Computational Intelligence, Volume 16, Number 4, 2000

PROBABILITY-BASED CHINESE TEXT PROCESSING AND RETRIEVAL

XIANGJI HUANG, STEPHEN ROBERTSON[1]

Department of Information Science, City University, London EC1V 0HB, United Kingdom

NICK CERCONE AND AIJUN AN

Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1 Canada

We discuss the use of probability-based natural language processing for Chinese text retrieval. We focus on comparing different text extraction methods and probabilistic weighting methods. Several document processing methods and probabilistic weighting functions are presented. A number of experiments have been conducted on large standard text collections. We present the experimental results that compare a word-based text processing method with a character-based method. The experimental results also compare a number of term-weighting functions including both single-unit weighting and compound-unit weighting functions.

Key words: information retrieval, word-based and character-based Chinese text processing, single-unit and compound-unit weighting.

1. INTRODUCTION

Modern information retrieval (IR) systems should be able to take natural language queries and retrieve documents written in natural languages. One way to process natural language texts in IR is statistical. In this approach, linguistic units are extracted from the documents and queries. Those units extracted from the documents are used to index the documents. Those extracted from a query are weighted according to their degrees of importance. The most important units are chosen as query terms. An information retrieval system takes the selected query terms, matches the terms with the indexes of documents, calculates a retrieval status value for each matched document using a term-weighting method, and presents the user with potentially relevant documents from a collection of documents. The performance of the information retrieval system in identifying relevant documents greatly depends on the text processing method used.

We investigate the effectiveness of different text extraction methods and different term weighting methods in the context of Chinese information retrieval. It is a well-known problem that there is no separator between Chinese words, so Chinese words cannot be used easily to index or search texts as is possible in English. Therefore, some people use characters or *n*-grams as searchable tokens instead of words. We discuss two text extraction methods. One extraction method uses words, compound words, and phrases in the document and query texts as indexing terms to represent the texts. We refer to this method as the *word-based approach*. For this approach, text segmentation, which divides both document and query texts into linguistic units, is regarded not only as a necessary precursor but also as a bottleneck of this kind of system (Wu and Tseng 1993). The other extraction method is based on single Chinese characters, in which texts are indexed by the characters appearing in the texts (Chen 1992). By using single character approaches, a search could be conducted for any multi-character word or phrase identified at search time, no matter whether this word or phrase is in the dictionary. Both word-based and character-based methods have been used in information

Address correspondence to Xiangji Huang at the Department of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada; e-mail: jhuang@math.uwaterloo.ca.

[1]Microsoft Research, Ltd., 1 Guildhall Street, Cambridge, CB2 3NH, U.K.

retrieval systems. Cornell's SMART system (Buckley et al. 1996) uses character-based retrieval augmented with character bigrams. The Chinese retrieval system in Berkeley (Gey et al. 1996) is a purely word-based system, which uses a dictionary of 140,000 words to segment the Chinese collection and queries. Queens College's PIRCS system (Kwok et al. 1997) applies a combination of dictionary and statistical techniques to detect two-, three-, and occasionally four-character words. Its aim was to obtain good indexing features rather than "correct" segmentation. We use both word-based and character-based segmentation methods in our Okapi retrieval system (Beaulieu et al. 1996; Robertson et al. 1995). In this paper we present the two methods used in Okapi and provide empirical results that compare the two methods in terms of their effectiveness for Chinese text retrieval in Okapi.

One other issue in text retrieval is how to make use of the extracted terms in the retrieval process. A usual way is to weight the query terms and calculate a score of relevance for each matched document based on the terms' weights. Statistical term-weighting methods for IR traditionally have taken two forms: formal approaches, where an exact formula is derived theoretically, and ad hoc approaches, where formulas are tried because they seem to be plausible. Both categories have had some notable successes (Buckley et al. 1996; Gey et al. 1996; Kwok et al. 1997; Robertson et al. 1995). A problem with the formal model approach is that it is very difficult for a model to take into account the wide variety of variables that are thought or known to influence retrieval. A problem with the ad hoc approach is that there is little guidance as to how to deal with specific variables. In Okapi, a complex formal probabilistic model is used to suggest some much simpler formulas. In this paper we present several weighting formulas used in Okapi and provide empirical results that compare these formulas coupled with either word-based or character-based text processing method in terms of their effectiveness for Chinese retrieval.

For the experiments presented in this paper, we use a standard collection of Chinese documents and queries provided by NIST (the National Institute of Standards and Technology). The document collection and the queries have been used by participants in the Text REtrieval Conferences (TREC). TREC is an annual conference organized by NIST, starting in 1992. Its purpose is to support research within the information retrieval community by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies. Evaluation of Chinese information retrieval systems was included at the fifth and sixth TREC conferences (TREC-5 and TREC-6), where a large collection of Chinese documents and two sets of queries (one for TREC-5 and the other for TREC-6) were provided.

This paper is organized as follows. In Section 2 we present Chinese text segmentation methods for document and query processing. Text retrieval based on our probabilistic model is described in Section 3, where two single-unit weighting functions and a number of compound-unit weighting functions are presented. In Section 4 we provide experimental results and performance comparison of using the different document processing methods and the retrieval functions presented in the paper. We conclude the paper in Section 5.

2. CHINESE TEXT SEGMENTATION

2.1. Document Processing

Most modern Chinese words consist of more than one ideographic character, and the number of characters in a word varies. Since a Chinese text is a linear sequence

of nonspaced or equally spaced ideographic characters, we must either apply a Chinese word segmentation method to the queries and documents or index and search in terms of single Chinese characters. There are different requirements on Chinese text segmentation for different applications. For example, machine translation requires correct segmentation of Chinese text according to Chinese syntax. For the purpose of information retrieval, we may not have to segment Chinese text completely correctly.

Word-based segmentation is normally accomplished using a dictionary and one of the three methods. The first is the longest match, for which text is sequentially scanned to match a word dictionary. The longest matched strings are taken as indexing and search tokens, and shorter tokens within the longest matched strings are discarded. Since longer tokens in the dictionary are more specific, longest match will generate fewer tokens with more specific meaning. The second is the shortest match, for which text is sequentially scanned to match the dictionary. The first matched tokens are selected, and the match process resumes from the next character. With the shortest match method, the segmentation process will generate more tokens with less specific meaning. The third is the overlap match, for which tokens generated from the text can overlap each other across the matching boundary.

Single-character-based segmentation is a purely mechanical procedure that segments Chinese texts into single characters. A huge inverted file is usually generated to index documents. In our experiments, we use both word-based and character-based methods to process documents. For the word-based method, we used the longest match algorithm to segment Chinese texts. By applying this algorithm to the Chinese collection with which we conduct experiments, approximately 43.6 million tokens were identified. These segmented tokens are used to index the documents in the collection for the retrieval purpose. For the character-based method, an inverted file of about 1 gigabyte is generated.

2.2. Query Processing

Both character-based and word-based segmentations for query processing have been used independently in our Okapi system. The character-based method uses characters, character pairs, and multicharacter adjacencies as retrieval keywords. Character pairs, and multi character adjacencies are similar to the bigrams and *n*-grams investigated by some other researchers (Chien 1995; Willett 1979). Our word-based method uses similar techniques that allow phrases to contribute to the matching. In this paper we report the experimental results for a word-based method for query processing. The method uses words and word pairs as retrieval keywords. Only pairs of the adjacent segmented terms are regarded as new potential phrases. After words and phrases are extracted from the query text, they are weighted by using an approximation to inverse collection frequency (ICF) (Spark Jones 1979) as follows:

$$w_{qt} = \log \frac{N - n + 0.5}{n + 0.5}$$

where N is the number of indexed documents in the collection and n is the number of documents containing a specific term. These words and phrases are then ranked by the values of their weights multiplied by the within-query frequencies. The top 19 terms are selected as keywords for searching the word index and for searching the character index.

3. RETRIEVAL BASED ON PROBABILISTIC MODEL

After documents and the query are processed, the selected keywords from the query are sent to the retrieval module to find documents that are most relevant to the query. In Okapi, we use a probability-based retrieval model that weights the keywords according to some information obtained from the documents. The keywords are then matched with the documents, which are indexed by characters or segmented words depending on the extraction method used in document processing. For each document that matches a keyword, a retrieval status value (RSV) is calculated by summing up the weights of all the keywords that match with the document. Finally, the documents are ranked by their RSVs and are presented to the user in that ordered sequence. The performance of the retrieval system therefore depends on the weighting method used to weight the keywords. In this section we describe the weighting functions used in our Chinese Okapi system. Later, in the experimental section, we will compare character-based and word-based document processing methods in terms of these weighting functions.

Keyword weighting can be classified into single-unit weighting and compound-unit weighting. A single-unit weighting function is for weighting a single linguistic unit in a keyword. A single linguistic unit is the linguistic unit that is used to build the index of documents. For example, if documents are indexed using words, a word is a single linguistic unit; if documents are indexed using characters, a character is a single linguistic unit. A compound-unit weighting function is for weighting a compound linguistic unit that consists of two or more single units. For example, if the documents are indexed using words, then a phrase is a compound linguistic unit; if the documents are indexed using characters, then both words and phrases are compound units.

3.1. Single-Unit Weighting

BM25 Weighting Function. We used two single-unit weighting functions in the experiments presented in this paper. They are both extended versions of ICF, 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. The first function is called BM25 (Beaulieu 1996), given as follows.

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

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 $k_1 * ((1-b) + b * dl/avdl)$, 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 the *correction factor* and was designed to take into account the length of a document. The value of the correction factor decreases with dl, from a maximum as

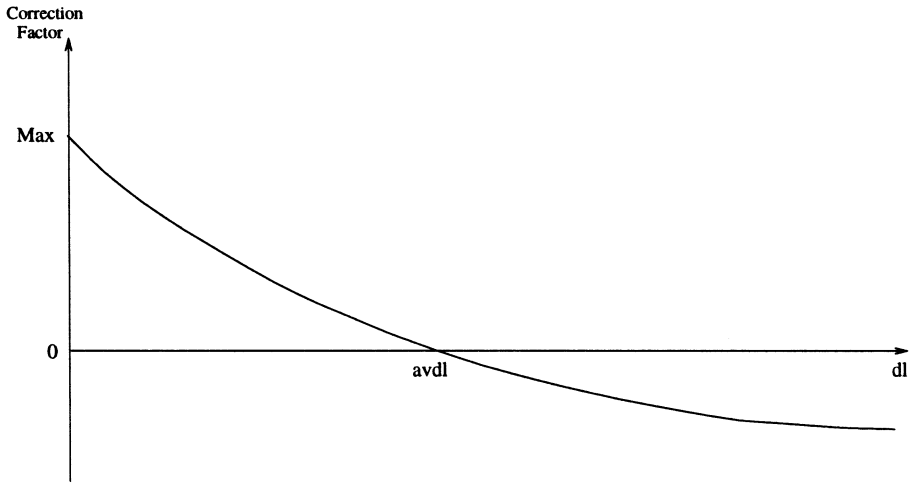

FIGURE 1. Curve for BM25's correction factor with respect to document length.

$dl \rightarrow 0$, through zero, at which $dl = avdl$, and to a minimum as $dl \rightarrow \infty$, as shown in Figure 1. 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.

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.

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 designed on the basis of the following assumptions:

Assumption 1. Too short documents are not relevant.

Assumption 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.

These assumptions can be justified by some statistics in the Chinese TREC-5 and TREC-6 experiments. Table 1 describes the Chinese collection for the TREC-5 and TREC-6 experiments in terms of the minimum, maximum, average length of documents, and total number of documents. Figure 2 depicts the frequency distribution of the documents in the collection with respect to the length of documents. In the figure, the document length is discretized by using equal-interval binning, where the length of each interval is 500 bytes. Table 2 shows the statistics on the *relevant* documents for the TREC-5 queries. Figure 3 depicts the frequency distribution of the TREC-5 *relevant* documents with respect to the document length. The statistics for the TREC-6 relevant documents are shown in Table 3 and Figure 4.

TABLE 1. Whole Data Set for TREC-5

Min. length	0 byte
Max. length	294,056 bytes
Average length	891 bytes
Total number of documents	164,768 documents

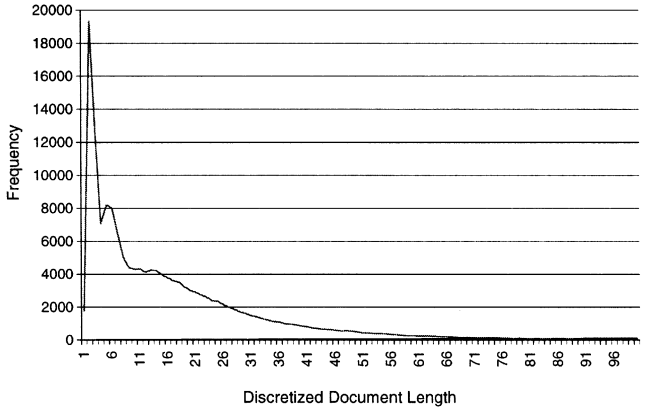

FIGURE 2. Distribution curve for the whole Chinese TREC data set.

TABLE 2. Relevant Data Set for TREC-5

Min. length	58 bytes
Max. length	22,718 bytes
Average number of relevant documents per query	83.9 documents
Average length	1399 bytes
Standard deviation	1675.31 bytes

From these statistics we can observe that

1. The average length of relevant documents for both TREC-5 and TREC-6 queries is longer than the average length of the documents in the whole TREC document collection. A closer look at the documents also revealed that very small documents, such as the title of an article, are usually considered as irrelevant by the human appraisers for the TREC evaluation.
2. On average, only 0.05 to 0.064% of the documents in the collection are relevant for a query. Since a very small portion of the whole documents is relevant, Figure 2 can be roughly regarded as the distribution curve of irrelevant documents.
3. The curve for the correction factor of BM25, shown in Figure 1, does not match relevant document distributions but roughly matches the distribution curve for the whole data set (Figure 2), which means that Figure 1 matches the distribution curve of document length for the irrelevant dataset.

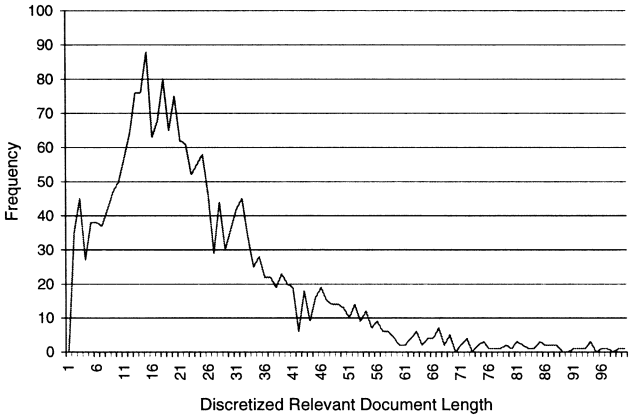

FIGURE 3. Distribution curve for the relevant TREC-5 data set.

TABLE 3. Relevant Data Set for TREC-6

Min. length	60 bytes
Max. length	294,056 bytes
Average number of relevant documents per query	105.6 documents
Average length	1987 bytes
Standard deviation	6194.5 bytes

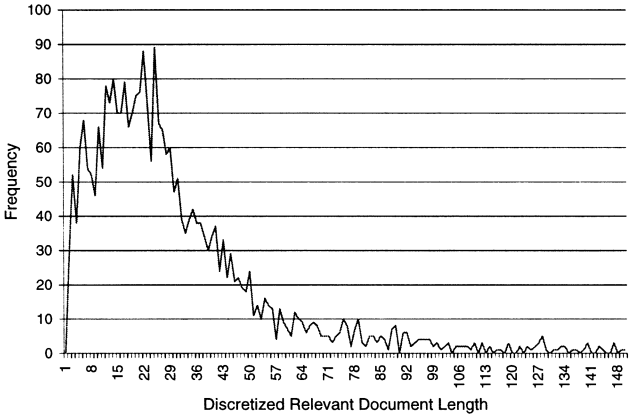

FIGURE 4. Distribution curve for the relevant TREC-6 data set.

Based on these observations, we can conclude that the BM25 correction factor function is suitable for the irrelevant data set but not for the relevant data set. To correct this problem, we define BM26 as follows:

$$w = \frac{(k_1+1)*tf}{K+tf} * \log\frac{N-n+0.5}{n+0.5} * \frac{(k_3+1)*qtf}{k_3+qtf} \oplus \quad k_d * y \tag{2}$$

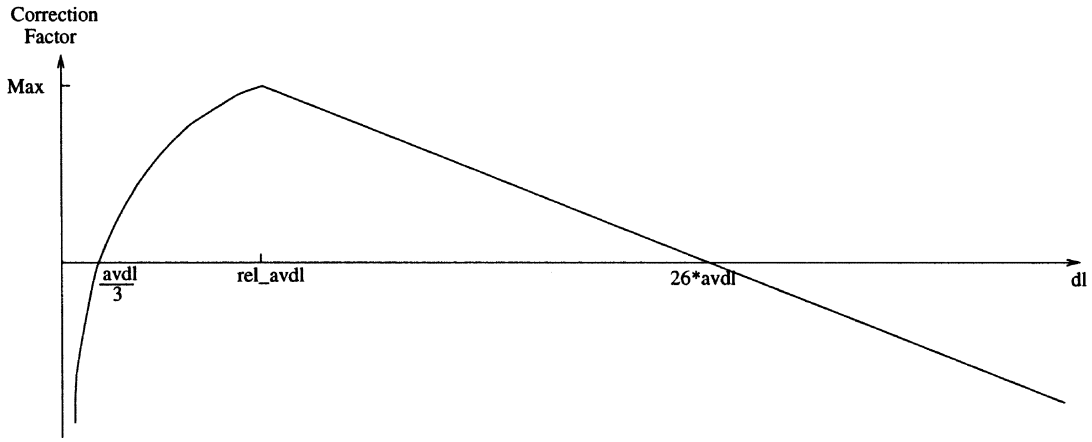

FIGURE 5. Curve for the new correction factor with respect to document length.

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

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

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, and x_1 and x_2 are two parameters to be set.

The difference between BM26 and BM25 is in the y bit of the correction factor. In BM26, y will reach a maximum as $dl \to rel\_avdl$, through zero when $dl = avdl/x_1$ (or $dl = x_2 * avdl$), and to 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 5.

3.2. Compound Unit Weighting

Compound units (phrases or words) contain more information than single units (words or characters). It is reasonable to consider that search for compound units during retrieval is one of the most powerful combination techniques that could improve the retrieval performance. In our Chinese Okapi system, documents are indexed by single units (words or characters), and a query keyword could be either a single unit (words) or a compound unit (words or phrases). Whether a document matches with a compound unit is determined at the search time using position information in the index file. In our system, query keywords may include both compound terms and all or some of their constituent elements, e.g., a word pair and both or one of the member single words, depending on whether all or only some of constituent elements are selected during the query processing. Therefore, both documents containing the compound unit and documents containing any of its member terms or containing all the member terms but not in the phrasal relationship could match with the query. Intuitively, preference should be given to the matches on the compound unit. This preference should be reflected in

designing weighting functions for compound units; i.e., it is reasonable to assume that for a compound unit consisting of two single terms t_1t_2,

$$w(t_1), w(t_2) < w(t_1 \wedge t_2) = w(t_1) + w(t_2) < w(t_1 \ adj \ t_2) \tag{3}$$

where $\wedge$ is the *and* operator and *adj* is the adjacency operator. The equation in the middle represents the usual scoring method for the $\wedge$ (and) operator: The score assigned to a document is the sum of the weights of the matching terms. The assumption is that two adjacent units carry a higher score than two separate terms.

The problem of devising such a method consistent with the probabilistic model generally has not been tackled in text retrieval in English. But for text retrieval in Chinese, the problem is likely to be more serious than it is in English. This would be so in a word-based system, since there are likely to be considerable differences between Chinese speakers as to whether a given combination of characters is considered to be a single word or a phrase. But it is even more serious in a character-based system, where one would want a match on a complete word or phrase to carry a higher score than matches on the component characters.

Suppose that we have a sequence of j adjacent single units $t_1, t_2, \ldots, t_j$ (characters or words) constituting a larger compound unit $t_1t_2 \cdots t_j$ (word or phrase) and that each single unit and the compound unit are included in the selected list of query keywords. Each unit (large or small) has a "natural" weight, given by a single-unit weighting formula (such as BM25 or BM26). Let w_{t_i} be the natural weight for term $t_i (i = 1, \ldots, j)$ and $w_{t_1t_2\cdots t_j}$ be the natural weight for the whole unit $t_1t_2 \cdots t_j$. If both the compound unit and its constituent single units are given weights in the usual fashion, we have

$$w(t_1 \wedge t_2 \wedge \cdots \wedge t_j) = \sum_{i=1}^{j} w_{t_i}$$

$$w(t_1 \ adj \ t_2 \ adj \cdots adj \ t_j) = w_{t_1t_2\cdots t_j} + \sum_{i=1}^{j} w_{t_i}$$

The weight for $w(t_1 \ adj \ t_2 \ adj \cdots adj \ t_j)$ contains both $w_{t_1t_2\cdots t_j}$ and $\sum_{i=1}^{j} w_{t_i}$ because $t_1 \ adj \ t_2 \ adj \cdots adj \ t_j$ implies $t_1 \wedge t_2 \wedge \cdots \wedge t_j$. This natural assignment of weights satisfies the preceding assumption expressed in equation (3). Here, $\sum_{i=1}^{j} w_{t_i}$ can be considered as a "boost weight." Furthermore, for consistency, we also could reduce the scores of those documents which contain the component single units but not the compound unit, e.g., by giving a small negative weight to the logical conjunction of the component units (i.e., reducing $w(t_1 \wedge t_2 \wedge \cdots \wedge t_j)$). However, design of this negative weight with the restriction to satisfy the left inequality $(w(t_1), w(t_2) < w(t_1 \wedge t_2))$ in the assumption in equation (3) is not an easy task. To avoid this difficulty, we can add an extra boost weight to $w(t_1 \ adj \ t_2 \ adj \cdots adj \ t_j)$ and let $w(t_1 \wedge t_2 \wedge \cdots \wedge t_j)$ remain naturally designed. Based on this consideration, we suggest a number of weighting functions that satisfy the condition specified in equation (3). Table 4 gives six of such functions (denoted as $Weight_1$, $Weight_2$,..., and $Weight_6$). In each set of the functions, the formula for $w(t_1 \ adj \ t_2 \ adj \cdots adj \ t_j)$ contains an extra boost weight, such as j^k in $Weight_2$. Among the six formulas, $Weight_1$ and $Weight_5$ are given the biggest extra boost, $Weight_3$ has no extra boost weight, and the others are in between. All these formulas are used in our experiments, each of which is coupled with BM25 or BM26 for single-unit weighting.

TABLE 4. Compound Unit Weighting Methods

Weight Methods	$w(t_1\ adj\ t_2\ adj \cdots adj\ t_j)$	$w(t_1 \wedge t_2 \wedge \cdots \wedge t_j)$
$Weight_1$	$2\sum_{i=1}^{j} w_{t_i} + w_{t_1 t_2 \cdots t_j}$	$\sum_{i=1}^{j} w_{t_i}$
$Weight_2$	$\sum_{i=1}^{j} w_{t_i} + w_{t_1 t_2 \cdots t_j} + j^k$	$\sum_{i=1}^{j} w_{t_i}$
$Weight_3$	$\sum_{i=1}^{j} w_{t_i} + w_{t_1 t_2 \cdots t_j}$	$\sum_{i=1}^{j} w_{t_i}$
$Weight_4$	$\sum_{i=1}^{j} w_{t_i} + w_{t_1 t_2 \cdots t_j} + \log \frac{\#(t_1 \wedge t_2 \wedge \cdots \wedge t_j)}{\#(t_1\ adj\ t_2\ adj \cdots adj\ t_j)}$	$\sum_{i=1}^{j} w_{t_i}$
$Weight_5$	$\sum_{i=1}^{j} w_{t_i} + w_{t_1 t_2 \cdots t_j} \times \log_2 j$	$\sum_{i=1}^{j} w_{t_i}$
$Weight_6$	$\sum_{i=1}^{j} w_{t_i} + w_{t_1 t_2 \cdots t_j} + \frac{\sum_{i=1}^{j} w_{t_i}}{d}$	$\sum_{i=1}^{j} w_{t_i}$

Where $\#(t)$ indicates the number of documents containing the term t and k ($k \in [0, 2]$) and d are tuning constants.

4. EXPERIMENTAL RESULTS AND PERFORMANCE COMPARISON

Extensive experiments have been done to investigate the effect of word-based and character-based document processing on Chinese text retrieval and the effect of different weighting functions and of varying their parameters. A large collection of Chinese documents, which contain 164,768 documents, was used in the experiments. The collection was obtained from the National Institute of Standards and Technology (NIST) for participating in the Text REtrieval Conferences (TREC). Fifty-four Chinese queries (28 for TREC-5 and 26 for TREC-6) were used in our experiments.

In our experiments, the relevance judgments for each query come from the human assessors of NIST. Statistical evaluation was done by means of the latest version 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 Voorhees and Harman, (1997).

4.1. Experimental Results for TREC-5 Queries

In this section we report our test results for the 28 TREC-5 queries. A number of versions of our Chinese Okapi retrieval system are tested, which use different document processing methods and four different compound-unit weighting methods. The single-unit weighting method used here is BM25. We do not use BM26 for the TREC-5 queries because BM26 requires a parameter, the average relevant document length, to be calculated from previous queries, and there are no previous queries with evaluation results for this Chinese document collection. Table 5 illustrates the results in terms of average precision, total number of relevant documents retrieved, R precision, and precision at 100 docs. Average precision, R precision, and precision at 100 docs are the average numbers over the 28 queries, and total number of relevant documents is the summation over the 28 queries of the number of relevant documents in the first 1000 retrieved documents for each query. All the numbers were calculated by the TREC evaluation program. The results in Table 5 indicate that $Weight_2$ is the best

TABLE 5. Results for the TREC-5 Queries

Run	Document Processing	Compound unit Weighting	Average Precision	Total Rel Retrieved	R Precision	Precision at 100 docs
T5w1.BM25	Word	$Weight_1$	0.3691	1995	0.3873	0.3164
T5w2.BM25	Word	$Weight_2$	0.3775	2003	0.3865	0.3189
T5w3.BM25	Word	$Weight_3$	0.3762	2002	0.3860	0.3204
T5w4.BM25	Word	$Weight_4$	0.3773	2005	0.3864	0.3193
T5w5.BM25	Word	$Weight_5$	0.3657	1992	0.3812	0.3164
T5c1.BM25	Character	$Weight_1$	0.3475	2004	0.3611	0.2918
T5c2.BM25	Character	$Weight_2$	0.4126	2056	0.4251	0.3368
T5c3.BM25	Character	$Weight_3$	0.3795	1986	0.3963	0.3189
T5c4.BM25	Character	$Weight_4$	0.3863	2011	0.4017	0.3275
T5c5.BM25	Character	$Weight_5$	0.3507	1992	0.3619	0.2968

compound-unit weighting formula among the five tested formulas for both word-based and character-based Chinese retrieval and also that the character-based method is better than the word-based method with three of the five tested compound unit weighting formulas. In the TREC-5 experiments, the value of k in $Weight_2$ is 0.5.

4.2. Experimental Results for TREC-6 Queries

We also run different versions of Okapi on the 26 TREC-6 Chinese queries. For these queries we use BM26 as the single-unit weighting method, since we can set the average relevant document length parameter (rel_avdl) based on the TREC-5 results. 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 value of k in $Weight_2$ is set to be 1. Table 6 shows the results for using word-based document processing. Table 7 shows the results for using character-based document processing. Figure 6 illustrates in bar plots a comparison of single-unit weighting functions (BM25 and BM26 with different values of parameter k_d) when word-based document processing is used. Comparison of compound-unit weighting methods for word-based document processing is illustrated in Figure 7, in which the last group of bars represents the results for the $Weight_6$ method with different values for parameter d ($d = 2, 10, 20, 50,$ and 100, respectively). Figures 8 and 9 illustrate the same comparisons for character-based document processing. Figure 10 shows in bar plots the comparison of word-based and character-based methods in terms of average precision over the 45 runs described in Tables 6 and 7. In the figure, darker bars represent the results for the character-based method and lighter bars for the word-based method.

In terms of single-unit weighting, both the result from the word-based method (Figure 6) and the result from the character-based method (Figure 8) indicate that BM26 with $k_d > 0$ is better than BM25 (BM26 with $k_d = 0$). In terms of parameter setting for BM26, the results show that the best performance is achieved when k_d is set to be 20 for word-based methods and when k_d is set to 20 or 15 for character-based methods. In terms of compound-unit weighting, the results (see Figures 7 and 9) confirm that $Weight_2$ is the best compound-unit weighting formula for both words and character-based methods. This is more obvious in the results for character-based methods. We can say that character-based methods are more sensitive to the compound

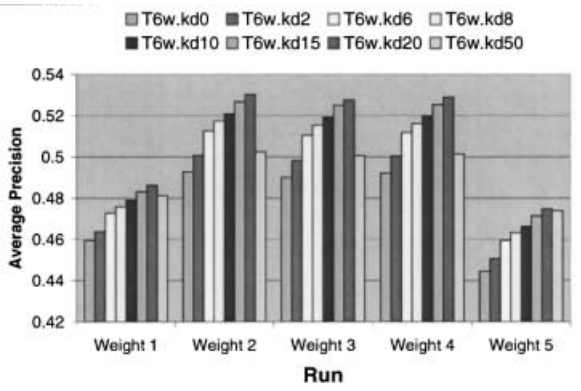

FIGURE 6. Comparison of single-unit weighting functions using word methods.

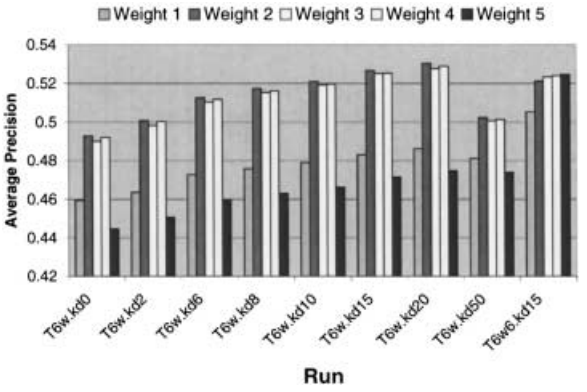

FIGURE 7. Comparison of compound-unit weighting functions using word methods.

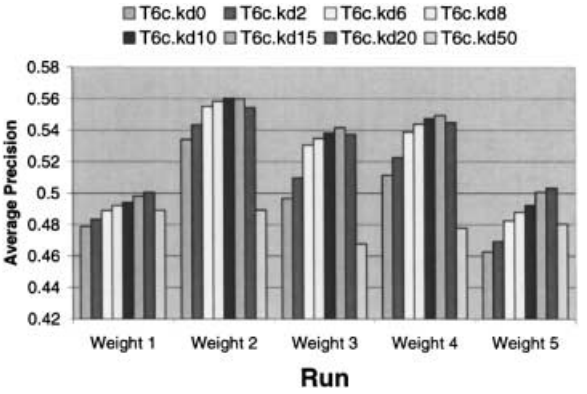

FIGURE 8. Comparison of single-unit weighting functions using character methods.

weighting functions. In addition, the results (see Figure 10) consistently show that the character-based method is better than the word-based method except when k_d is set to 50. Table 8 shows the improvements of BM26 over BM25 and character-based over word-based methods, in which $Weight_2$ is used for compound unit weighting.

TABLE 6. Results for TREC-6 Queries with the Word-Based Approach

	Run	Weighting Method	k_d	Average Precision	Total Rel Retrieved	R Precision	Precision at 100 docs
1	T6w1.kd0	$Weight_1$	0	0.4594	2516	0.4740	0.4519
2	T6w2.kd0	$Weight_2$	0	0.4927	2546	0.5127	0.4773
3	T6w3.kd0	$Weight_3$	0	0.4900	2542	0.5106	0.4746
4	T6w4.kd0	$Weight_4$	0	0.4921	2548	0.5124	0.4777
5	T6w5.kd0	$Weight_5$	0	0.4446	2502	0.4635	0.4385
6	T6w1.kd2	$Weight_1$	2	0.4636	2528	0.4772	0.4585
7	T6w2.kd2	$Weight_2$	2	0.5008	2557	0.5197	0.4842
8	T6w3.kd2	$Weight_3$	2	0.4982	2551	0.5202	0.4842
9	T6w4.kd2	$Weight_4$	2	0.5004	2558	0.5200	0.4862
10	T6w5.kd2	$Weight_5$	2	0.4507	2513	0.4693	0.4492
11	T6w1.kd6	$Weight_1$	6	0.4727	2547	0.4883	0.4688
12	T6w2.kd6	$Weight_2$	6	0.5126	2577	0.5294	0.4988
13	T6w3.kd6	$Weight_3$	6	0.5104	2574	0.5278	0.4977
14	T6w4.kd6	$Weight_4$	6	0.5118	2580	0.5287	0.5004
15	T6w5.kd6	$Weight_5$	6	0.4595	2530	0.4795	0.4608
16	T6w1.kd8	$Weight_1$	8	0.4758	2517	0.4911	0.4708
17	T6w2.kd8	$Weight_2$	8	0.5174	2590	0.5288	0.4992
18	T6w3.kd8	$Weight_3$	8	0.5154	2588	0.5282	0.5000
19	T6w4.kd8	$Weight_4$	8	0.5162	2588	0.5277	0.4996
20	T6w5.kd8	$Weight_5$	8	0.4631	2532	0.4839	0.4631
21	T6w1.kd10	$Weight_1$	10	0.4789	2548	0.4923	0.4738
22	T6w2.kd10	$Weight_2$	10	0.5209	2593	0.5309	0.5027
23	T6w3.kd10	$Weight_3$	10	0.5192	2592	0.5296	0.5035
24	T6w4.kd10	$Weight_4$	10	0.5198	**2595**	0.5309	0.5027
25	T6w5.kd10	$Weight_5$	10	0.4662	2533	0.4868	0.4658
26	T6w1.kd15	$Weight_1$	15	0.4831	2548	0.4939	0.4765
27	T6w2.kd15	$Weight_2$	15	0.5267	2589	0.5331	0.5023
28	T6w3.kd15	$Weight_3$	15	0.5251	2589	0.5328	0.5031
29	T6w4.kd15	$Weight_4$	15	0.5253	2583	0.5308	0.5019
30	T6w5.kd15	$Weight_5$	15	0.4714	2533	0.4872	0.4715
31	T6w1.kd20	$Weight_1$	20	0.4862	2534	0.4924	0.4785
32	T6w2.kd20	$Weight_2$	20	**0.5303**	2575	**0.5342**	0.5035
33	T6w3.kd20	$Weight_3$	20	0.5276	2575	0.5320	0.5038
34	T6w4.kd20	$Weight_4$	20	0.5289	2575	0.5321	0.5027
35	T6w5.kd20	$Weight_5$	20	0.4748	2522	0.4905	0.4769
36	T6w1.kd50	$Weight_1$	50	0.4812	2429	0.4928	0.4831
37	T6w2.kd50	$Weight_2$	50	0.5024	2415	0.5162	0.4938
38	T6w3.kd50	$Weight_3$	50	0.5006	2408	0.5147	0.4923
39	T6w4.kd50	$Weight_4$	50	0.5013	2410	0.5133	0.4931
40	T6w5.kd50	$Weight_5$	50	0.4739	2427	0.4939	0.4765
41	T6w6.kd15.d2	$Weight_6(d = 2)$	15	0.5053	2566	0.5111	0.4931
42	T6w6.kd15.d10	$Weight_6(d = 10)$	15	0.5212	2584	0.5314	0.5038
43	T6w6.kd15.d20	$Weight_6(d = 20)$	15	0.5233	2586	0.5311	**0.5050**
44	T6w6.kd15.d50	$Weight_6(d = 50)$	15	0.5241	2587	0.5319	0.5031
45	T6w6.kd15.d100	$Weight_6(d = 100)$	15	0.5247	2588	0.5336	0.5031

TABLE 7. Results for TREC-6 Queries with the Character-based Approach

	Run	Weighting Method	k_d	Average Precision	Total Rel Retrieved	R Precision	Precision at 100 docs
1	T6c1.kd0	$Weight_1$	0	0.4789	2550	0.4767	0.4704
2	T6c2.kd0	$Weight_2$	0	0.5341	2637	0.5416	0.5108
3	T6c3.kd0	$Weight_3$	0	0.4967	2537	0.5175	0.4812
4	T6c4.kd0	$Weight_4$	0	0.5113	2558	0.5244	0.4923
5	T6c5.kd0	$Weight_5$	0	0.4627	2493	0.4666	0.4458
6	T6c1.kd2	$Weight_1$	2	0.4833	2562	0.4813	0.4738
7	T6c2.kd2	$Weight_2$	2	0.5434	2653	0.5482	0.5158
8	T6c3.kd2	$Weight_3$	2	0.5096	2565	0.5279	0.4942
9	T6c4.kd2	$Weight_4$	2	0.5227	2568	0.5356	0.5035
10	T6c5.kd2	$Weight_5$	2	0.4693	2504	0.4735	0.4504
11	T6c1.kd6	$Weight_1$	6	0.4891	2566	0.4875	0.4769
12	T6c2.kd6	$Weight_2$	6	0.5551	**2655**	0.5505	0.5169
13	T6c3.kd6	$Weight_3$	6	0.5306	2575	0.5412	0.5077
14	T6c4.kd6	$Weight_4$	6	0.5389	2581	0.5504	0.5104
15	T6c5.kd6	$Weight_5$	6	0.4825	2516	0.4884	0.4619
16	T6c1.kd8	$Weight_1$	8	0.4921	2573	0.4864	0.4792
17	T6c2.kd8	$Weight_2$	8	0.5582	2647	0.5518	0.5177
18	T6c3.kd8	$Weight_3$	8	0.5348	2569	0.5404	0.5035
19	T6c4.kd8	$Weight_4$	8	0.5439	2577	0.5488	0.5135
20	T6c5.kd8	$Weight_5$	8	0.4880	2520	0.4965	0.4685
21	T6c1.kd10	$Weight_1$	10	0.4942	2574	0.4894	0.4800
22	T6c2.kd10	$Weight_2$	10	**0.5603**	2647	0.5544	0.5208
23	T6c3.kd10	$Weight_3$	10	0.5383	2560	0.5422	0.5058
24	T6c4.kd10	$Weight_4$	10	0.5476	2573	0.5526	0.5154
25	T6c5.kd10	$Weight_5$	10	0.4925	2515	0.5001	0.4742
26	T6c1.kd15	$Weight_1$	15	0.4981	2575	0.4956	0.4808
27	T6c2.kd15	$Weight_2$	15	0.5599	2621	**0.5613**	**0.5227**
28	T6c3.kd15	$Weight_3$	15	0.5417	2546	0.5494	0.5131
29	T6c4.kd15	$Weight_4$	15	0.5494	2566	0.5548	0.5173
30	T6c5.kd15	$Weight_5$	15	0.5007	2511	0.5106	0.4804
31	T6c1.kd20	$Weight_1$	20	0.5005	2570	0.4960	0.4781
32	T6c2.kd20	$Weight_2$	20	0.5545	2590	0.5540	0.5181
33	T6c3.kd20	$Weight_3$	20	0.5374	2518	0.5466	0.5096
34	T6c4.kd20	$Weight_4$	20	0.5450	2539	0.5488	0.5145
35	T6c5.kd20	$Weight_5$	20	0.5032	2491	0.5119	0.4800
36	T6c1.kd50	$Weight_1$	50	0.4892	2491	0.4985	0.4823
37	T6c2.kd50	$Weight_2$	50	0.4893	2325	0.5119	0.5004
38	T6c3.kd50	$Weight_3$	50	0.4677	2225	0.4995	0.4788
39	T6c4.kd50	$Weight_4$	50	0.4779	2256	0.5036	0.4850
40	T6c5.kd50	$Weight_5$	50	0.4805	2349	0.5037	0.4742
41	T6c6.kd15.d2	$Weight_6(d = 2)$	15	0.5144	2618	0.5102	0.4873
42	T6c6.kd15.d10	$Weight_6(d = 10)$	15	0.5421	2626	0.5387	0.5073
43	T6c6.kd15.d20	$Weight_6(d = 20)$	15	0.5471	2611	0.5460	0.5108
44	T6c6.kd15.d50	$Weight_6(d = 50)$	15	0.5484	2608	0.5474	0.5142
45	T6c6.kd15.d100	$Weight_6(d = 100)$	15	0.5488	2605	0.5493	0.5150

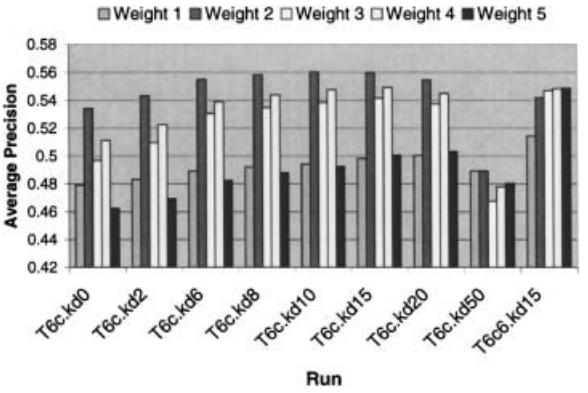

FIGURE 9. Comparison of compound-unit weighting functions using character methods.

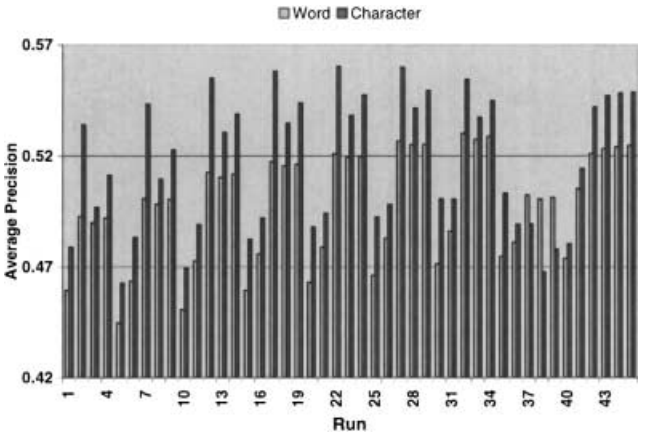

FIGURE 10. Comparison of character and word methods.

TABLE 8. Results Comparison

Run	k_d	Indexing Method	Average Precision
T6w2.kd0 (BM25)	0	*word*	0.4927
T6c2.kd0 (BM25)	0	*character*	0.5341 (+8.40%)
T6w2.kd10 (BM26)	10	*word*	0.5209 (+5.72%)
T6c2.kd10 (BM26)	10	*character*	0.5603 (+13.72%)

4.3. Comparison with Other TREC Participating Systems

To see how our system performs, we compare our results with the automatic run results from other systems participating in TREC-5 and TREC-6 experiments. Since we only have 19 TREC-5 queries' evaluation results for other participating systems, our comparison on the TREC-5 queries is based on these 19 queries. The comparison on TREC-6 is based on all the TREC-6 queries. Figure 11 shows the comparison on TREC-5, and Figure 12 on TREC-6. In both figures we includes the best automatic run from almost every participating institution. These runs are compared with two of

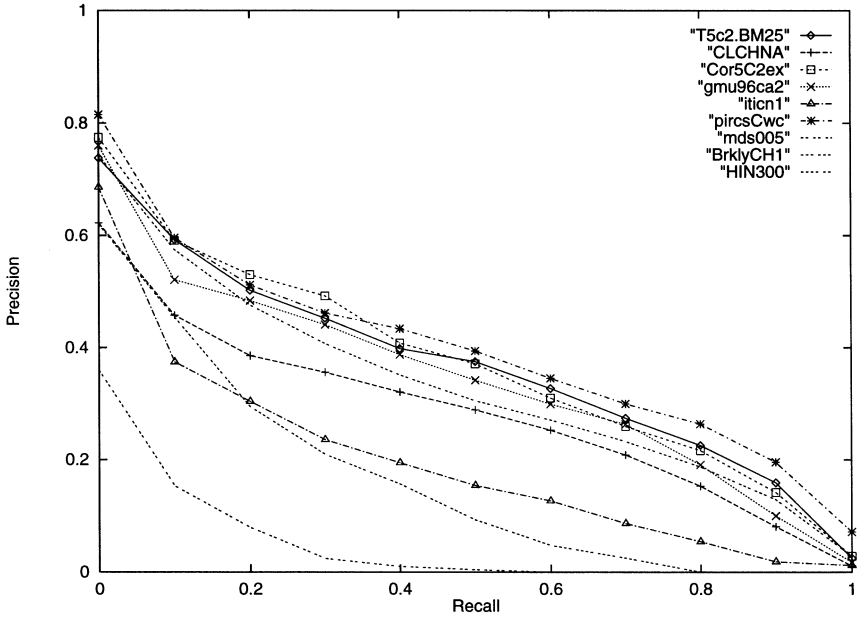

FIGURE 11. Precision-recall curves for some automatic runs at TREC-5.

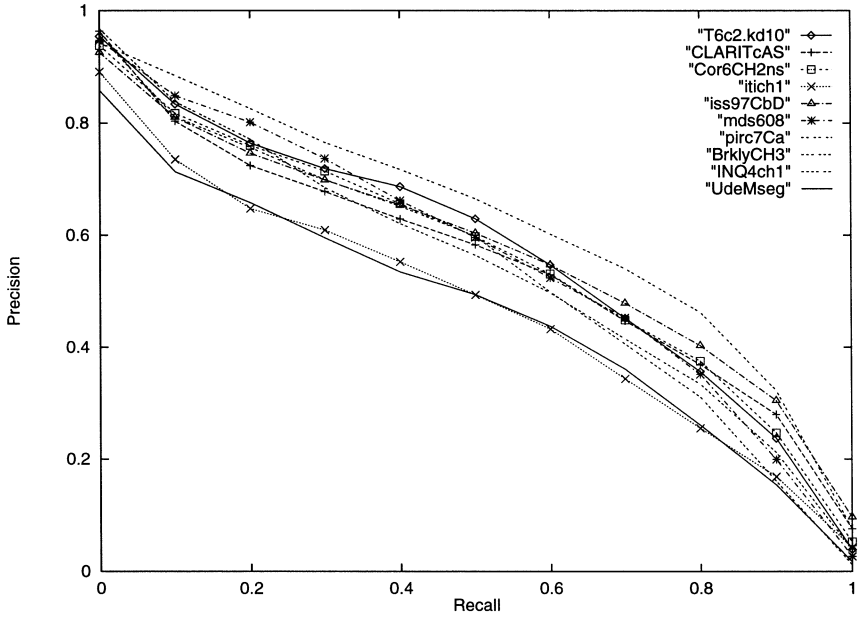

FIGURE 12. Precision-recall curves for some automatic runs at TREC-6.

our runs (T5c2.BM25 using the TREC-5 queries and T6c2.kd10 using the TREC-6 queries). From these performance statistics we can see that our results compare well with the best reported at TREC. In terms of average precision, the results at TREC-5 range from 0.027 to 0.38, with just two systems giving results better than 0.35, and our

official result (Beaulieu et al. 1996) from T5c2.BM25 is 0.3541. The average precision results at TREC-6 range from 0.34 to 0.62, with four systems giving results better than 0.55, and our latest result from T6c2.kd10 is 0.5603. Detailed comparison of these runs can be found in Huang et al. (1999).

5. CONCLUSION

We have presented and evaluated word-based and character-based text segmentation methods for Chinese text processing, two single-unit weighting methods (BM25 and BM26), and a number of compound-unit weighting methods for text retrieval. The evaluation results have demonstrated that character-based document processing is better than the word-based approach for Chinese text retrieval using probabilistic models. As to single-unit weighting, the results indicate that using BM26 weighting function makes a significant positive contribution to the quality of retrieval compared with using BM25. Concerning the use of compound-unit weighting methods, we can draw a conclusion that the method of $Weight_2$ is the best among the six tested methods in terms of average precision. The results for compound-unit weighting also indicate that the extra boost weight in the formulas for $w(t_1\ adj\ t_2\ adj \cdots adj\ t_j)$ plays an important role in the performance of the retrieval system. Neither a big extra boost weight as in $Weight_1$ and $Weight_5$ nor a too small extra boost as in $Weight_3$ that has zero amount of extra boost leads to the best results. A moderate extra boost weight such as the extra boost weights in $Weight_2$ and $Weight_4$ produces better retrieval results. More experiments and analyses need to be done in the future to confirm our findings in this paper and to establish the best form of the extra boost weight for compound unit weighting.

ACKNOWLEDGMENTS

This research was supported by an ORS award from the Committee of Vice-Chancellors and Principals of United Kingdom and a Centenary Scholarship from City University. We would like to thank the anonymous reviewers for their valuable comments. We also would like to thank University of Waterloo and University of Regina for providing computer facilities for conducting the experiments reported in the paper.

REFERENCES

BEAULIEU, M. M., M. GATFORD, X. HUANG, S. E. ROBERTSON, S. WALKER, and P. WILLIAMS. 1996. Okapi at TREC-5. *In* Proceedings of the Fifth Text REtrieval Conference (TREC-5), Gaithersburg. *Edited by* D. K. Harman. NIST Special Publication, pp. 143–166.

BUCKLEY, C., A. SINGHAL, and M. MITRA. 1996. Using query zoning and correlation within SMART: TREC-5. *In* Proceedings of the Fifth Text REtrieval Conference (TREC-5), Gaithersburg. *Edited* by D. K. Harman. NIST Special Publication, pp. 105–118.

CHEN, G. 1992. On single Chinese character retrieval system. Journal of Information, **11**(1):11–18 (in Chinese).

CHIEN, L. F. 1995. Fast and quasi-natural language search for gigabits of Chinese texts. SIGIR, **95**:112–120.

CROFT, W. B., and D. T. HARPER. 1979. Using probabilistic models of document retrieval without relevance information. Journal of Documentation, **35**(4):285–295.

GEY, F., A. CHEN, J. HE, L. XU, and J. MEGGS. 1996. Term importance, Boolean conjunct training, negative terms, and foreign language retrieval: probabilistic algorithms at TREC-5. *In* Proceedings

of the Fifth Text REtrieval Conference (TREC-5), Gaithersburg. *Edited by* D. K. Harman. NIST Special Publication, pp. 181–190.

HUANG, X., and S. E. ROBERTSON. 1997. Application of probabilistic methods to Chinese text retrieval. Journal of Documentation, 53(1):74–79.

HUANG, X., S. E. ROBERTSON, N. CERCONE, and A. AN. 1999. The probability-based Chinese text processing and retrieval. *In* Proceedings of the Conference of Pacific Association for Computational Linguistics (PACLING'99). University of Waterloo, Waterloo, Canada, pp. 223–235.

KWOK, K. L., J. H. FRUNDELD, and J. H. XU. 1997. TREC-6 English and Chinese retrieval experiments using PIRCS. *In* Proceedings of the Sixth Text REtrieval Conference (TREC-6), Gaithersburg. *Edited by* D. K. Harman. NIST Special Publication, pp. 207–114.

ROBERTSON, S. E., and K. SPARCK JONES. 1976. Relevance weighting of search terms. Journal of the American Society for Information Science, **27**:129–146.

ROBERTSON, S. E., S. WALKER, and M. HANCOCK-BEAULIEU. 1995. Large test collection experiments on an operational, interactive system: OKAPI at TREC. Information Processing and Management, **31**(3):345–360.

SPARCK JONES, K., 1979. Search relevance weighting given little relevance information. Journal of Documentation, **35**(1):30–48.

VOORHEES, E. and D. HARMAN. 1997. Overview of the Sixth Text REtrieval Conference (TREC-6). *In* Proceedings of the Sixth Text REtrieval Conference (TREC-6), Gaithersburg. NIST Special Publication, 1–24.

WILLETT, P. 1979. Document retrieval experiments using indexing vocabularies of varying size: II. Hashing, truncation, digram and trigram encoding of index terms. Journal of Documentation, **35**(4):296–305.

WU, Z., and G. TSENG. 1993. Chinese text segmentation for text retrieval: Achievement and Problems. Journal of the American Society for Information Science, 44(9):532–541.