Using Self-Supervised Word Segmentation in Chinese Information Retrieval

Fuchun Peng[1] Xiangji Huang[1] Dale Schuurmans[1] Nick Cercone[1] Stephen Robertson[2]
[1]Department of Computer Science, University of Waterloo, Waterloo, Canada
{f3peng, jhuang, dale, ncercone}@uwaterloo.ca
[2]Microsoft Research, Cambridge, U.K. and City University, London, U.K.
ser@microsoft.com

ABSTRACT

We propose a self-supervised word-segmentation technique for Chinese information retrieval. This method combines the advantages of traditional dictionary based approaches with character based approaches, while overcoming many of their shortcomings. Experiments on TREC data show comparable performance to both the dictionary based and the character based approaches. However, our method is completely language independent and unsupervised, which provides a promising avenue for constructing accurate multilingual or cross-lingual information retrieval systems that are flexible and adaptive.

1. INTRODUCTION

The increasing interest in cross-lingual and multilingual information retrieval has created the challenge of designing accurate information retrieval systems for Asian languages such as Chinese, Thai and Japanese. For multilingual information retrieval it is important to have an adaptable system which can be easily ported to new domains and languages. However, in designing information retrieval systems for these languages one faces the challenge of addressing the word segmentation problem as part of the retrieval process. That is, unlike English, in Asian languages, words are not explicitly delimited by whitespace. This creates significant problems both in interpreting queries and in indexing the text corpus.

The word segmentation problem in Asian languages has been heavily researched in the past decade [1, 3, 6, 12, 16, 21, 22, 26]. In the retrieval task for these languages, the first step to indexing is to tokenize the collection. Traditionally there have been two approaches taken to tokenization: the dictionary based approach and the character based approach [15, 19, 20]. In the dictionary based approach, one pre-defines a lexicon containing a large number of Chinese words and then uses heuristic methods such as maximum matching to segment the Chinese sentences. In the character based approach, sentences are tokenized simply by taking each character to be a basic unit. Both of these approaches have advantages and disadvantages. The dictionary based approach is the earliest and most widely used method in Chinese IR. It has the advantage of requiring smaller inverted index file, achieving faster retrieval speed, and flexibly allowing additional linguistic information to be incorporated into into the retrieval system (e.g. synonyms). The most prominent disadvantage of the dictionary based approach is that it requires a large pre-defined lexicon, which normally must be constructed by hand with significant amount of labor and time. Moreover, the lexicon constructed in one language/domain is not portable to another languge/-domain, and it is virtually impossible to list all the Chinese words in a dictionary since the set of words is open-ended [7]. An additional shortcoming of the traditional maximum matching method used in the dictionary approach is that a character sequence is always segmented the same way regardless of context, which clearly violates the true nature of Chinese text. In the character based approach, the most prominent advantage is that it does not require pre-defined lexicon. Each character is considered as a basic unit. However, the disadvantages include huge index file, much slower retrieval speed, and the fact that it is difficult to incorporate linguistic information of any kind.

Both the dictionary based and the character based approaches have been successfully applied to Chinese information retrieval in recent work [4, 7, 14]. Overall, the character based approach has tended to yield better retrieval precision [5, 15, 28], and therefore there is an argument about whether word segmentation is necessary at all for Chinese IR. However, some researchers [19, 20] have argued that there exits some inherent difficulties in the character based approach. For example, a modern Chinese IR system should be able to take into account more than just character information, but should also be able to exploit sophisticated techniques such as latent semantic indexing [13] and query expansion (as in English IR), which are all based on *words*. Thus, research on Chinese word segmentation remains an important open problem for IR systems.

In this paper, we propose an EM-based method for Chinese information retrieval which has many of the advantages of both the character based approach and the dictionary based approach, while overcoming many of the shortcomings of both methods. We call our approach *self-supervised segmentation*. In *self-supervised segmentation*, no predefined lexicon is required. Instead, all that is needed is a large unsegmented training corpus—which is almost always easy to

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 2001 ACM X-XXXXX-XX-X/XX/XX ...$5.00.

obtain. We automatically learn a lexicon and lexical distribution from the training corpus by using the EM algorithm [10], and then segment the collections using the Viterbi algorithm [23]. Unlike previous EM word segmentation methods [12], where one lexicon is learnt, we learn two lexicons (for reasons outlined below). Since our segmentation approach is completely unsupervised and language independent, it can be easily adapted to other languages.

The rest of the paper is organized as follows. Section 2 first introduces the self-supervised word segmentation algorithm, and Section 3 then briefly describes the weighting methods we use in the experiments. Section 4 then presents the experiments we have conducted on the TREC data set. Conclusions are given in Section 5.

2. SELF-SUPERVISED SEGMENTATION

For the convenience of non-Chinese speaking reader, we illustrate the Chinese segmentation problems by employing an artificial English segmentation problem where the whitespace has been removed between words and the task is to recover the proper word segmentation from the conjoined text. To understand how one can begin to make progress on this problem, note that in a general word segmentation task where there are no identifying markers between words, one could effectively exploit *known* words to guide the segmentation of unknown words [9]. For example, if the word "*computer*" is already known then upon seeing the text "*computerscience*" it is natural to segment "*science*" as a possible new word. To exploit this observation, we develop an EM based word discovery method that is a variant of standard EM training, but avoids getting trapped in local maxima by keeping two lexicons: a *core* lexicon which contains words that are judged to be trustworthy, and a *candidate* lexicon which contains all other candidate words that are not in the core lexicon. The remainder of this section describes our unsupervised word segmentation algorithm in detail.

2.1 EM segmentation and training

Assume we have a sequence of characters $C = c_1 c_2 ... c_T$ that we wish to segment into chunks $S = s_1 s_2 ... s_M$, where T is the number of characters in the sequence and M is the number of words in the segmentation. Here chunks s_i will be chosen from the core lexicon $V_1 = \{s_i, i = 1, ..., |V_1|\}$ or the candidate lexicon $V_2 = \{s_j, j = 1, ..., |V_2|\}$. If we already have the probability distributions $\theta = \{\theta_i | \theta_i = p(s_i), i = 1, ..., |V_1|\}$ defined over the core lexicon and $\phi = \{\phi_j | \phi_j = p(s_j), j = 1, ..., |V_2|\}$ over the candidate lexicon, then we can recover the most likely segmentation of the sequence $C = c_1 c_2 ... c_T$ into chunks $S = s_1 s_2 ... s_M$ as follows. First, for any given segmentation S of C, we can calculate the joint likelihood of S and C by

$$prob(S, C|\theta, \phi) = \prod_{i=1}^{M_1} \frac{p(s_i)}{2} \prod_{j=1}^{M_2} \frac{p(s_j)}{2} = \frac{1}{2^M} \prod_{k=1}^{M} p(s_k)$$

where M_1 is the number of chunks occurring in the core lexicon, M_2 is the number of chunks occurring in the candidate lexicon, and s_k can come from either lexicon. (Note that each chunk s_k must come from exactly one of the core or candidate lexicons.) Our task is to find the segmentation S^* that achieves the maximum likelihood:

$$S^* = \underset{S}{argmax}\{prob(S|C; \theta, \phi)\}$$
$$= \underset{S}{argmax}\{prob(S, C|\theta, \phi)\} \quad (1)$$

Given a probability distribution defined by θ and ϕ over the lexicon, the Viterbi algorithm [23] can be used to efficiently compute the best segmentation S of character string C. However, *learning* which probabilities to use given a training corpus is the job of the EM algorithm. Following [10], the parameter re-estimation formulas are as follows.

$$\theta_i^{k+1} = \frac{\sum_S \#(s_i, S) \times prob(S, C|\theta^k, \phi^k)}{\sum_{s_i} \sum_S \#(s_i, S) \times prob(S, C|\theta^k, \phi^k)} \quad (2)$$

$$\phi_j^{k+1} = \frac{\sum_S \#(s_j, S) \times prob(S, C|\theta^k, \phi^k)}{\sum_{s_j} \sum_S \#(s_j, S) \times prob(S, C|\theta^k, \phi^k)} \quad (3)$$

where $\#(s_i, S)$ is the number of times s_i occurring the segmentation S. In both cases the denominator is a weighted sum of the number of words in all possible segmentations, the numerator is a normalization constant, and (2) and (3) therefore are weighted frequency counts. Thus, the updates can be efficiently calculated using the forward and backward algorithm.

2.2 Self-supervised lexicon growing

The above algorithm requires two lexicons. Here we describe how they can be constructed automatically. Let us define C_1, C_2 as the training corpus and the validation corpus respectively, and let V_1 and $V2$ be the core candidate lexicons respectively. Initially, V_1 is set to be empty and V_2 is initialized to contain all candidate "words" that are generated from the training corpus by enumerating contiguous character strings of length 1 to some predefined maximum length L. In a first pass, starting from the uniform distribution, EM is used to increase the likelihood of the training corpus C_1. When the training process stabilizes, the M words with highest probability are selected from V_2 and moved to V_1, after which all the probabilities are rescaled so that V_1 and V_2 each contain half the total probability mass. EM is then run again. The rationale for shifting half of the probability mass to V_1 is that this increases the influence of core words in determining segmentations and allows them to act as more effective guides in processing the training sequence. We call this procedure of successively moving the top M words to V_1 *forward selection*.

Forward selection is repeated until the segmentation performance of Viterbi on the validation corpus C_2 leads to a decrease in F-measure (which means we must have included some erroneous words in the core lexicon). After forward selection terminates, M is decremented and we carry out a process of *backward deletion*, where the M words with the lowest probability in V_1 are moved back to V_2, and EM training is successively repeated until F-measure again decreases on the validation corpus C_2 (which means we must have deleted some correct core words). The two procedures of forward selection and backward deletion are alternated, decrementing M at each alternation, until $M \leq 0$; as shown in Fig. 1. As with EM, the outcome of this self-supervised training procedure is the probability distributions over the lexicons that can be used by Viterbi to segment test sequences.

```
0. Input
     Completely unsegmented training corpus C1
     Validation corpus C2

1. Initialize
     V1 = empty;
     V2 contains all potential words;
     OldFmeasure = infinite small;
     bForwardSelection=true;
     set M to a fixed number;

2. Iterate
     while (M > 0){
        Perform EM based on current V1 and V2 until convergence;

        Calculate NewFmeasure on validation corpus C2;

        if(NewFmeasure < OldFmeasure){
           // change selection direction
           bForwardSelection = ¬bForwardSelection;
           M = M-5;
        }

        OldFmeasure = NewFmeasure;

        //SelectCoreWords(true) do forward selectoin
        //SelectCoreWords(false) do backward selectoin
        SelectCoreWords(bForwardSelection);
     }
```

Figure 1: Self-supervised Learning

2.3 Mutual information lexicon pruning

Note that the likelihood is defined by a product of individual chunk probabilities (making the standard assumption that segments are independent), which means that the more chunks a segmentation has, the smaller its likelihood will tend to be. For example, in English, given a character sequence *sizeofthecity* and a uniform distribution over multigrams, the segmentation *sizeof|thecity* will have higher likelihood than segmentation *size|of|the|city*. Therefore, maximum likelihood will tend to prefer fewer chunks in its segmentation and consequently put large probability on long non-word sequences like *sizeof* and *thecity*. These words are called erroneous agglomerations. Another difficulty is that EM is known to have trouble escaping local maxima in similar sequence models [23]. To eliminate erroneous agglomerations and to pull EM out of the local maxima, we employ a mutual information based criterion to prune the lexicon.

Recall that the mutual information between two random variables is given by

$$MI(X,Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x) \times p(y)} \quad (4)$$

where a large value indicates a strong dependence between X and Y, and zero indicates independence. In our case, we want to test the dependence between two chunks s_1 and s_2. Given a long word, say s="*abcdefghijk*", we consider splitting it into its most likely two-chunk segmentation, say s_1="*abcd*" and s_2="*efghijk*". The *pointwise mutual information* [18] between s_1 and s_2 is

$$MI(s_1,s_2) = \log \frac{p(s)}{p(s_1) \times p(s_2)}. \quad (5)$$

To apply this measure to pruning, we set two thresholds $\gamma_1 > \gamma_2$. If the mutual information is higher than the threshold γ_1, we say s_1 and s_2 are strongly correlated, and do not split s. (That is, we do not remove s from the lexicon.) If mutual information is lower than the lower threshold γ_2, we say s_1 and s_2 are independent, and remove s from the lexicon and redistribute its probability to s_1 and s_2. If mutual information is between the two thresholds, we say s_1 and s_2 are weakly correlated, and therefore shift some of the probability from s to s_1 and s_2, by keeping a portion of s's probability for itself (1/3 in our experiments) and distributing the rest of its probability to the smaller chunks, proportional to their probabilities. The idea is to shift the weight of the probability distribution toward shorter words. This splitting process is carried out recursively for s_1 and s_2.

2.4 Summary

We have described our approach to automatically learning a lexicon given a corpus of Chinese text. Note that once the lexicon is established in this manner it can be used to segment the Chinese corpus for use in future IR tasks. That is, our automatic segmentation technique facilitates a dictionary based approach to IR without having to construct the dictionary by hand, making the ease and flexibility with which a retrieval system can be constructed comparable to that of a character based approach. Below we evaluate how well a dictionary based approach built on an automatically constructed lexicon compares to using a hand built dictionary and how it compares to a character based approach.

3. PROBABILISTIC TERM WEIGHTING

To construct a dictionary based IR system we considered two different single unit weighting functions. They are both extended versions of ICF,[1] which include document length and within-document and within-query frequencies as providing further evidence. Adding this evidence makes the term-weighting dependent on the document, which has been shown to be highly beneficial in English text retrieval [24].

3.1 BM25 weighting function

The first function, called BM25 [2], is given as

$$w = \frac{(k_1+1)*tf}{K+tf} * \log \frac{N-n+0.5}{n+0.5} * \frac{(k_3+1)*qtf}{k_3+qtf} \ \oplus \ k_2 * nq * \frac{(avdl-dl)}{(avdl+dl)} \quad (6)$$

where N is the number of indexed documents in the collection, n is the number of documents containing a specific term, *tf* is within-document term frequency, *qtf* is within-query term frequency, *dl* is the length of the document, *avdl* is the average document length, *nq* is the number of query terms, the k_is are tuning constants (which depend on the database and possibly on the nature of the queries and are empirically determined), K equals to $k_1*((1-b)+b*dl/avdl)$, and $\oplus$ indicates that its following component is added only once per document, rather than for each term. The component:

$$k_2 * nq * \frac{(avdl-dl)}{(avdl+dl)}$$

is called *correction factor* which was designed to take into account the length of a document. The value of the correction factor decreases with *dl*, from a maximum as $dl \to 0$, at which $dl = avdl$, and to a minimum as $dl \to \infty$, as shown in Figure 2.

This design of the correction factor assumes that, the shorter the document is, the more value the correction factor should have, i.e., the more possible the document is relevant.

[1] ICF is defined as $w_{qt} = log\frac{N-n+0.5}{n+0.5}$, where N is the number of indexed documents in the collection and n is the number of documents containing a specific term [25].

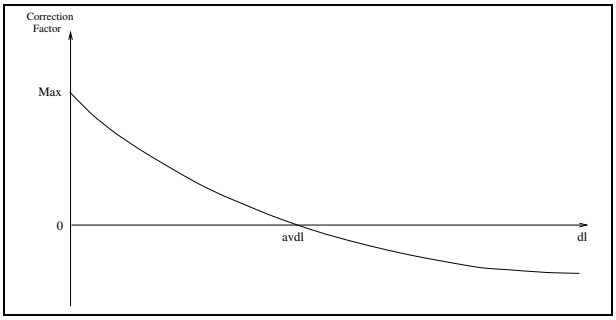

Figure 2: Curve for BM25's correction factor

In our experiments, the values of k_1, k_2, k_3 and b in the BM25 function are set to be 2.0, 0, 5.0 and 0.75 respectively. Note that we set k_2 to be 0, which means that the correction factor is not considered. The setting of these numbers was obtained from previous extensive experiments for English text retrieval and from initial experiments for Chinese text retrieval. For example, we found that the system produces better results if we set k_2 to be 0.

3.2 BM26 weighting function

The fact that better performance of BM25 is achieved when k_2 is set to be zero (i.e., the correction factor is ignored) indicates that the correction factor in BM25 is not designed properly. To tackle this problem, we propose an enhanced version of BM25, referred to as BM26, which is based on the following two assumptions: (1) overly short documents are not relevant; (2) the function curve for the correction factor should be consistent with the distribution of relevant documents in the standard text collection provided by the TREC conferences. BM26 is defined as follows:

$$w = \frac{(k_1+1)*tf}{K+tf} * \log\frac{N-n+0.5}{n+0.5} * \frac{(k_3+1)*qtf}{k_3+qtf} \quad \oplus \quad k_d * y \qquad (7)$$

where all the parameters have the same meaning as in BM25 except k_d is a tuning constant and

$$y = \begin{cases} \ln(\frac{dl}{avdl}) + \ln(x_1) & \text{if } 0 < dl <= rel_avdl; \\ (\ln(\frac{rel_avdl}{avdl}) + \ln(x_1))(1 - \frac{dl-rel_avdl}{x_2*avdl-rel_avdl}) & \text{if } rel_avdl < dl < \infty. \end{cases} \qquad (8)$$

in which dl is the length of the document, $avdl$ is the average document length, rel_avdl is the average relevant document length calculated from previous queries based on the same collection of documents, x_1 and x_2 are two parameters to be set. The parameter k_d in BM26 is set to have different values in our experiments. When k_d is 0, BM26 becomes BM25 since we set the parameter k_2 in BM25 to be 0 in our experiments.

The difference between BM26 and BM25 is in the y bit of the correction factor. In BM26, y will reach a maximum as $dl \to rel_avdl$, and a minimum as $dl \to 0$ (or $dl \to \infty$). In our experiments, x_1 and x_2 were set to 3 and 26 respectively. The curve of this new correction function is shown in Figure 3.

4. EXPERIMENTS AND ANALYSES

We conducted a number of experiments to investigate the effect of segmented length on Chinese text retrieval, the effect of different indexing methods on Chinese text retrieval, the effect of different weighting functions (setting different k_d in equation 7), and the effect of different training corpus on Chinese text retrieval. 54 TREC Chinese topics (28 for TREC-5 and 26 for TREC-6) were used in our experiments.

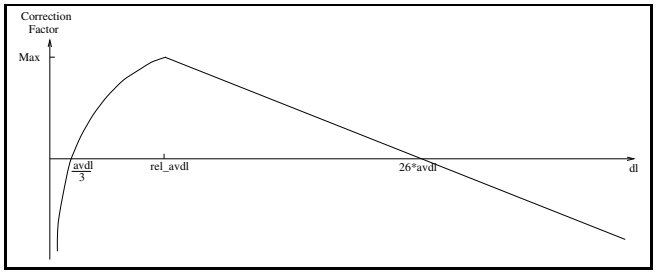

Figure 3: Curve for the new correction factor

4.1 Experiment setup and data sets

The test collection we use is from TREC-5 and TREC-6 (Text REtrieval Conferences) [27]. It contains 164,768 documents and consists of 139,801 articles selected from the *People's Daily* newspaper and 24,988 articles selected from the *Xinhua newswire*, with 0 bytes as the minimum file size, 294,056 bytes as the maximum size and 891 bytes as the average file size. 54 Chinese topics (28 for TREC-5 and 26 for TREC-6) were used in our experiments. The document collection used in TREC-6 Chinese track was identical to the one used in TREC-5. All the original articles were tagged using SGML. Chinese characters inside these articles were encoded using the GB (GuoBiao) coding scheme.

In our experiments, the TREC relevance judgments for each topic came from the human assessors of the National Institute of Standards and Technology (NIST). Statistical evaluation was done by means of the TREC evaluation program. Several measures are used to evaluate the retrieval result which is an ordered set of retrieved documents. The measures include Average Precision: average precision over all 11 recall points (0.0, 0.1, 0.2,..., 1.0); R Precision: precision after the number of documents retrieved is equal to the number of known relevant documents for a query; and Precision at 100 docs: precision after 100 documents have been retrieved. Detailed descriptions of these measures can be found in [27].

Our segmenter is trained on the training set C_1 with validation set C_2 (see subsection 2.2). The second corpus C_2 used here is a randomly selected 2000 sentence subset of the Chinese Treebank from LDC[2] which is has been segmented by hand. To evaluate the effect of different training sets C_1 (see subsection 4.3), we trained our segmented using two different sets: C_1^1 and C_1^2, where C_1^1 is 90M data which contains one year of *People's Daily* news service stories (www.snweb.com) and C_1^2 is the 130M TREC data itself with SGML marks removed. Our segmentation accuracy is around 70-74% on the Chinese Treebank. The parameter k_d in Table 1, Table 2, Table 3 and Table 4 is a tuning constant in BM26 (see equation 7). When k_d is 0, BM26 becomes BM25 in our experiments.

4.2 Influence of maximum word length L

According to *Frequency dictionary of modern Chinese,1980*, (see [11]), among the top 9000 most frequent words: 26.7% are unigrams, 69.8% are bigrams, 2.7% are trigrams, 0.007%

[2]http://www.ldc.upenn.edu/ctb

are 4-grams, and 0.002% are 5-grams. So most Chinese words are within 4 characters long. In our training algorithm, we set a maximum word length constraint L. To evaluate the effect of L, we experimented with L set to 2, 3 or 4. Table 1 and Table 2 shows the *average precision/R-precision*— results on TREC-5 and TREC-6 data sets.

k_d	$L = 2$	$L = 3$	$L = 4$
0	0.3264/0.3639	0.3422/0.3857	0.3246/0.3550
2	0.3326/0.3707	0.3504/0.3901	0.3319/0.3585
6	0.3430/0.3819	0.3613/0.3981	0.3416/0.3692
8	0.3453/0.3849	0.3641/0.3996	0.3419/0.3692
10	0.3450/0.3832	0.3661/0.4027	0.3422/0.3733
15	0.3403/0.3773	0.3601/0.3923	0.3412/0.3747
20	0.3320/0.3744	0.3536/0.3836	0.3368/0.3737
50	0.2756/0.3271	0.2982/0.3444	0.2865/0.3316

Table 1: Influence of L on TREC-5 using different weighting methods

k_d	$L = 2$	$L = 3$	$L = 4$
0	0.4363/0.4572	0.4660/0.4849	0.4531/0.4652
2	0.4459/0.4579	0.4754/0.4897	N/A
6	0.4595/0.4693	0.4906/0.4949	0.4781/0.4792
8	0.4635/0.4687	0.4950/0.4939	0.4822/0.4822
10	0.4661/0.4667	0.4968/0.4973	0.4841/0.4839
15	0.4659/0.4685	0.4970/0.5001	0.4820/0.4799
20	0.4603/0.4624	0.4928/0.4976	0.4758/0.4798
50	0.3990/0.4244	0.4186/0.4566	0.4101/0.4459

Table 2: Influence of L on TREC-6 using different weighting methods

Here one can see that the best results were achieved when $L = 3$. An explanation of this observation is that although more than 96% of Chinese words are within 2 characters long (which is the reason why the bi-gram indexing works so effectively [7]), there are still many words that are longer than 2 characters, and breaking them down reduces the precision. Obviously, 3-grams and 4-grams will have less ambiguity than bi-grams. However, 4-grams will include many more combinations than 3-grams, which reduces the reliability with which their occurrence probabilities can be estimated relative to 3-grams. Therefore, statistical over-fitting explains why $L = 4$ yields worse performance than $L = 3$.

Once can also see that the different weighting methods have a large effect on performance. Using the BM26 weighting function makes a significant positive contribution to the quality of retrieval compared to using BM25. By setting k_d to 10, the best retrieval performance can be obtained for all the EM-based word segmentation methods. Here we find that the performance on TREC-6 data set is much better than on TREC-5.

4.3 Influence of the training corpus

The previous section used data collected from the web (People's Daily new service) to train the segmenter. However, training the segmenter with web data has the possibility of adding noise and perhaps adversely influence retrieval performance. To evaluate the effect of different training corpora, we re-train the segmenter with TREC data C_1^2 (the union of the TREC-5 and TREC-6 corpora) and repeat the experiments. Tables 3 and 4 show the test results on TREC-5 and TREC-6 data respectively.

k_d	$L = 2$	$L = 3$	$L = 4$
0	0.3371/0.3707	0.3333/0.3739	0.3064/0.3493
2	0.3448/0.3788	0.3418/0.3784	0.3179/0.3569
6	0.3562/0.3855	0.3548/0.3830	0.3300/0.3671
8	0.3579/0.3877	0.3564/0.3882	0.3316/0.3701
10	0.3581/0.3911	0.3568/0.3858	0.3324/0.3743
15	0.3528/0.3892	0.3517/0.3777	0.3257/0.3649
20	0.3421/0.3854	0.3406/0.3742	0.3170/0.3539
50	0.2840/0.3343	0.2768/0.3229	0.2581/0.3094

Table 3: Influence of training corpus on TREC-5 using different weighting methods

k_d	$L = 2$	$L = 3$	$L = 4$
0	0.4582/0.4818	N/A	0.4472/0.4696
2	0.4687/0.4854	0.4711/0.4850	0.3432/0.4192
6	0.4828/0.4940	0.4867/0.4947	0.4716/0.4846
8	0.4868/0.4920	0.4906/0.4965	0.4746/0.4863
10	0.4898/0.4891	0.4932/0.4937	0.4770/0.4848
15	0.4899/0.4888	0.4925/0.4925	0.4745/0.4819
20	0.4847/0.4811	0.4861/0.4873	0.4668/0.4758
50	0.4181/0.4439	0.4106/0.4421	0.3951/0.4293

Table 4: Influence of training corpus on TREC-6 using different weighting methods

Comparing the corresponding columns of Table 1 and Table 3 (or Table 2 and Table 4), we find the performance on $L = 2$ increased but the performance on $L = 3$ and $L = 4$ decreased. One possible reason is that we stopped early in training the segmenter with $L = 3$ and $L = 4$. (One could hope that when the segmenter is fully trained, it should behave similarly on $L = 2$ and perhaps make some improvements.) Nevertheless, the experimental results show that the training corpus does indeed have some influence on the retrieval performance.

4.4 Comparisons

In this section, we compare the performance of our IR method against two standard text extraction methods. The first extraction method uses a hand built dictionary of words, compound words and phrases to index the texts. We refer to this method as the dictionary-based approach. The dictionary we use in the experiments contains 69,353 Chinese words and phrases. The second extraction method we compare to is a standard character based approach, in which documents are indexed by the single Chinese characters that appear in the text. However, we would like to emphasize that using single characters for indexing does not imply that we use single characters as keywords in search. For the character based approach we used in our experiments, a search could be conducted for any multi-character

word or phrase identified at search time, whether or not this word or phrase appeared in the dictionary. Therefore the experimental results we report for the character based approach use the character based method for indexing and use a dictionary based method for topic processing, because segmenting topics is much easier than segmenting the whole document collection and we found that using the character based topic processing yields far worse results than using dictionary based topic processing.

The motivation for proposing the self-supervised segmentation method is to incorporate the advantages of both the character based and the dictionary based approaches, while overcoming their shortcomings. For the TREC-5 comparison, we choose one best run in terms of Average Precision from the EM based method, one best run using only single words as retrieval keywords from the dictionary based method, and one run from the character based method where only BM25 was used. For the TREC-6 comparison, we also compare one best run in terms of Average Precision from the EM based method, one best run using only single word from the dictionary based method, and one run of the character based method. No phrase weighting has been used in the above six selected runs.

The topic processing method we used in the experiments is automatic and simple. First we rank the words extracted from topics by the values of their weights multiplied by the within-query frequencies, and then choose the top 19 ranking as retrieval keywords. The size of the character based index built for Chinese TREC collection is about one gigabyte, which is about twice the size of the document collection. The size is doubled since because it stores positional information about each character's occurrence. The index for the EM-based methods are at the same level as the dictionary based method. Detailed information about the size of the dictionary based and EM based methods is given in Table 5. In terms of retrieval time, the EM based methods are at the same level as the dictionary based methods. Both of these methods are about three times faster than the character based approach.

	character	dictionary	EM2	EM3	EM4
index	139,734	1,691,575	2,087,509	1,431,104	1,868,539
invert	1,077,393,536	678,257,616	648,472,816	667,634,000	677,900,752

Table 5: Size of Index Files (unit is byte)

We show the experimental results of the three methods on TREC-5 data in Tables 6 and 7, and the results on TREC-6 data sets in Tables 8 and 9. Here the *relevant retrieved* is the number of relevant documents retrieved out of the 2182 or 2958 documents in the collection for TREC-5 or TREC-6 respectively. We set the dictionary based method as the baseline.

On TREC-5 data, we find the EM based segmentation gives a 5.57% improvement in average accuracy over the dictionary based method, but it does a little worse than the character based method. In terms of R-precision, the EM based method yields better performance than both the character based and dictionary based methods. On TREC-6 data, the EM based method yields slightly worse results than both the dictionary based and the character based methods.

4.5 Discussion

Recall	character	dictionary	EM-based
0.00	0.7764	0.7681	0.7358
0.10	0.6243	0.6261	0.6249
0.20	0.5507	0.5075	0.5429
0.30	0.4987	0.4531	0.4998
0.40	0.4458	0.4034	0.4406
0.50	0.4247	0.3558	0.3954
0.60	0.3591	0.3180	0.3343
0.70	0.2711	0.2463	0.2803
0.80	0.2236	0.1760	0.1859
0.90	0.1408	0.1154	0.1082
1.00	0.0266	0.0082	0.0157
average precision	0.3795	0.3468	0.3661
improvement	9.43%	baseline	5.57%
relevant retrieved	1986	1883	1939

Table 6: TREC-5: comparing precision at specified recall rate

R	character	dictionary	EM-based
5	0.5571	0.5429	0.5500
10	0.5429	0.5143	0.5107
15	0.4881	0.4810	0.4881
20	0.4732	0.4732	0.4857
30	0.4369	0.4321	0.4595
100	0.3189	0.3150	0.3243
200	0.2380	0.2302	0.2418
500	0.1272	0.1216	0.1269
1000	0.0709	0.0672	0.0693
R-Precision	0.3963	0.3863	0.4027
improvement	2.59%	baseline	4.25%

Table 7: TREC-5: comparing R-precision

Previous research [4, 15] has suggested that exact segmentation may not be necessary in the IR task. The results that we have obtained here also support this point: although the segmentation accuracy is not necessarily very high (around 70-74%), it achieves comparable performance to a hand built dictionary approach. Strictly speaking, the *character based method* we use in the paper is mixture of a pure character based method and dictionary based method, i.e., it uses the character based method for indexing but uses a dictionary based method for topic processing. This hybrid system yields far superior results to the pure character based method, which we did not present in this paper.

Our work is most related to the work of Chen [7]. There too it was proposed that Chinese IR could be conducted without using a dictionary. In their method, one first collects occurrence frequencies for uni-grams and bi-grams from the corpus, and then uses a mutual information based criterion to segment Chinese text [26]. To use mutual information, they limit the word length to at most 2 characters. Similarly, we also use the frequencies from the corpus and also use mutual information during the process. However, our work differs from theirs in many respects. First we do not limit the word length to 2 characters. The maximum word length could be set arbitrarily to suit the application.

Recall	character	dictionary	EM-based
0.00	0.9604	0.9558	0.9110
0.10	0.8144	0.8217	0.8021
0.20	0.7396	0.7351	0.7482
0.30	0.6957	0.6586	0.6580
0.40	0.6662	0.5958	0.5935
0.50	0.5937	0.5507	0.5267
0.60	0.5195	0.4708	0.4608
0.70	0.4284	0.3844	0.3779
0.80	0.3224	0.2892	0.2738
0.90	0.1966	0.1485	0.1846
1.00	0.0239	0.0023	0.0297
average precision	0.5348	0.5044	0.4970
improvement	6.03%	baseline	-1.47%
relevant retrieved	2569	2536	2540

Table 8: TREC-6: comparing precision at specified recall rate

R	character	dictionary	EM-based
5	0.7615	0.8077	0.7538
10	0.7731	0.7885	0.7846
15	0.7615	0.7667	0.7564
20	0.7385	0.7404	0.7346
30	0.6910	0.6936	0.6833
100	0.5035	0.4923	0.4831
200	0.3615	0.3521	0.3406
500	0.1832	0.1808	0.1798
1000	0.0988	0.0975	0.0977
R-Precision	0.5404	0.5055	0.5001
improvement	6.90%	baseline	-1.07%

Table 9: TREC-6: comparing R-precision

In fact, our best results are achieved when $L = 3$. Second, the statistics we used were optimized by an iterative EM process, which is guaranteed to achieve at least a local optimum. This approach should be more reliable than the statistics direct from the corpus.

Using EM for word segmentation has many advantages, and has in fact been considered in previous research [12, 21]. However, due to the low segmentation accuracies these methods obtain, they still do not tend to be regarded as good methods for Chinese IR. Nevertheless, the results presented in this paper suggest that this need not be the case. In fact, we have shown that state of the art Chinese word segmentation, limited as it is in terms of accuracy, still facilitates retrieval performance that it comparable with the best of the current Chinese IR systems.

5. CONCLUSIONS AND FUTURE WORK

We have proposed a novel EM based method for segmenting Chinese words for the purposes of Chinese information retrieval, and presented experimental results on recent TREC data. Our method has the advantages of both the character based and dictionary based methods, while overcoming many of their shortcomings. Although our EM based segmentation method does not yield completely accurate segmentations by itself, it nevertheless performs well as a basis for Chinese IR. We achieve retrieval performance that this comparable (and sometimes even better) than the manual dictionary based and character based methods. Our results demonstrate the machine learning techniques can be successfully applied to Chinese IR to build an adaptable system. We also have submitted evaluation runs to NTCIR[3] for evaluation on a Chinese collection of 381,681 documents and 50 topics. The results of these evaluations are not yet known.

Building a Chinese IR system encompasses many research problems, and the performance of such systems can be influenced by several factors. In this paper, we have experimented with a single word weighting strategy, and are currently investigating alternative strategies involving word-pair weighting, which can greatly increase the performance [7, 15, 17]. Keyword extraction also plays an important role in IR systems. Our current keyword extraction method is very rough, and we are investigating more sophisticated extraction methods such as those used in [7, 8]. Finally, a successful Chinese IR system should employ many of the same techniques that are effective for English IR, such as latent semantic indexing [13] and query expansion. Combining our unsupervised Chinese word segmentation technique with latent semantic indexing has the potential to make an adaptable and high performance Chinese IR system. This is ongoing work.

6. REFERENCES

[1] R. Ando and L. Lee. Mostly-Unsupervised Statistical Segmentation of Japanese: Application to Kanji. In *Proceedings ANLP-NAACL*, 2000.

[2] M. Beaulieu, M. Gatford, X. Huang, S. Robertson, S. Walker, and P. Williams. Okapi at TREC-5. In *D.K.Harman (ed): Proceedings of TREC-5*, pages 143–166, 1997.

[3] M. Brent and X. Tao. Chinese Text Segmentation With MBDP-1: Making the Most of Training Corpora. In *Proceedings of ACL2001*, France, 2001.

[4] C. Buckley, A. Singhal, and M. Mitra. Using Query Zoning and Correlation within SMART: TREC-5. In *Proceedings of TREC-5*, pages 105–118, 1997.

[5] C. Buckley, J. Walz, M. Mitra, and C. Cardie. Using Clustering and SuperConcepts Within SMART: TREC-6. In *Proceedings of TREC-6*, pages 107–124, 1998.

[6] J.-S. Chang and K.-Y. Su. An Unsupervised Iterative Method for Chinese New Lexicon Extraction. *International Journal of Computational Linguistics & Chinese Language Processing*, 1997.

[7] A. Chen, J. He, L. Xu, F. C. Gey, and J. Meggs. 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. ACM, 1997.

[8] L.-F. Chien, T.-I. Huang, and M.-C. Chien. Pat-tree-based Keyword Extraction for Chinese Information Retrieval. In *Proceedings of the 20th Annual International ACM SIGIR Conference on*

[3] http://research.nii.ac.jp/ntcir/workshop/

Research and Development in Information Retrieval, pages 50–58. ACM, 1997.

[9] D. Dahan and M. Brent. On the Discovery of Novel Word-like Units from Utterances: An Artificial-language Study with Implications for Native-language Acquisition. *Journal of Experimental Psychology: General*, 128:165–185, 1999.

[10] A. Dempster, N. Laird, and D. Rubin. Maximum-likelihood from Incomplete Data via the EM algorithm. *J. Royal Statist. Soc. Ser.*, B(39), 1977.

[11] P. Fung. Extracting Key Terms from Chinese and Japnese text. *International Journal on Computer Processing of Oriental Language,Special Issue on Information Retrieval on Oriental Languages*, 99-121, 1998.

[12] X. Ge, W. Pratt, and P. Smyth. Discovering Chinese Words from Unsegmented Text. In *Proceedings of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 271–272, 1999.

[13] T. Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1999.

[14] X. Huang and S. Roberton. Okapi Chinese Text Retrieval Experiments at TREC-6. In *Proceedings of TREC-6*, pages 137–142, 1998.

[15] X. Huang and S. Roberton. A Probabilistic Approach to Chinese Information Retrieval: Theory and Experiments. In *Proceedings of the BCS-IRSG 2000: the 22nd Annual Colloquium on Information Retrieval Research*, Cambridge, England, 2000.

[16] W. Jin. Chinese Segmentation and its Disambiguation. In *MCCS-92-227*, Computing Research Laboratory, New Mexico State University, Las Cruces, New Mexico, 1992.

[17] K. L. Kwok. Comparing Representations in Chinese Information Retrieval. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 34–41. ACM, 1997.

[18] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, 1999.

[19] J. Nie and F. Ren. Chinese information retrieval: using characters or words? *Information Processing and Management*, 35:443–462, 1999.

[20] J. Nie, X. Ren, and M. Brisebois. On Chinese text retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 225–233. ACM, 1996.

[21] F. Peng and D. Schuurmans. Self-supervised Chinese Word Segmentation. In *F. Hoffman et al. (Eds.): Advances in Intelligent Data Analysis, Proceedings of the Fourth International Conference (IDA-01), LNCS 2189*, pages 238–247, Cascais, Portugal, 2001. Springer-Verlag Berlin Heidelberg.

[22] J. Ponte and W. Croft. Useg: A Retargetable Word Segmentation Procedure for Information Retrieval. In *Symposium on Document Analysis and Information Retrival 96 (SDAIR)*, 1996.

[23] L. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of IEEE*, 77(2), 1989.

[24] S. E. Robertson and S. Walker. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241, 1994.

[25] K. Sparck-Jones. Search Relevance Weighting Given Little Relevance Information. *Journal of Documentation*, 35(1), 1979.

[26] R. Sproat and C. Shih. A statistical method for finding word boundaries in chinese text. *Computer Proceedings of Chinese and Oriental Languages*, 4:336–351, 1990.

[27] E. Voorhees and D. Harman. Overview of the Sixth Text REtrieval Conference (TREC-6). In *Proceedings of the sixth Text REtrieval Conference (TREC-6)*. NIST Special Publication, 1998.

[28] R. Wilkinson. Chinese Document Retrieval at TREC-6. In *Proceedings of TREC-6*, 1998.