# Discovering Chinese Words from Unsegmented Text

Xianping Ge, Wanda Pratt, Padhraic Smyth
Information and Computer Science,
University of California, Irvine
{xge,pratt,smyth}@ics.uci.edu

**Abstract**

In English written text, words are separated by spaces, but in written Chinese text, there are no such separators between words. (See Figure 1.) Thus, effective information retrieval of Chinese text first requires good word segmentation. In this paper, we investigate an efficient algorithm to discover the words and their occurrence probabilities from a corpus of unsegmented text without using a dictionary. Using the probabilities of the words, word segmentation is done according to the maximum likelihood principle. Comparing the segmentation output by the algorithm with the correct segmentation, recall/precision of 65.65%/71.91% is achieved. If some simple post-processing is performed, recall/precision can be boosted up to 97.72%/91.05%.

## 1 Introduction and related work

Segmentation of Chinese text into words is a nontrivial task because the words have variable lengths, the same character may occur in many different words, and many characters are single-character words by themselves. [1] reviews previous works on Chinese word segmentation and studies the effect on information retrieval.

Sproat and Shih [2] develop a purely statistical method that utilizes the mutual information between two characters: $I(x,y) = \log \frac{p(x,y)}{p(x)p(y)}$. The character pair with largest mutual information is found and assumed to be a word. Then the algorithm is applied recursively to the rest of the sentence. The limitation of this method is that it can only deal with words of length 1 or 2.

Most other algorithms require a pre-compiled word list (dictionary, or lexicon). Some simply match the substrings with the words in the dictionary using a heuristic such as longest matching, which segments a sentence in such a way that the number of words is minimized. As noted in [1] and [3], the coverage of the dictionary is critical for the dictionary-based methods. Automatically learning new words from text is an unsolved problem.

In this paper we present a simple probabilistic model of Chinese text based on the occurrence probability of the words which make the following assumptions:

墨与欧盟谈判自由贸易

今天，墨西哥与欧盟就签署自由贸易协定问题在此间正式开始谈判。

在为期一周的谈判中，双方将围绕市场准入、服务与资本流动、政府采购、公平竞争、知识产权保护和纠纷仲裁等问题进行谈判。

Figure 1: An example of Chinese written text. There are no separators between the words except punctuation marks such as commas, periods. A word commonly contains 1 to 4 characters, while the same character can occur in many different words. There are about 6,000 frequently used characters, and a typical word dictionary contains more than 100,000 words.

1. There are a finite (although very large) number of words of length $1, 2, \ldots k$. (e.g. $k = 4$).
2. Each word has an unknown probability of occurrence.
3. Words are independent of each other, i.e., any two words can occur together, governed only by their respective probability of occurrence.

Given the probabilities of the words, according to the maximum likelihood principle, a sentence should be segmented into $w_1, w_2, \ldots, w_k$ such that $\prod p(w_i)$ is maximized, where $p(w_i)$ is the probability of word $w_i$. This can be easily done using dynamic programming.

Our model can be seen as a zero-th order hidden Markov Model (HMM). HMM models for word segmentation are studied in Ponte and Croft [3] and An and Wong [4]. In [3], a dictionary is used; in [4], the words in the corpus are presegmented and tagged with part-of-speech information. In this paper, we take a different approach to train the model. In the following sections, we investigate how to discover the words and their probabilities from a corpus of unsegmented text without using a dictionary.

## 2 Method

The unknown parameters of the model are the probabilities of the individual words. If we had a training corpus of segmented texts, we could count the words to compute the probabilities of the words. Conversely, if we knew the probabilities of the words, we could segment the sentences into words. This situation is similar to the question "which came

first, the chicken or the egg?" This dilemma is solved by the Expectation-Maximization (EM) algorithm. Metaphorically, the EM algorithm puts an egg there to "jump-start" the process. Specifically, the EM algorithm randomly assigns an initial value to the probabilities of the words. Using the current value of the probabilities of the words, the sentences in the corpus are segmented. From the segmentation results the probabilities of the words are re-estimated. This process is repeated a number of iterations until the probabilities converge. The convergence is guaranteed by the general property of the EM algorithm.

For convenience we denote by **sentence** a string between two neighboring punctuation marks (although it might be only a fragment of a sentence). Given a sentence of length $n$, there are $2^{n-1}$ possible ways to segment it, and we do not yet know which is the correct segmentation. But, from the current (although imperfect) estimate of the probabilities of the words, we can compute the likelihood $p_i$ of each segmentation. This sentence will be "shared", for the purpose of word counting, by all the segmentations according to their individual likelihood. E.g., in a segmentation with likelihood $p_i$, we increase the word count for each word by $\frac{p_i}{\sum_{j=1}^{2^{n-1}} p_j}$. We call this way of counting words "**soft-counting**" because all the possible words are counted. For comparison, [3] only counts words in the segmentation with the highest likelihood.

The soft-counting is done efficiently by dynamic programming. The input is a sentence $C_1C_2C_3 \ldots C_n$. For any word $C_{j_1} \ldots C_{j_2}$ inside this sentence, its count should be increased by $S_{j_1}^{left} p(C_{j_1} \ldots C_{j_2}) S_{j_2}^{right} / \alpha$, where

- $S_{j_1}^{left}$ is the sum of the likelihood of all the possible segmentations of the substring to the left of $C_{j_1}$,
- $p(C_{j_1} \ldots C_{j_2})$ is the current estimate of the probability of the word $C_{j_1} \ldots C_{j_2}$,
- $S_{j_2}^{right}$ is the sum of the likelihood of all the possible segmentations of the substring to the right of $C_{j_2}$,
- $\alpha$ is the normalizing constant, which is the sum of the likelihood of all the possible segmentations of this sentence. It is equal to $S_{n+1}^{left}$.

$S_{j_1}^{left}$ and $S_{j_2}^{right}$ are computed by dynamic programming. For example, the recursive function for $S_i^{left}$ is

$$S_i^{left} = \begin{cases} 1 & \text{if } i = 1 \\ p(C_1) & \text{if } i = 2 \\ \sum_{j=1}^{i-1} p(C_j \ldots C_{i-1}) S_j^{left} & \text{if } i > 2 \end{cases}$$

We compute $S_i^{left}$ for $i = 1, 2, \ldots, n+1$ from left to right in the first pass, at the end of which we get $\alpha = S_{n+1}^{left}$. Then we compute $S_i^{right}$ for $i = n, n-1, \ldots, 3, 2, 1$ from right to left; at the same time, we output the count of each word.

The complexity of the algorithm is $O(kIN)$ where $k$ is the maximum word length, $I$ is the number of iterations (usually 5-10), $N$ is the size of the corpus.

## 3 Results

We train our model on a corpus of (unsegmented) Chinese text about 100 MBytes in size. We report the performance of our algorithm (soft-counting) in Table 1, where we also list the results of the word-based method of [3] on a different corpus. **Recall** and **precision** compare the $n_1$ segmented words output by the algorithm with $n_2$ words in the correct segmentation (i.e. segmentation by hand). Let $c$ be the number of words in common. Then recall $= \frac{c}{n_2}$, precision $= \frac{c}{n_1}$.

| Segmenter | Recall(%) | Precision(%) |
|---|---|---|
| Soft-counting | 65.65 | 71.91 |
| (after postprocessing) | 97.72 | 91.05 |
| Word based [3] | 87.80 | 84.40 |
| (Perfect Lexicon) | 93.63 | 95.87 |

Table 1: Accuracy of segmentation algorithms

Although we do not use a dictionary, our results are quite good. We find that most of the errors of our algorithm come from 20 single-character auxiliary words (approximately equivalent to English words "of", "and", "or", "to", etc) that occur together with other words so often that our algorithm cannot tell them apart. After a simple post-processing step that separates these few words from other words, recall/precision increases from 65.65%/71.91% to 97.72%/91.05%.

## 4 Future work and Conclusions

We have been concentrating on the purely statistical approach, i.e., assuming no knowledge of Chinese other than that the Chinese words are 1, 2, ..., 4 characters long. In the future, we are interested in incorporating prior knowledge, e.g., lexicon, the distribution of word length, syntactic constraints, etc.

In conclusion, we presented a simple zero-th order Markov model of the words in Chinese text. We developed an efficient algorithm to train this model on an unsegmented corpus. The segmentation results are comparable to other dictionary-based methods.

## References


[1] Aitao Chen et al. Chinese text retrieval without using a dictionary. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, 1997.

[2] R. Sproat and C. Shih. A statistical method for finding word boundaries in Chinese text. *Computer Processing of Chinese and Oritental Languages*, 4:336–351, March 1990.

[3] Jay M. Ponte and W. Bruce Croft. Useg: A retargetable word segmentation procedure for information retrieval. In *Symposium on Document Analysis and Information Retrieval 96 (SDAIR)*, 1996.

[4] Q. An and W. S. Wong. Automatic segmentation and tagging of Hanzi text using a hybrid alogrithm. In *Proceedings of the 9th International Conference on Industrial & Engineering Applications of AI & Expert Systems*, June 4-7 1996.