 shows an example of a lattice for the Chinese sentence: “陳德銘答記者問” (Chen Deming answers to journalists’ questions). The correct path is marked with bold lines. The upper part of the lattice (word-level nodes) represents known words from the system’s lexicon,

Table 2 Word representation with a 6-tag tagset: S, B, B_2, B_3, M, E

Word Length	1	2	3	4	5	6	7 or more
Tags	S	BE	BB_2E	BB_2B_3E	BB_2B_3ME	BB_2B_3MME	$BB_2B_3M \dots ME$

Fig. 2 A Word-character hybrid lattice of a Chinese sentence. Correct path is represented by bold lines.

and the lower part of the lattice (character-level nodes) represents unknown words. A sequence of character-level nodes are considered as an unknown word if and only if the sequence of position-of-character tags forms one of the cases listed in Table 2. This table also illustrates the permitted transitions between adjacent character-level nodes. We have used a dynamic programming algorithm to search for the best path in the lattice.

Note that there are some character-level nodes that are not reachable. For instance, the first character of the sentence combined with a POC tag “ B_2 ”, or the last character of the sentence combined with a POC tag “ M ”. We have omitted unreachable nodes from the lattice.

We apply the feature templates described in Table 3. Note that if the part-of-speech tags are not available, we omit those templates involving POS tags.

3.2 Maximized Substring Features

We have incorporated the list of extracted maximized substrings into the baseline system by using a technique which discriminatively learns their features. For every word-level and character-level node in the lattice, the method checks the maximized substring list for entries that satisfy the following two conditions:

- (1) The node matches the maximized substring at the beginning, the end, or both boundaries.
- (2) The length of the node is shorter than or equal to that of the entry.

For example, consider the lattice in Figure 2 with a maximized substring “陈德铭”. All of the character-level nodes of “陈” and “铭” are encoded with maximized substring features. A

Table 3 Feature templates. The “Condition” column describes when to apply the templates: W_{-1} and W_0 denote the previous and the current word-level node; C_{-1} and C_0 denote the previous and the current character-level node; N_{-1} and N_0 denote the previous and the current node of any types. $begin(\cdot)$ and $end(\cdot)$ denote the first and last character in a word-level node, respectively. Word-level nodes represent known words that can be found in the system’s lexicon. p_i, w_i, l_i denote the POS, surface form, and length of the node at position i , respectively.

Category	Template	Condition
Unigram	$\langle w_0 \rangle \langle p_0 \rangle \langle w_0, p_0 \rangle \langle l_0, p_0 \rangle \langle begin(w_0), p_0 \rangle \langle end(w_0), p_0 \rangle$ $\langle begin(w_0), end(w_0), p_0 \rangle$	W_0
	$\langle c_{-2}, p_0 \rangle \langle c_{-1}, p_0 \rangle \langle c_0, p_0 \rangle \langle c_1, p_0 \rangle \langle c_2, p_0 \rangle$ $\langle c_{-2}, c_{-1}, p_0 \rangle \langle c_{-1}, c_0, p_0 \rangle \langle c_0, c_1, p_0 \rangle \langle c_1, c_2, p_0 \rangle \langle c_{-1}, c_1, p_0 \rangle$	C_0
Bigram	$\langle w_{-1}, w_0 \rangle \langle p_{-1}, p_0 \rangle \langle w_{-1}, p_0 \rangle \langle p_{-1}, w_0 \rangle \langle w_{-1}, p_{-1}, w_0 \rangle \langle w_{-1}, w_0, p_0 \rangle$ $\langle w_{-1}, p_{-1}, p_0 \rangle \langle p_{-1}, w_0, p_0 \rangle \langle w_{-1}, p_{-1}, w_0, p_0 \rangle \langle l_{-1}, p_{-1}, l_0 \rangle \langle l_{-1}, l_0, p_0 \rangle$ $\langle l_{-1}, p_{-1}, p_0 \rangle \langle p_{-1}, l_0, p_0 \rangle \langle l_{-1}, p_{-1}, l_0, p_0 \rangle \langle end(w_{-1}), p_0 \rangle$ $\langle p_{-1}, begin(w_0) \rangle \langle end(w_{-1}), p_{-1}, p_0 \rangle \langle p_{-1}, begin(w_0), p_0 \rangle$	$W_{-1} \times W_0$
	$\langle c_{-1}, c_0 \rangle \langle p_{-1}, p_0 \rangle \langle c_{-1}, p_{-1}, c_0 \rangle \langle c_{-1}, c_0, p_0 \rangle$ $\langle c_{-1}, p_{-1}, p_0 \rangle \langle p_{-1}, c_0, p_0 \rangle \langle c_{-1}, p_{-1}, c_0, p_0 \rangle$	$C_{-1} \times C_0$
	$\langle p_{-1}, p_0 \rangle$	Otherwise

segmenter will only obtain information on those possible word boundaries that are identified by maximized substrings. The maximized substrings are not directly treated as single words, because a maximized substring can sometimes be a compound word or phrase.

For each match with a maximized substring entry, the technique encodes the following features on a node.

Basic: A binary feature that indicates whether the node matches at the beginning or end of the maximized substring. It is encoded both individually and as a combination with each other feature types.

Lexicon: There is a particular kind of noise in the extracted list of maximized substrings, namely, those like the substring “中美經”, which has resulted from the two phrases “中美經濟” (China and U.S. economy) and “中美經貿” (China and U.S. economic and trade). This happens when the boundary of a maximized substring is a shared boundary character of multiple other words. In this example, the last character “經” of the maximized substring is the character at the beginning of “經濟” (economy) and “經貿” (economic and trade). This kind of noise can be identified by checking the context of maximized substrings in system’s lexicon.

For a node that matches the maximized substring, we check the context of the maximized substring in the input sentence and compares it with the system’s lexicon. If any item in the lexicon is found that forms a positional relation with the maximized substring entry (as listed in

Table 4) then the corresponding features are encoded on the node.

It should be noted that, the positional relations listed in Table 4 is in fact a subset of the thirteen interval relations defined in (Allen 1983); the omitted relations are: “equal”, “before”, “before-inverse”, “during” and “during-inverse”. “equal” means that the maximized substring matches a lexicon entry, in which case it becomes redundant; and in all the other omitted relations, the maximized substring and the lexicon entry do not have either shared or adjacent characters on their boundaries, which would give us little clue to indicate the reliability of a extracted maximized substring.

Lexicon Composition: When a maximized substring is a match to more than one item in the lexicon, a combination of multiple lexicon features is more informative than individual features. We encode the combinations of lexicon features listed as in Table 5.

Frequency: We sort the list of maximized substrings by their frequencies. If a maximized substring is among the 10% most frequent it is classed as “highly frequent”, if it is among the top 30% it is “normal”, and all other cases are “infrequent”.

Table 4 Lexicon features

Sentence $s = \dots c_{-1}c_0c_1 \dots c_{n-1}c_nc_{n+1} \dots$		Representation
Maximized substring $x = c_0c_1c_2 \dots c_n$
Lexicon entry $l = c_ic_{i+1}c_{i+2} \dots c_j$
ID	Positional Relation	
L1	$0 = i < j < n$
L2	$0 < i < j = n$
L3	$0 = i < n < j$
L4	$i < 0 < j = n$
L5	$0 < i < n < j$
L6	$i < 0 < j < n$
L7	$i = n + 1$
L8	$j = -1$

Each one represents a positional relation between a maximized substring and a contextual substring which exists in system’s lexicon.

Table 5 Lexicon composition features

ID	At Beginning	ID	At Ending
B1	$\langle L1, L6 \rangle$	E1	$\langle L2, L5 \rangle$
B2	$\langle L6, L8 \rangle$	E2	$\langle L5, L7 \rangle$
B3	$\langle L1, L8 \rangle$	E3	$\langle L2, L7 \rangle$

Each one represents a combination of two Lexicon features that fire simultaneously.

The extracted maximized substrings cannot be directly treated as words, for two reasons: first, many of the extracted substrings are chains of words, including compounds and short phrases; while this is fine in the task of word segmentation as long as they still correctly identify word boundaries, it is necessary to break them into atoms for word extraction. Second, random noise that violates word boundaries can be introduced during the extraction process. In this section, we introduce two post-processing strategies for unknown word extraction that are effective in filtering out redundant compound words, phrases, and substrings that violate word boundaries.

4 Unknown Word Extraction

The extracted maximized substrings cannot be directly treated as words, for two reasons: first, many of the extracted substrings are chains of words, including compounds and short phrases; while this is fine in the task of word segmentation as long as they still correctly identify word boundaries, it is necessary to break them into atoms for word extraction. Second, random noise that violates word boundaries can be introduced during the extraction process. In this section, we introduce two post-processing strategies for unknown word extraction that are effective in filtering out redundant compound words, phrases, and substrings that violate word boundaries.

4.1 Redundancy Reduction

The extracted list of maximized substrings contains not only single words but also compound words and phrases, which are detrimental when extracting unknown words. With the help of a lexicon, we have applied a simple remedy by discarding all those extracted substrings that contain multiple non-overlapping multi-character words.

4.2 Lexicon-based Voting

The errors described in Section 3.2 are another influence of the performance of maximized substrings in unknown word extraction. We need to filter out these errors before evaluation,

unlike the case of word segmentation where we can encode lexicon-based features in the context of test sentences. If there is any word found in the lexicon that forms the positional relation L3, L4, L5, or L6 (Table 4) with an occurrence of a maximized substring, a vote is cast to discard the maximized substring. We remove the maximized substring from the extracted list if at least 50% of the occurrences vote for discarding.

5 Evaluation

5.1 Experiments on Chinese Word Segmentation

5.1.1 Setting

To evaluate our approach, we have conducted Chinese word segmentation experiments on several datasets. The first experiment is conducted on Chinese Treebank 5 (CTB5) and Chinese Treebank 7 (CTB7), which are two widely used datasets for word segmentation evaluation. The statistics of CTB5 and CTB7 are shown in Table 6: we have adopted the same setting of data division as (Wang et al. 2011): the training set, dev set and test set; the OOV rate is defined as the percentage of tokens in the datasets that are not found in the corresponding training set.

The second experiment is conducted on the second international Chinese word segmentation bakeoff (SIGHAN Bakeoff-2005) (Emerson 2005), which has four independent subsets: the Academia Sinica Corpus (AS), the Microsoft Research Corpus (MSR), the Hong Kong City University Corpus (CityU) and the Peking University Corpus (PKU). Since POS tags are not available in this dataset, we have omitted all templates that include them.

We have used two different types of unlabeled data. One is the test set itself, which means there is no external resource included in the system. Another is the Chinese Gigaword Second Edition (LDC2007T03), a large-scale dataset, this dataset is a collection of news articles from 1991 to 2004 published by Central News Agency (Taiwan), Xinhua News Agency and Lianhe Zaobao Newspaper. It includes both simplified Chinese characters and traditional Chinese characters, with a total amount of over 1.2 billion. In order to make fair comparison with the previous work, we have adopted the same setting as (Wang et al. 2011) in the experiments conducted on CTB5

Table 6 Statistics of CTB5 and CTB7

Dataset	Number of sentences			OOV rate per token	
	Training	dev	test	dev	test
CTB5	18,089	350	348	0.0811	0.0347
CTB7	31,131	10,136	10,180	0.0549	0.0521

and CTB7: we only used the XIN_CMN portion of the Chinese Gigaword, and we excluded the news articles published between 1994 and 1998 which overlap with CTB data. The resulted unlabeled data has approximately 204 million words.

Table 7 shows some of the extracted maximized substrings from the Chinese Gigaword. We have chosen these instances to demonstrate the approximate distribution of different types of maximized substrings in the entire extracted list. As we can see from the table, many of these maximized substrings are named entities, including names of persons such as “黄玉斌”, “诗琳通公主” and “小阪善太郎”, organizations such as “温布尔登俱乐部”, “内蒙古自治区政协” and “贝纳通车队”, and locations such as “丁村文化遗址”, “克拉玛依油田”, and “普济禅院”. Some of them are technical terms, such as “骨质疏松症” and “颗粒体病毒”. These strings are commonly accepted words in Chinese. Other instances in this table are compound nouns such as “黑海舰队问题”, and verb phrases such as “加快教育改革”. Although these strings are not single words, they still correctly identify word boundaries and are thus helpful in the task of word segmentation.

We have trained all models using the averaged perceptron algorithm (Collins 2002), which we selected because of its efficiency and stability. To learn the characteristics of unknown words, we

Table 7 Examples of extracted maximized substrings from Chinese Gigaword

Maximized Substring	Translation
内蒙古自治区政协	Political Consultative Conference of Inner Mongolia Autonomous Region
温布尔登俱乐部	Wimbledon Football Club
火星环球勘探者	Mars Global Surveyor
丁村文化遗址	Ruins of Dingcun culture
克拉玛依油田	Karamay oil field
黑海舰队问题	Black Sea Fleet problem
骨质疏松症	Osteoporosis
博斯曼法案	Bosman ruling
加快教育改革	Accelerate education reform
贝纳通车队	Benetton racing team
小阪善太郎	Kosaka Zentaro
黎以和谈	Israeli-Lebanese peace talks
普济禅院	Puji Temple
军事五项	Military pentathlon
利率管理体制	Interest rate regulation system
诗琳通公主	Princess Sirindhorn
珞巴族	Lhoba people
黄玉斌	Huang Yu-bin
颗粒体病毒	Granulosis virus

built the system’s lexicon using only the words in the training data with a frequency higher than a threshold . This threshold was tuned using the development data. In order to use the maximized substring features, we have used training data as unlabeled data for supervised models, and used both the training data and Chinese Gigaword for semi-supervised learning models.

We have applied the same parameters for all models, which are tuned on the CTB7 dev set: $\lambda = 50$ and $\theta = 3$.

We have used precision, recall and the F-score to measure the performance of segmentation systems. Precision p is defined as the percentage of words that are segmented correctly, and recall r is the percentage of words in the gold-standard data that are recognized in the output. The balanced F-score is defined as $F = \frac{2pr}{p+r}$.

We have also evaluated statistical significance of the results we obtained. We use McNemar’s test with Yates correction with correction factor 0.5 to compare the correctness (correct vs. incorrect) between the output of proposed systems against the output of baselines.

5.1.2 Experimental results on in-domain data

We have compared the performance between the baseline system and our approach. The results are shown in Table 8. Each row in this table shows the performance of the corresponding system. “Baseline” refers to our baseline hybrid word segmentation and POS-tagging system. “MaxSub-Test” refers to the method that uses only the test set as unlabeled data; “MaxSub-L” refers to the method that uses only the large-scale unlabeled data (Chinese Gigaword); “MaxSub-U” refers to the method that uses both. It should be noted that, though the test set is being used as unlabeled data during the evaluation in systems “MaxSub-Test” and “MaxSub-U”, it is not being used in any way during the training stage; we are applying such a setting due to the fact that, in a real world application, it is sometimes the case that the data to be segmented (which is equivalent to the test set) exists as a batch of sentences instead of a stream. We therefore want to make full use of the test set to see whether, by including it as a source for substring extraction,

Table 8 Evaluation results on CTB5 and CTB7

System	CTB5				CTB7			
	P	R	F	OOV Recall	P	R	F	OOV Recall
Baseline	97.48	98.44	97.96	75.28	95.19	95.86	95.52	74.90
MaxSub-Test	97.52	98.55	98.03	76.47	95.30	95.97	95.63	76.40
MaxSub-L	97.70	98.64	98.17	81.19	95.45	96.16	95.80	80.35
MaxSub-U	97.73	98.79	98.26	81.86	95.53	96.15	95.88	80.78

it can improve the segmentation accuracy over the baseline. In our evaluation, we have focused on the segmentation performance of our systems.

The results show that, using the test data as an additional source of information, “MaxSub-Test” outperforms the baseline method by 0.07 and 0.11 points in F-score on CTB5 and CTB7 test set, respectively. This result indicates that our method of using maximized substrings can enhance the segmentation performance even with a purely supervised approach. By adding the maximized substrings extracted from large scale unlabeled data, “MaxSub-L” increased the improvements to 0.21 and 0.28 points in F-score, and “MaxSub-U” further increased the improvements to 0.30 and 0.36 points in F-score. Furthermore, we have observed that “MaxSub-Test” increased the recall of OOV words by 1.19 and 1.50 on CTB5 and CTB7 test set, respectively. These numbers are further increased to 5.91 and 5.45 by “MaxSub-L”, and 6.58 and 5.88 by “MaxSub-U”. These results show that systems enhanced with maximized substrings are much more effective in handling OOV words compared to the baseline. Since the OOV rate in the test set of CTB5 and CTB7 are both less than 6%, it is likely that the improvement can be higher in a dataset with larger percentage of OOV words; this assumption is verified in our experiment in Section 5.1.4.

We have compared our approach with previous work in Table 9. Two methods from (Kruengkrai et al. 2009a; 2009b) are referred to as “Kruengkrai 09a” and “Kruengkrai 09b”, and are taken directly from the report of (Wang et al. 2011); “Wang 11” refers to the semi-supervised system in (Wang et al. 2011); “Zeng 13” refers to the semi-supervised system in (Zeng, Wong, Chao, and Trancoso 2013).

We have observed that our system “MaxSub-U” achieves the better accuracy compared to “Kruengkrai 09a”, “Kruengkrai 09b” and “Wang 11”. Our proposed method however does not show better accuracy compared to “Zeng 13”. “Zeng 13” processes raw sentences with the mixture

Table 9 F-measure on CTB5 and CTB7 test sets compared with previous work

System	CTB5	CTB7
Baseline	97.96	95.52
MaxSub-Test	98.03	95.63*
MaxSub-L	98.17*	95.80*
MaxSub-U ⁺	98.26*	95.88*
Kruengkrai 09a	97.87	95.40
Kruengkrai 09b	97.98	95.46
Wang 11 ⁺	98.11	95.65
Zeng 13 ⁺	98.27	96.72

⁺ semi-supervised systems. * $p < 0.05$ in McNemar’s test vs. Baseline.

of a character-based model and a word-based model; The two models are trained on labeled data, and then further updated using unlabeled examples, where the segmentation agreements are used as constraints to bias the models. On the other hand, our method and the one of “Wang 11” both collect substrings from unlabeled data which have high statistical likelihood to be words, and the information of these substrings can be used as feature and incorporated into character-based, word-based, or hybrid word segmentation models.

The results for the SIGHAN Bakeoff-2005 dataset are shown in Table 10. The first three rows (“Tseng 05”, “Asahara 05” and “Chen 05”) show the results of systems that have reached the highest score on at least one corpus (Tseng, Chang, Andrew, Jurafsky, and Manning 2005; Asahara, Fukuoka, Azuma, Goh, Watanabe, Matsumoto, and Tsuzuki 2005; Chen, Kikui, and Sumita 2005). “Best closed” summarizes the best official results on all four corpora. “Zhao 07” and “Zhang 06” represent the supervised segmentation systems in (Zhao and Kit 2007; Zhang, Zhou, Zhang, and Sun 2006). “Baseline”, “Maxsub-Test” and “MaxSub-U” refer to the same systems as in Table 8. For the unlabeled data, we have used the test sets of corresponding corpora for “MaxSub-Test”, and the Chinese Gigaword for “MaxSub-U”. Other parameters were left unchanged. The results do not indicate that our approach performs better than other systems. However, the joint model we used in this paper relies heavily on part-of-speech annotation in the training data; Since part-of-speech information is unavailable in the SIGHAN Bakeoff-2005 dataset, the advantages of our joint model is largely compromised. We therefore mainly focus on comparing the performance of the systems before and after adding maximized substring features. When compared with the baseline, our approach has yielded consistent improvements across the four corpora, and on the PKU corpus we have performed better than previous work.

Table 10 F-measure on SIGHAN Bakeoff-2005 test set compared with previous work

System	AS	CityU	MSR	PKU
Baseline	95.15	94.52	96.31	95.15
MaxSub-Test	95.22	94.59	96.43*	95.30
MaxSub-U ⁺	95.36*	94.82*	96.64*	95.57*
Tseng 05	94.7	94.3	96.4	95.0
Asahara 05	95.2	94.1	95.8	94.1
Chen 05	94.5	94.0	96.0	95.0
Zhang 06	95.1	95.1	97.2	95.1
Zhao 07	95.5	95.6	97.5	95.4
Best closed	95.2	94.3	96.4	95.0

⁺ semi-supervised systems. * $p < 0.05$ in McNemar’s test vs. Baseline.

5.1.3 Impacts of semi-supervised learning features and short-term store

In Table 11, we show the effects of the different maximized substring feature types proposed in this paper. We activated different combinations of feature types in turn and trained separate models. We also investigated the impact of the short-term store by training models without this feature. The rows of this table represent models and corresponding F-score, trained and tested on CTB5 and CTB7 with different configurations. The row “Baseline” is baseline system as in Table 8. “+Basic&Freq” represents the system “MaxSub-U” with only basic and frequency features activated, and STS turned off. The row “+All” represents a system activating all maximized substring features but still without STS. The last row “+All+STS” is identical to the system “Maxsub-U”. It is clear that lexicon-based features are effective in discriminating unreliable maximized substring from reliable ones, and the short-term store improves the segmentation performance by filtering out noises during the extraction of maximized substrings. The combination of these two techniques yields an improvement of 0.22 point in F-score, and thus are essential when using maximized substrings.

Moreover, to further investigate the effect of short-term store, we have compared “+All+STS” with another system, “+All+Naive”, where we replaced short-term store with a naive strategy to limit the context of maximized substring extraction: We split the unlabeled data into small chunks of 50 sentences and extracted maximized substrings from each chunk; then we combined all extracted maximized substrings into a single list and merged their occurrence lists. The result shows that short-term store outperforms this simple strategy on both CTB5 and CTB7 test sets. By comparing the maximized substring lists extracted with these two strategies, we found that the simple strategy left out many good word candidates which are infrequent. We think that this is a result of the fixed length chunking of the unlabeled data, as the occurrences of an infrequent maximized substring can be distributed across two or more adjacent chunks and therefore be dropped by the extraction algorithm. On the other hand, short-term store can adjust the context length of maximized substring extraction based on extracted occurrences, which avoided this problem.

Table 11 Influence of activated feature types and short-term store on CTB5 and CTB7 test data

System	CTB5	CTB7
Baseline	97.96	95.52
+Basic&Freq	98.04	95.61
+All	98.10	95.76
+All+Naive	98.15	95.79
+All+STS	98.26	95.88

Table 12 F-measure on out-of-domain data

System	Scientific articles corpus
Baseline	92.01
MaxSub-Test	92.52*

* $p < 0.01$ in McNemar’s test vs. Baseline.

5.1.4 Experimental results on out-of-domain data

To demonstrate the effectiveness of our method on out-of-domain text, we have conducted an experiment on a test set that was drawn from a corpus of scientific articles. This test set contains 510 sentences that have been manually segmented by a native Chinese speaker. We used the test set as the unlabeled data.

As the results show (Table 12), the system “MaxSub-Test” exceeded the baseline method by 0.51 in F-score, and the improvement is statistically significant ($p < 0.01$ in McNemar’s test vs. Baseline). Considering that the amount of unlabeled data is relatively small, it is likely that acquiring large-scale unlabeled data in the same domain will further benefit the accuracy.

5.2 Experimental Results on Unknown Word Extraction

5.2.1 Setting

We have also conducted unknown word extraction experiments. It is difficult to directly evaluate the precision and recall of a list of extracted unknown words, since there is no complete list of unknown words to be compared with. Previous studies have adopted evaluation methods based on hand annotation (Feng et al. 2004; Zhang et al. 2011). We instead used the word list of CTB7 as gold-standard data for evaluation. We used the entire CTB7 dataset as an input text for maximized substring extraction, which has 51,447 sentences. We used the same lexicon that has been used in previous studies (Feng et al. 2004; Zhang et al. 2011), which has 119,803 Chinese words of two to seven characters². With the words in the lexicon being known, there are 11,722 unknown words remaining in the word list of CTB7. The result is compared with the reduced N-gram method (Lin and Yu 2001) as well as the accessor variety method (Feng et al. 2004), which is one of the most widely applied Chinese word recognition method (Zhao and Kit 2007; Zhao and Kit 2008; Sun and Xu 2011).

² <http://www.mandarintools.com/segmenter.html>

5.2.2 Performance comparison

In Figure 3 we show the performance of three maximized substring-based systems: “MaxSub” represents the maximized substring extraction method described in Section 2.2, “MaxSub+LV” represents the previous system plus the post-processing technique of lexicon-based voting, and “MaxSub+LV+STS” represents the second system, which uses the short-term store. The parameters of the short-term store are $\alpha = 0.5$ and $\beta = 0.5$, which are the optimum combination for this dataset. We have applied the redundancy reduction method described in Section 3.1 to all three systems. The number of substrings have been reduced by 22.0% for “MaxSub” after applying the redundancy reduction method, and by 39.1% and 28.3% respectively for “MaxSub+LV” and “MaxSub+LV+STS” after applying both the redundancy reduction and lexicon-based voting method.

We have also implemented the methods of reduced N-gram (Lin and Yu 2001) and accessor variety (Feng et al. 2004) for comparison. The reduced N-gram method and the accessor variety method are represented as “Reduced N-gram” and “AV” respectively; “AV+LV” represents the

Fig. 3 Precision-Recall curves of maximized substring extraction and the accessor variety method on CTB7.

method of “AV” plus the lexicon-based voting technique, where 14.9% of the substrings are removed compared to “AV”. We have calculated the reduced N-gram and accessor variety for substrings of two to seven characters; for the accessor variety methods, we have also applied the adhesive judge rules (Feng et al. 2004) to reduce errors.

The plot points in Figure 3 are obtained by varying the threshold of minimum frequency of substrings that we treat as words, i.e., we only regard an extracted substring as a word candidate if and only if its frequency is no less than the threshold. This threshold varies from 2 to 8 for “MaxSub”, “MaxSub+LV”, and “MaxSub+LV+STS”, and from 2 to 9 for “Reduced N-gram”, “AV”, and “AV+LV”. The result shows that our method outperforms the methods of reduced N-gram and accessor variety, and both the short-term store and the lexicon-based voting techniques can largely contribute to the overall performance of unknown word extraction.

It should be noticed that the precision and recall of these systems are generally lower than those reported by (Feng et al. 2004), which is due to two reasons. We have considered only out-of-vocabulary words in the evaluation, while Feng et al. consider both known words and unknown words. Also, a major drawback of our automatic evaluation method is that, although commonly acceptable as single words by native Chinese speakers, named entities and compound nouns are often regarded as multiple words in the gold-standard data. So the precision and recall values tend to be much lower than in the case of human evaluation.

5.2.3 Efficiency

To demonstrate the efficiency of our approach for processing large-scale data, in Figure 4 we have compared the processing time of our system (“MaxSub+STS”, no post-processing) against the method of accessor variety. We have used the first four million sentences of as the Xinhua Newswire section in Chinese Gigaword Second Edition. The results show that up to 12 times the processing time can be required for this dataset. In addition, our method shows a quasi-linear time complexity while the time taken for the accessor variety method is $O(n^2)$.

5.2.4 Error analysis

In the extracted list of maximized substrings, we regard instances representing single words, compound words and phrases as meaningful strings, as they correctly identify word boundaries. Instances which violate word boundaries are regarded as errors. During the analysis of the output of maximized substring extraction, we found that the majority of errors involves person’s name. A typical instance is the string “舍维奇” (šević), which is a commonly used suffix in the transliteration of person’s names like “米洛舍维奇” (Milošević) and “伊比舍维奇” (Ibišević).

Fig. 4 Processing time comparison between maximized substring extraction and the accessor variety method on a large-scale text.

Since instances of transliteration are relatively rare in a Chinese lexicon, it leaves no trace for lexicon-based voting or features to reject this kind of instances.

In fact, this kind of errors happens not only in the transliteration of foreign person’s names. For instance, a maximized substring “记者马” (the journalist Ma) is extracted in our experiment from the context “记者马维坤” (the journalist Ma Weikun) and “记者马建国” (the journalist Ma Jianguo). Unlike in western languages, it is not common to split the surname and the given name in person’s names in Chinese, in which case the character “马” (Ma) violates word boundaries in person’s name “马维坤” (Ma Weikun) and “马建国” (Ma Jianguo). Like the case in transliteration, lexicon-based techniques are not helpful in rejecting this instance. More unfortunately, the given name “建国” in “记者马建国” (the journalist Ma Jianguo) also means “founding of a nation” and is therefore regarded as a lexicon entry, so it actually “supported” the right boundary identified by the instance “记者马” by applying lexicon-based techniques. This problem is not rare in our experiment, since given names in Chinese are often meaningful strings registered in a lexicon.

Many instances of this error pattern is however a result of word segmentation standard adopted in CTB. For example, the same maximized substring “记者马” (the journalist Ma) would not be considered as violating word boundaries if the gold-standard word segmentation follows the standard in the PKU corpus, as it treats Chinese family names and given names as separate words.

Another type of errors we observed in the CTB5 and CTB7 word segmentation experiments is that, although a maximized substring correctly identified the boundaries of a phrase, the segmenter still made incorrect segmentation decisions regarding word boundaries inside that phrase. This could happen when an extracted maximized substring corresponds to a phrase in the gold-standard dataset, while the system has no knowledge regarding the internal word boundaries of that phrase from either the lexicon or extracted maximized substrings. For example, “麦格赛赛奖” (Magsaysay Prize) is a maximized substring extracted from CTB5 test set, while the OOV person’s name “麦格赛赛” (Magsaysay) is not included in the list of maximized substrings. The best model in our experiments segmented this phrase as “麦格赛赛奖” which is incorrect. We think this problem can be largely relieved by adding word formation knowledge in Chinese, e.g., a person’s name and the word for “prize” can form a phrase, into the model as features; it is however outside the scope of this paper to discuss the methods of obtaining such knowledge.

6 Comparison to Related Work

The idea of using frequent strings collected from text in word segmentation and recognition have been long employed by previous works. Nagao and Mori (1994) proposed a method for collecting character N-grams from large scale texts. They investigated the characteristics of the resulted frequent strings, and demonstrated that it is possible to automatically recognize compounds, set phrases and idiomatic phrases in Japanese by merging adjacent frequent strings. Lin and Yu (2001) proposed a method which extracts unknown words from a corpus based on string frequency. They showed that by discounting a string’s frequency from its superstrings (“reduced N-gram”), the resulted set of frequent strings can be used as reliable indicators of word boundaries. This type of frequent strings has also been exploited by Zhao and Kit (2007) for enhancing Chinese word segmentation, and Sung et al. (2008) for calculating more reliable term frequencies.

Another line of research studied the properties of frequent substrings from the information theory aspect. Feng et al. (2004) proposed the accessor variety-based method, a widely applied method in Chinese word segmentation and word recognition (Sun and Xu 2011; Yang et al. 2011;

Zhao and Kit 2007; Zhao and Kit 2008), which is another view of measuring the likelihood of a string being a Chinese word. The accessor variety of a string s is defined as

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

where $L_{av}(s)$ and $R_{av}(s)$ are the number of distinct surrounding characters of s from its left and right-hand side, respectively. Accessor variety is closely related to the concept of branching entropy (Jin and Tanaka-Ishii, 2006) and can be seen as the continuous version of it.

Though accessor variety is a good indicator of word candidates in unknown word extraction, the candidates extracted with a high threshold of accessor variety can still sometimes be meaningless strings. For example, consider the string “常谈” which is not generally considered as a word or phrase in contemporary Chinese language. It is a suffix of the Chinese idiom “老生常谈” (cliche), so the value $R_{av}(\text{常谈})$ should be high in a Chinese corpus which contains this idiom. On the other hand, $L_{av}(\text{常谈})$ should also be high if the same corpus contains the phrase “常谈起” (often talk about). By the definition, we can draw the conclusion that $AV(\text{常谈})$ is high in this corpus and therefore should be a word, which is incorrect. This problem can happen because in the definition of accessor variety, the numbers of distinct surrounding characters of a string are counted from its left and right-hand side independently. On the other hand, this problem is avoided in maximized substring extraction, since all occurrences of a maximized substring must have distinct surrounding characters from both left and right-hand sides at the same time according to the definition.

Maximized substring extraction does not require the extra step of statistical substring reduction (SSR), which makes the implementation simple. Unlike the above methods where it is necessary to collect statistics from all possible strings in a corpus that takes quadratic time (or quasilinear time if implemented with suffix arrays), maximized substring extraction only scans the corpus once which is a linear time algorithm in average case. The efficiency advantage of maximized substring extraction is demonstrated in Section 5.2.3.

Another advantage of maximized substring extraction is its ability to utilize the short-term store technique, which enhances the performance of unknown word extraction and Chinese word segmentation. The effect of short-term store is demonstrated in Section 5.1.3. and 5.2.2.

7 Conclusion

We propose a simple yet effective approach for extracting maximized substrings from unlabeled data. These are a particular type of substrings that provide good estimations of unknown

word boundaries. We have evaluated our approach in two tasks. In the task of Chinese word segmentation, the extracted maximized substrings are incorporated with a supervised segmentation system through discriminative learning. We have demonstrated the effectiveness of our approach through experiments in both in-domain and out-of-domain data and have achieved improvements over the baseline systems across all datasets. In the task of Chinese unknown word extraction, our approach has also been demonstrated to outperform the previous work of accessor variety in both the accuracy and efficiency.

Acknowledgement

The authors would like to thank the reviewers as well as the editors who have given us valuable suggestions and pointed out insufficiencies in this paper. Their insights have greatly helped to improve this work.

Reference

- Agrawal, R. and Srikant, R. (1994). “Fast Algorithms for Mining Association Rules.” In *Proceedings of 20th International Conference on Very Large Data Bases, VLDB*, Vol. 1215, pp. 487–499.
- Allen, J. F. (1983). “Maintaining Knowledge about Temporal Intervals.” *Communications of the ACM*, **26** (11), pp. 832–843.
- Asahara, M. (2003). *Corpus-based Japanese Morphological Analysis*. Ph.D. thesis, Citeseer.
- Asahara, M., Fukuoka, K., Azuma, A., Goh, C.-L., Watanabe, Y., Matsumoto, Y., and Tsuzuki, T. (2005). “Combination of Machine Learning Methods for Optimum Chinese Word Segmentation.” In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*, pp. 134–137.
- Chen, A., Zhou, Y., Zhang, A., and Sun, G. (2005). “Unigram Language Model for Chinese Word Segmentation.” In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*, pp. 138–141. Association for Computational Linguistics Jeju Island, Korea.
- Chen, K.-J. and Ma, W.-Y. (2002). “Unknown Word Extraction for Chinese Documents.” In *Proceedings of the 19th International Conference on Computational linguistics-Volume 1*, pp. 1–7. Association for Computational Linguistics.
- Collins, M. (2002). “Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms.” In *Proceedings of the ACL-02 Conference on Em-*

- pirical Methods in Natural Language Processing-Volume 10*, pp. 1–8. Association for Computational Linguistics.
- Emerson, T. (2005). “The Second International Chinese Word Segmentation Bakeoff.” In *Proceedings of the 4th SIGHAN workshop on Chinese language Processing*, Vol. 133.
- Feng, H., Chen, K., Deng, X., and Zheng, W. (2004). “Accessor Variety Criteria for Chinese Word Extraction.” *Computational Linguistics*, **30** (1), pp. 75–93.
- Fischer, J., Heun, V., and Kramer, S. (2005). “Fast Frequent String Mining using Suffix Arrays.” In *null*, pp. 609–612. IEEE.
- Goh, C.-L., Asahara, M., and Matsumoto, Y. (2005). “Training Multi-classifiers for Chinese Unknown Word Detection.” *Journal of Chinese Language and Computing*, **15** (1), pp. 1–12.
- Grossi, R. and Vitter, J. S. (2005). “Compressed Suffix Arrays and Suffix Trees with Applications to Text Indexing and String Matching.” *SIAM Journal on Computing*, **35** (2), pp. 378–407.
- GuoDong, Z. (2005). “A Chunking Strategy towards Unknown Word Detection in Chinese Word Segmentation.” In *Natural Language Processing—IJCNLP 2005*, pp. 530–541. Springer.
- Jiang, M. T.-J., Liu, S.-H., Sung, C.-L., and Hsu, W.-L. (2010). “Term Contributed Boundary Feature using Conditional Random Fields for Chinese Word Segmentation Task.” In *ROCLING*, pp. 143–156.
- Jin, Z. and Tanaka-Ishii, K. (2006). “Unsupervised Segmentation of Chinese Text by Use of Branching Entropy.” In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, pp. 428–435. Association for Computational Linguistics.
- Kärkkäinen, J. and Sanders, P. (2003). “Simple Linear Work Suffix Array Construction.” In *Automata, Languages and Programming*, pp. 943–955. Springer.
- Kärkkäinen, J., Sanders, P., and Burkhardt, S. (2006). “Linear Work Suffix Array Construction.” *Journal of the ACM (JACM)*, **53** (6), pp. 918–936.
- Kim, D. K., Sim, J. S., Park, H., and Park, K. (2003). “Linear-time Construction of Suffix Arrays.” In *Combinatorial Pattern Matching*, pp. 186–199. Springer.
- Ko, P. and Aluru, S. (2003). “Space Efficient Linear Time Construction of Suffix Arrays.” In *Combinatorial Pattern Matching*, pp. 200–210. Springer.
- Kruengkrai, C., Uchimoto, K., Kazama, J., Wang, Y., Torisawa, K., and Isahara, H. (2009a). “An Error-driven Word-character Hybrid Model for Joint Chinese Word Segmentation and POS Tagging.” In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pp. 513–521. Association for Computational Linguistics.
- Kruengkrai, C., Uchimoto, K., Kazama, J., Yiou, W., Torisawa, K., and Isahara, H. (2009b).

- “Joint Chinese Word Segmentation and POS Tagging using an Error-driven Word-character Hybrid Model.” *IEICE Transactions on Information and Systems*, **92** (12), pp. 2298–2305.
- Kuhn, R. and De Mori, R. (1990). “A Cache-based Natural Language Model for Speech Recognition.” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **12** (6), pp. 570–583.
- Li, Z. and Sun, M. (2009). “Punctuation as Implicit Annotations for Chinese Word Segmentation.” *Computational Linguistics*, **35** (4), pp. 505–512.
- Lin, Y.-J. and Yu, M.-S. (2001). “Extracting Chinese Frequent Strings Without Dictionary From a Chinese corpus, its Applications.” *Journal of Information Science and Engineering*, **17** (5), pp. 805–824.
- Ling, G. C., Asahara, M., and Matsumoto, Y. (2003). “Chinese Unknown Word Identification using Character-based Tagging and Chunking.” In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 2*, pp. 197–200. Association for Computational Linguistics.
- Lü, X., Zhang, L., and Hu, J. (2004). “Statistical Substring Reduction in Linear Time.” In *Natural Language Processing-IJCNLP 2004*, pp. 320–327. Springer.
- Ma, W.-Y. and Chen, K.-J. (2003). “A Bottom-up Merging Algorithm for Chinese Unknown Word Extraction.” In *Proceedings of the 2nd SIGHAN workshop on Chinese language processing-Volume 17*, pp. 31–38. Association for Computational Linguistics.
- Nagao, M. and Mori, S. (1994). “A New Method of N-gram Statistics for Large Number of N and Automatic Extraction of Words and Phrases from Large Text Data of Japanese.” In *Proceedings of the 15th conference on Computational linguistics-Volume 1*, pp. 611–615. Association for Computational Linguistics.
- Nakagawa, T. (2004). “Chinese and Japanese Word segmentation using Word-level and Character-Level Information.” In *Proceedings of the 20th international conference on Computational Linguistics*, p. 466. Association for Computational Linguistics.
- Nakagawa, T. and Uchimoto, K. (2007). “A Hybrid Approach to Word Segmentation and Pos Tagging.” In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pp. 217–220. Association for Computational Linguistics.
- Nelson, M. (1996). “Fast String Searching with Suffix Trees.” In *Dr. Dobbs’s Journal*, pp. 115–119.
- Nong, G., Zhang, S., and Chan, W. H. (2011). “Two Efficient Algorithms for Linear Time Suffix Array Construction.” *Computers, IEEE Transactions on*, **60** (10), pp. 1471–1484.
- Sproat, R. and Emerson, T. (2003). “The First International Chinese Word Segmentation Bake-off.” In *Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing-Volume*

- 17, pp. 133–143. Association for Computational Linguistics.
- Sun, W. and Xu, J. (2011). “Enhancing Chinese Word Segmentation using Unlabeled Data.” In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 970–979. Association for Computational Linguistics.
- Tseng, H., Chang, P., Andrew, G., Jurafsky, D., and Manning, C. (2005). “A Conditional Random Field Word Segmenter for Sighan Bakeoff 2005.” In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*, Vol. 171.
- Wang, Y., Kazama, J., Tsuruoka, Y., Tsuruoka, Y., Chen, W., Zhang, Y., and Torisawa, K. (2011). “Improving Chinese Word Segmentation and POS Tagging with Semi-supervised Methods Using Large Auto-Analyzed Data.” In *IJCNLP*, pp. 309–317.
- Wu, H.-J. P., Ng Hong, I., and Ruibin, G. (2003). “Efficient Methods for Multigram Compound Discovery.” In *Proceedings of the 17th Pacific Asia Conference*, pp. 257–268.
- Yamamoto, M. and Church, K. W. (2001). “Using Suffix Arrays to Compute Term Frequency and Document Frequency for All Substrings in a Corpus.” *Computational Linguistics*, **27** (1), pp. 1–30.
- Yang, T.-h., Jiang, T.-J., Kuo, C.-h., Tsai, R. T.-h., and Hsu, W.-l. (2011). “Unsupervised Overlapping Feature Selection for Conditional Random Fields Learning in Chinese Word Segmentation.” In *Proceedings of the 23rd Conference on Computational Linguistics and Speech Processing*, pp. 109–122. Association for Computational Linguistics.
- Ye, Y., Wu, Q., Li, Y., Chow, K., Hui, L. C., and Yiu, S.-M. (2013). “Unknown Chinese word extraction based on variety of overlapping strings.” *Information Processing & Management*, **49** (2), pp. 497–512.
- Zeng, X., Wong, D. F., Chao, L. S., and Trancoso, I. (2013). “Co-regularizing Character-based and Word-based Models for Semi-supervised Chinese Word Segmentation.” In *ACL (2)*, pp. 171–176.
- Zhang, K., Wang, R., Xue, P., and Sun, M. (2011). “Extract Chinese Unknown Words from a Large-scale Corpus Using Morphological and Distributional Evidences.” In *IJCNLP*, pp. 837–845. Citeseer.
- Zhang, K., Liu, Q., Zhang, H., and Cheng, X.-Q. (2002). “Automatic Recognition of Chinese Unknown Words Based on Roles Tagging.” In *Proceedings of the 1st SIGHAN Workshop on Chinese Language Processing-Volume 18*, pp. 1–7. Association for Computational Linguistics.
- Zhang, R., Kikui, G., and Sumita, E. (2006). “Subword-based Tagging for Confidence-Dependent Chinese Word Segmentation.” In *Proceedings of the COLING/ACL on Main Conference*

Poster Sessions, pp. 961–968. Association for Computational Linguistics.

- Zhao, H., Huang, C.-N., Li, M., and Lu, B.-L. (2006). “Effective Tag Set Selection in Chinese Word Segmentation via Conditional Random Field Modeling.” In *Proceedings of PACLIC*, Vol. 20, pp. 87–94. Citeseer.
- Zhao, H. and Kit, C. (2007). “Incorporating Global Information into Supervised Learning for Chinese Word Segmentation.” In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pp. 66–74. Citeseer.
- Zhao, H. and Kit, C. (2008). “Exploiting Unlabeled Text with Different Unsupervised Segmentation Criteria for Chinese Word Segmentation.” *Research in Computing Science*, **33**, pp. 93–104.

Mo Shen: Mo Shen received his B.S. degree in Mathematics from Hong Kong Baptist University, Hong Kong, China, in 2010. He received the M.S. and Ph.D. degree in Informatics from Kyoto University, Kyoto, Japan, in 2012 and 2016. His current research interests include syntactic parsing, Chinese morphological analysis, and cognitive modeling of language.

Daisuke Kawahara: Daisuke Kawahara received his B.S. and M.S. in Electronic Science and Engineering from Kyoto University in 1997 and 1999, respectively. He obtained his Ph.D. in Informatics from Kyoto University in 2005. He is currently an associate professor at the Graduate School of Informatics at Kyoto University. His research interests center on natural language processing, particularly knowledge acquisition and text understanding.

Sadao Kurohashi: Sadao Kurohashi received the B.S., M.S., and Ph.D. in Electrical Engineering from Kyoto University in 1989, 1991 and 1994, respectively. He has been a visiting researcher of IRCS, University of Pennsylvania in 1994. He is currently a professor of the Graduate School of Informatics at Kyoto University. His research interests include natural language processing, knowledge acquisition/representation, and information retrieval. He received the 10th anniversary best paper award from journal of natural language processing in 2004, 2009 Funai IT promotion award, and 2009 IBM faculty award.

(Received May 6, 2015)

(Revised August 10, 2015)

(Rerevised October 21, 2015; January 15, 2016)

(Accepted February 11, 2016)