Effective and efficient document ranking without using a large lexicon

OGAWA Yasushi
ogawa@ic.rdc.ricoh.co.jp

Information and Communication R&D Center, RICOH Co., Ltd.
3-2-3 Shin-yokohama, Kouhoku-ku, Yokohama, 222 JAPAN

Abstract

Although a word-based method is commonly used in document retrieval, it cannot be directly applicable to languages that have no obvious word separator. Given a lexicon, it is possible to identify words in documents, but a large lexicon is troublesome to maintain and makes retrieval systems large and complicated. This paper proposes an effective and efficient ranking that does not use a large lexicon; words need not be identified during document registration because a character-based signature file is used for the access structure. A user request, during document retrieval, is statistically analyzed to generate an appropriate query, and the query is evaluated efficiently in a word-based manner using the character-based index. We also propose two optimizing techniques to accelerate retrieval.

1 Introduction

Best match retrieval, which ranks retrieved documents in order of their relevance to the user request, is more effective than the conventional exact match retrieval [5][8][30]. Most retrieval models, such as the vector space and probabilistic models, compute the relevance values of documents by using three term frequencies: the term's *document frequency*, the number of documents containing the term; the *in-query frequency*, the number of times the term occurs in the query; and the *in-document frequency*, the number of times the term occurs in the target document.

As the size of document databases has increased, the efficient implementation of ranking models has been extensively studied recently[5][30]. Most implementations use a word-based inverted file, but word-based indexing is not easily applicable to some Asian languages such as Chinese and Japanese; a large lexicon is necessary for identifying words in texts because word boundaries in those languages are not indicated by spaces. Such a lexicon requires troublesome maintenance and makes retrieval systems larger and more complicated.

Therefore we chose not to use a large lexicon in designing a retrieval system for Japanese documents.[1] As for retrieval methods without using a large lexicon, character-based or n-gram-based[2] methods have been proposed by some researchers [3][6][16][28]. Those methods ignore word boundaries, and rank documents according to frequencies of characters (or n-grams) instead of words. They were shown to be effective, but our experimental results[19] suggest that they were still inferior to a word-based method because they miss some word-level semantics. Therefore, we developed a new method which combines the two approaches by adopting a character-based signature file in indexing and a word-based query evaluation[19][22].

A processing flow is illustrated in Figure 1. During registration, in addition to the insertion of new documents in the document file, the character-based signature file is updated; word segmentation is not required. During retrieval, a user's natural language request is processed to generate a query, and a ranked list of retrieved documents is returned. The query generation uses a newly developed statistical word segmentation method to process the request, so a large lexicon is not required. In query evaluation, the candidate documents are obtained by using the signature file, and relevance values are computed in a word basis only for a small portion of candidates to identify the top-ranked documents.

Moreover, we propose two optimizing techniques to speed up retrieval. One reduces the number of document accesses by relaxing a condition that determines the top-

Proceedings of the 22nd VLDB Conference
Mumbai(Bombay), India, 1996

[1]General issues concerning Japanese IR are summarized by Fujii[6].

[2]N-gram-based indexing uses n-grams, overlapping series of n successive characters, as indexing units. Because n-gram-based indexing is considered as an extension of character-based indexing, we include, in the following, the former in the latter.

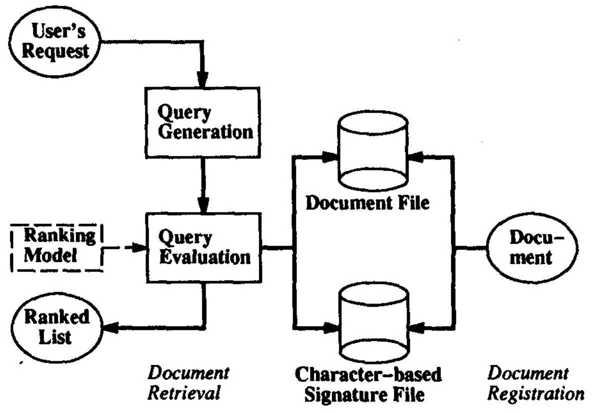

Figure 1: Overview of processing flow.

ranked documents. The other accelerates retrieval by limiting the number of query terms used to determine ranking candidates.

This paper is organized as follows. The next section briefly describes the character-based signature file used in our system. Section 3 explains query generation without using a large lexicon. Sections 4 and 5 detail a query evaluation method and its optimization techniques. Section 6 presents the results of an experiment that evaluated our proposals with regard to retrieval effectiveness and efficiency.

2 Character-based Signature File

Character-based indexing is preferred in kanji-based Asian languages, since there is no need to identify words [6][16][28][29]. Because in-document frequencies of characters are much larger than those of words, a character-based index tends to become large. A signature file is a kind of indexing methods, in which a document is described by a fixed-length bitstring or *signature*, which has less space overhead than an inverted file [5]. Thus a character-based signature file is especially widely used in kanji-based languages [3][7][13][17].

In a character-based signature file, a signature is computed not from words but from characters. For example, a signature for the text "…半導体の生産…"(the production of semiconductors) is computed from all the characters included in the text, instead of from the words "半導体"(semiconductors), "の"(of), and "生産"(production) as shown in Figure 2.

The signature file used in our retrieval system has the following features:

- A document/query signature is obtained from the bitwise OR of signatures computed from single characters and from character pairs [20]. To reduce false drops, we use more than one hash functions, each of which is used for a certain character class [11] and makes the document counts among the signature bits nearly the same [21].

- To attain faster retrieval, bitmap data is organized in a bit-sliced manner and each bit slice is compressed using the Exp-Golomb algorithm [11].

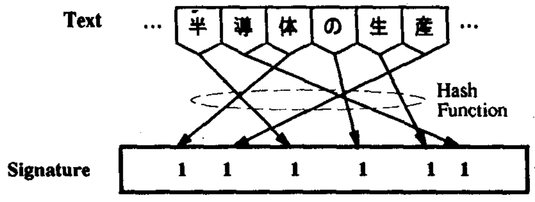

Figure 2: Character-based signature.

3 Query Generation

3.1 Processing Flow

Our retrieval system allows users to write a request in natural language. The main subject of query generation is how to identify appropriate words in a user request without a large lexicon.

Japanese has a two word forms: *simple words* and *compound words* composed of several simple words [6][20]. There are in Japanese several classes of characters (kanji, katakana and hiragana) each of which has different functions and which sometimes indicates word boundaries [6][24]. Some types of words can therefore be identified by using a closed lexicon that consists of a small number functional words [12][24]. Component simple words in a compound word, however, generally cannot be identified without a large lexicon. However, since the meaning of a compound word can be expressed using other phrases or passages made up of its component simple words, simple words need to be identified in retrieval. Thus a statistical method of segmenting compound words has been developed. It will be explained later.

A user request is at first segmented into compound words by using a closed lexicon parser called QJP [12]. The size of the lexicon used in QJP is about 5000 words (50 KB), mainly composed of functional words such as particles. Then, words which are not particles or auxiliary verbs are further segmented into their component words.

3.2 Statistical Compound Word Segmentation

To segment a compound word into component words, breaks between the component words have to be found. For a given word, the probability that a given character pair is a break between two component words is computed for every adjacent character pair in it. Then the word is segmented at points where the probabilities are greater than a segmentation threshold, P.

To compute the segmentation probability of a character pair, it has been assumed that the segmentation probability is a product of the tail probability of the former character and the head probability of the latter character. Here, the character's head or tail probability are probabilities that a given character appears at the head or tail of words. These values can be compiled automatically by counting the number of times a character occurs in a large corpus. It should be noted that the data size is small, i.e., proportional to the size of the character set.

For example, if the tail probability of "平" is 0.20 and the head probability of "和" is 0.09, the segmentation probability of a character pair "平和", which forms word "peace," becomes $0.20 \times 0.09 = 0.018$. Given a compound word "平和維持活動"(peace keeping operation), segmentation probabilities between all character pairs are computed in the same way. If the result is given as shown below, the word is correctly segmented into "平和"(peace), "維持"(keeping), and "活動"(operation) by setting $P = 0.1$.

平 0.018 和 0.104 維 0.047 持 0.2265 活 0.0290 動

As you may understand from this example, the threshold value controls segmentation and therefore greatly influences both the effectiveness and efficiency of retrieval. That is, when the value is too small, words are divided into many small parts. Effectiveness thus increases because fewer documents are missed, but retrieval takes longer because a query includes more words, which generate many more documents to be ranked.

4 Query Evaluation

4.1 Evaluation Procedure using Upper Bounds

When one uses a word-based signature file, because there is no room to store the in-document frequencies, retrieval effectiveness deteriorated[25]. One attempt to solve this replaces the term's in-document frequency by the number of logical blocks, in a given document, that contain the term[4]. Another uses several signature files, each of which corresponds to a certain in-document frequency[31]. Those methods are not true solutions, however, because the former is even less effective and the latter entails a large space overhead.

Yet another method uses the signature file to determine the documents to be ranked, and the final relevance values are computed using a non-inverted description file that records the frequencies of terms[15][26]. The introduction of the description file, however, requires another disk access, which might decrease the retrieval speed. Considering observations that users generally assess only a limited number of top-ranked documents in retrieval results [4][32], Knaus and Schäuble focused on speeding up the identification of top ranking documents. In their method, the upper bound of the relevance values is computed for all retrieved documents using the signature file. There upper bounds are, as shown below, used to determine the order of computing the final relevance values, and to judge whether the top-ranked documents have been identified.

> Let $s(D)$ be the upper bound of the relevance value, and $o(l)$ be the identifier of the l-th preranked document. Now there is the relationship "$r(D_i) \geq s(D_j) \Rightarrow r(D_i) \geq r(D_j)$" for documents D_i and D_j. Given an upper bound list for the ranking candidates, when the top l documents in the upper bound list have been evaluated, the $(l+1)$st document has the largest upper bound among documents that have not been evaluated. Thus, one can determine the final ranks of documents belonging to the document set:
>
> $$R_l = \{D_i | r(D_i) \geq s(D_{o(l+1)})\}. \quad (1)$$
>
> Therefore, to determine the top k documents of the final ranking, it is enough to evaluate documents until $|R_l| \geq k$ ($|X|$ denotes the number of items in set X). □

The following procedure implements the above idea in two distinct phases. One has first to prepare a formula that computes an upper bound without using in-document frequencies. In the *pre-ranking phase*, candidate documents that contain at least one query term are identified using the signature file, and their upper bounds are computed. In the *reevaluation phase*, the candidate documents are, in the order of their upper bounds, reevaluated one-by-one until the top k documents are fixed. In each iteration, the exact relevance value is computed using the query terms' in-document frequencies obtained from the description file. The iteration terminates immediately when $|R_l| \geq k$.

Figure 3 illustrates the reevaluation phase. In this figure, each document is identified by a letter of the alphabet, and the dark hatched bars represent documents whose final ranks are determined. In the l-th iteration, the l-th document is evaluated and the all exact relevance values of the evaluated documents are compared with the upper bound of the $(l+1)$st document. At the 5th step, for example, final ranking is determined for three documents b, d, and e. If $k = 3$, the iteration can be stopped at the 5th step.

It should be noted that the above procedure is quite different from other query evaluation methods using upper bounds[2][27][32] in the complete separation of the upper bound and the relevance value computations. This separation comes from the fact that in-document frequencies cannot be obtained from the signature file index.

4.2 Modifications for Character-based Indexing

The above procedure is developed for word-based signature files, and cannot be directly applied to the character-based

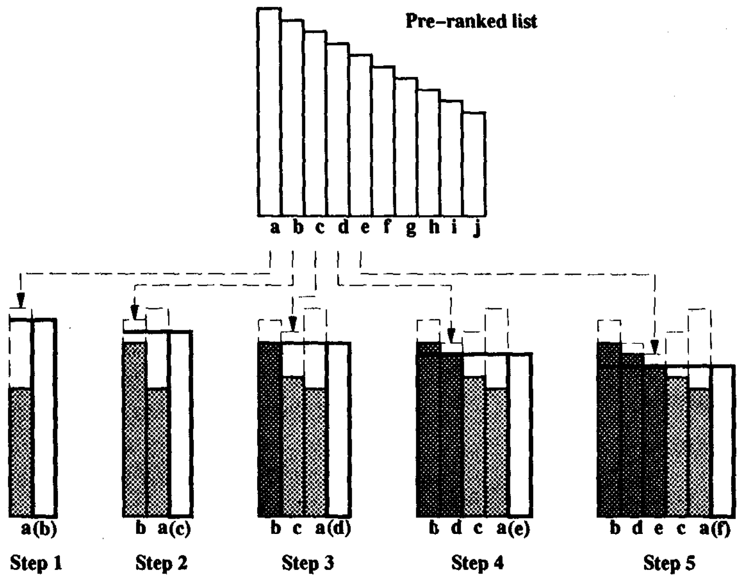

Figure 3: Reevaluation process.

ones. Therefore, we have made the following modifications to the upper bound method.

The first modification is in the way the document frequencies are obtained. In the character-based organization, because there is no word table (lexicon), there is no space to store the document frequencies in the signature file. To solve this problem, the term's document frequency is replaced by the number of retrieved documents that are judged to contain the term by using the signature file. Although the document frequency is greater than its real value because of false drops, the effect on the ranked results is negligible as the document frequencies are taken logarithm in calculating the relevance value.

The second modification is that the non-inverted description file is not used. The reason is that because terms are not identified at document registration in our implementation, in-document frequencies are not established and thus the description file cannot be created. On the other hand, the original method is modified to directly access documents in order to establish the term frequencies at retrieval. This modification is also useful in reducing space overhead because the description file is now not used.

The final modification is that the processing strategy in the pre-ranking phase is changed. In the original method [26], the evaluation is carried out in the "document at a time" way [27], where the upper bounds are computed in sequential order of document identifier, because the signature file is organized sequentially to increase updating efficiency. But, because the file is organized in a bit-sliced manner in order to maximize retrieval performance, the evaluation is processed in the "term at a time" way [27], where terms are picked up from the query one-by-one and the upper bounds for documents containing the term are updated simultaneously.

4.3 Ranking Model

We used Robertson's model, a probabilistic ranking model which assumes that a word's frequency has a Poisson distribution[23]. We adopted this model for two reasons; (1) It achieved good performance in TREC, a large-scale IR system evaluation contest for English document collections [8]. (2) Most retrieval models require establishing in-document frequencies for all the terms in a target document to normalize them, but Robertson's model does not. It is a very attractive feature for our method that no word is identified at document registration.

To compute the relevance value, Robertson presented a basic model BM15 and another model BM11 that incorporates the effect of the document length in normalizing the term's in-document frequencies. [3] In our experiment, the relevance value $r(D)$ of a document D is computed by the following formula that merges both model.

$$r(D) = \sum_i \log(\frac{N}{df_i}) \cdot \frac{qf_i}{Kq + qf_i} \cdot \frac{tf_i}{Kd(\lambda \frac{L}{L_{ave}} + (1-\lambda)) + tf_i}, \quad (2)$$

where df_i is the document frequency of term t_i, qf_i is the t_i's in-query frequency, and tf_i is the t_i's in-document frequency. Kq and Kd are constants for normalizing qf_i and tf_i, and N is the number of documents in the collection. L and L_{ave} are the length of the target document and the average document length in the collection, and λ is a constant for control of the effect of the document length.

[3]There is, in formulas for the BM11 and BM15 models, another component related to the document length, but it has a negative (for BM11) or only a slight positive effect (for BM15). Therefore we ignore this component.

Note that $\lambda = 0$ and 1 correspond to the BM15 and BM11 models.

To compute the upper bounds of the relevance values, we have established the formula,

$$s(D) = \sum_i \log(\frac{N}{df_i}) \cdot \frac{qf_i}{Kq + qf_i} \cdot \delta_i. \qquad (3)$$

where δ_i is 1 if the signature file judges that the term t_i exists in the document. Because $tf_i/(Kd + tf_i) \leq \delta_i$, Formula (3) gives the upper bound of the relevance value given by Formula (2).

5 Optimizing Techniques

5.1 Relaxing the Stop Condition

In the above query evaluation method, the rank of a given document D_i is determined when the condition, $r(D_i) \geq s(D_{o(l+1)})$, is satisfied. However, this condition seems too strict, because there sometimes exists a document D_i whose final ranking can be determined, i.e. $r(D_i) \geq r(D_{o(l+1)})$, but which does not fulfill the stop condition, i.e. $r(D_i) < s(D_{o(l+1)})$.

Let's consider a relaxed stop condition for the l-th iteration, $r(D) \geq \alpha \cdot s(D_{o(l+1)})$ where $\alpha(0 \leq \alpha \leq 1)$, and denote a document set that satisfies the new condition by

$$R'_l = \{D_i | r(D_i) \geq \alpha \cdot s(D_{o(l+1)})\}. \qquad (4)$$

Because $|R'_l| \geq |R_l|$, we can expect $|R'_m| \geq k$ for m which is smaller than l that fulfills $|R_l| \geq k$. This means that fewer documents need to be reevaluated to determine the top-ranked documents, and the modified condition therefore speeds up the reevaluation phase.

Relaxing the stop condition affects the ranking results because there possibly exists a D_i such that $r(D_i) \geq \alpha \cdot s(D_{o(l+1)})$ but $r(D_i) < r(D_{o(l+1)})$. However, since the term's in-document frequency is usually not large, the contribution of tf_i in Formula (2) is smaller than that of δ_i in Formula (3). Thus, by controlling α, the effect on ranking results and the decrease in retrieval effectiveness can be made negligible.

5.2 Selecting Query Terms

5.2.1 Methodology

One way to speed up ranking retrieval is to limit the number of query terms actually used by selecting only such terms as have large impact on relevance values [9]. The term's impact is usually measured by *idf* (the inverse of the term's document frequency). This term selection reduces not only the amount of index access but also the number of candidate documents, resulting in reduction of relevance value computation and the memory required to store intermediate results [1][18][27]. If terms with low *idf*s are simply discarded, however, their contribution to a relevance value is lost and retrieval becomes less effective. Query terms should therefore be used only to determine the ranking candidate, and the discarded terms should be used to compute the relevance values for the candidate documents [9][18].

One can apply this idea to the proposed query evaluation method as follows. The system at first selects query terms with *idf*s greater than $\beta \cdot idf_{max}$, where β is a constant between 0 and 1 and idf_{max} is the maximum *idf* among all the query terms. Note that *idf* in our ranking model is given by the first component of Formula (2). In the pre-ranking phase, only these selected terms are used to access the signature file to determine candidate documents, and their upper bounds are computed. In the reevaluation phase, all the query terms, including the discarded terms, are used to compute the relevance values of the candidate documents.

This simple application, however, causes another problem. Since our signature file is organized in a bit-slice manner, which of the not-selected terms a candidate document contains is unknown in the pre-ranking phase. Because the estimated value needs to be an upper bound of the relevance value, all the remaining terms must be assumed to occur in a candidate document, and thus δs for these terms must be set to 1. This assumption, however, requires performance degradation. That is, some δs are set to 1 even though their real values are 0, so the estimated value in this case is greater than the *real* upper bound that is computed using all the terms. This increase in upper bounds causes more reevaluation iterations before the stop condition is satisfied. In the worst case, the lengthening of the reevaluation phase is greater than the shortening of the pre-ranking phase, and the total response time increases.

To avoid this problem, δ_i is set to $\gamma(0 \leq \gamma \leq 1)$ for the remaining terms.[4] As γ decreases, the upper bounds decrease, and thus the reevaluation phase terminates earlier. Of course, the retrieval effectiveness is affected by the modification since it violates the upper bound condition, and estimated values sometimes become lower than the corresponding relevance values. But because the possibility that a remaining term appears in a document is usually small, the decrease in the effectiveness can be made small by controlling γ appropriately.

5.2.2 Estimation of Document Frequency

When one uses the query term selection optimization, the character-based signature file generates another problem. To select query terms, the document frequencies of terms must be established to compute *idf*s before signature file access. Since the document frequencies are obtained by accessing the signature file in our implementation, however, it is impossible to simply use the optimized query term

[4] δ_i is either 0 or 1 by definition, but it is quite easy to modify the upper bound formula to take real values between 0 and 1 for δ_i.

selection. The document frequencies of terms need to be obtained without accessing the signature file.

We have, therefore, developed a method of estimating the document frequency using character-level statistical information. Let $p(t)$ be the term t's occurrence probability which is obtained by dividing the number of all the occurrences of term t by the total size in the collection, and let C be the length of a document. Since df_i is roughly equal to $C \times p(t_i)$, what we have to do is to estimate $p(t)$. For this estimation, it is assumed that a word is generated in such a way that the occurrence probability of one character is determined by the character class of its proceeding character.

Let term t has n characters, $t = c_1 \cdots c_n$. When m_i denotes the character class of the i-th character c_i, $p(c_i)$ denotes the occurrence probability of the character, and $p(c_i|m_{i-1})$ denotes the conditional occurrence probability of the character after the character belonging to the character class m_{i-1}, the assumption is expressed as

$$p(t) = p(c_1)\prod_{i=2}^{n} p(c_i|m_{i-1}). \qquad (5)$$

When the relationships $p(c_i) = p(c_i|m_i)p(m_i)$ and $p(c_i|m_{i-1}) = p(c_i|m_i)p(m_i|m_{i-1})$ given by Basian inference are used, the above equation becomes

$$p(t) = p(c_1|m_1)p(m_1)\prod_{i=2}^{n} p(c_i|m_i)p(m_i|m_{i-1}). \qquad (6)$$

6 Evaluation

6.1 Test Condition

The proposed ranking method was evaluated from the viewpoints of retrieval effectiveness and efficiency.

Effectiveness was measured using *recall*, the ratio of the number of relevant documents retrieved to the relevant documents in the entire collection, and *precision*, the ratio of the number of relevant documents retrieved to the total number of retrieved documents[30]. Measured results are shown in interpolated recall vs. precision graphs [8][30]. The BMIR-J1,[5] whose statistics are shown in Table 1, was used. It should be noted that in the baseline evaluation without the optimization, recall and precision are computed using the entire ranking list, but for the optimized cases they are computed from the top 20 documents; optimization parameters α and γ (in case of $\beta = 1.0$) only control the termination of the reevaluation phase so that they do not affect the performance results that are measured from the entire ranking list.

[5]The data used in this paper were provided to the Working Group for Benchmark Database for Evaluation of Information Retrieval Systems (in SIG Database System of the Information Processing Society of Japan) contents of the Nihon Keizai Shinbun, Inc., and is based on articles of that appeared in the Nikkei newspaper between Sept. 1, 1993 and Dec. 31, 1993[14].

Table 1: Statistics.

	BMIR-J1 queries	BMIR-J1 articles	1993 N.K. articles
number of items	47	600	163110
ave. length(chr.)	11	703	494
min. length(chr.)	2	102	11
max.length(chr.)	28	3802	16545
total size	—	872KB	159MB

Table 2: Parameters used in the experiments.

Word segmentation probability threshold	P	0.00,0.05,0.10,1.00
In-document frequency normalization factor	Kd	0.0,0.5,1.0,2.0
Document length factor	λ	1.00,0.75,0.50,0.25
Stop condition relaxing factor	α	1.00,0.75,0.50,0.25
Query term selection factor	β	1.00,0.75,0.50,0.25
Unselected term weight	γ	1.0,0.5,0.1,0.01
Number of ranked documents	k	0,10,20,50,100

Efficiency was measured by the response time to get the top k documents. The BMIR-J1 is too small to evaluate retrieval efficiency, so we used all the articles of the 1993 Nihon Keizai newspaper CDROM (statistics also shown in Table 1). Note that articles in BMIR-J1 is a subset of this collection. The average response time for all the 47 queries was used as the measurement result. A Sun SPARCstation20 model 70 running Solaris 2.4 with a local SCSI disk was used in the evaluation.

The parameters used in the evaluation are summarized in Table 2. Note that Kq in Formula (2) is fixed at 0.0, since the retrieval requests are rather short and Kq has no impact on effectiveness.

6.2 Baseline Results

We first evaluated the basic performance of the proposed ranking method without any optimization. There are three parameters, P, Kd and λ, related to baseline performance. Although we have measured performance for all possible combinations, it is hard to show all the results here. Thus, in this subsection, we illustrate the effect of each of these parameters by showing the results for cases in which the remaining parameters are fixed to appropriate values.

6.2.1 Effect of P

Figure 4 shows the effect of P, which controls how compound words are segmented in the query generation, when Kd and λ were fixed to 1.0 and 0.0. The left side of Figure 4 represents the retrieval effectiveness. Precisions at $P = 1.00$ were almost always worst among various P

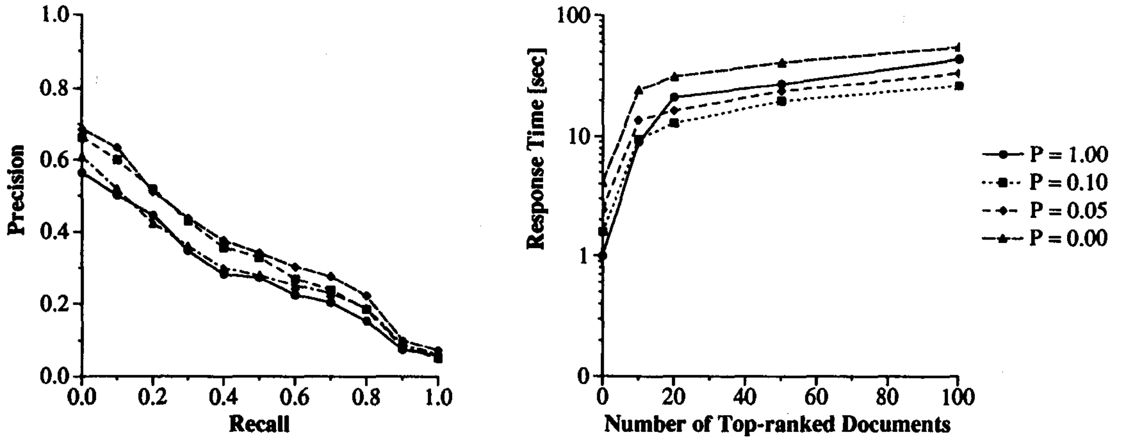

Figure 4: Effect of $P(Kd = 1.0)$.

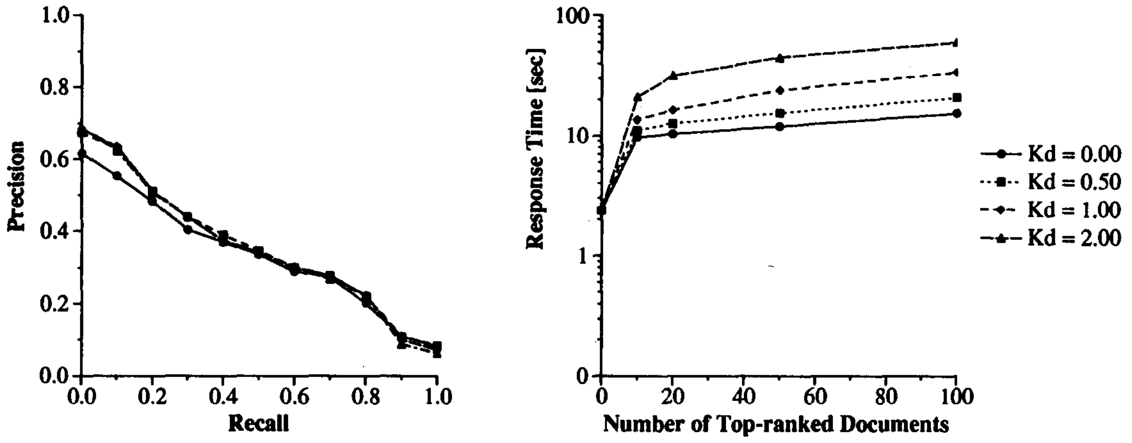

Figure 5: Effect of $Kd(P = 0.05)$.

values, which corresponds to the case without segmentation, and increased in accordance with P. When $P = 0.05$, the performance reached at the best, where precision at recall = 1.0 is 22% higher than that without segmentation. However, precision decreased from $P = 0.05$ to 0.00, because a compound word is divided into almost single characters at $P = 0.00$, so the possibility that irrelevant documents receive higher relevance values by chance increases. The proposed word segmentation method is, from these results, confirmed to be effective.

The response times for various P values are plotted in the right side of the figure. The response time increased as P increased because the number of query terms and candidate documents increased in accordance with P. Actually, for $P = 1.00$, 0.10, 0.05, and 0.00, the average number of query terms generated were 2.74, 3.60, 4.40, and 7.72, and the average number of candidate documents were 23291, 45356, 75686, 120712. For $k \geq 20$, the response time of $P = 1.0$ was worse than that of $P = 0.1$; At least k documents have to receive relevance values higher than the upper bound value of a certain document to finish a ranking. Thus the range of upper bound values needs to be wide in order to terminate the reevaluation iteration quickly. However, only a few query terms are generated and the range of upper bounds becomes very small when $P = 1.0$. As a result, all of the candidate documents sometimes needed to be evaluated, and the response time increased.

6.2.2 Effect of Kd

The effect of Kd when $P = 0.05$ and $\lambda = 0$ is illustrated in Figure 5. The effectiveness was minimum at $Kd = 0.0$, which corresponds to cases without using the in-document frequency. This result means in-document frequency plays an important role in ranking. Although the best performance was achieved at $Kd = 1.0$, Kd has less impact than P.

As for the response time, the ranking result was established quickly as Kd became smaller. That is because the difference between the final score and the upper bound becomes smaller according to Kd. We noticed that while the processing time for the pre-ranking phase (which is given at $k = 0$) stayed at the same level for all Kds, the re-

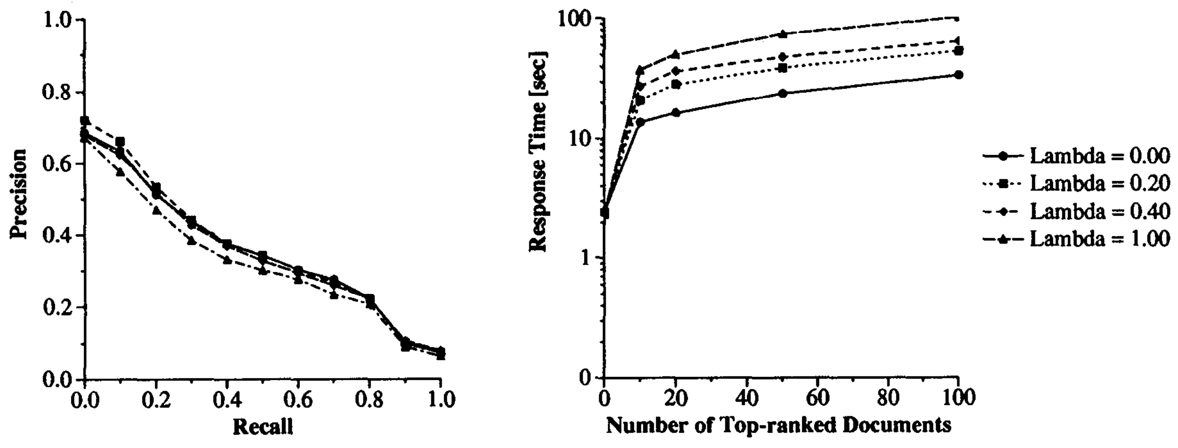

Figure 6: Effect of $\lambda(P = 0.05, Kd = 1.0)$.

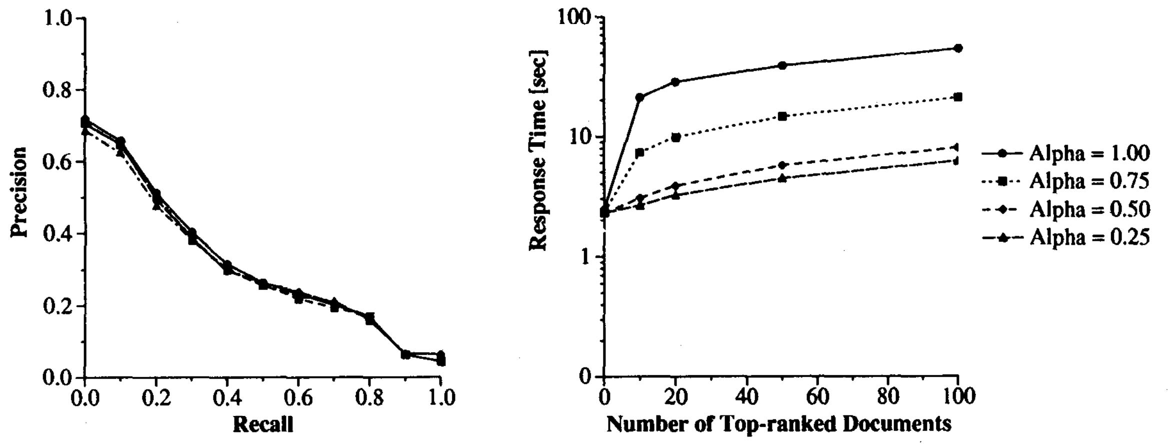

Figure 7: Effect of α.

sponse time increased greatly as Kd increased for larger k. This is because Kd only affects the number of reevaluation iterations according to decreases in the third component of Formula (2), and does not affect the pre-ranking phase.

6.2.3 Effect of λ

The effect of the document length factor λ is shown in Figure 6. The retrieval effectiveness attained the best performance when $\lambda = 0.2$, and after that decreased to minimum at $\lambda = 1.0$. This result is incompatible with other experimental results which showed that the document length factor had a quite positive effect; the above-mentioned BM11 model, which corresponds to $\lambda = 1.0$, attained higher performance than the BM15 model, which corresponds to $\lambda = 0.0$[23]. Although the BM11 model requires, to get the same relevance value, the term's occurrence frequency must be proportional to the document length, this requirement is hard to be maintained because a longer document frequently has more than one topic [10]. Thus, we believe that our results in which the best result was given at a point between the two extreme cases would coincide with our intuition.

As for retrieval efficiency, λ decreased the performance. This is because larger in-document frequencies are obtained in general in longer documents, so that the normalization makes the effective frequencies small and thus the processing becomes take much time.

In summary, $P = 0.05$, $Kd = 0.5$ and $\lambda = 0.2$ look like the best parameter settings from the above experiments. This combination of parameters was, therefore, used in the following experiments.

6.3 Effect of Stop Condition Relaxation

Measurement results are shown in Figure 7, in which $\alpha = 1.0$ corresponds to the baseline method. It might be difficult to see from this figure, the retrieval effectiveness for $\alpha = 1.0$ was slightly lower than the result at the full ranking given in Figure 6. This is because the effectiveness here was measured at $k = 20$ as described in Section 6.1, so some documents that should have been listed in the top 20 were missed. Precision was decreased by this optimization, but the decrease was small.

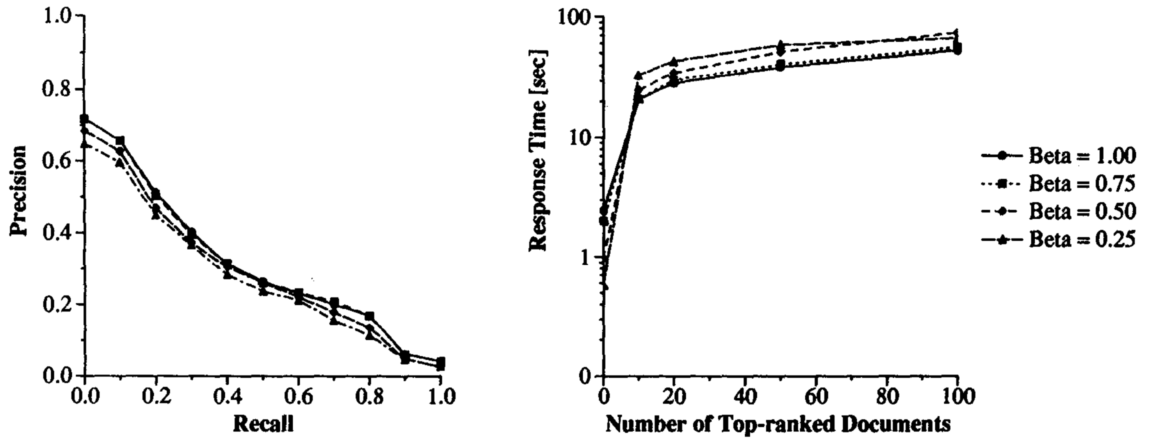

Figure 8: Effect of β.

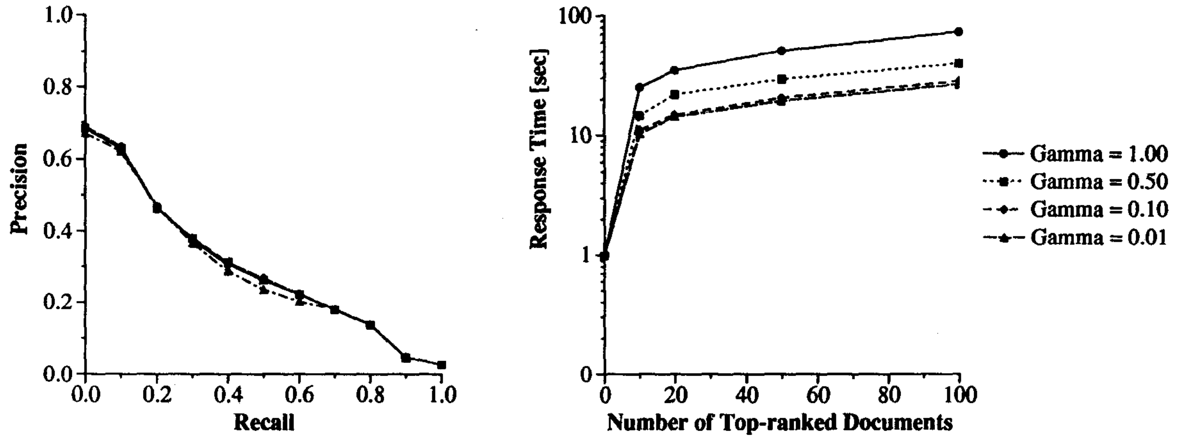

Figure 9: Effect of $\gamma(\beta = 0.5)$.

On the other hand, this optimization was quite effective to speed retrieval as shown in the right graphs in Figure 7. The top 10 or 20 documents were, for example, identified 6 or 7 times more quickly when $\alpha = 0.5$. Because the improvement was almost the same for $\alpha = 0.50$ and 0.25, it seems better to set α at 0.50.

6.4 Effect of Query Term Selection

The query term selection optimization was evaluated by changing β and γ.

First, the effect of β is shown in Figure 8. Recall-precision graphs show that precision decreased in accordance with β, and became considerably worse when $\beta = 0.25$. The reason for this decrease is that document candidates that only contained many terms with smaller idfs were missed when β became low.

The effect on the response time was complicated, as shown in the left graph. Decrease in β serves to speed up the pre-ranking phase, which was indicated by a decrease in the response time at $k = 0$, by limiting the query words used in the pre-ranking and the number of candidate documents. Actually, the number of query words decreased as follows: 4.40 ($\beta = 1.00$), 4.09 ($\beta = 0.75$), 2.57 ($\beta = 0.50$) and 1.81 ($\beta = 0.25$). However, as mentioned in Section 5.2.1, the response time for $k > 0$ went up as β became smaller because the number of accessed documents in the reevaluation phase increased. In consequence, no performance gain was achieved by simply setting as $\beta < 1$, and γ must be smaller.

Figure 9 illustrates the effect of γ when β was fixed at 0.5. As we expected, the system responded faster in accordance with decrease in γ throughout the range for all ks, but the speed up saturated at $\gamma = 0.1$. Note that the response time at $k = 0$ did not change, since γ did not change the number of query words and was not effective in accelerating the pre-ranking phase.

The effect on recall and precision was also shown in the same figure, and we found that γ had a smaller effect. In conclusion, $\beta = 0.5$ and $\gamma = 0.1$ seems the best parameter settings for the query term selection optimization.

Finally, the performance was measured when the two optimizing methods were combined. In Figure 10, RSC and QTS stand for the stop condition relaxation and the

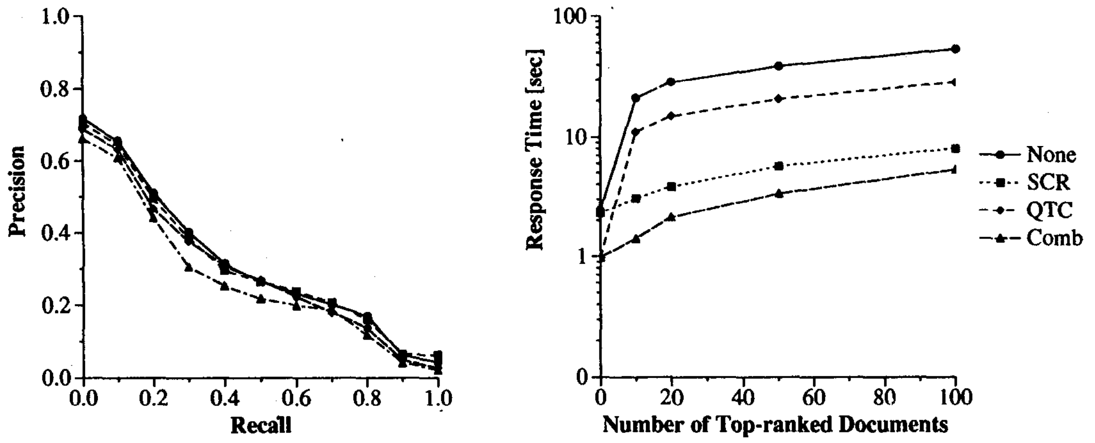

Figure 10: Effect of combination of two optimizations.

query term selection optimizations, and the parameters were set to $\alpha = 0.50$ for RSC, and $\beta = 0.5$ and $\gamma = 0.1$ for QTS. Although the precision decreased by several percent, especially in the middle ranges in recall, the combined optimization speeded retrieval by a factor of 10.

7 Conclusion

This paper proposes an effective and efficient ranking method that does not require a large lexicon. By using a character-based signature file and developing a statistical method to segment compound words, the need for a large lexicon is eliminated. In addition, we have developed an efficient query evaluation strategy using upper bounds, as well as two optimizing techniques. Evaluation results confirmed the effectiveness of both the query evaluation method and the optimizing techniques.

The application range of the strategy is not limited to ranking models that use term frequencies. Ranking effectiveness could be increased by using information other than term frequency, but it would be almost impossible to store all the information necessary for ranking in an index. Because such information can be acquired directly from documents when necessary, the proposed processing strategy is practical for such ranked models.

References

[1] E.W. Brown. Fast evaluation of structured queries for information retrieval. In *Proc. of 18th ACM SIGIR Conf.*, pages 30–38, 1995.

[2] C. Buckley and A.F. Lewit. Optimization of inverted vector searches. In *Proc. of 8th ACM SIGIR Conf.*, pages 97–110, 1985.

[3] L.F. Chien. Fast and quasi-natural language search for gigabits of Chinese texts. In *Proc. of 18th ACM SIGIR Conf.*, pages 112–121, 1995.

[4] W. B. Croft and P. Savino. Implementing ranking strategies using text signatures. *ACM Trans. on Office Information Systems*, 6(1):42–62, 1988.

[5] W.B. Frakes and R. Basza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, New Jersey, 1992.

[6] H. Fujii and W. B. Croft. A comparison of indexing techniques for Japanese text retrieval. In *Proc. of 16th ACM SIGIR Conf.*, pages 237–246, 1993.

[7] K. Furuse et al. Implementation of signature file access method in a DBMS (*in Japanese*). In *Technical Report DE94-58*, pages 23–30. Institue of Electronics, Information and Communication Engineers, Japan, 1994.

[8] D. Harman, editor. *The 3rd Text REtrieval Conference (TREC-3)*. National Institute of Standards and Technology, 1995.

[9] D. Harman and G. Candela. Retrieving records from a gigabyte of text on a minicomputer using statistical ranking. *Journal of the American Society for Information Science*, 41(8):581–589, 1990.

[10] M.A. Hearst and C. Plaunt. Subtopic structuring for full-length document access. In *Proc. of 16th ACM SIGIR Conf.*, pages 59–68, 1993.

[11] M. Iwasaki and Y. Ogawa. A new character-based indexing method for Japanese texts using reduced adjacent character bitmap tables. In *Int. Symp. on Next Generation Database Systems and Their Applications*, pages 145–150, 1993.

[12] M. Kameda. QJP: a portable and quick Japanese processing tool (*in Japanese*). In *Proc. of 1st Annual Meeting*, pages 349–352. Japan Natural Language Processing Society, 1995.

[13] Y. Kawashimo et al. Development of full text search system Bibliotheca/TS (*in Japanese*). In *Proc. of the 45th Zenkoku Taikai, 3*, pages 241–242. Information Processing Society of Japan, 1992.

[14] I. Keshi et al. Overview of benchmark for Japanese IR system ver. 1.0. In *SIG notes, No. DBS-106*, pages 139–145. Information Processing Society of Japan, 1996.

[15] D. Knaus and P. Schäuble. Effective and efficient retrieval from large and dynamic document collections. In *Proc. of 2nd TREC*, pages 163–170, 1994.

[16] J.H. Lee et al. N-gram-based indexing for effecitve retrieval of Korean texts. In *Proc. of 1st Australian Document Computing Conf.*, pages 90–94, 1996.

[17] T. Liang and W.P. Wang. Signature method in Chinese text retrieval. In *Proc. of Int. Symp. on Digital Libraries '95*, pages 97–104, 1995.

[18] A. Moffat and J. Zobel. Fast ranking in limited space. In *Proc. of Int. Conf. on Data Engineering*, pages 428–437, 1994.

[19] Y. Ogawa. An efficient document ranking method using a character-based signature file(*in Japanese*). In *Proc. of Advanced database system symp. '95*, pages 29–38. Information Processing Society of Japan, 1995.

[20] Y. Ogawa, A. Bessho, M. Iwasaki, M. Hirose, and M. Nishimura. A new indexing and text ranking method for Japanese text databases using simple-word compounds as keywords. In *Proc. of 3rd Int. Symp. on Database Systems for Advanced Applicataion*, 1993.

[21] Y. Ogawa and M. Iwasaki. A new character-based indexing method using frequency data for Japanese documents. In *Proc. of 18th ACM SIGIR Conf.*, pages 121–129, 1995.

[22] Y. Ogawa, M. Kameda, and T. Matsuda. Inforium: A user-friendly document retrieval system. In *Proc. of Int. Workshop on Information Retrieval with Oriental Languages*, (*to appear*), 1996.

[23] S.E. Robertson and S. Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proc. of 17th ACM SIGIR Conf.*, pages 232–241, 1994.

[24] Y. Sakamoto. An automatic segmentation for Japanese text (*in Japanese*). *Mathematical Linguistics*, 11(6), 1978.

[25] G. Salton, E.A. Fox, and H. Wu. Parallel text search methods. *Communications of the ACM*, 31(2):202–215, 1988.

[26] P. Schäuble. SPIDER: A multiuser information retrieval system for semistructured and dynamic data. In *Proc. of 16th ACM SIGIR Conf.*, pages 318–327, 1993.

[27] H. Turtle and J. Flood. Query evaluation: Strategies and optimizations. *Information Processing and Management*, 31(6):831–850, 1995.

[28] P. Vines et al. Indexing for Chinese text retrieval. In *Proc. of 1st Australian Document Computing Conf.*, pages 85–89, 1996.

[29] Y. Watanabe and M. Nagao. Document classification using important kanji characters extracted by χ^2 method. In *SIG Notes FI39*, pages 25–32. Information Processing Society of Japan, 1995.

[30] I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold, 1994.

[31] W.Y.P. Wong and D.L. Lee. Signature file methods for implementing a ranking strategy. *Information Processing and Management*, 26(5):641–653, 1990.

[32] W.Y.P. Wong and D.L. Lee. Implementations of partial document ranking using inverted files. *Information Processing and Management*, 29(5):647–669, 1993.