Xiaofei Lu

A Hybrid Model for Chinese Word Segmentation

This paper describes a hybrid model that combines machine learning with linguistic and statistical heuristics for integrating unknown word identification with Chinese word segmentation. The model consists of two major components: a tagging component that annotates each character in a Chinese sentence with a position-of-character (POC) tag that indicates its position in a word, and a merging component that transforms a POC-tagged character sequence into a word-segmented sentence. The tagging component uses a support vector machine (Vapnik, 1995) based tagger to produce an initial tagging of the text and a transformation-based tagger (Brill, 1995) to improve the initial tagging. In addition to the POC tags assigned to the characters, the merging component incorporates a number of linguistic and statistical heuristics to detect words with regular internal structures, recognize long words, and filter non-words. Experiments show that, without resorting to a separate unknown word identification mechanism, the model achieves an F-score of 95.0% for word segmentation and a competitive recall of 74.8% for unknown word identification.

1 Introduction

Word segmentation is a prerequisite for all word-based Chinese language processing systems. For example, for speech synthesis, many phonological rules depend on correct word segmentation, and text-to-speech systems need to compute boundaries between intonational phrases in long utterances and assign relative prominence to words in those utterances (Sproat et al., 1996). For information retrieval, accurate word segmentation has been shown to measurably improve retrieval performance (Palmer and Burger, 1997; Foo and Li, 2004).

A Chinese word consists of one or more Chinese characters, or *zi*. In languages where word boundaries are clearly delimited by whitespace and punctuation marks, word segmentation is relatively straightforward. However, the absence of unambiguous word boundary markers in real Chinese text makes Chinese word segmentation a nontrivial challenge. The task is further complicated by the lack of a commonly accepted definition of word in Chinese among theoretical linguists. In practice, the Chinese language processing community adopts a pragmatic approach to this issue, where the definition of word varies depending on the purpose of the natural language processing system and the segmentation standard it adopts (Sproat et al., 1996; Wu, 2003; Gao et al., 2005).

A primary source of difficulty for Chinese word segmentation comes from segmentation ambiguities, including covering ambiguity and overlapping ambiguity (Liang, 1987). Covering ambiguity refers to the case where two segments may or may not be combined to form a larger segment. For example, the string 要害 may be segmented into two units 要/害 will/hurt 'will hurt' or one unit 要害 'vitals', depending on context. Overlapping ambiguity refers to the case where a segment may combine with either its preceding or following segment. For example, in the string 布什在谈话中指出, the segment 指 can potentially combine with either the preceding segment 中 or the following segment 出, as shown in (1) and (2) respectively. In this case, however, only the segmentation in (2) is acceptable.

∗ 布什	在	谈话	中指	出	(1)
Bush	at	talk	middle-finger	out	

布什	在	谈话	中	指出	
Bush	at	talk	middle	point-out	(2)
'Bush pointed out in his talk'					

Unknown words constitute a second source of difficulty for Chinese word segmentation. These are words that are not registered in the dictionary used by the word segmenter and/or are not found in the training data used to train the segmenter. While the size and domain specificity of the dictionary and training data may well affect the proportion of unknown words in real texts, unknown words will always exist, both because any dictionary creation effort has limited resources and because new words are constantly created. Chen and Bai (1998) report that 3.11% of the words in the Sinica Corpus (Chen et al., 1996), one of the largest word-segmented and POS-tagged Chinese corpora, are not listed in the CKIP lexicon, a Chinese lexicon with over 80,000 entries used for processing the corpus. These include unknown words of the categories of noun, verb, and adjective only, but not numeric type compounds or non-Chinese words. Xue (2003) partitions the 250K-word Penn Chinese Treebank (Xue et al., 2002) into training and test sets at a rather skewed ratio of 9.5:0.5 and finds that 4% of the words in the test set are unknown. Meng and Ip (1999) partition a smaller 72K-word corpus from Tsinghua University (Bai et al., 1992) at a 9:1 ratio, and report that 13% of the words in the test set are unknown.

Most previous studies treat word segmentation and unknown word identification as two separate problems, using a mechanism to identify unknown words in a post-processing step after word segmentation is done. However, determining where word boundaries are necessarily involves understanding how characters relate to and interact with each other in context, and it is desirable to capture this dynamic interaction by integrating unknown word identification with word segmentation. Several recent studies have taken a unified approach to unknown word identification and word segmentation (e.g., Sproat et al., 1996; Xue, 2003; Gao et al., 2005)

We describe a hybrid model that combines machine learning with linguistic and statistical heuristics for integrating unknown word identification with Chinese word segmentation. We adopt the notion of character-based tagging (Xue, 2003) to directly model the combinatory power of Chinese characters, i.e., the tendency for characters to combine with adjacent characters to form words, either known or unknown, in different contexts. The model consists of two components. First, a tagging component tags each character in a Chinese sentence with a position-of-character (POC) tag that indicates its position in a word. This component uses a support vector machine based tagger (Vapnik, 1995) to produce an initial tagging of the text and a transformation-based tagger (Brill, 1995) to improve the initial tagging. Second, a merging component transforms a POC-tagged character sequence into a word-segmented sentence, using a number of linguistic and statistical heuristics to detect words with regular internal structures, recognize long words, and filter non-words. Without resorting to a sophisticated mechanism for unknown word identification or additional resources other than a word-segmented training corpus, the model achieves an F-score of 95.0% for word segmentation and a recall of 74.8% for unknown word identification.

The rest of the paper is organized as follows. Section 2 reviews previous approaches to Chinese word segmentation. Section 3 describes the two components of the proposed model. Section 4 reports the experiment results of the model. Section 5 concludes the paper and points to avenues for future research.

2 Previous Approaches

Previous approaches for word segmentation generally fall into the following four categories: dictionary-based approaches, statistical approaches, statistical and dictionary-based approaches, and machine learning approaches.

In dictionary-based approaches, only words listed in the dictionary are identified. Most studies in this tradition use some variation of the maximum matching algorithm (e.g., Liang, 1987; Nie et al., 1995; Feng, 2001). Simply put, a maximum matching algorithm starts at the beginning of a sentence and inserts a word delimiter after the longest character string that matches a dictionary entry; it then moves to the character after the word delimiter and repeats the process until it reaches the end of the sentence. The algorithm can also start from the end of a sentence and search backwards. As the algorithm always favors the longest word, it in effect ignores segmentation ambiguities. To address this problem, some methods produce all possible segmentations first and then use certain criteria to select the best segmentation, e.g., using syntactic and semantic constraints (Yeh and Lee, 1991), favoring the segmentation in which each word has about the same length (Chen and Liu, 1992), etc. The performance of dictionary-based approaches is heavily dependent on the size and domain specificity of the dictionary used. These approaches also require the application of a separate unknown word identification mechanism in a post-processing step.

Relatively few studies have adopted purely statistical approaches to Chinese word segmentation. These studies use information-theoretical or probabilistic measures to determine whether adjacent characters form words or which segmentation is most likely for a sentence (e.g., Sproat and Shih, 1990; Ge et al., 1999). One advantage of statistical approaches is that they do not require any dictionary or word-segmented training data, but only a large raw corpus, which is relatively easy to obtain. However, they incorporate little linguistic knowledge and generally perform worse than other approaches. As Sproat et al. (1996) point out, such statistical methods can work as good unknown word identification mechanisms and can be used to augment existing electronic dictionaries and boost the performance of other word segmenters.

Statistical and dictionary-based approaches attempt to benefit from both worlds, using both information about words in the dictionary and statistical information derived from corpora to compute the most likely segmentation of a sentence. A good example of these approaches is Sproat et al. (1996). They represent a dictionary as a weighted finite state transducer, where weights or costs for word-strings are estimated based on their frequency in a large corpus. For each input sentence, they choose the path with the lowest cost as the best segmentation. This model requires separate mechanisms to estimate the probabilities of different types of unknown words. The authors recognize that the quality of the base lexicon is more important than the model and that unknown words constitute the greatest challenge.

With the availability of word-segmented training corpora, a number of supervised machine learning algorithms have been applied to Chinese word segmentation, including, for example, transformation-based learning (e.g., Hockenmaier and Brew, 1998; Florian and Ngai, 2001), maximum entropy (e.g., Xue, 2003; Low et al., 2005), conditional random fields (e.g., Tseng et al., 2005; Zhou et al., 2005), and linear mixture models (Gao et al., 2005). These approaches integrate unknown word identification with word segmentation and have achieved fairly competitive results.

3 Proposed Approach

This section describes a hybrid model that integrates unknown word identification with Chinese word segmentation. The major hypothesis tested in this study is that it is possible to directly model the combinatory power of individual characters, i.e., the tendency for individual characters to combine with adjacent characters to form words, either known or unknown, in different contexts. Furthermore, such modeling should have good potential for integrating unknown word identification with Chinese word segmentation. To this end, we adopt the notion of character-based tagging that has been employed for Chinese word segmentation and/or unknown word identification in a few recent studies (Zhang et al., 2002; Goh et al., 2003; Xue, 2003).

The proposed model consists of two major components. First, a tagging component annotates each character in a character sequence with a position-of-character (POC) tag that indicates the position of the character in a word. This component is based on the transformation-based learning (TBL) algorithm , using a tagger based on support vector machines (SVMs) to produce an initial tagging of a character sequence. Second, a merging component transforms the output of the tagging component, i.e., a POC-tagged character sequence, into a word-segmented sentence. In addition to the POC tags assigned to the character sequence, the merging component also makes use of a number of linguistic and statistical heuristics generalized from the training data to detect words with regular internal structures, recognize long words, and filter non-words. The overall architecture of the system is represented in Figure 1. An input Chinese sentence is first segmented into a character sequence, with a boundary marker after each character. The segmented character sequence is then processed by the POC-tagging component, where it is tagged first by the SVM-based initial tagger and then by the TBL tagger. Finally, the POC-tagged character sequence is transformed into a word-segmented sentence by the merging component.

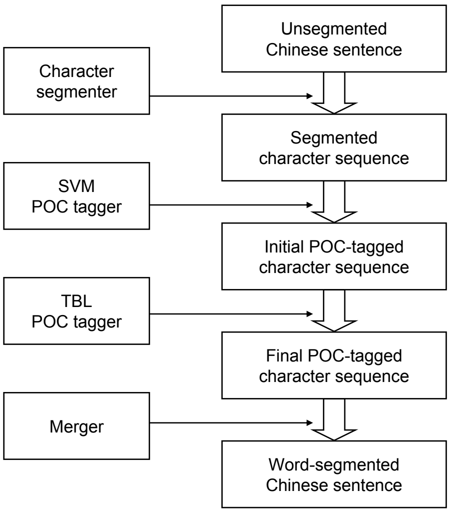
Unsegmented Chinese sentence
Character segmenter
Segmented character sequence
SVM POC tagger
Initial POC-tagged character sequence
TBL POC tagger
Final POC-tagged character sequence
Merger
Word-segmented Chinese sentence

Figure 1: System architecture.

3.1 The POC Tagging Component

Table 1 summarizes the tagset defined for the POC tagging component. The tagset consists of four tags: L, M, R, and W, each of which indicates that the character is in a word-initial, word-middle, or word-final position or is a monosyllabic word.

A TBL Tagger. The transformation-based learning algorithm is adopted for the POC tagging component. This is done because, as Brill (1995) argues, it captures linguistic knowledge in a more explicit and direct fashion without compromising performance, an advantage over other promising, statistical machine learning algorithms. The implementation of the algorithm used in this task is fnTBL (Ngai and Florian, 2001), which is more efficient than Brill's original implementation.

The TBL algorithm requires a POC-tagged training corpus (the *truth*) and its corresponding raw version. A POC-tagged corpus can be converted from a word-segmented

Tag	Description
L	Character in word-initial position
M	Character in word-middle position
R	Character in word-final position
W	Monosyllabic word

Table 1: POC tagset.

corpus by assigning each character a POC tag based on its position in the word containing it. The conversion process is illustrated by the following example, where the word-segmented sentence in (3) is converted into a POC-tagged character sequence in (4).

今天	是	星期一	.
Today	is	Monday	.

‘Today is Monday.’ (3)

今/L 天/R 是/W 星/L 期/M 一/R ./W (4)

In addition to the POC-tagged training corpus and its corresponding raw corpus, the algorithm requires three crucial components. The first is some initial tagging of the raw training corpus produced by an initial tagger. The algorithm itself places no requirement on the initial tagger. For example, it could assign the tag W (monosyllabic word) to each character in the corpus; or it could tag each character with its most likely tag derived from the training corpus. However, there is nothing that prevents one from using a more sophisticated tagger, and previous studies have shown that a better initial tagger leads to better final results and shorter learning time for the algorithm (Hockenmaier and Brew, 1998). For this reason, this study uses a more sophisticated initial tagger, an SVM-based tagger. A second initial tagger based on the hidden Markov model (HMM) is used as a comparison. Both of the two initial taggers are discussed in detail below.

The second component required by the TBL algorithm is the space of transformations allowed. The training process produces an ordered list of rules that can be applied to new text for tagging, but the kinds of transformations that can be learned need to be specified. Each transformation consists of a rewrite rule and a triggering environment. The set of transformations used in this study is similar to the set Ngai and Florian (2001) define for the task of base noun phrase chunking. In this case, however, the triggering context is defined over characters and POC tags instead of words and part-of-speech (POS) tags. The triggering context considered include the character and tag in the current position and the characters and tags in the three positions immediately preceding or following the current position. The specific rule templates defined are given in (5), where t_0 and c_0 denote the current tag and character respectively; t_{-i} and t_i denote the ith tag preceding or following t_0; c_{-i} and c_i denote the ith character preceding or following c_0; x, y, and z are tag variables; l, m, and n are character variables. The templates are categorized into three groups, depending on whether they require the identity of the current tag or character.

Change t_0 to *Tag*$_j$ if (5)

1. $t_0 = x$, c$_0 = l$, and one of the following is true:
   a) $c_{-1} = m$
   b) $c_1 = m$
   c) $c_{-2} = n$
   d) $c_2 = n$
   e) $c_{-1} = m$ and $c_{-2} = n$
   f) $c_1 = m$ and $c_2 = n$
   g) $c_{-1} = m$ and $c_1 = n$
   h) $t_{-1} = y$
   i) $t_1 = y$
   j) $t_{-2} = z$
   k) $t_2 = z$
   l) $c_{-1} = m$ and $t_{-1} = y$
   m) $c_1 = m$ and $t_1 = y$
   n) $c_{-1} = m$ and $t_1 = y$
   o) $c_1 = m$ and $t_{-1} = y$
2. $t_0 = x$ and one of the following is true:
   a) $c_0 = l$
   b) $c_{-1} = m$
   c) $c_1 = m$
   d) $c_{-2} = n$
   e) $c_2 = n$
   f) $c_{-1} = m$ or $c_{-2} = m$
   g) $c_1 = m$ or $c_2 = m$
   h) $c_{-1} = m$ or $c_{-2} = m$ or $c_{-3} = m$
   i) $c_1 = m$ or $c_2 = m$ or $c_3 = m$
   j) $t_{-1} = y$
   k) $t_1 = y$
   l) $t_{-2} = z$
   m) $t_2 = z$
   n) $t_{-1} = y$ or $t_{-2} = y$
   o) $t_1 = y$ or $c_2 = y$
   p) $t_{-1} = y$ or $t_{-2} = y$ or $t_{-3} = y$
   q) $t_1 = y$ or $t_2 = y$ or $t_3 = y$
   r) $t_{-1} = y$ and $t_1 = z$
   s) $t{-}_1 = y$ and $t_{-2} = z$
   t) $t_1 = y$ and $t_2 = z$
   u) $c_1 = m$ and $t_1 = y$ and $t_2 = z$

3. $c_0 = l$ and one of the following is true:

   a) $t{-1} = y$ and $t_{-2} = z$
   b) $c{-1} = m$ and $t{-1} = y$
   c) $c_1 = m$ and $t_1 = y$

The third component of the algorithm is a scoring function, which is used to compare the corpus to the *truth* and determine which transformation should be learned. In this study, the scoring function used is the number of POC tagging error reductions achieved after applying a transformation.

Once all the components are in place, the training process is iterative and takes place as follows. At each iteration, the learner applies each possible instantiation of the transformation templates to the text (starting with the text tagged by the initial tagger), counts the number of tagging error reductions each transformation achieves, and chooses the transformation that achieves the greatest number of tagging error reductions. That transformation is then applied to the text, and the learning process repeats, until no more transformations reduce errors beyond a pre-determined threshold. The output of the algorithm is a ranked list of transformations. Once the system is trained, a new sentence is tagged by first applying the initial tagger and then by applying the learned transformations to the sentence in the right order.

An SVM-Based Initial Tagger. SVMs (Vapnik, 1995) are binary classifiers on a feature vector space $\Re^L$. Given a set of training data, $\{(x_i\ ,y_i)|x_i{\in}\Re^L,\ y_i{\in}\{\pm 1\},\ 1{\leq}i{\leq}l\}$, where x_i is the ith sample in the training data and y_i is the label of x_i, a hyperplane, given in (6), separates the set into two classes in such a way that the constraints in (7) are satisfied:

$$w \cdot x + b = 0, w \in \Re^L, b \in \Re \tag{6}$$

$$y_i \cdot (w \cdot x_i + b) \geq 1 \tag{7}$$

In (6) and (7), w is a weight vector with one weight for each feature, and b is a bias, which is the distance of the hyperplane to the origin. Among all hyperplanes that separate the training data into two sets, SVMs find the optimal hyperplane with maximal margin, i.e., maximal distance between the hyperplane and the nearest positive and negative samples, because it is expected to minimize expected test error. Giving a test example x, its label y is determined by the sign of a discrimination function $f(x)$ given by the SVMs classifier as follows:

$$f(x) = sgn(\sum_{z_i \in SV} \alpha_i y_i K(x, z_i) + b) \tag{8}$$

where $b \in \Re$, z_i is a support vector, which receives a non-zero weight α_i, $K(x,\ z_i)$ is a polynomial kernel function of degree d given by $K(x, z_i)$ = $(x \cdot z_i\ +\ 1)^d$, which maps vectors into a higher dimensional space where all combinations of up to d features are considered, and SV denotes the set of support vectors, i.e., the vectors that receive a non-zero weight. The support vectors and the parameters are determined by quadratic

Characters	$c_{-2}, c_{-1}, c_0, c_1, c_2$
POC tags	t_{-2}, t_{-1}
Ambiguity class	a_0, a_1, a_2
Character bigrams	$(c_{-2}, c_{-1}), (c_{-1}, c_1), (c_{-1}, c_0), (c_0, c_1), (c_1, c_2)$
POC bigrams	$(t_{-2}, t_{-1}), (t_{-1}, a_1), (a_1, a_2)$
Character trigrams	$(c_{-2}, c_{-1}, c_0), (c_{-2}, c_{-1}, c_1), (c_{-1}, c_0, c_1), (c_0, c_1, c_2), (c_1, c_1, c_2)$
POC trigrams	$(t_{-2}, t_{-1}, a_0), (t_{-2}, t_{-1}, a_1), (t_1, a_0, a_1), (t_{-1}, a_1, a_2)$

Table 2: Feature set for the SVM-based POC tagger.

programming. If $f(x) = +1$, then x is a positive member, otherwise it is a negative member.

The implementation of the SVM classifier used in this study is SVMTool (Giménez and Màrquez, 2004). As SVMs are binary classifiers, some adaptation is necessary to make them suitable for multi-class classification tasks. Giménez and Màrquez use a one-per-class binarization, where they train an SVM for each class to determine whether an example is of this class or not. At classification time, the most confident tag given by all SVMs is selected.

The features used for POC tagging include character and POC tag n-grams. The two characters and POC tags preceding and following the current character and POC tag are considered. At running time, the POC tags of the characters to the right of the current character are not known. In SVMTool, a general ambiguity class tag, i.e., a label that concatenates all the possible POC tags for a character, is used for the right context characters. Table 2 summarizes the feature set used for POC tagging. For unknown characters, i.e., characters that are not found in the training data, only the POC features are used.

An HMM-Based Initial Tagger. To compare the impact of the initial tagger on the performance of the TBL algorithm, we implement a simple first-order HMM tagger (Charniak et al., 1993). This model computes the most likely tag sequence $t_1 ... t_n$ for a character sequence $c_1 ... c_n$, as follows:

$$\arg\max_{t_1 ... t_n} \prod_{i=1}^{n} p(t_i | t_{i-1}) p(t_i | c_i) \tag{9}$$

where t_i and c_i denote the ith tag in the tag sequence and the ith character in the character sequence respectively, $p(t_i | t_{i-1})$ denotes the transition probability, i.e., the probability of a tag given its previous tag, and $p(t_i | c_i)$ denotes the lexical probability, i.e., the probability of a tag given a character. The transition and lexical probabilities are estimated from the training corpus. For unknown characters, the lexical probabilities are uniformly distributed among the four POC tags defined in the tagset.

The transition probabilities are not smoothed, as all unseen POC tag bigrams are impossible combinations. Once the model is trained, the Viterbi algorithm (Rabiner, 1989) is used to tag new text.

3.2 The Merging Component

The output of the tagging component, i.e., a POC-tagged character sequence, becomes the input of the merging component. The merging component transforms the POC-tagged character sequence into a word-segmented sentence. By default, it concatenates all the characters in the character sequence, inserting a word boundary marker after each character tagged *R* (word-final position) or *W* (monosyllabic word). In addition to the POC tags assigned to the character sequence, the merging component also makes use of a number of linguistic and statistical heuristics for 1) detecting various types of unknown words with regular internal structures, i.e., numeric type compounds and non-Chinese words, reduplicated and derived words, and transliterated foreign names; 2) recognizing long words that have appeared in the training corpus; and 3) filtering non-words. The specific heuristics used in the merging component are detailed below.

Numeric Type Compounds and Non-Chinese Strings. The merging component uses regular expressions to detect numeric type compounds and non-Chinese character strings. The types of numeric type compounds handled using regular expressions include dates, times, percentages, fractions, numbers, etc. Non-Chinese character strings include email addresses, words or acronyms in foreign alphabets, etc. The patterns for each of these types of words are generalized from the training corpus. A character string that fits one of these patterns is grouped into one segmentation unit, and a word boundary marker is inserted after it. Gao et al. (2005) show that the performance of detecting such words using their internal properties alone is comparable with that of using contextual information.

Reduplicated and Derived Words. The merging component also uses heuristics to detect reduplicated words that are three or more characters long and a few types of derived words with predictable internal structures. In Chinese, many monosyllabic and disyllabic words can reduplicate in various patterns, e.g., 读书 'read' and 读读书 'read a little' (AAB pattern), and 高兴 'happy' and 高高兴兴 'very happy' (AABB pattern). If a string of characters fits one of the reduplication patterns in Chinese, it is grouped as one word.

For derived words, many morphemes are ambiguous between an affix and a word. For example, the morpheme 家 can be either an affix '-ist' or a common noun 'family'. Detection of the correct use of such ambiguous morphemes can be especially hard in disyllabic words. To avoid overgeneration, the heuristics only attempt to detect derived words formed with a root word that is two or more characters long and an unambiguous affix, such as 学家 study-expert '-ist'. If an unambiguous prefix or suffix is detected, it is attached to its following or previous word, if that word is at least two characters long.

Transliterated Foreign Names. Foreign names are usually transliterated using a sequence of Chinese characters, and most transliterated foreign names are three or more characters long. The error analysis of the results of the default merger indicates that these words pose a challenge to the POC tagger. Since only a subset of Chinese characters are used in transliterations, it is possible to use heuristics to detect a substantial part of transliterated names.

A list of characters used for transliterations is acquired from the wordlist generated from the training corpus. The algorithm starts with a simple observation, i.e., there is a dot used exclusively within transliterated foreign names to indicate different parts of a name, such as first, middle, or last name. For example, 'George Bush' is transliterated as 乔治·布什, where the dot within the name indicates that 乔治 'George' and 布什 'Bush' are different parts of the same name. Based on this observation, the algorithm acquires a list of characters for transliterations in the following three steps:

1) Extract names with the dot from the wordlist and store all characters in these names in a seed list.

2) Extract all candidate names that are four or more characters long with all but one character from the seed list. For each candidate, add the one character not on the seed list to the seed list. Repeat until no more characters can be acquired.

3) Filter out characters acquired in step 2 that have appeared only in one candidate.

In processing new text, the algorithm first uses the final character list to identify candidate names and then filters candidates using contextual information, as follows:

1) Add characters immediately before or after the dot in the text to the final list of characters, if they are not already included.

2) Identify character n-grams ($n \geq 3$) whose component characters are all from the list. For each n-gram, do steps 3-7.

3) Extract 5 characters to its left and right.

4) Strip the leftmost character of the n-gram if it forms a word with the 1, 2, 3, 4, or 5 left context characters and add it to the left context. Iterate until the leftmost character of the n-gram no longer forms a word with its left context characters.

5) Strip the leftmost 2 characters of the n-gram if they form a word with the 1, 2, 3, 4, or 5 left context characters and add them to the left context. Iterate until the leftmost 2 characters of the n-gram no longer form a word with its left context characters.

6) Do steps 4 and 5 with the rightmost characters of the n-gram and the right context characters.

7) If the final n-gram has three or more characters, it is considered a transliterated foreign name.

Long Words. In addition to the three types of heuristics used to detect unknown words with regular internal structures discussed above, the merging component also uses heuristics to detect known long words. One hypothesis tested in this research is that longer words behave more consistently and introduce less ambiguity than shorter words. In particular, we hypothesize that if a string of four or more characters has appeared in the training corpus as a word, it is also likely to be a word in the test corpus. Whereas there are foreseeable counterexamples for this hypothesis, its usefulness will be empirically tested in the experiments. There is no dictionary that is adopted across all segmentation standards and corpus annotation projects. Given the fuzzy line between compounds and phrases in Chinese, a character string may be considered as a compound in one segmentation standard but a phrase consisting of two or more words in another segmentation standard. For example, the string 个人所得税 'personal income tax' may be considered one word or three words, 个人 'personal', 所得 'income', and 税 'tax', depending on the definition of the word in the segmentation standard. Within the same segmentation standard, however, such strings should be treated consistently. To adapt to the segmentation standard following which the corpus is segmented, the merging component uses the wordlist generated from the segmented training corpus instead of any existing dictionary. If a string of four or more characters is found in the wordlist, the merging component considers it as one word.

Non-Word Filtering. The last type of heuristics is for filtering non-words, i.e., false new word candidates. The first of these is used to detect bisyllabic non-words. If the POC tagger tags two adjacent characters as *L* (word-initial) and *R* (word-final), the string is a candidate new word, if it has not occurred in the training data. The candidate is filtered in two steps.

First, it is checked against two short lists of characters that contain frequent, unproductive morphemes that do not occur in 1) word-initial position of new bisyllabic words, e.g., 而 'but', or 2) word-final position of new bisyllabic words, e.g., 这 'this', respectively. If the first or second character of the candidate is on the first or second list, respectively, the candidate is split into two words.

Second, the probability for a character to appear in word-initial or word-final position is estimated from the training data as in (10):

$$P(C, Pos) = \frac{F(C, Pos)}{F(C)} \tag{10}$$

where *C* denotes a character, *Pos* denotes a position in a word, $P(C,Pos)$ denotes the probability that *C* occurs in *Pos*, $F(C,Pos)$ denotes the number of times *C* occurs in *Pos*, and $F(C)$ denotes the number of times *C* appears in any position of any word. Given a candidate new word, the probability for it to be a word is computed as the joint probability of $P(C,Pos)$ for both of its component characters. If the joint probability is below a pre-determined threshold, then the candidate is considered a non-word and is split into two words.

Finally, the following two rules are used to merge or split character strings of certain patterns, based on the error analysis in the development stage. First, if two adjacent characters are both tagged *W* (monosyllabic word) but have always occurred in the training data as a word, they are merged into a word. Second, if three adjacent characters are tagged *W* (monosyllabic word), *L* (word-initial), and *R* (word-final) respectively, but the first two characters form a known word and the last two characters do not, then the first two are grouped into a word and the last one is left as a monosyllabic word.

4 Results

The model is developed and tested using the Contemporary Chinese Corpus developed at Peking University in mainland China (Yu et al., 2002). This corpus contains all the news articles published in January, 1999 in *People's Daily*, a major newspaper in mainland China. The corpus has a total of over 1.12 million tokens and is word-segmented and POS-tagged. The corpus is randomly partitioned into three sets, with 80% used for training, 10% used for development, and 10% reserved for testing. The final model is trained on the union of the training and development sets and results are reported on the test set.

4.1 POC Tagging Results

As discussed earlier, two initial taggers are used to compare the impact of the initial tagger on the TBL algorithm, namely, a first-order HMM tagger and an SVM-based tagger. In both cases, the same rule templates described in section 3.1 are used. The threshold for the scoring function of the TBL algorithm is set to 1, i.e., all rules that achieve two or more tagging error reductions are learned. In addition, fnTBL makes it possible to learn rules that, at the end of the training process, result in no negative application but a number of positive applications greater than a pre-determined threshold. This threshold is set to 1 as well, i.e., all rules that achieve two or more positive applications with no negative application at the end of the training process are also learned. Both thresholds are set empirically in the development stage. The results of the two initial taggers as well as the improved results achieved by the TBL algorithm over their output are summarized in Table 3.

POC Tagger	Accuracy
HMM tagger	0.814
HMM + TBL	0.936
SVM tagger	0.931
SVM + TBL	0.946

Table 3: POC tagging results.

The HMM tagger achieves an accuracy of 81.4%, which is improved to 93.6% by the TBL algorithm. This amounts to an absolute accuracy improvement of 12.2%, which equals to an impressive tagging error reduction rate of 65.6%. The SVM-based tagger achieves a much better initial tagging than the HMM tagger, with an accuracy of 93.1%. The TBL algorithm achieves an absolute accuracy improvement of 1.5%, or a tagging error reduction rate of 21.7%. The better initial tagging achieved by the SVM-based tagger results in less improvement for the TBL algorithm, but better overall tagging accuracy.

4.2 Segmentation Results

The output of the merging component is evaluated using the scoring algorithm adopted for the SIGHAN Chinese segmentation bakeoffs (Sproat and Emerson, 2003; Emerson, 2005). This algorithm evaluates word segmenters in terms of recall, precision, F-score, recall for out-of-vocabulary (OOV) words (i.e., unknown words), and recall for in-vocabulary (IV) words.

The results for word segmentation and unknown word identification are summarized in Table 4. As shown in the second row of the figure, the default merger, which uses the POC tags assigned to the character string alone without resorting to any heuristics, achieves an F-score of 93.5% along with a recall rate of 73.8% for unknown word identification. Rows three through five indicate the improvement achieved with each of the three types of heuristics incorporated in the merging component. Row three shows that the heuristics for identifying unknown words with regular internal structures (UWI) improve the recall rate for unknown words (R_{OOV}) by 2.2%; row four shows that the heuristics for long word identification (LWI) slightly boost performance on known words; and row five shows that the heuristics for non-word filtering (NWF) improve performance on known words, although the recall rate for unknown word identification is brought down by 1.2%. When all the heuristics are used together with the POC tags, the F-score for word segmentation is improved by 1.5% to 95.0%, and the recall rate for unknown word identification is also improved by 1% to 74.8%.

Merger	R	P	F	R_{OOV}	R_{IV}
Default	0.932	0.938	0.935	0.738	0.944
+UWI	0.933	0.939	0.936	0.760	0.944
+LWI	0.933	0.940	0.937	0.737	0.945
+NWF	0.942	0.944	0.943	0.726	0.955
+ALL	0.947	0.952	0.950	0.748	0.959

Table 4: Word segmentation results.

Since the model uses no additional resources other than the training data, the results are directly comparable with the results of the systems that participated in the closed track of the Peking University Corpus in the second SIGHAN Chinese Segmentation

Bakeoff. Table 5 summarizes the results of our model and the top three systems in the bakeoff, namely, Chen et al. (2005), Tseng et al. (2005), and Zhang et al. (2005). As these results indicate, without a complicated unknown word recognition mechanism, the final model performs at the state of the art for word segmentation along with a competitive recall rate for unknown word identification.

System	R	P	F	R_{OOV}	R_{IV}
Ours	0.947	0.952	0.950	0.748	0.959
Chen	0.953	0.946	0.950	0.636	0.972
Tseng	0.946	0.954	0.950	0.787	0.956
Zhang	0.952	0.945	0.949	0.673	0.969

Table 5: Comparison of word segmentation results.

5 Discussion and Conclusion

This paper presents a new hybrid model that integrates unknown word identification with Chinese word segmentation using the notion of character-based tagging. The major hypothesis tested in this model is that character-based tagging has good potential for integrating the two, as it allows one to directly model the tendency for individual characters to combine with adjacent characters to form words in different contexts, regardless of whether the words formed are known or unknown. This hypothesis is confirmed by the results. With the default merger, the model achieves a competitive rate for unknown word identification without resorting to any complicated unknown word recognition mechanism.

One advantage of the use of the notion of character-based tagging is that since it is essentially a classification problem, it can directly benefit from improvements in classifiers. A second advantage of the two-component setup of the model is that it allows easy integration of additional linguistic insights as well as statistical heuristics in the merging stage. The current merging component of the model incorporates three types of linguistic and statistical heuristics. The heuristics for the recognition of unknown words with regular or predictable internal properties, including numeric type compounds, reduplicated words and certain derived words, and transliterated foreign names, prove useful for enhancing the performance on unknown words. The heuristics for the recognition of known long words tests the hypothesis that long words behave more consistently and introduce less segmentation ambiguity than shorter words. The hypothesis is supported by the improvement achieved by the heuristics. Finally, the heuristics for non-word filtering successfully catch a substantial portion of false unknown words detected by the POC tagger.

The current model does not make use of any additional resources other than the training data. Various lexical resources have been used in different word segmentation systems. For example, some of the resources used in Gao et al. (2005), which is by far the most sophisticated word segmentation system with access to the richest resources, include a 98,668-entry lexicon, a semi-automatically compiled morpho-lexicon that contains 59,960 morphologically derived words with information about morphological patterns and stems for each entry, a list of 373 family name characters, a list of 30,000 location names, a list of 1,355 organization names, a list of 618 transliterated name characters, a list of 151 characters for single-character person names, and a list of 177 single-character location names. Such lexical resources can be used to improve the tagging component by enriching the feature set used for POC tagging. They can also be used to improve and enrich the heuristics used in the merging component.

References

Bai, S. H., Xia, Y., and Huang, C. (1992). Automatic part of speech tagging system for Chinese. Technical report, Tsinghua University, Beijing, China.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

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*, pages 138–141.

Chen, K.-J. and Bai, M.-H. (1998). Unknown word detection for Chinese by a corpus-based learning method. *International Journal of Computational Linguistics and Chinese Language Processing*, 3(1):27–44.

Chen, K.-J., Huang, C.-R., Chang, L.-P., and Hsu, H.-L. (1996). Sinica corpus: Design methodology for balanced corpora. In *Proceedings of the 11th Pacific Asia Conference on Language, Information, and Communication*, pages 167–176.

Chen, K.-J. and Liu, S.-H. (1992). Word identification for Mandarin Chinese sentences. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 101–107.

Emerson, T. (2005). The second Chinese word segmentation bakeoff. In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*, pages 123–133.

Feng, Z. (2001). Hybrid approaches for automatic segmentation and annotation of a Chinese text corpus. *International Journal of Corpus Linguistics*, 6:35–42.

Florian, R. and Ngai, G. (2001). Multidimensional transformation-based learning. In *Proceedings of the 5th Workshop on Computational Language Learning*, pages 1–8.

Foo, S. and Li, H. (2004). Unsupervised Chinese word segmentation and its effect on information retrieval. *Information Processing and Management: An International Journal*, 40(1):161–190.

Gao, J., Li, M., Wu, A., and Huang, C.-N. (2005). Chinese word segmentation and named entity recognition: a pragmatic approach. *Computational Linguistics*, 31(4):531–574.

Ge, X., Pratt, W., and Smyth, P. (1999). Discovering Chinese words from unsegmented text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 271–272.

Giménez, J. and Màrquez, L. (2004). SVMTool: A general POS tagger generator based on support vector machines. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 43–46.

Goh, C.-L., Asshara, M., and Matsumoto, Y. (2003). Chinese unknown word identification using character-based tagging and chunking. In *Proceedings of the Interactive Posters and Demonstration Session at the 41st Annual Meeting of the Association for Computational Linguistics*, pages 197–200.

Hockenmaier, J. and Brew, C. (1998). Error-driven segmentation of Chinese. In *Proceedings of the 12th Pacific Asia Conference on Language, Information and Computation*, pages 218–229.

Liang, N. (1987). Shumian hanyu zidong fenci xitong-CDWS [an automatic segmentation system for written Chinese-CDWS]. *Journal of Chinese Information Processing*, 1(2):44–52.

Low, J., Ng, H., and Guo, W. (2005). A maximum entropy approach to Chinese word segmentation. In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*, pages 161–164.

Meng, H. and Ip, C. (1999). An analytical study of transformational tagging for Chinese text. In *Proceedings of the 12th Research on Computational Linguistics Conference*, pages 101–122.

Ngai, G. and Florian, R. (2001). Transformation-based learning in the fast lane. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 40–47.

Nie, J.-Y., Hannan, M.-L., and Jin, W. (1995). Unknown word detection and segmentation of Chinese using statistical and heuristic knowledge. *Communications of COLIPS*, 5(1/2):47–57.

Palmer, D. and Burger, J. (1997). Chinese word segmentation and information retrieval. In *Proceedings of the 1997 AAAI Spring Symposium on Cross-Language Text and Speech Retrieval*, pages 175–178.

Rabiner, L. (1989). A tutorial of hidden Markov models and selected applications in speech recognition. In *Proceedings of IEEE 1989*, pages 257–286.

Sproat, R. and Emerson, T. (2003). The first international Chinese word segmentation bakeoff. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*.

Sproat, R. and Shih, C. (1990). A statistical method for finding word boundaries in Chinese text. *Computer Processing of Chinese and Oriental Languages*, 4(4):336–351.

Sproat, R., Shih, C., Gale, W., and Chang, N. (1996). A stochastic finite-state word segmentation algorithm for Chinese. *Computational Linguistics*, 22(3):377–404.

Tseng, H., Chang, P., Andrew, G., Jurafsky, D., and Manning, C. (2005). A conditional random fields word segmenter for SIGHAN Bakeoff 2005. In *Proceedings of the 4th SIG-HAN Workshop on Chinese Language Processing*, pages 168–171.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin.

Wu, A. (2003). Customizable segmentation of morphologically derived words in Chinese. *International Journal of Computational Linguistics and Chinese Language Processing*, 8(1):1–28.

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

Xue, N., Chiou, F.-D., and Palmer, M. (2002). Building a large annotated Chinese corpus. In *Proceedings of the 19th International Conference on Computational Linguistics*.

Yeh, C.-L. and Lee, H.-J. (1991). Rule-based word identification for Mandarin Chinese sentences - a unification approach. *Computer Processing of Chinese and Oriental Languages*, 5(2):97–118.

Yu, S., Duan, H., Zhu, X., and Sun, B. (2002). The basic processing of contemporary Chinese corpus at Peking University. *Journal of Chinese Information Processing*, 16(5):49–64.

Zhang, H., Liu, T., Ma, J., and Liao, X. (2005). Chinese word segmentation with multiple post-processors in HIT-IR Lab. In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*, pages 172–175.

Zhang, K., Liu, Q., Zhang, H., and Cheng, X. (2002). Automatic recognition of Chinese unknown words based on roles tagging. In *Proceedings of the 1st SIGHAN Workshop on Chinese Language Processing*, pages 71–78.

Zhou, J., Ni, B., and Chen, J. (2005). A hybrid approach to Chinese word segmentation around CRFs. In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*, pages 196–199.