

# An Effective Chinese Indexing Method Based on Partitioned Signature Files



Department of Systems Engineering  
& Engineering Management



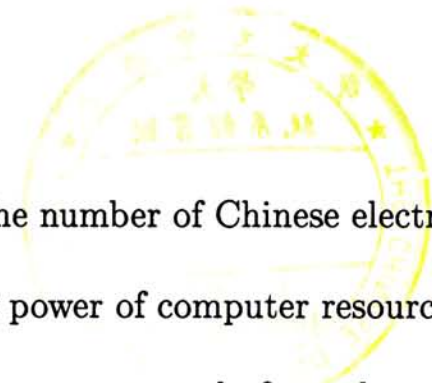
香港中文大學  
The Chinese University of Hong Kong

Submitted in partial fulfillment of requirements for  
the degree of Master of Philosophy

*June 1998*



# Abstract



In the past few years, the number of Chinese electronic documents grows enormously with the increasing power of computer resources and the popularity of the Internet. These documents are composed of mostly non-structured texts. In order to seek for appropriate information from a large Chinese document collection, there is an increasing demand for information retrieval techniques to retrieve relevant and useful documents quickly and accurately. Although there are many indexing methods and retrieval models proposed, they are mostly designed for English texts. The linguistic difference between English and Chinese hinders from applying them directly on Chinese. We first investigate the effect of three different segmentation methods on Chinese text retrieval performance in the vector space retrieval model. From the experimental results, we find that the word-based segmentation approach has a better retrieval performance than other segmentation approaches.

Until now, there are many Chinese information retrieval systems using the inverted file to index Chinese texts due to its fast retrieval performance. Unfor-

tunately, this approach does not allow insertion or deletion of terms in dynamic environment and consumes a large amount of storage for its index file, about 100% - 200% of the corpus size. We conduct research on an effective signature files for Chinese text indexing and retrieval. The advantages of the signature file approach over the inverted files are effective file structure for insertion or deletion of terms as well as small storage requirement. However, one drawback is the introduction of false drops. We address this problem by proposing a new partitioning method for the signature file based on Chinese characteristics so as to reduce the false drops as well as the time for retrieval.

In addition, the number of false drops is also affected by various signature generation methods, such as lookup table and hashing functions. We investigate some existing hashing functions on our partitioning method. Then we develop two new hashing methods to improve the retrieval performance. The experimental results show that the performance of our new hashing methods can achieve a better retrieval precision than simple hashing methods.



## 摘要

近年隨著電腦科技日漸進步以及國際互聯網絡 (Internet) 的普及，中文電子文件的數目與日俱增，而這些文件大部份都沒有特定的格式，要從這些數目龐大的中文文件中找尋所須要的資料，並非一件容易的事。所以對於準確而又快速的中文文件檢索系統的需求是非常之大。雖然現今已有很多文件索引和搜尋的技術，但是它們都是針對英文文件而設計，而且由於中文和英文在語言表達上的分別，導致這些技術不能直接使用於中文文件之上。我們比較了三種不同的中文句子分段方法對文件檢索系統的影響。從我們的實驗結果指出「單詞模式分段法」比另外兩種分段方法有較佳的表現。

現今的中文文件檢索系統大部份都是使用「倒排文件」(Inverted file) 的索引方法，雖然它擁有快速和準確的搜尋性能，但它也需要大量的空間來儲存它的索引檔 (大約是原來文件檔的一至兩倍)，而且它的檔案結構複雜並不適合動態環境下使用。所以我們的研究是發展中文「簽字檔」(Signature files) 的使用，使用「簽字檔」的好處在於它的檔案結構簡單，使儲存新增文件的索引資料時更方便，而且它使用較小的空間來儲存索引檔，但這方法亦

會導致「誤檢」的情況出現。我們針對這個缺點提出以中文單詞的特點來分割「簽字檔」—「中文單詞簽字檔分割法」(PSFC)，目的是要減少誤檢的數目以及加快搜尋速度。

「簽字」(Signature)的產生方法有兩種—使用查尋表和散列函數。不同的產生方法是會影響誤檢的數目，所以我們使用「中文單詞簽字檔分割法」比較了幾種「簽字」產生方法對檢索性能的影響。爲了提高檢索性能，我們亦提出了兩種新的散列函數方法來產生「簽字」。從我們的實驗中，這兩種新的散列函數對檢索性能測試都有很好的成績。

# Acknowledgements

I would like to thank all the people who have helped and guided me through my research work. In particular, I would like to express my deepest gratitude to Professor Wai Lam, my research supervisor, for his invaluable guidance and encouragement through my whole research work. Moreover, I would like to thank Professor Kam-Fai Wong for his invaluable assistance and suggestions.

Harry Chi-Yin Wong.

*June 1998*

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Introduction to Chinese IR . . . . .	1
1.2 Contributions . . . . .	3
1.3 Organization of this Thesis . . . . .	5
<b>2 Background</b>	<b>6</b>
2.1 Indexing methods . . . . .	6
2.1.1 Full-text scanning . . . . .	7
2.1.2 Inverted files . . . . .	7
2.1.3 Signature files . . . . .	9
2.1.4 Clustering . . . . .	10
2.2 Information Retrieval Models . . . . .	10
2.2.1 Boolean model . . . . .	11
2.2.2 Vector space model . . . . .	11
2.2.3 Probabilistic model . . . . .	13
2.2.4 Logical model . . . . .	14
<b>3 Investigation of Segmentation on the Vector Space Retrieval Model</b>	<b>15</b>
3.1 Segmentation of Chinese Texts . . . . .	16
3.1.1 Character-based segmentation . . . . .	16
3.1.2 Word-based segmentation . . . . .	18
3.1.3 <i>N</i> -Gram segmentation . . . . .	21
3.2 Performance Evaluation of Three Segmentation Approaches . . . . .	23
3.2.1 Experimental Setup . . . . .	23
3.2.2 Experimental Results . . . . .	24
3.2.3 Discussion . . . . .	29



<b>4</b>	<b>Signature File Background</b>	<b>32</b>
4.1	Superimposed coding . . . . .	34
4.2	False drop probability . . . . .	36
<b>5</b>	<b>Partitioned Signature File Based On Chinese Word Length</b>	<b>39</b>
5.1	Fixed Weight Block (FWB) Signature File . . . . .	41
5.2	Overview of PSFC . . . . .	45
5.3	Design Considerations . . . . .	50
<b>6</b>	<b>New Hashing Techniques for Partitioned Signature Files</b>	<b>59</b>
6.1	Direct Division Method . . . . .	61
6.2	Random Number Assisted Division Method . . . . .	62
6.3	Frequency-based hashing method . . . . .	64
6.4	Chinese character-based hashing method . . . . .	68
<b>7</b>	<b>Experiments and Results</b>	<b>72</b>
7.1	Performance evaluation of partitioned signature file based on Chinese word length . . . . .	74
7.1.1	Retrieval Performance . . . . .	75
7.1.2	Signature Reduction Ratio . . . . .	77
7.1.3	Storage Requirement . . . . .	79
7.1.4	Discussion . . . . .	81
7.2	Performance evaluation of different dynamic signature generation methods . . . . .	82
7.2.1	Collision . . . . .	84
7.2.2	Retrieval Performance . . . . .	86
7.2.3	Discussion . . . . .	89
<b>8</b>	<b>Conclusions and Future Work</b>	<b>91</b>
8.1	Conclusions . . . . .	91
8.2	Future work . . . . .	95
<b>A</b>	<b>Notations of Signature Files</b>	<b>96</b>
<b>B</b>	<b>False Drop Probability</b>	<b>98</b>
<b>C</b>	<b>Experimental Results</b>	<b>103</b>
	<b>Bibliography</b>	<b>107</b>



# List of Figures

2.1	Relationship between record vectors and inverted index. . . . .	8
3.1	Recall-precision curve of the three segmentation approaches. . . . .	25
4.1	Example of superimposed coding. . . . .	35
5.1	Design of the partitioning in PSFC. . . . .	47
5.2	Example of partitioned signature files based on Chinese word length. . . . .	47
5.3	Superimposing example. . . . .	52
6.1	Using random number assisted division method in signature generation. . . . .	63
6.2	An example of frequency-based hashing method. . . . .	65
6.3	Frequency-group hashing method. . . . .	67
6.4	An example of Chinese character-based hashing method. . . . .	69
6.5	Chinese character-group hashing method. . . . .	70
7.1	Average number of false drops. . . . .	76
7.2	Average search space. . . . .	78
7.3	Total storage space of the index files. . . . .	80
7.4	Average number of false drop ( $\alpha = 1$ ). . . . .	87
7.5	Average number of false drop ( $\alpha = \frac{l_x}{l}$ ). . . . .	87
7.6	Average number of false drop ( $\alpha = 1.5$ ). . . . .	88
7.7	Average number of false drop ( $\alpha = \frac{l_x}{2l}$ ). . . . .	88
C.1	Average number of false drops of $w = 10$ . . . . .	104
C.2	Average number of false drops of $w = 20$ . . . . .	104
C.3	Average search space $w = 10$ . . . . .	105
C.4	Average search space $w = 20$ . . . . .	105
C.5	Total storage space of the index files $w = 10$ . . . . .	106
C.6	Total storage space of the index files $w = 20$ . . . . .	106

# List of Tables

3.1	11-point average precision of the three segmentation approaches. . .	26
3.2	Time required for various operations. . . . .	27
3.3	Number of terms indexed by each approach after segmentation. . .	28
3.4	Storage requirement of the three segmentation approaches. . . . .	29
5.1	Percentage of occurrence of Chinese words. . . . .	46
7.1	Statistics of Chinese words in corpus. . . . .	74
7.2	Index to corpus ratio. . . . .	79
7.3	Statistics of collision using different hashing methods. . . . .	84
A.1	Notations used in signature files. . . . .	97

# Chapter 1

## Introduction

### 1.1 Introduction to Chinese IR

Information retrieval (IR) systems process requests for information so as to identify and retrieve the relevant records from a huge document collection in response to a query. One major form of information handled in IR is text databases which are very different from the traditional structured databases. A structured database contains a specific set of attributes used to characterize each record and the values of the attributes are assumed to describe the records unequivocally and completely. Thus, the retrieval of structured databases can be achieved by an exact match between the attribute values used in a query and the records.

Text databases involve unformatted texts without specific attributes. The vocabulary of the texts is allowed to vary widely and the subject matter is unre-

stricted. Such databases include newspaper articles, newswire dispatches, textbooks, dictionaries, encyclopedia, and so on. It is difficult to select the terms that can describe the text content for indexing from the unformatted text. There exists several indexing methods, namely, the full-text scanning [71], the inverted files [70, 71], the signature files [24], and the clustering. These indexing methods provide a way to locate documents containing a particular term. Apart from indexing, we also need a method for determining the degree of relevance of a document given a query. Exact match is insufficient because the documents are unformatted and there is no specific set of attributes for matching. Therefore, a retrieval model is needed for determining the degree of relevance between the set of identifiers attached to the query and documents. Some existing models are the Boolean model, the vector space model [69], the probabilistic model [61, 65, 66, 87], and the logical model [62].

Although information retrieval has been studied for decades and many indexing methods and retrieval models have been proposed, they are mainly designed for English texts. Unlike English, a Chinese sentence consists of a continuous string of characters without explicit word boundaries, for example “計算機已經用於各個領域” (Computers have been used in every area). In order to process Chinese texts, a task known as *segmentation* needs to be performed on the text before indexing. Segmentation extracts appropriate terms from Chinese sentences



for indexing.

Many Chinese information retrieval systems use the inverted file method for indexing because this method has a fast retrieval response and high precision. However, it does not allow insertion or deletion of terms in dynamic environment and consumes a very large amount of storage space for its index file. Recently, there has been much attention of using signature files for Chinese indexing. A major advantage of the signature file indexing approach is that it supports dynamic insertion and deletion easily and it consumes less storage space than that of the inverted file. However, one drawback is the introduction of false drops. We address this problem by proposing a new partitioning method for the signature file based on Chinese characteristics so as to reduce the false drops as well as the time for retrieval. In addition, the number of false drops is also affected by various signature generation methods, such as lookup table and hashing functions. We investigate some existing hashing functions on our partitioning method. Then we develop two new hashing methods to improve the retrieval performance. The experimental results show that the performance of our new hashing methods can achieve a better retrieval precision than simple hashing methods.

## 1.2 Contributions

The contributions achieved by our research are summarized as follows:



1. There exists three commonly used segmentation approaches for Chinese texts, namely, the character-based approach, the word-based approach, and the  $N$ -gram approach. Each of them has their own advantages and disadvantages, but there is no indication of which of them has a better retrieval performance over the others. We investigate the effect of the three segmentation approaches on the performance of the vector space retrieved model which is an advanced information retrieval model. This study also provides a guideline for the choice of a more effective segmentation approach in the subsequent part of our research.
2. We investigate the design of the signature file in Chinese information retrieval. Signature files allow dynamic insertion and deletion of terms or documents easily and reduce a large amount of storage than the inverted files. However, the search time of a signature file is proportional to the corpus size due to the sequential search of all block signatures. To alleviate this problem, we propose a new partitioning method for the signature file, referred to as PSFC, based on some characteristics of Chinese. PSFC can reduce the retrieval time and reduce the number of false drops. We devise a general scheme for our method so that we can control the trade-off between the storage overhead and the retrieval precision based on the system resources availability.
3. As mentioned in [27], signature generation based on the look-up table has a

better retrieval performance than the dynamic hashing function. However, the lookup table approach requires extra storage space. We evaluate two existing hashing functions for the signature generation. We then propose two new hashing methods to improve the retrieval performance based on the term statistics and Chinese characteristic.

### 1.3 Organization of this Thesis

The rest of this thesis is organized as follows: In Chapter 2, the background of information retrieval will be provided including the description of various existing indexing methods and retrieval models. Chapter 3 presents three different commonly used Chinese segmentation approaches. It includes a comparison of their effect on the retrieval performance. Based on our comparison result, we employ the more effective segmentation method on Chinese signature files. Before the description of our model, a description of the background of the signature files will be provided in Chapter 4. Chapter 5 describes our proposed partitioning method of Chinese signature file, referred to as PSFC. We have also investigated four signature generation methods using hashing function and their effects on applying our proposed partitioning method. The description of each signature generation method will be provided in Chapter 6. Chapter 7 presents the results of various experiments. Chapter 8 gives a conclusion as well as some future directions.

# Chapter 2

## Background

Indexing free text documents requires techniques that are different from that of structured databases because there is no specific set of attributes in free text. In this chapter, some existing indexing methods and retrieval models will be described.

### 2.1 Indexing methods

In the retrieval process a way to facilitate the retrieval of documents that satisfy a query is required. This functionality resembles the index in the back of books indicating which pages contain an index word. Several existing indexing methods [26, 29] are reviewed in the following sections.



### 2.1.1 Full-text scanning

Full-text scanning [71] is the simplest method to search for the qualified documents. Actually, this method does not involve indexing the original corpus. It is based on a comparison of individual characters in query formulations with words in the stored document texts. Given a query or a search pattern, the whole raw document collection is scanned to locate which documents contain the given pattern. This method does not require storage overhead to store the index file. In addition, it is very simple to perform the insertion and updates. However, for each query pattern, the whole corpus has to be scanned. Although there exist some fast string searching algorithms [2, 4, 40], it still takes a considerable amount of time to scan through the whole corpus in retrieval [34]. As a result, it is impractical for dealing with large corpus.

### 2.1.2 Inverted files

An inverted file is one of the methods for implementing an index file [70, 71]. Each entry of the index consists of a term, which may be a stemmed word or a concept, along with a list of pointers to indicate which documents in the collection contain this term. For example, in Figure 2.1, when four documents are indexed, the indexed entry consists of the document along with a list of terms it contains. This is a simple method for indexing, however, it is inefficient for searching the

qualified documents given a query term. To alleviate this problem, we can transpose the original record array in such a way that each record in the array consists of an indexed term and a list of document pointers. It will be much faster to obtain a list of documents that contain the query term. This technique is called inverted files.

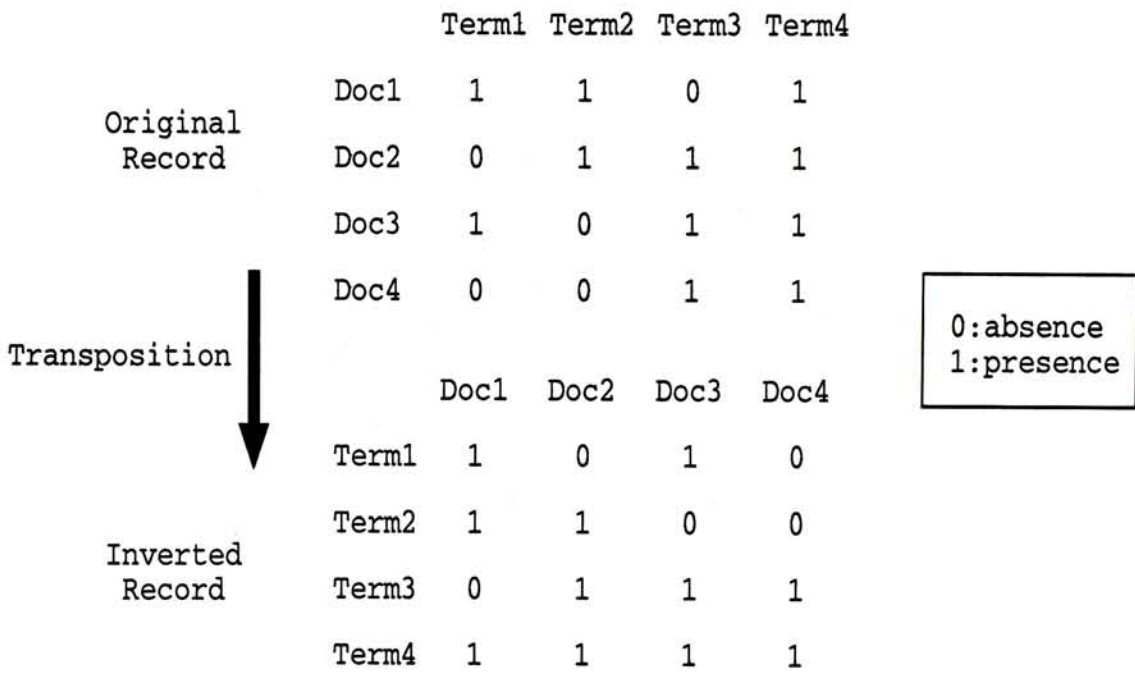


Figure 2.1: Relationship between record vectors and inverted index.

Inverted files have the advantage of fast retrieval. Given a query term, we can locate the qualified documents very easily. Thus, many commercial systems have adopted this approach, such as, STAIRS [35, 70], MEDLARS [70], ORBIT, LEXIS [70]. However, for a large collection, it will be very time consuming to perform the transposition. Owing to its complex index structure, an inverted file requires



expensive updating and reorganizing the index for dynamic environment. It also requires a large amount of time to merge the files. Since the index file contains all the indexed terms together with their document pointers, it requires a large storage overhead typically around 50% - 300% of the initial corpus size [17, 32].

### 2.1.3 Signature files

In signature files [24], each indexable term from documents and queries will be transformed to a sequence of bits, i.e., the word signature, instead of the original term for storing in the index file. In order to reduce the storage overhead of the index file, superimposed coding technique [64] is used to combine several word signatures into a block signature. As a result, the storage overhead for the index file will be less than that of the inverted files. For retrieval, the query terms are transformed into query signatures in the same way as that of document terms. These query signatures will be used to match each block signature in the index file. Since each block signature in the index file is required to be checked, the retrieval time of the signature file is longer than that of the inverted files but shorter than that of the full-text scanning.

Besides the search speed, this method has a drawback due to the application of superimposed coding technique. It leads to the introduction of false drops. A false drop occurs when a block signature is evaluated as relevant but actually it is not.

Thus, this method can eliminate most, but not all, of the block signatures that do not match the query. It can be used as a filtering mechanism. The truly irrelevant blocks can be eliminated by further comparing the query term with the text blocks. Detailed description of the signature files will be provided in Chapter 4.

### 2.1.4 Clustering

In this method, similar documents are grouped together to form a *cluster* and they are physically contiguous in storage. The underlying reason of the cluster hypothesis is that closely associated documents tend to be relevant to the same request. Thus grouping similar documents can accelerate the searching process. Clustering involves two procedures, namely, the cluster generation and the cluster search. Clustering is the common access method in the area of library science [61, 70].

## 2.2 Information Retrieval Models

The above mentioned indexing methods can locate the documents which contain the query terms. Next, we need to determine which of the located documents are really relevant to the query. There are four commonly used information retrieval models, namely, the Boolean model, the vector space model, the probabilistic model, and the logical model for determining the degree of relevance of

documents with respect to a query.

### 2.2.1 Boolean model

The Boolean model is the earliest information retrieval model. This model makes use of Boolean operators (e.g. AND, OR) to formulate the query. It basically only checks the presence or absence of query terms in the documents. Hence, the retrieved result only indicates a binary decision in which the documents are classified as relevant or non-relevant. There is no indication of the degree of relevance of the retrieved documents to the query. Besides, the query in this model is also restricted to the form of Boolean operators. It is difficult to construct a query with complex meaning. This problem becomes serious when queries are in the form of free texts similar to natural language.

### 2.2.2 Vector space model

In the vector space model proposed by Salton [69], both queries and documents are represented in the form of vectors [71]. Specifically, a document  $d$  will be transformed into a  $n$ -dimensional vector in the form of:

$$d = (w_{d1}, w_{d2}, \dots, w_{dn})$$

where  $w_{di}$  denotes the weight of the term  $t_i$  in the document  $d$ . Ideally each term represents a concept that can characterize the content of the document. These



terms are usually extracted automatically from a document. For instance, a term may correspond to a stemmed word in the document  $d$ . The weight  $w_{di}$  captures the importance of the term  $t_i$  in the corresponding document  $d$ . Thus, if a term is absent in the document, its weight will be zero.

Similarly, the query in vector space model is also transformed into its term vector in the same way as documents vectors. A query  $q$  is also represented in the form of vectors  $(w_{q1}, w_{q2}, \dots, w_{qn})$  where  $w_{qi}$  represents the weight of the term  $t_i$  in the query  $q$ . The dimension of the query vector is the same as that of the document vector. The same position of these two vectors represent the same term. The weight of each term in the document and the query can be determined from its term frequency and inverse document frequency as follows:

$$w_{ij} = \frac{tf_{ij} \times idf_j}{\sqrt{\sum_{k=1}^n (tf_{ik} \times idf_k)^2}} \quad (2.1)$$

where  $w_{ij}$  is the weight of the term  $j$  in the document  $i$ ;  $tf_{ij}$  is the term frequency of the term  $j$  in the document  $i$ ;  $idf_j$  is the inverse document frequency of the term  $j$  in the corpus. The denominator is used for the weight normalization which restricts the weight within the range between 0 and 1. Thus it reduces the dominant effect of rare terms with large inverse document frequency.

By using the query and document vectors, we can determine the degree of relevance of the document to the query by using the inner product of the corresponding

vectors as follows:

$$Sim(d, q) = \sum_{i=1}^n (w_{di} \times w_{qi}) \quad (2.2)$$

The result of the above equation gives the degree of similarity of each stored document to the query. The retrieved result can be ranked in decreasing order according to their degree of similarity to the query. This model has been adopted in some information retrieval systems such as SMART [70].

### 2.2.3 Probabilistic model

The probabilistic model [61, 65, 66, 87] is based on probabilistic ranking principle in which documents should be ranked according to their probability of relevance with respect to the actual request. INQUERY [5] is one of the information retrieval systems using this approach. This model is based on two main parameters, namely  $Pr(x|rel)$ , the probability that a relevant item has representation  $x$ , and  $Pr(x|nonrel)$ , the probability that a non-relevant item has representation  $x$ .

Assuming that terms are independently assigned to the relevant and to the non-relevant documents of a corpus. Binary term weights to documents are restricted to 0 and 1. Each document  $D$  is represented in the form  $D = (d_1, d_2, \dots, d_n)$ , where  $d_i$  indicates the absence or presence of the  $i$ -th index term. The similarity value between a document and the query can then be derived from  $\log \frac{Pr(x|rel)}{Pr(x|nonrel)}$ . Two parameters, namely  $p_i$  and  $u_i$ , represent the probabilities that the  $i$ -th item



has a value 1 in a relevant and non-relevant document respectively. The degree of similarity between query  $Q$  and document  $D$  can be calculated as follows:

$$Sim(D, Q) = \sum_{i=1}^n d_i \log \frac{p_i(1 - u_i)}{u_i(1 - p_i)} + constants \quad (2.3)$$

where  $p_i = Pr(x_i = 1|relevant)$

$u_i = Pr(x_i = 1|nonrelevant)$

#### 2.2.4 Logical model

A logic based information retrieval model uses logical framework to model different functions of an information retrieval system. Most logic based models consist of sentences joined by connectors to construct complex sentences. For example, the implication ( $\rightarrow$ ) connector establishes a relationship between two sentences. Precisely, the truth of  $\phi \rightarrow \psi$  means that the sentence  $\phi$  implies the sentence  $\psi$  whenever  $\phi$  is true. For modeling information retrieval system,  $d \rightarrow q$  means that the information captured by document  $d$  is sufficient to infer the information represented by query  $q$ , such that the document  $d$  satisfies the query  $q$ . There are various logic based models, such as the classical logic model [62], situation theory IR model [63], and the possible world-based model [57].

## Chapter 3

# Investigation of Segmentation on the Vector Space Retrieval Model

Owing to the linguistic difference between Chinese texts and English texts, the indexing methods mentioned in Chapter 2 cannot be applied directly on Chinese texts. It is because these models are originally designed for handling English texts. An English sentence consists of words that are separated by word boundaries such as space or punctuations. Therefore, it is easy to extract meaningful words for indexing. However, Chinese is an agglutinative language. A Chinese sentence is a continuous character string without any explicit word boundaries. It is not easy to extract meaningful words or phrases from sentence for indexing. To handle Chinese text for information retrieval system, a sentence must be divided into indexable

terms suitable for the subsequent indexing process. The process of extracting indexable terms from Chinese text is called *segmentation*. For example, “計算機已經用於各個領域” (Computers have been used in every area) should be segmented as “計算機 已經 用於 各個 領域” (Computers, have been, used, every, area). However, this kind of accurate segmentation is a difficult task. Even Chinese speakers may disagree over how a sentence should be segmented because of the lack of a clear-cut definition on what constitutes a Chinese word. There are three commonly used segmentation approaches, namely, the character-based, the word-based, the  $N$ -gram approaches to perform the Chinese segmentation automatically. Among these three approaches, each of them has their own advantages and disadvantages on the performance of information retrieval system. We investigate the retrieval performance of each segmentation approach integrated with the vector space model which is an advanced information retrieval model [41].

## 3.1 Segmentation of Chinese Texts

### 3.1.1 Character-based segmentation

The character-based segmentation approach is the simplest approach to segment Chinese texts. It treats each individual character as an indexable term because a Chinese character is a ideogram which can be used individually with descriptive



meaning. In this approach, a continuous character string will be divided into its individual single character components, and then each of the component will be indexed separately. For example, “中國文學” (Chinese literature) will be segmented into “中國文學” (middle, country, language, study). An advantage of the character-based segmentation is that the indexing and retrieval processes are fast because this approach segments a sentence mechanically and does not require the use of a dictionary. In addition, the size of the Chinese character set is small (about 6,763 for GB coding scheme and 13,053 for BIG5 coding scheme) compared with that of meaningful Chinese words (over hundred thousands of words). This approach has been used in several experimental systems for Chinese [16] and Japanese text retrieval [30, 58, 59].

However, this approach is only appropriate for text retrieval using concepts that may be expressed by a set of single characters. In general, the character-based segmentation approach will introduce some kinds of errors in retrieval [51, 56]. The first one is the problem of incorrect parsing. Part of the character string may be falsely matched with the query due to the arbitrary combination of characters in sentences. For example, a document contains “發展中國家” (developing country) will be qualified with respect to the query “中國” (China). This kind of error is intrinsic and cannot be eliminated in the character-based segmentation method. In addition to the parsing error, it has the problem of losing the character



sequence. As the matching between query and document is performed at the character level, there is no positional information about character sequence in the query and in the document. For example, “日本” (Japan) will also be matched with “本日” (today).

The character-based approach also does not allow the incorporation of linguistic knowledge into the search process. Thus the division of word into characters will lose the original meaning of word. For example, “中國文學” (Chinese literature) will be indexed as “中國文學” (middle, country, language, study) rather than the actual meaning “中國文學” (China, literature). Moreover, Chinese language has a rich vocabulary such that a concept may often be expressed in multiple ways. However, the incorporation of thesaurus is difficult to achieve using the character-based approach.

### 3.1.2 Word-based segmentation

To avoid the problems encountered in the character-based approach, some studies [1, 82, 83] have suggested the word-based segmentation. The idea is to locate suitable word boundaries so that meaningful words can be extracted which are closed to our general knowledge about Chinese words. The word boundaries can also prevent the system from matching a word against parts of different words as in the character-based approach. The word-based approach is able to filter out

common function words easily by using a stopword list. It also facilitates the incorporation of thesaurus into the retrieval process. Moreover, this approach does not require a fixed morphology for concepts [56].

There are two existing methods that are derived from the word-based approach, namely, the dictionary-based method and the statistical method. The dictionary-based (or called rule-based) method [13, 33, 38, 48, 50, 79, 84, 85, 86, 89] segments the sentence into many legitimate words for indexing based on a predefined dictionary. The dictionary usually contains all the legitimate words that may be indexed. When segmenting a sentence, it first looks up the dictionary to find any consecutive characters in the sentence that can form a word. The resulting word sequence forms the indexable terms. However, in general, several legitimate word sequences may be obtained. The *maximum-matching* (or the longest matching) algorithm is often used to select the word sequence which contains the longest (or equivalently, the fewest) words. Some experiments have shown that this method has a good segmentation accuracy (above 98% correctly segmented [81]). This method has the advantage that the adopted lexical knowledge corresponds closely to our general knowledge about Chinese words. Moreover, the dictionary can also be reused in different context without much modification. Therefore, there are some systems adopted this segmentation approach, such as ABWS<sup>1</sup> [49], CASS<sup>2</sup> [37, 49], and

---

<sup>1</sup>ABWS - Association-Backtracting Word Segmentation.

<sup>2</sup>CASS - Chinese Automatic Segmentation System.



CDWS<sup>3</sup> [49]. These systems also have a good result in the segmentation accuracy (above 98% correct segmentation).

Statistical method [10, 14, 21, 53, 73, 76] is another word-based segmentation approach which relies on statistical information such as word and character (co-)occurrence frequencies in the training data. The training data is usually a set of manually segmented text. Given a set of manually segmented training texts, the probability  $P(S)$  of a character string  $S$  forming a word is calculated by the following equation.

$$P(S) = \frac{f_1}{f_2} \quad (3.1)$$

where  $f_1$  is the number of occurrence of  $S$  being segmented as a word in the training set and  $f_2$  is the number of occurrence of  $S$  in the training set. Therefore, given an input string to be segmented, the best solution is composed of a sequence of potential words  $S_i$  such that the value of  $\prod_i P(S_i)$  is the largest. Since statistical method requires the use of a manually segmented training set, this method is highly corpus-dependent. It is more sensitive to the word usage in a particular application area. The statistical information cannot be reused in other contexts. Moreover, it is very costly to prepare the manually segmented training set and the training set may be inconsistent because of the absence of a precise definition of words in Chinese.

---

<sup>3</sup>CDWS - The Modern Written Chinese Distinguishing Word System.

Nie [56] proposed to use a hybrid method in which dictionary is considered as the background and the statistical information as foreground knowledge. In our retrieval performance evaluation, we used the dictionary-based method because it does not need a manually segmented training set and it has a good segmentation performance (above 98% accuracy).

### 3.1.3 *N*-Gram segmentation

As shown in Section 3.1.2, the word-based approach requires a dictionary for segmentation, but a complete dictionary is not feasible for accurate segmentation in practice because there are many new words generating from time to time. To cope with the problems of the word-based approach, there is another segmentation method called the *N*-gram techniques [7, 8, 19, 46]. The advantages of this approach is that it is lexical knowledge independent because it does not require a dictionary or training set for segmentation. As a result, this approach does not require any linguistic processing such as part-of-speech tagging and stopword removal.

The main idea of *N*-gram approach is to divide a sentence into many overlapping *N*-character slices where *N* determines the number of characters in each slice. When segmenting a sentence, it starts from the first character of the sentence and groups the first *N* characters into a slice. Then this process repeats from the



second character and continues until the end of the sentence. As a result a set of overlapping  $N$ -character slices from the sentence will be generated. In Chinese, bi-gram (2-gram) are usually used because it has been estimated that about 73.6% of meaningful Chinese words are composed of two characters [12, 42]. For example, the sentence “關於地震的消息及其拯救工作” (News about the earthquake and the rescue task) will be divided into “關於 於地 地震 震的 的消 消息 息 及 及其 其拯 拯救 救工 工作” and each of the segmented term will be indexed individually.

One drawback of  $N$ -gram approach is the large amount of storage space used. Since in the character-based and the word-based approaches, the number of distinct words are fixed and some of the segmented terms can be removed by using a stopword list. Nevertheless, some of  $N$ -gram slices are meaningless and cannot be removed by using stopword list, such as the terms “於地”, “震的”, “的消”, “息及”, “其拯”, and “救工” in the example. Moreover, each  $N$ -gram slice is made up of different combination of the  $N$  characters. As the size of slice  $N$  increases, the number of different combination of these  $N$ -gram slice also increases considerably. As a result, the number of distinct terms for indexing also increases. These two reasons cause the  $N$ -gram approach uses much more storage space than the other two approaches.

## 3.2 Performance Evaluation of Three Segmentation Approaches

Since each of three segmentation approaches has their own advantages and disadvantages in storage and retrieval precision, we have performed a series of experiments to investigate the effect of each segmentation approach on the performance of an information retrieval system [41]. We adopted the vector space model as the retrieval model since it is a popular and advanced one. We have chosen three system characteristics for evaluation, namely, the retrieval performance, the time efficiency, and the amount of storage space.

### 3.2.1 Experimental Setup

To conduct vector space model retrieval, we employed a system known as SMART [70]. As the SMART system is used to handle English texts, we modified it to handle Chinese characters. The three segmentation approaches were applied on the same Chinese corpus and the same set of Chinese queries. The experiments were conducted on a SUN SPARC-20 workstation. The document corpus is extracted from the news articles of a local newspaper, *Wen Wei Po* (文匯報). This collection contains 6,127 articles and its topics vary from world politics to local entertainment occupying about 10 Megabytes in size. 10 queries were used to evaluate the retrieval performance on the collection. For example, one of the

queries was “第二次世界大戰和抗日戰爭的歷史以及全球各地的記念活動” (The history and memorial activities of the Second World War and the anti-Japan War). The relevance judgment of each query against every document in the corpus is available. The size of the dictionary used for the word-based approach contains 54,203 Chinese words.

### 3.2.2 Experimental Results

#### Retrieval Performance

The retrieval performance of an information retrieval system can be measured quantitatively by the recall and precision values [71]. Recall ( $R$ ) is defined as the proportion of relevant material that are retrieved from the collection and precision ( $P$ ) is the proportion of the retrieved material that is relevant. Thus, the higher the recall and precision value, the better the performance of the information retrieval system will be.

$$\text{Recall } (R) = \frac{\text{No. of retrieval relevant documents}}{\text{Total no. of relevant documents}}$$

$$\text{Precision } (P) = \frac{\text{No. of retrieved relevant documents}}{\text{Total no. of retrieved documents}}$$

To measure the retrieval performance of the three segmentation approaches, we use their precision at various recall values and their 11-point average precision.



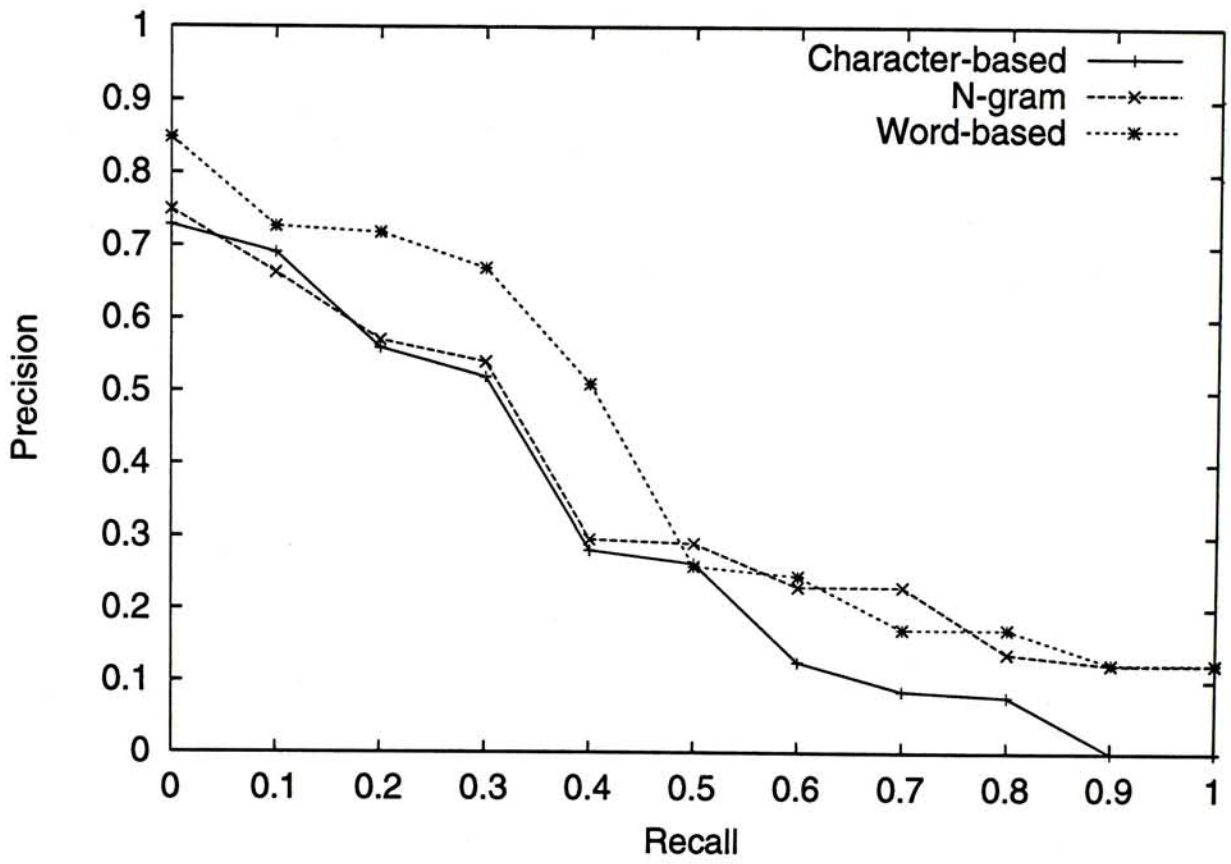


Figure 3.1: Recall-precision curve of the three segmentation approaches.



Figure 3.1 shows the recall-precision curves of the three approaches, and Table 3.1 gives their 11-point average precision values. The retrieval result shows that the word-based segmentation approach has a better overall retrieval effectiveness at most of the recall values. Hence, it outperforms the other two approaches. Moreover, it also has the largest 11-point precision value, as shown in Table 3.1. It is due to the fact that in the word-base segmentation, the segmented results are legitimate words. Therefore, their occurrence frequency are more discrete than that in the character-based and the  $N$ -gram approaches. As a result, it is easy to extract the correct words from the collection.

	11-point average precision	% of improvement
Character-based	0.3027	0%
Word-based	<b>0.4151</b>	<b>+37%</b>
Bi-gram	0.3592	+19%

Table 3.1: 11-point average precision of the three segmentation approaches.

Although the character-based segmentation approach is the simplest approach, it was the least effective than the other two approaches. The character-based approach has the smallest 11-point average precision value and its recall-precision curve is below the other two. It is because the character-based segmentation approach has sustained the parsing and sequencing errors.

The performance of the bi-gram approach varies at different recall values. At lower recall value (0.0 - 0.5), the bi-gram approach is slightly better than the character-based approach except at the recall value of 0.1. However, at high recall

value (0.5 - 1.0), the bi-gram approach is much better than the character-based approach and its curve is closed to that of word-based approach.

### Time Efficiency

Besides the accuracy of the retrieval performance, the response time of various operations is also an important factor affecting the performance. We compared the time required for three main operations in Chinese information retrieval system, namely, segmentation, indexing, and retrieval. Table 3.2 shows the CPU time recorded for various operations performed on the same SUN SPARC-20 workstation.

	Character-based	Word-based	Bi-gram
Segmentation	518.68 secs	2,119.56 secs	550.37 secs
Indexing	145.62 secs	103.43 secs	515.45 secs
Retrieval	1.90 secs	1.14 secs	0.99 secs

Table 3.2: Time required for various operations.

From the result shown in Table 3.2, the word-based approach used the longest time for segmentation. It is because the segmentation method of the word-based approach involves searching the appropriate words of each sentence from the dictionary. This searching process requires a considerable amount of time. On the other hand, the time required for the character-based and the bi-gram approaches were similar and much less than that of the word-based approach. Owing to the fact that these two approaches only divide a sentence in certain pattern mechanically



without the need of searching a dictionary.

	Number of indexed terms
Character-based	3,236,281
Word-based	2,148,257
Bi-gram	4,093,808

Table 3.3: Number of terms indexed by each approach after segmentation.

The amount of time used in indexing is proportional to the amount of the terms to be indexed. The more the terms for indexing, the longer time it needs. So, as shown in Table 3.2, the word-based approach consumed the least amount of time for indexing. Since the maximum-matching algorithm of the word-based segmentation approach gives the least amount of words for each sentence, it will produce the least amount of indexable terms after segmentation, as shown in Table 3.3. In this way, it consumed the least amount of time for indexing.

As mentioned in Section 3.1.3, the *N*-Gram approach will produce a large amount of distinct terms for indexing which cannot be removed by a stoplist. Thus, as shown in Table 3.3, this approach has the largest amount of terms to be indexed. As a result, it takes the longest time for indexing.

### Storage Requirement

Similar to the time efficiency, the amount of storage space consumed is also proportional to the amount of distinct terms for indexing. As shown in Table 3.4, the bi-gram approach requires the largest amount of storage than the other ap-

proaches. Since, as mentioned in Section 3.1.3, there is free combination of characters in bi-gram slices, the number of distinct terms is much more than that of the word-based and the character-based approaches. Since both the word-based and the character-based approaches have a fixed size of distinct indexable terms, their storage requirement are less than that of  $N$ -gram approach. The word-based approach, on the other hand, consumes the least amount of storage space because the maximum-matching algorithm and the stopword removal reduce the actual number of distinct terms for indexing.

	Index Storage Size
Character-based	10,760 kbytes
Word-based	10,070 kbytes
Bi-gram	23,762 kbytes

Table 3.4: Storage requirement of the three segmentation approaches.

### 3.2.3 Discussion

From the performance results of the three different segmentation approaches in the retrieval performance, the time efficiency, and the storage requirement, the character-based approach has the least performance than the other two approaches. Since many meaningful Chinese words are composed of more than one characters, character-based matching will lead to poor retrieval effectiveness because of the parsing and sequencing errors. The word-based approach, on the other hand, gives



a better result except the time for segmentation because it requires to search the dictionary during segmentation. The  $N$ -gram approach will be a better choice when a dictionary is not available because its performance is better than the character-based approach. The drawbacks of this approach are larger storage required and longer time for indexing.

Kwok [31] also conducted experiments to evaluate the performance of these three segmentation approaches using the TREC-5 [78] Chinese document collection. His results are similar to our experimental results in that the character-based approach is the least effective one among the others. Moreover, the 11-point average precision of the word-based approach (0.452) is about 1% better than that of the bi-gram approach (0.448). It is because the word-based approach in his experiments divided Chinese sentence into short words composed of one to four characters while the segmented results in our experiments include long compound words. Since, in Chinese, long compound words have more complex and unambiguous meaning than that of short words, the resulting precision of the word-based approach in our experiments will be higher than that of Kwok's experiments. In addition to using short word for indexing, these short words were also represented in the form of their single-character components in Kwok's experiments. Although this method increases the recall of the retrieved results, the precision will also be sacrificed because of the mentioned parsing and sequencing errors of the character-

based approach. These two reasons reduce the precision of the word-based approach in such a way that it is only about 1% better than the bi-gram approach.

As shown in Table 3.4, the storage size of the index files are about 100%-200% of the original document collection size. In order to reduce the size of the storage overhead, we use the signature files and propose a partitioning method so as to increase the search speed. Our proposed method is based on the word-based segmentation approach because of its superior performance. The detailed presentation will be provided in Chapter 4 and Chapter 5.

## Chapter 4

# Signature File Background

As mentioned in Section 2.1.2, the inverted file indexing method consumes a large amount of storage overhead for its index file, about 100%-200% of the original corpus size, as in our experiments (Table 3.4). Cardenas [6] also mentioned that the inverted directory or index becomes a large database itself in large, highly inverted databases. Besides, it is also expensive to perform update and transposition of the inverted file in a dynamic environment. In our research, we investigate alternatives to the inverted file technique for Chinese text indexing and retrieval. Our technique is based on the signature files. We propose a new partitioning method for signature files so as to reduce the retrieval response time as well as the number of false drops. Before we give a detailed description of our proposed model, an overview of the signature file technique will be presented first.

Signature files were originally used to save space on the edge-notched cards



system [39], but some earlier studies [22, 39, 77] suggested using the signature files on information retrieval systems, and this principle has been applied on textual databases such as the news databases [75], multiattribute retrieval in relational databases [3, 44], multimedia office filing [18], and chemical databases for DNA matching [54]. The use of the signature files is motivated by its advantages of small storage overhead, and relatively simple data structure for insertion [26].

A signature file is basically an abstraction of the corpus and acting as an index file. It will be searched first so as to eliminate most of the unqualified documents. The main purpose of a signature file is to reduce the search space in the primary database. In this approach, each index term from documents is transformed into a fixed-length binary bit pattern, called a word signature. The mapping between the index term and the word signature can be transformed by two methods, namely, hashing function or lookup table.

In order to reduce the storage overhead, several word signatures are combined to form a block signature when stored in signature file. There are two existing coding schemes to perform the signature combination, namely, the disjoint coding [60], and the superimposed coding [64]. In the disjoint coding scheme, word signatures will be concatenated together to form a block signature. The resulting signature length will be longer than that of the word signatures. This coding scheme has been applied on indexed descriptor file [60]. In superimposed coding scheme, the

word signatures will be superimposed (inclusive OR operation) together to form a block signature such that the resulting block signature length will be equal to that of the word signature. Superimposed coding is more commonly used to generate the block signature in the signature file approach. Our proposed technique is based on the superimposed coding scheme.

## 4.1 Superimposed coding

In this approach, a document is divided into many block signatures containing the same number of unique, nontrivial words. These words will be first transformed into their word signatures which are bit-string of length  $m$  consisting of exactly  $w$  1's and  $(m - w)$  0's. The position of the bits set are randomly generated within the  $m$  positions. Then for each  $s$  word signatures, they will be superimposed (inclusive ORed) to form a block signature which has the same signature length  $m$  as that of its word signature components. It results in the reduction of the storage overhead in the index file since each entity in signature file contains more than one index term. For example, as shown in Figure 4.1, a document contains two words, "chinese" and "database". These two words are transformed into their word signatures, respectively. Then they are superimposed (inclusive ORed) to form a block signature for storage.

When a query is submitted to retrieve relevant documents, the query term

Indexed Word	Word Signature		
chinese	0001 1100 0001		
database	0010 1101 0000		
<hr/>			
Block Signature (OR):	0011 1101 0001		
Query	Query Signature		Result
1. chinese	0001 1100 0001		Match
2. data	1010 0010 1000		Not Match
3. index	0010 1001 0001		False Drop

Figure 4.1: Example of superimposed coding.

is first transformed to a query signature in a similar way and then every block signature in the signature file will be compared with the query signature by using simple AND operation. If the query word is one of the words that composed the block signature, then all the positions of 1's in the query signature must also be set in the block signature. Thus, the corresponding block signature will be identified as qualified block. In the first retrieval example of Figure 4.1, the block signature will be qualified as containing the first query term, "chinese", since the query term is indeed in the text. In retrieving the second query term, "data", since some of the bit set in the query signature are not set in the block signature, this block signature will be regarded as not containing the query term. This block signature will be filtered out. In the third case, the block is qualified as containing the query term because every bit set in the query signature are also set in the block



signature. However, this block signature, actually, does not contain the query term, "index", this block signature is falsely identified as containing the query term. This is because of the combinational error of the superimposition coding scheme. The corresponding matched bit set in the block signature are set by irrelevant words in the document other than the query word during signature superimposition. This is called false drop, which is an inherent property in signature file.

As shown in this example, any block signatures containing the query terms will be retrieved together with some false drops. In summary, a query signature acts as a filter to reject most, but not all, of the unqualified block signatures from the corpus. These false drops can be eliminated by further matching between the retrieved documents with the query term or they can be returned to the users as false drops. Nevertheless, both of them degrade the performance of system.

## 4.2 False drop probability

As shown in Section 4.1, the introduction of false drops in the signature file technique degrades the retrieval performance. False drop is defined as the block signature that is qualified to be containing the query term, but actually it does not. We can measure it quantitatively by the false drop probability  $P$  [55] which is defined as the ratio of false drops to the number of unqualified signatures. The number of unqualified signatures depends on the size of the signature and also on

the storage overhead.

$$\text{False Drop Probability} = \frac{\text{No. of retrieved irrelevant block signatures}}{\text{Total no. of irrelevant block signatures}}$$

For a signature file with the signature length  $m$ , the weight of word signature  $w$  (number of 1's in a word signature), and each block signature is the result of superimposing  $s$  word signatures. When all of its block signatures are checked with a simple query signature, the false drop probability  $P$  of this signature file will be [64, 67]:

$$P = \left(1 - \left(1 - \frac{w}{m}\right)^s\right)^w \quad (4.1)$$

The derivation of the false drop probability  $P$  (Equation 4.1) has been provided in Appendix B. In order to reduce the number of false drops, we have to minimize the false drop probability  $P$ . The false drop probability  $P$  depends on the signature length, the weight of the block signature and the weight of the query signature. When the same number of distinct words superimposed to form a block signature, the "density" of 1 in the signature decreases as the length of the signature  $m$  increases. The chance of getting false drops will decrease. Thus, the number of false drops will decrease, but the increase in the signature length  $m$  will also increase the storage overhead. As the weight of query signature increases, the number of false drops will also decrease since there are more bit positions in block signature have to be matched in order to pass through the filtering process. However, the increase in the weight of block signature will increase the number of false drops



because there are more bit positions set to 1 in block signature. It increases the chance of matching a query signature. It has been shown that in order to minimize the false drop probability  $P$ , the expected number of zeros and ones in a signature must be the same [17, 28]. Thus, the weight of block signature should be half of signature length  $m$ . As the formation of block signature is the result of superimposing fixed number  $s$  of word signatures, the weight of block signature is only an expected number due to the collision of bit position in superimposition. The optimal condition is only satisfied on statistical grounds.

Since the signature length  $m$  and the weight of word signature  $w$  are the factors that will affect the false drop probability  $P$ . We can make use of the Equation 4.1 to determine both  $m$  and  $w$  at optimal condition given the storage overhead, the derivation is shown in Appendix B. Equation 4.2 and Equation 4.3 show the signature length  $m$  and the weight of word signature  $w$  at the given false drop probability  $P$  under optimal condition.

$$w = \left(\frac{1}{\log 2}\right) \log \left(\frac{1}{P}\right) \quad (4.2)$$

$$m = \left(\frac{1}{\log 2}\right)^2 s \log \left(\frac{1}{P}\right) \quad (4.3)$$

Since Equation 4.1 only shows the probability of the occurrence of false drops, the actual number of false drops will be proportional to the number of block signatures as well as to the size of the collection. The elimination of false drops affects the system performance seriously, especially for large document collections [47].



## Chapter 5

# Partitioned Signature File Based On Chinese Word Length

As mentioned in Chapter 4, the design of signature files needs to consider two issues: false drops and search speed. The introduction of false drops can seriously degrade the search speed since a further matching is needed between the query term and the filtered documents. On the other hand, if the documents are returned directly to the users without false drop elimination, the precision of the retrieval can be affected. As the number of false drops produced by a query is proportional to the number of unqualified documents. It increases with the corpus size. The low search speed of signature files is the result of its simple file structure. As the query signature is checked sequentially against all block signatures in the signature

file, the search time depends on the number of block signatures in the signature file which, in turn, is directly proportional to the size of the corpus. It results in an unacceptable performance when the corpus is sizable.

Since false drops affect system performance, some methods have been proposed to reduce the false drops probability [27, 28, 47]. To alleviate the problem of low search speed, some efficient searching methods have been proposed [45], including the indexed descriptor file method [60] and its variant S-tree [20], the two-level superimposed coding method [11, 67], and the partitioned signature file method [23, 43, 44, 88]. These methods, in general, organize the signature file in a way such that only a small number of the signatures are accessed in response to a query.

We investigate the use of the signature files in Chinese information retrieval system such that the storage overhead used will be less than that of the inverted file. We propose a new partitioning method for Chinese signature file, referred to as PSFC<sup>1</sup> [80], which is based on the characteristic of Chinese words. Although some partitioning methods of signature file have been previously proposed, e.g. horizontal partitioning [23], vertical partitioning [23], and deterministic algorithm [43], they are mainly used for reducing the search time (running on a parallel environment) or for reducing the search space. The problem of false drops has not been addressed. Compare with the traditional single signature file approach, our partitioning method, PSFC, can improve the search speed over the signatures.

---

<sup>1</sup>PSFC - Partitioned Signature File for Chinese.

This partitioning method has an advantage in sequential environment since not every partition needs to be searched in response to a query. Thus, the number of block signatures required to be checked can be reduced dramatically. Not only can it reduce the time for the further matching between the filtered documents and the query, it can improve the precision of the retrieved result as well. We devise a general scheme for controlling the trade-off between false drops and storage overheads while maintaining the search space reduction. An analytical study is presented to support the claims of our method. We have implemented PSFC and evaluated it using the TREC-5 (Text REtrieval Conference) Chinese collection [78]. The experimental results, presented in Chapter 7, confirm that the advantages of our method and demonstrate its superiority over the traditional single signature file approach.

## 5.1 Fixed Weight Block (FWB) Signature File

Owing to the linguistic difference between Chinese and English, Chinese text has to be segmented into indexable terms (i.e. words) before indexing. From the performance results of the three commonly used segmentation approaches in Chapter 3, we make use of the word segmentation method to integrate with the Chinese signature file. Since the word segmentation approach uses the least amount of storage and has a better retrieval performance [41]. Moreover, the signatures of



dictionary words can be generated first, and assigned to the document terms during segmentation. It reduces the time to generate the signature during indexing and also eliminates signature duplication of different dictionary words. It can reduce the number of false drops in retrieval rather than using hashing function to generate the signature at runtime [27].

In the traditional signature file, all block signatures are stored sequentially in a single signature file. This simple file structure has the advantage for insertion of new signatures easily. However, it also reduces the search speed. As the whole corpus is stored in a single signature file, every block signature in the signature file has to be checked sequentially with the query signature. The time taken for the retrieval would be very long if the corpus is sizable. We propose a partitioned signature file method that can improve the search speed in Chinese texts.

The superimposed coding component in our method is based on the Fixed Weight Block (FWB) method proposed by Leng and Lee [47]. The traditional superimposed coding method, as described in Chapter 4, is called the Fixed Size Block (FSB) which has its block signatures containing the same number of distinct terms after stopword and duplicate removal. Under the optimal condition, only the expected weight of the block signature is constant, but there is a nonzero probability in the FSB method for block signatures with a large number of ones to be generated. Thus, it increases the number of false drops and the optimal

condition is only satisfied on statistical ground. The FWB method alleviates this problem by fixing the weight of each block signature instead of fixing the number of word signatures superimposed together. The simulation of Leng and Lee [47] shows that the FWB method is superior to the FSB method in reducing false drops.

Before we present our partitioning method, PSFC, we first review the characteristic of employing the traditional single signature file using FWB method on Chinese document indexing. The notations used in this section are provided in Appendix A. Indexable terms are usually generated after the word segmentation process. A block signature is formed by superimposing various number of word signatures together until the weight of a block signature reached. Therefore, the weight of each block signature is, or almost, constant. Since the number of word signatures composing a block is not fixed, the false drop probability as shown in Equation 4.1 cannot be used to measure the number of false drops. Instead, we use the random false drop rate to measure the chance of false drops. The random false drop rate (*RFDR*), which is defined as the probability that an unqualified document emerges as a random false drop, is estimated as follows [72]:

$$\begin{aligned}
 RFDR &= \frac{\binom{W_{B_i}}{w_Q}}{\binom{m}{w_Q}} \\
 &= \frac{W_{B_i}(W_{B_i} - 1) \cdots (W_{B_i} - w_Q + 1)}{m(m - 1) \cdots (m - w_Q + 1)} \\
 &\approx \left(\frac{W_{B_i}}{m}\right)^{w_Q}
 \end{aligned} \tag{5.1}$$

Since in FWB method, the weight of a block signature is fixed. We have  $W_{B_1} =$

$W_{B_2} = \dots = W_{B_i} = W_{B_{avg}}$ . Equation 5.1 becomes:

$$\begin{aligned} RFDR &\approx \left( \frac{W_{B_{avg}}}{m} \right)^{w_Q} \\ &= (b)^{w_Q} \end{aligned} \quad (5.2)$$

To retrieve a query word  $v_i$ , the expected number of false drops of the query word  $v_i$  is:

$$\begin{aligned} \text{Expected No. of False Drops of } v_i &= (b)^{w_{v_i}} p_i \\ &= (b)^{w_{v_i}} (l - f_i) \end{aligned} \quad (5.3)$$

The term  $f_i$  in Equation 5.3 is the document frequency of the word  $v_i$  in the corpus and it also indicates the number of block signatures containing this word  $v_i$ . The expected number of false drops for any word from a dictionary containing  $V$  unique words, is:

$$\begin{aligned} &\text{Expected No. of False Drops} \\ &= \sum_{i=1}^V (b)^{w_Q} p_i K(Q_i) \\ &= \sum_{i=1}^V (b)^{w_Q} (l - f_i) K(Q_i) \\ &= \frac{1}{V} (b)^{w_Q} (Vl - \sum_{i=1}^V f_i) \end{aligned} \quad (5.4)$$

assume uniform query probability, i.e.  $K(Q_i) = \frac{1}{V}$

In the single signature file method, each query signature has to be matched with all block signatures. The search space is equal to the number of block signatures,  $l$ ,



in the signature file. The search space is a measure of the response time since the retrieval time is proportional to the number of block signatures in the signature file required to be checked. Hence, we have:

$$\text{Search Space} = l \quad (5.5)$$

## 5.2 Overview of PSFC

PSFC is a partitioning method of Chinese signature file based on the length of the Chinese words. Since there are some characteristics of Chinese words that are different from its English counterpart, we make use of these Chinese words characteristics in our proposed model.

First of all, the variation in the length of Chinese words is much less than that in English. In English, it is very common that the word length varies from one to more than ten characters. Each English word is composed of a number of alphabets while each alphabet does not have meaning individually. However, Chinese characters, called *hanzi*, are ideograms which can be used individually with descriptive meanings. They can also be strung together to form a longer word with more complex meanings. Consequently, the variation in length of Chinese word is very small. From the statistical result of Lau [42], as shown in Table 5.1, most of the Chinese words, above 99%, are composed of less than five characters, and a large portion of them contains two characters only.

	% of occurrence
1-character words	12.1%
2-character words	73.6%
3-character words	7.6%
4-character words	6.4%
>5-character words	0.2%

Table 5.1: Percentage of occurrence of Chinese words.

Second, in general, the longer is a Chinese word, the more specific its meaning will be. For instance, the word “貨車” (truck) has a more specific meaning than the word “車” (vehicle). As a longer word in Chinese usually has a more specific meaning, a higher retrieval precision is generally achieved for a query entailing a long Chinese word.

Owing to the above characteristics of Chinese words, we propose a method called PSFC as shown in Figure 5.1. The whole signature file is partitioned into many signature files, each of them contains a set of block signatures of Chinese words with a specified length. Specifically, only the partition containing the specified Chinese word length will be searched given a query term. The reduction in searching the number of block signatures cannot only shorten the response time but it can also increase retrieval precision. It guarantees that the query word will not appear in other partitions, thus there is no need to compare the query signature with other words with different lengths as in the traditional signature files.

In PSFC, the signature file design parameters, such as the signature length, the weight of each signature, the block weight, etc, of each partition can be different.

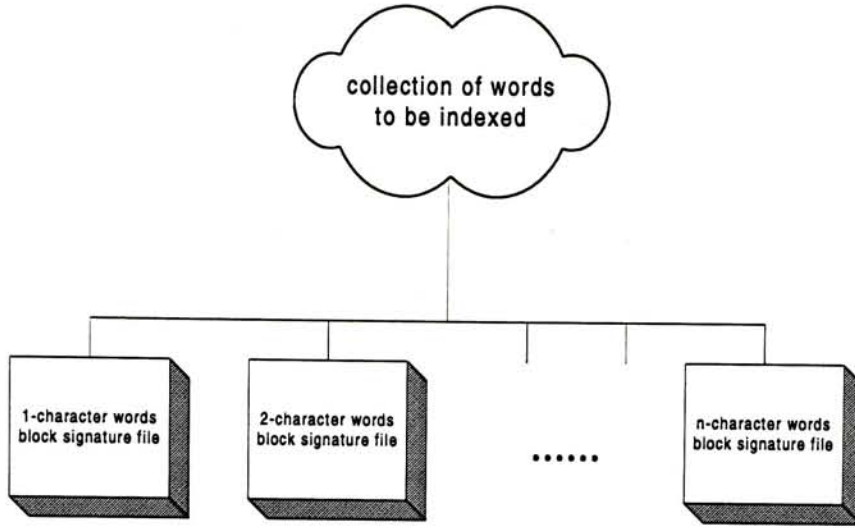


Figure 5.1: Design of the partitioning in PSFC.

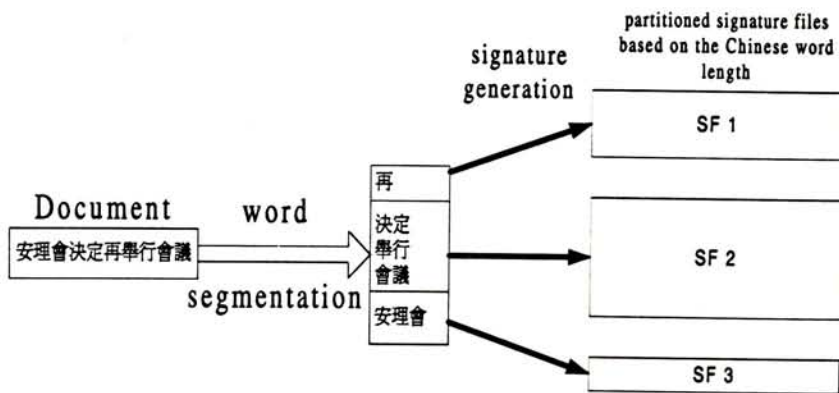


Figure 5.2: Example of partitioned signature files based on Chinese word length.



The next issue is how to determine these parameters so that the false drop or the storage overhead can be reduced while maintaining the advantage in search space reduction.

The number of partitions in PSFC can be determined based on the distribution of the word length in the documents to be indexed. It usually uses less than five partitions since Chinese words longer than five characters are rare, as shown in Table 5.1. Figure 5.2 is an example of the signature file generation using the PSFC method.

We first analyze the reduction in search space for a partitioning design that uses the same signature length, the same weight of signature, and the same weight of block signature for every partition. As the single signature file is partitioned into  $n$  signature files, and each of them only contains the block signatures of Chinese words with a specified length. In this way, the number of block signatures  $l_i$  ( $i = 1, 2, \dots, n$ ) in each partition is less than that in the single signature file (i.e.  $l$ ) because each of them only contains a portion of the block signatures of the single signature file. In query time, the query signature will match against only the partition which contains the block signatures of words with the same length. Therefore, the expected search space is:

$$\text{Expected Search Space} = \sum_{i=1}^n \frac{V_i}{V} l_i \quad (5.6)$$

The term  $\frac{V_i}{V}$  is the probability that the partition  $i$  is searched and it depends on

the number of words in the dictionary having the specified length  $i$ . Since the block signatures in each partition are only part of the single signature file, it is clear that the expected search space (Equation 5.6) of our method is less than the expected search space (Equation 5.5) of the traditional single signature file method.

The expected number of false drops in PSFC is:

Expected no. of false drops in PSFC

$$\begin{aligned}
 &= \frac{1}{V} (b)^{w_Q} \sum_{i=1}^V p_i \\
 &= \frac{1}{V} (b)^{w_Q} \left( \underbrace{p_{11} + \dots + p_{1V_1}}_{V_1} + \underbrace{p_{21} + \dots + p_{2V_2}}_{V_2} + \dots + \underbrace{p_{n1} + \dots + p_{nV_n}}_{V_n} \right) \\
 &= \frac{1}{V} (b)^{w_Q} (p_{11} + \dots + p_{1V_1}) + \frac{1}{V} (b)^{w_Q} (p_{21} + \dots + p_{2V_2}) + \dots + \\
 &\quad \frac{1}{V} (b)^{w_Q} (p_{n1} + \dots + p_{nV_n}) \\
 &= \frac{1}{V} (b)^{w_Q} \sum_{i=1}^{V_1} (l_1 - f_{1i}) + \frac{1}{V} (b)^{w_Q} \sum_{i=1}^{V_2} (l_2 - f_{2i}) + \dots + \frac{1}{V} (b)^{w_Q} \sum_{i=1}^{V_n} (l_n - f_{ni}) \\
 &\hspace{20em} (5.7') \\
 &= \frac{1}{V} (b)^{w_Q} \left( (V_1 l_1 + V_2 l_2 + \dots + V_n l_n) - \left( \sum_{i=1}^{V_1} f_{1i} + \sum_{i=1}^{V_2} f_{2i} + \dots + \sum_{i=1}^{V_n} f_{ni} \right) \right) \\
 &= \frac{1}{V} (b)^{w_Q} \left( \sum_{j=1}^n V_j l_j - \sum_{i=1}^V f_i \right) \\
 &\hspace{20em} (5.7)
 \end{aligned}$$

where the term  $\frac{1}{V} (b)^{w_Q} \sum_{i=1}^{V_k} (l_k - f_{ki})$  in Equation 5.7' is the weighted expected number of false drop in partition  $k$ . As we have just explained, the following

inequality holds:

$$\begin{aligned} \sum_{i=1}^n \frac{V_i}{V} l_i &< l \\ \frac{1}{V} \sum_{i=1}^n (V_i l_i) &< l \\ \sum_{i=1}^n (V_i l_i) &< V l \end{aligned} \quad (5.8)$$

Therefore, the expected number of false drops of PSFC, as expressed in Equation 5.7, is less than that of the single signature file, as shown in Equation 5.4.

As a result, the expected number of false drop and the expected search space in PSFC is in general less than that in the single signature file. Moreover, its storage overhead is:

$$\text{Storage overhead of PSFC } (S_1) = \sum_{i=1}^n m l_i \quad (5.9)$$

### 5.3 Design Considerations

The above partitioning method can shorten the response time by reducing the search space and also reduce the number of false drops, but it causes more incomplete block signatures than that in the single signature file method. Incomplete block signatures are the signatures which have its weight less than the fixed block signature weight. They come from short documents or from the remaining words of document composing a block signature. The sum of the block signatures in all



partitions is more than that in the single signature file. As a result, it leads to a larger storage overhead than the single signature file.

Keeping the number of block signatures in each partitions, i.e.  $l_i, i = 1, 2, \dots, n$  to be the same as the above design can maintain the reduction of the search space. Under this design, if we decrease the signature length in each partition, we can reduce the storage overhead. Unfortunately, at the same time, it will increase the random false drop rate and thus increasing the expected number of false drops.

We aim to devise a general scheme for controlling the trade-off between the reduction of false drop and the reduction of storage overhead while maintaining the advantage of significant search space reduction. To achieve this purpose, we first determine the bound of the expected number of false drops. It is reasonable to set the bound of the expected number of false drops of PSFC equal to that of the single signature file. In this way, we can guarantee that our method has a better precision.

In order to maintain the number of block signatures to be the same as the above design, we have to fix the blocking factor of each partition to the same as in the single signature file. Since in FWB method, each block signature contains different number of word signatures, the blocking factor is defined as the average number of word signatures composing a block signature. To find the blocking factor, we need to consider the weight of the block signature ( $W_B$ ) and the sum of the term

weights in the block signature ( $W_B^+$ ).

text word	word signature
chinese	1100 1010
text	1010 1100
block signature	1110 1110

(superimposition)

Figure 5.3: Superimposing example.

For example, in Figure 5.3, the two words "chinese" and "text" superimposed to form a block signature. The weight of the block signature ( $W_B$ ) is 6 while the sum of the weight in this block signature ( $W_B^+$ ) is  $4+4=8$ . Since each block signature in FWB method is composed of different number of block signatures, it is better to use the  $W_B^+$  value to calculate the blocking factor rather than the  $W_B$  value, it was because there is some collision during superimposition, as shown in Figure 5.3.

For a block signature  $B_j$  which is composed of  $N$  word signatures, and  $w_{i,j}$  is the weight of word signature of the  $i$ -th term in the block signature  $B_j$  such that  $i \in [1, N]$ . Then the probability for a bit to be '0' in word signature of term  $i$  will equal to  $1 - \frac{w_{i,j}}{m}$ . The probability of a bit which is set to '1' within the block

signature  $B_j$  will be:

$$\begin{aligned}
 & 1 - \left[ \left(1 - \frac{w_{1,j}}{m}\right) \left(1 - \frac{w_{2,j}}{m}\right) \cdots \left(1 - \frac{w_{N,j}}{m}\right) \right] \\
 & \approx 1 - \left(1 - \frac{1}{m}\right)^{w_{1,j} + w_{2,j} + \cdots + w_{N,j}} \\
 & = 1 - \left(1 - \frac{1}{m}\right)^{W_{B_j}^+}
 \end{aligned} \tag{5.10}$$

Equation 5.10 is valid only if  $m \ll w_{i,j}$ . Then, we can base on the above equation to formulate the expected weight  $W_{B_j}$  of the block signature  $B_j$ , as discussed in Leng and Lee [47].

$$\begin{aligned}
 W_{B_j} &= m \left[ 1 - \left(1 - \frac{1}{m}\right)^{W_{B_j}^+} \right] \\
 W_{B_j}^+ &= \frac{\ln\left(1 - \frac{W_{B_j}}{m}\right)}{\ln\left(1 - \frac{1}{m}\right)} \\
 \frac{W_{B_j}^+}{W_{B_j}} &= \frac{\ln\left(1 - \frac{W_{B_j}}{m}\right)}{W_{B_j} \ln\left(1 - \frac{1}{m}\right)} \\
 \frac{W_{B_j}^+}{W_{B_j}} &= \mu
 \end{aligned} \tag{5.11}$$

In Fixed Weight Block (FWB) method, the weight of the block signature is fixed in all blocks,  $W_{B_j}$  is equal to  $W_B$  and thus  $W_{B_j}^+ = W_B^+$ . Moreover, it has been shown that the ratio  $\mu$  is constant as long as the ratio between the signature length  $m$  and the weight of block signature  $W_B$  is fixed. Using Equation 5.11, we can define the blocking factor as follows:

$$\text{Blocking Factor} = \frac{W_B^+}{W_B} \tag{5.12}$$



For a partition  $x$ , with a vocabulary size  $V_x$ , the expected number of false drops in this partition is:

$$E_x = \frac{1}{V_x} (b)^{w_x} \sum_{i=1}^{V_x} p_{xi} \quad (5.13)$$

To calculate the total expected number of false drops of all partitions,  $E$ , we calculate the sum of the product of the expected number of false drops in each partition  $E_x$  and the probability of occurrence of the words with specified length as follows:

$$\begin{aligned} E &= \sum_{i=1}^n \frac{V_i}{V} E_i \\ &= \frac{V_1}{V} E_1 + \frac{V_2}{V} E_2 + \cdots + \frac{V_n}{V} E_n \\ &= \frac{V_1}{V} \frac{1}{V_1} (b)^{w_1} \sum_{i=1}^{V_1} p_{1i} + \frac{V_2}{V} \frac{1}{V_2} (b)^{w_2} \sum_{i=1}^{V_2} p_{2i} + \cdots + \frac{V_n}{V} \frac{1}{V_n} (b)^{w_n} \sum_{i=1}^{V_n} p_{ni} \quad (5.14) \end{aligned}$$

From Equation 5.14, if the expected number of false drops  $E_x$  ( $x = 1, 2, \dots, n$ ) in each partition is the same as that of the single signature file, then the total expected number of false drops,  $E$ , is also equal to that of the single signature file as expressed in Equation 5.4.

$$\frac{1}{V} (b)^w \sum_{i=1}^V p_i = \frac{1}{V_1} (b)^{w_1} \sum_{i=1}^{V_1} p_{1i} = \frac{1}{V_2} (b)^{w_2} \sum_{i=1}^{V_2} p_{2i} = \cdots = \frac{1}{V_n} (b)^{w_n} \sum_{i=1}^{V_n} p_{ni}$$

The weight of the word signature of different partitions ( $w_1, w_2, \dots, w_n$ ) can be

calculated. For example the weight of word signature  $w_x$  of partition  $x$  is:

$$\begin{aligned}
 \frac{1}{V_x} (b)^{w_x} \sum_{i=1}^{V_x} p_{xi} &= \frac{1}{V} (b)^w \sum_{i=1}^V p_i \\
 (b)^{w_x} &= \frac{V_x}{V} (b)^w \frac{\sum_{i=1}^V p_i}{\sum_{i=1}^{V_x} p_{xi}} \\
 (b)^{w_x} &= \frac{V_x}{V} (b)^w \frac{\sum_{i=1}^V (l - f_i)}{\sum_{i=1}^{V_x} (l_i - f_{xi})} \\
 (b)^{w_x} &\approx (b)^w \frac{l}{l_x} \\
 w_x &\approx w + \log_b \frac{l}{l_x} \tag{5.15}
 \end{aligned}$$

Since  $\frac{l}{l_x} > 1$  and  $b < 1$ , the resulting weight of the word signature in partition  $x$  is less than that of the single signature file because the term  $\log_b \frac{l}{l_x}$  is negative.

In fact, we can control the trade-off between the storage overhead and the expected number of false drops using the relationship between the random false drop rate (*RFDR*) of partition  $x$  ( $x = 1, 2, \dots, n$ ) and that of the single signature file as follows:

$$\begin{aligned}
 (b)^w &= \alpha (b)^{w_x} \\
 w_x &= w + \log_b \left( \frac{1}{\alpha} \right) \tag{5.16}
 \end{aligned}$$

The parameter  $\alpha$  can be varied by the user to achieve the desired trade-off. When the value of  $\alpha$  is set to  $\frac{l_x}{l}$ , Equation 5.16 will be equal to Equation 5.15. The

expected number of false drops of partition  $x$  is:

$$\begin{aligned}
 E_x &= \frac{1}{V_x} \alpha (b)^{w_x} \sum_{i=1}^{V_x} (l_x - f_{xi}) \\
 &= \frac{1}{V_x} \alpha (b)^{w_x} (V_x l_x - \sum_{i=1}^{V_x} f_{xi}) \\
 &\approx \alpha (b)^{w_x} l_x \quad l_x \ll f_{xi} \\
 &= (b)^w \frac{l}{l_x} l_x \quad (b)^w = \frac{l_x}{l} (b)^{w_x} \\
 &= (b)^w l
 \end{aligned} \tag{5.17}$$

When  $\alpha = \frac{l_x}{l}$ , the performance of the whole system can maintain a good reduction in search space and a small storage overhead. However, it has to sacrifice the expected number of false drops, it will be the same as in the single signature file approach. As the value of  $\alpha$  is larger than  $\frac{l_x}{l}$ , the search space and the expected number of false drops are guaranteed to be less than that of the single signature file. If the value of  $\alpha$  is set to 1, there will be no change in the design parameters in all partitions. The random false drop rate in each partition will equal to that of the single signature file,  $(b)^w = (b)^{w_x}$ . The expected number of false drops of



partition  $x$  in this circumstance will be:

$$\begin{aligned}
 E_x &= \frac{1}{V_x} \alpha (b)^{w_x} \sum_{i=1}^{V_x} (l_x - f_{xi}) \\
 &= \frac{1}{V_x} \alpha (b)^{w_x} (V_x l_x - \sum_{i=1}^{V_x} f_{xi}) \\
 &\approx \alpha (b)^{w_x} l_x \quad l_x \ll f_{xi} \\
 &\approx (b)^w l_x
 \end{aligned} \tag{5.18}$$

The expected number of false drops, as shown in Equation 5.18, and also the search space can be reduced at those value of  $\alpha$ , but it also bring about a larger storage overhead. If the value of  $\alpha$  is also larger than 1, the expected number of false drops can be further reduced thus increasing the retrieval performance. The storage overhead used is larger. Nevertheless, the overhead is still less than that of the inverted file approach because the storage required by inverted file is usually greater than 100% of the original corpus size, as in our experiments.

After calculating the weight of word signature in different partitions, we can determine the corresponding signature length  $m_x$  and the weight of block signature  $W_{B_x}$  for each partition. In order to maintain the search space reduction, the number of block signatures contained in all partitions should equal to the corresponding partitions in the previous partitioning method without alteration of  $m_i$  and  $W_{B_i}$ .

Thus, the weight of block signature  $W_{B_x}$  can be calculated from the following:

$$\begin{aligned}\frac{w}{W_B^+} &= \frac{w_x}{W_{B_x}^+} \\ W_{B_x}^+ &= \frac{w_x W_B^+}{w} \\ \mu W_{B_x} &= \frac{w_x \mu W_B}{w} \\ W_{B_x} &= \frac{w_x W_B}{w}\end{aligned}\tag{5.19}$$

Since the ratio  $\mu$  of Equation 5.11 is constant only if the bit density  $b$  is constant, we can find the signature length  $m_x$  of the corresponding partition.

$$\begin{aligned}b &= \frac{W_{B_x}}{m_x} \\ m_x &= \frac{W_{B_x}}{b}\end{aligned}\tag{5.20}$$

Also the storage overhead is:

$$\text{Storage overhead } (S_2) = \sum_{i=1}^n m_i l_i\tag{5.21}$$

When  $\alpha < 1$ , the signature length  $m_i$  for  $i = 1, 2, \dots, n$  of each partition is less than  $m$ . From Equation 5.15 and Equation 5.19, it has been shown that  $w_x$  and  $W_{B_x}$  are less than  $w$  and  $W_B$  respectively. Such that  $m_x$  will be less than  $m$ . Moreover, we maintain the reduction in search space by keeping the number of block signatures in each partition to be the same, thus  $l_i$ , ( $i = 1, 2, \dots, n$ ) will be the same in  $S_1$  and  $S_2$ . Under this condition the storage overhead  $S_2$  is always less than  $S_1$ .

## Chapter 6

# New Hashing Techniques for Partitioned Signature Files

In signature file approach, the word signature can be obtained by using lookup table or hashing function. In Chapter 5, we make use of a lookup table to assign the signature to each indexed term in our partitioning method since we know a fixed set of vocabulary in advance. Word signatures are pre-generated and stored with their corresponding words in the lookup table. Lookup tables can eliminate signature collision among words. Moreover, any query word which does not exist in the lookup table can be avoided searching the signature file.

On the other hand, there are some drawbacks in using the lookup table for signature assignment. First of all, the vocabulary of the corpus must be stable,



especially in the word-based segmentation approach. It is because any change in the dictionary requires the corpus to be segmented once again. Second, extra storage will be needed for storing the lookup table. It will increase the storage overhead in addition to the index file.

An alternative approach to the lookup table in Chinese signature file is to make use of hashing function [15, 50], such as ‘尋易’ (CSmart)[15]. The advantage of using hashing function for signature assignment is that it supports a growing vocabulary in dynamic environment. Moreover, it does not require extra storage space for the lookup table. Nevertheless, the retrieval performance of hashing function is usually less effective than that of lookup table [27]. As the word signatures are generated dynamically, there will be a non-zero probability that the same signature will be assigned to different words. It increases the number of false drops and thus degrades the retrieval performance. Therefore, the choice of hashing function is important. A suitable hashing function can provide a uniform distribution of 1's in the signature, such that the number of false drops can be minimized. However, it is difficult to find a suitable hashing function for this purpose.

A more sophisticated approach is to make use of a perfect hashing function [9, 36, 74] since it can completely avoid the collisions between the signatures of different words. However, even though each word has a unique signature, false drop still exists when a query word tries to match the block signature. It is also extremely

difficult and computationally demanding to obtain a perfect hashing function even for a small vocabulary size. Moreover, any change in the vocabulary requires a recomputation of a different perfect hashing function. Thus it is impractical to use such technique.

In our research, we review some existing hashing techniques and develop two new hashing approaches for generating the signatures for Chinese texts. The existing techniques we investigate are the direct division method and the random number assisted division method. We then propose a frequency-based hashing method and Chinese character-based hashing method. We have implemented all the four hash-based signature generation methods and integrated into our partitioning method. We have conducted experiments to compare the indexing performance of these methods as well as the lookup table method. The description of each hashing method will be presented in this chapter and the evaluation of their performance will be provided in Chapter 7.

## 6.1 Direct Division Method

Division method is one of the well-known and frequently used hashing function. In this method, the key value is divided by a positive number, which is usually the size of the address space, and the remainder will become the address of the key. The same technique can be applied in the signature generation. Liang [52]



used this simple division method to generate the bit sequence from characters and bi-gram slices. As a Chinese character is composed of two bytes, a Chinese word with  $k$  characters will be  $2k$  bytes long. The signature generation based on this direct division method is shown in Equation 6.1 as follows:

$$\left[ \sum_{i=1}^{2k} (B_i \times P_i) \right] \text{ mod } m \quad (6.1)$$

where  $B_i$  is the  $i$ -th bytes of the Chinese words;  $P_i$  is the prime number;  $m$  is the signature length. The hashed value of this equation is the bit position within the signature length  $m$ , we can alter the values of  $P_i$  so as to generate the other bit positions. If a particular bit position has been generated, we ignore this position. This process terminates when the number of distinct bit positions generated equals to the weight of a word signature.

## 6.2 Random Number Assisted Division Method

We investigate another approach based on the division method. We first make use of the division method to find the hashed value,  $h(w)$ , of a Chinese word  $w$  as follows:

$$h(w) = \left[ \sum_{i=1}^{2k} (B_i \times P_i) \right] \text{ mod } s \quad (6.2)$$

where  $B_i$  is the  $i$ -th byte of a Chinese word  $w$ ;  $P_i$  is the prime number that are used for multiplication with the value of the byte  $B_i$ ; and  $s$  is the address space.



The hashed value  $h(w)$ , which is obtained by hashing the Chinese word  $w$  using Equation 6.2, is used as an entry value of a random number generator to produce a sequence of bit position  $b$  over the interval  $1 \leq b \leq m$  as mentioned in [68]. The signature generation method is shown in Figure 6.1.

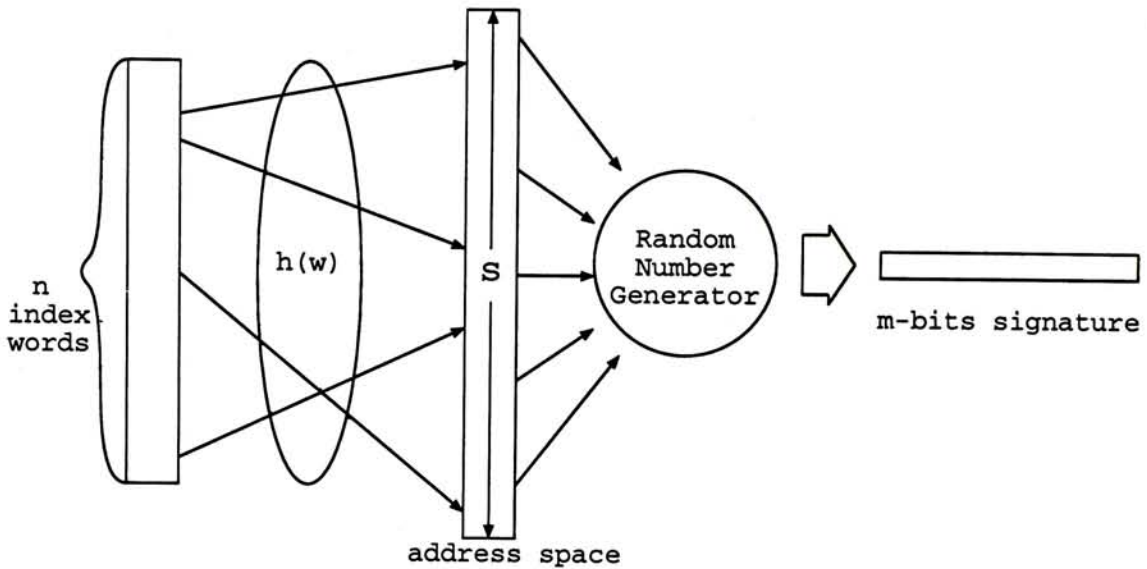


Figure 6.1: Using random number assisted division method in signature generation.

The difference between using division method as hashing function and as signature generation is the collision resolution. When the division method is used as a hashing function, the hashed value indicates an address space. So the collision between two different keys can be handled by using some collision resolution methods, such as open addressing, chaining, etc. However, collision resolution is not required in signature generation because the hashed value is not used as an address space. Instead, it is used as an entry value to generate a sequence of bit positions. Since this is a one-way process, we cannot obtain the hashed value from

the signature in reverse direction. Thus the collision of the hashed value will cause the signature duplication and it increases the number of false drops in signature files.

### 6.3 Frequency-based hashing method

False drops can be reduced by flattening the distribution of the index entry's frequencies in signature files [25, 68]. Ogawa and Iwasaki [59] proposed to hash the index words based on the frequency and developed a frequency-based hashing function to improve the precision. Since a conventional simple hashing function determines the hashed value from a word which has no relationship to frequency, the sum of frequencies of assigned words to each entry varies greatly according to changes in word frequency. Conversely, given the word frequencies, the hashed value can be assigned to words, so that the hashed value are equally balanced by the word frequency.

We develop a frequency-based hashing technique for our signature generation. We divide the dictionary words into groups and try to make the sum of frequencies in each group (i.e. the total frequency in each group) as even as possible. However, assigning words to each group so that the variation of the total frequencies among groups are as small as possible is a NP-complete problem [59]. Thus it takes a impractically large amount of time to find the optimum solution. We use a heuris-

tic algorithm for this group assignment problem. First, we collect the frequency information of each word from the corpus and then sort the words in descending order based on their total term frequencies. Afterwards, we assign the words to each group in order to make the sum of frequencies in each group be the same. It can be done by assigning the ungrouped word with the highest frequency to the group having the smallest sum of frequencies. This process repeats until all ungrouped words are processed.

Group	Words	Frequency	Total frequency of group
1	日本 (Japan)	70	70
2	足球 (soccer)	68	68
3	比賽 (match)	67	67
4	加拿大 (Canada)	65	65
5	快 (fast)	60	119
	國家 (country)	59	

#### Ungrouped words

Word	Frequency
檔案 (file)	58
馬 (horse)	55
家人 (family)	54

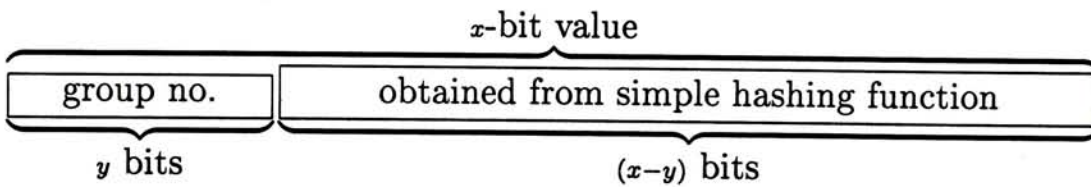
Figure 6.2: An example of frequency-based hashing method.

Figure 6.2 shows an example of assigning Chinese words into 5 groups. In this example, the ungrouped Chinese words are sorted in decreasing order according to their frequencies, such that the order of the ungrouped words according to their



frequencies will be “檔案” (file), “馬” (horse), and “家人” (family). The ungrouped word with the highest frequency, “檔案”, will then be assigned to the group having the smallest total frequency. This word will be assigned to Group 4 because its total frequency, which is 65, is the smallest among the others. After assigning the word “檔案”, Group 4 will no longer be the group having the smallest sum of frequencies because its value becomes 123. After that the next highest frequency ungrouped word “馬” will be considered. Since Group 2 has the smallest sum of frequencies now, this word will be assigned to Group 2. This process continues until all ungrouped words are assigned.

After assigning the words into groups, we hash each word into a  $x$ -bit value as shown below.



Unlike the division method, the hashed value in our frequency-based hashing function is composed of two parts. The first  $y$  bits of the hashed value indicates the group in which this Chinese word belongs. The value of the remaining  $(x - y)$  bits can be obtained by hashing the word using a simple hashing function, such as the division method. After the hashed value of each word is obtained, the signature can be generated by using a random number generator based on its hashed value similar to that of the random number assisted division method. The process of this signature generation is shown in Figure 6.3.

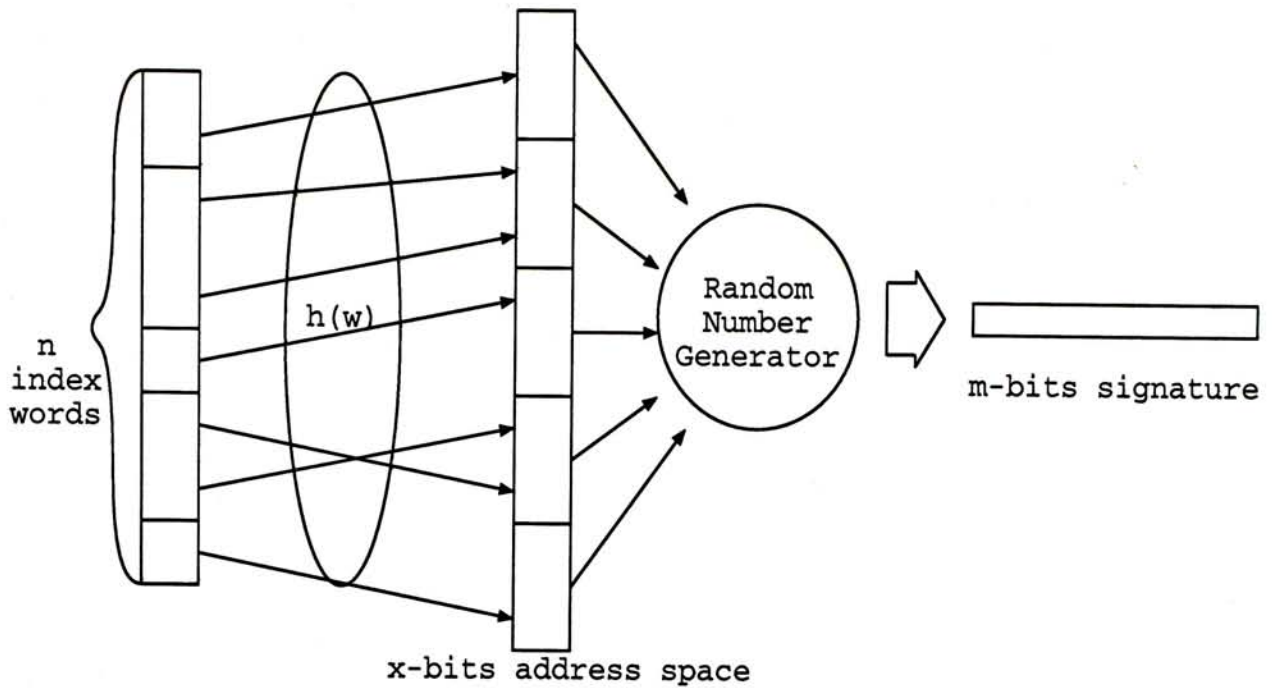


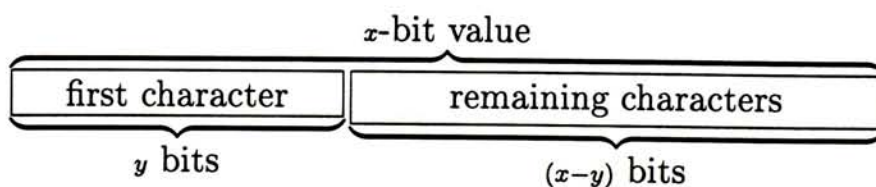
Figure 6.3: Frequency-group hashing method.

The advantage of using our frequency-based hashing method over the division method is that the precision can be improved by reducing the signature duplication. Since each group occupied a range of values, it reduces the collision of hashed value between words. Thus even two words have the same hashed value after the transformation of simple hashing function, the number of collisions can be reduced if they belong to different groups. This leads to the reduction of signature duplication which, in turn, reduces the number of false drops.

## 6.4 Chinese character-based hashing method

Although the frequency-based hashing function can reduce the collision of hashed value between words by dividing the Chinese words into groups, it requires extra processing and storage overhead. As mentioned before, we have to collect the term frequency of all words in the corpus for the group assignment before indexing. Therefore, the corpus is required to be processed twice leading extra processing overhead. In addition, extra storage is required to store the group information so that the query words can be transformed in the same way as the document words.

In our research, we propose another grouping method for hashing based on Chinese characteristics. In the Chinese character set, the number of unique characters is fixed (there are 6,763 distinct characters in GB coding scheme). We propose to divide the Chinese words into groups based on the first character of word, in such a way that the words in the same group will have the same first character.



In this method, each Chinese word will be mapped into a  $x$ -bit value, similar to that of the frequency-based hashing, in which the first  $y$  bit values are obtained by hashing the first character of the word. While the values of the remaining  $(x - y)$  bits are obtained by hashing the remaining characters in the word. The



transformation can be handled by applying some simple hashing functions. In this way, Chinese words with different first characters will be mapped into different range of values.

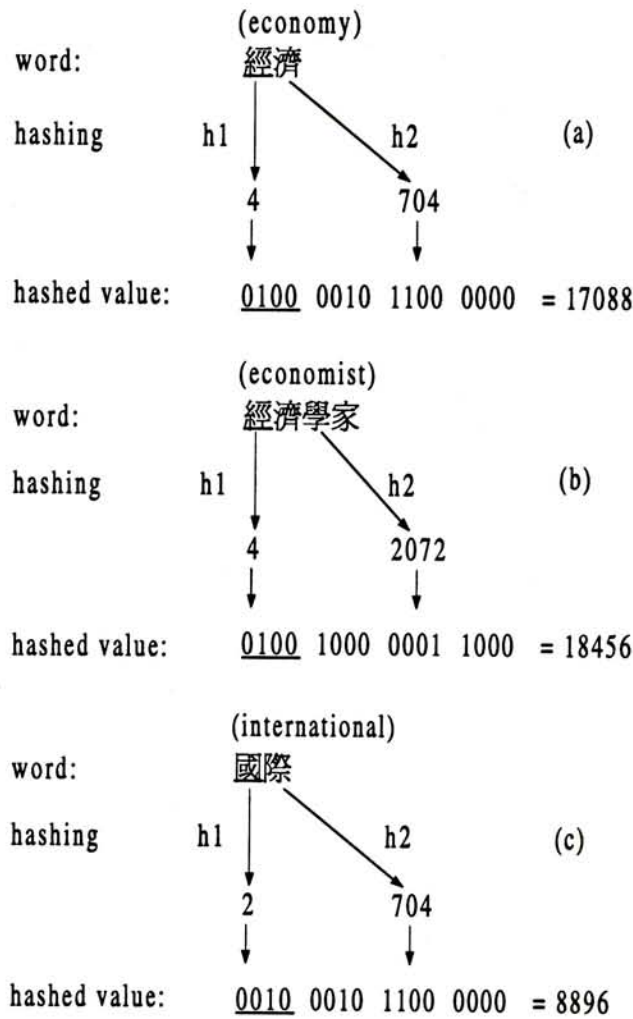


Figure 6.4: An example of Chinese character-based hashing method.

Figure 6.4 shows an example of mapping Chinese words into their respective hashed value in Chinese character-based hashing method. In this example, Chinese

words are hashed into a 16-bit value in which the value of the first 4 bits is obtained by hashing the first character of word using hashing function  $h_1$  and the value of the remaining 12 bits is obtained by hashing the remaining characters using hashing function  $h_2$ . From this example, both the words “經濟” (economy) and “經濟學家” (economist) have the same first character, they will be hashed into the values within the same range. The word “國際” (international) is mapped into different hashed value in spite of its last 12-bit value is equal to that of the “經濟”, which is 704. It was because the words “國際” and “經濟” have different first characters, their hashed values will fall into different range. It reduces the number of collision between words.

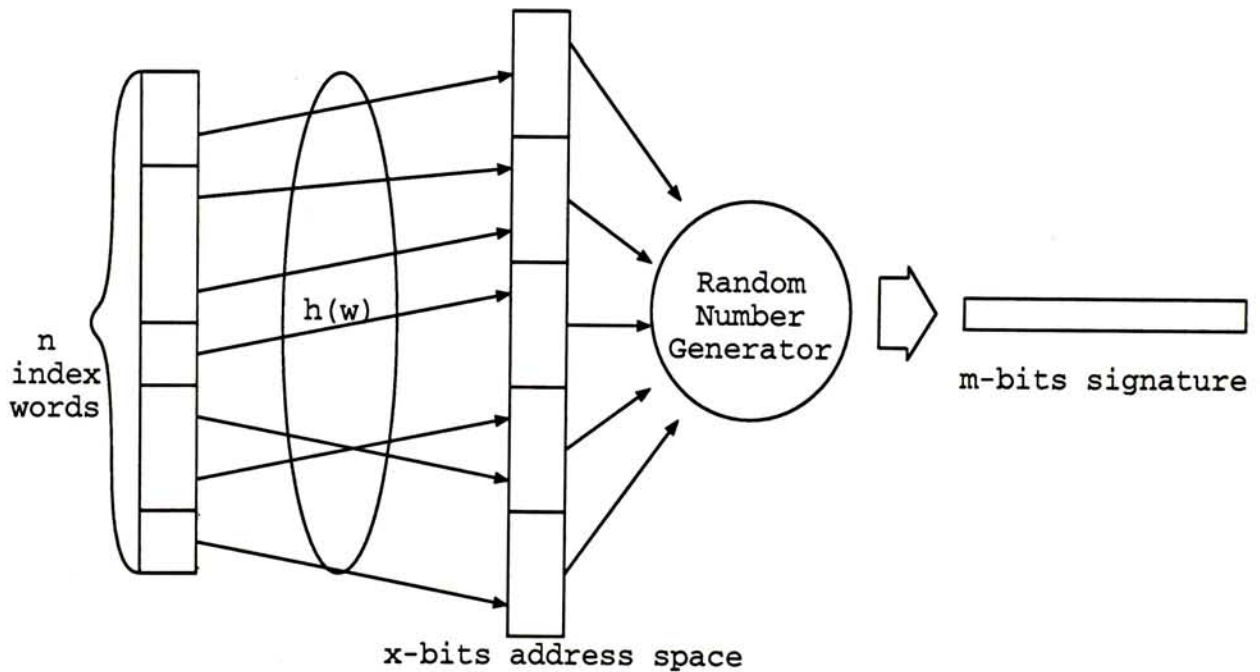


Figure 6.5: Chinese character-group hashing method.

After the hashed value is obtained, we can use it as a entry value for the random number generator to generate a signature for that word, as shown in Figure 6.5.

Our proposed method can reduce the collision of hashed value by dividing the word set into groups. Unlike the frequency-based hashing function, the group assignment and the hashed value generation in this method can be performed during the segmentation and indexing processes. Moreover, as the group assignment is based on the first character of each word, it does not require any extra space for storing the group information of each word.



## Chapter 7

# Experiments and Results

We have implemented both the single signature file method and our partitioning method, PSFC. Both of them have been used to conduct a series of experiments to compare the performance between the single signature file method and our partitioning method, and between different dynamic signature generation methods. All these experiments were conducted and evaluated using the TREC-5 (Text REtrieval Conference) [78] Chinese collection containing 164,789 news articles from two major news sources of China, including *People Daily* (人民日報) and *XinHua* (新華社). The size of this collection is about 170 Megabytes. This Chinese collection was first pre-processed by a word segmentation program [41]. The size of the dictionary used for the word segmentation contains 86,396 Chinese words. The segmented documents were then indexed by using both the single signature file and PSFC. We conducted the indexing under various signature length  $m$ , such as 304, 400, 504,

600 and 700, and also under various  $\alpha$  values in PSFC,  $\alpha = \frac{l_x}{2l}, \frac{l_x}{l}, 1$ , and 1.5, on a SUN UltraSparc-1 workstation. We used three values of the weight of word signatures, i.e. 10, 15, and 20. The effect of a larger weight reduces the number of false drops. The precision increases because an increased number of bits has to be checked in the filtering mechanism. However, the number of word signatures in a block decreases and so it consumes a larger amount of storage. While using a smaller values of this weight increases the number word signatures composing a block. The storage required can be reduced and the precision of the retrieved results will be sacrificed in this case. In our experiments, these three weights of word signature give the results in similar patterns, we will discuss the results using the weight of 15 in detail. The results of other values of weight are presented in the Appendix C.

These indexed documents were used to retrieve on 300 query terms, which were randomly selected from the dictionary, to obtain the retrieval performance of each methods. Moreover, the number of partitions used by PSFC in our experiments was three. It is because the word distribution of the TREC-5 Chinese corpus shows that words longer than four characters only occupies a small fraction of the corpus which is about 3.34%, as shown in Table 7.1. Thus we used one partition to store the signatures of words longer than two characters.

	% of occurrence
1-character words	26.82%
2-character words	62.69%
3-character words	6.88%
>4-character words	3.34%

Table 7.1: Statistics of Chinese words in corpus.

## 7.1 Performance evaluation of partitioned signature file based on Chinese word length

We first compare the performance between PSFC and the traditional signature file method which has been adopted by some systems using the signature file indexing method. Three performance measures for quantitative comparison are studied.

1. The retrieval performance which is measured by the average number of false drops. It allows a quantitative measure of the retrieval precision in each method.
2. The signature reduction ratio [43] which is defined as the average number of block signatures to be searched given a query signature and it is normalized by the total number of signatures. This ratio can be used to measure the reduction in search space, thus the retrieval time, compared with the single signature file method.



3. The amount of storage required for PSFC to index the Chinese corpus compared with that of the single signature file and the inverted file methods. Beside the storage requirement, the index-to-text ratio will also be studied which indicates the ratio of storage required compared with the original corpus.

### 7.1.1 Retrieval Performance

In the signature file approach, false drops are the only cause of degradation in retrieval precision. Thus, the smaller is the number of false drops, the better the retrieval performance will be. We compared the retrieval performance of PSFC with that of the traditional single signature file method by using their average number of false drops.

Figure 7.1 shows the average number of false drops of the single signature file and PSFC under various  $\alpha$  values. The plot of the single signature file can be used as the baseline to compare with the performance of PSFC. When  $\alpha = 1$ , the average number of false drops in PSFC is about 70% better than that of the single signature file. For each query signature, only one of the partitions in PSFC is accessed and it reduces the number of block signatures to be searched. Thus the resulting average number of false drops is less than that of the single signature file.

For  $\alpha = \frac{1}{7}$ , the average number of false drops is close to that of the single

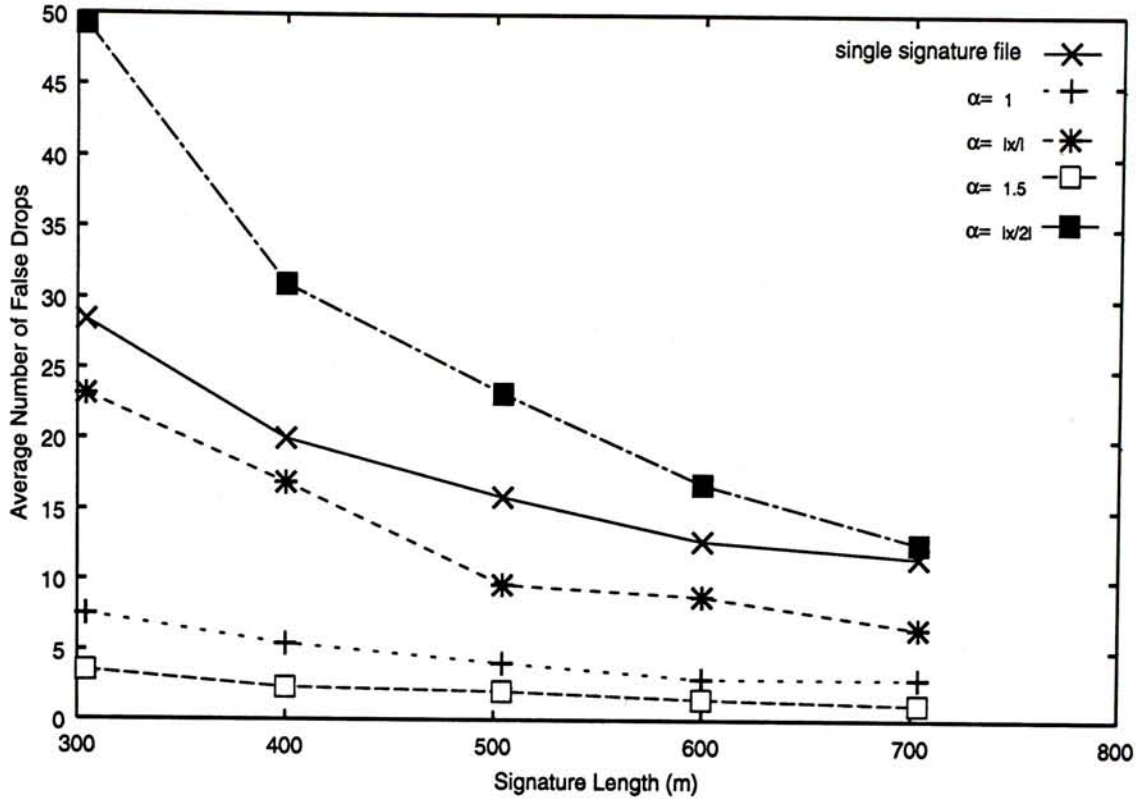


Figure 7.1: Average number of false drops.

signature file. The purpose of using this  $\alpha$  value is to maintain a search space smaller than that of the single signature file, and to reduce the storage overhead but at the expense of decreased retrieval precision. Nevertheless, the average number of false drops is still less than that in the single signature file case because the search space of our approach is actually less.

For  $\alpha$  value larger than 1, it further reduces the expected number of false drops as shown in the plot of  $\alpha = 1.5$ . However, the storage overhead used in this case will be more than that of  $\alpha = 1$ . The plot of  $\alpha = \frac{l_x}{2l}$ , shows a more average number of false drop than that of the single signature file. As  $\alpha = \frac{l_x}{l}$  is the bound for the expected number of false drop same as the expected number of false drop in the

single signature file.

$$\text{Expected no. of false drops in PSFC} = \frac{1}{V} (b)^{w_x} \left( \sum_{j=1}^n V_j l_j - \sum_{i=1}^V f_i \right) \quad (7.1)$$

$$w_x = w + \log_b \left( \frac{1}{\alpha} \right) \quad (7.2)$$

The results of various  $\alpha$  values are also in-line with our analytical finding in Chapter 5. As shown in Equation 7.2, when the value of  $\alpha$  increases, the weight of word signature ( $w_x$ ) in each partition will also increase. Therefore, the retrieval precision is improved because the expected number of false drops in PSFC decreases as  $\alpha$  increase, as shown in Equation 7.1. On the other hand, as the value of  $\alpha$  decreases, the retrieval precision decreases because of the reduction of the weight of word signature ( $w_x$ ) in each partition.

### 7.1.2 Signature Reduction Ratio

The reduction in search space can be measured by the signature reduction ratio which is defined as the ratio of the number of signature searched normalized by the total number of signatures. In the signature files method, the time spent on retrieval is directly proportional to the number of block signatures required to be checked. Thus, the search space can be used to represent the response time of the



system. In our experiments, the maximum number of signatures is equal to that of the single signature file.

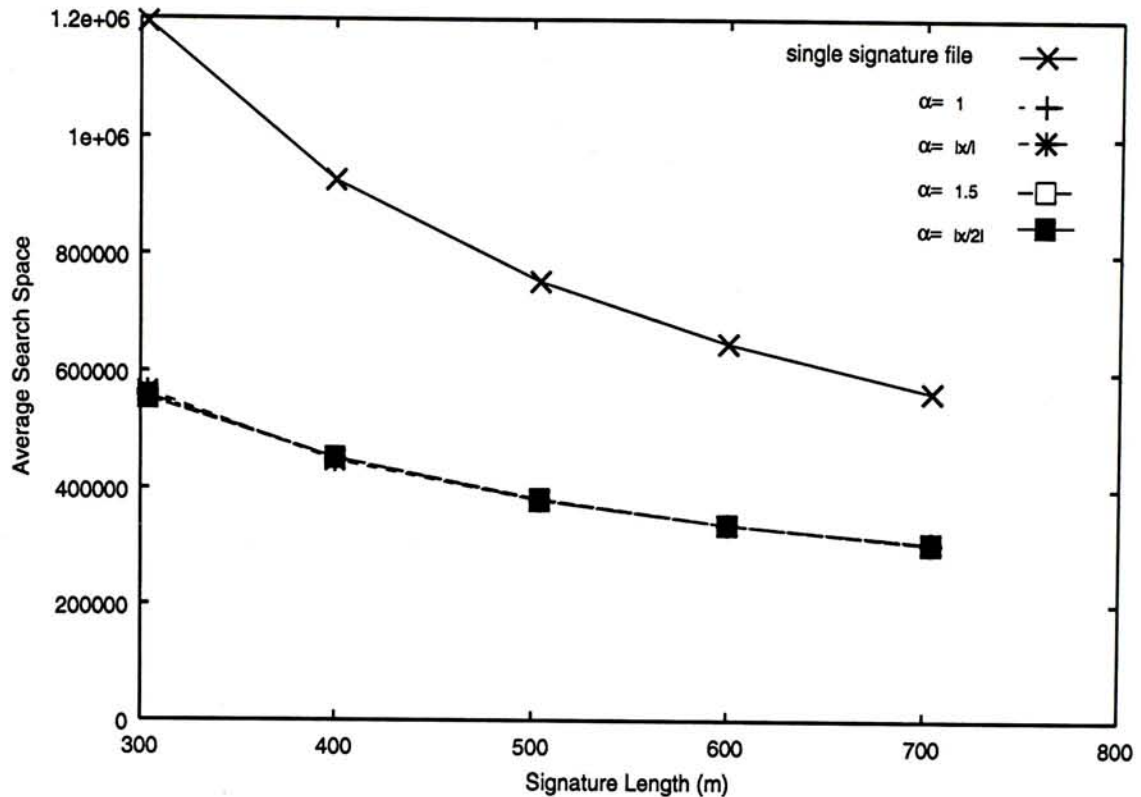


Figure 7.2: Average search space.

It is evident from Figure 7.2 that PSFC can reduce the average search space to about 50% of the single signature file in various  $\alpha$  values. Since only one of the partitions is required to be searched in response to a query signature, it dramatically reduces the number of block signatures to be checked. As we maintained the number of blocks near the same in various  $\alpha$  values so as to investigate the effects on the number of false drops and storage, the average number of search space will also be near the same. Therefore, the response time of PSFC in various  $\alpha$  values is faster than that of its single signature file counterpart because our analytical study

shows that the expected search space of PSFC ( $\sum_{i=1}^n \frac{V_i}{V} l_i$ ) is less than that of single signature file ( $l$ ).

### 7.1.3 Storage Requirement

The storage overhead used in the signature files depends on the number and the length of block signatures. In PSFC, we altered the value of  $\alpha$  to investigate its effect on the retrieval performance and storage requirement while maintaining the search space reduction. Thus the number of block signatures in various  $\alpha$  values are the same and the length of block signature is the only factor that will affect the storage requirement. In our experiments, we compare the storage overhead of PSFC under various  $\alpha$  values. We also use the modified SMART [70] system to index the same TREC-5 Chinese collection so as to investigate the storage requirement between PSFC and the inverted files.

	index/corpus ratio	file size (MB)
TREC-5 Chinese corpus	—	167
Inverted files	2.09	351
Single signature files	0.27 - 0.30	45 - 49
PSFC ( $\alpha = 1$ )	0.31 - 0.41	52 - 69
PSFC ( $\alpha = \frac{l_x}{l}$ )	0.28 - 0.36	46 - 61
PSFC ( $\alpha = 1.5$ )	0.33 - 0.44	55 - 74
PSFC ( $\alpha = \frac{l_x}{2l}$ )	0.26 - 0.34	43 - 56

Table 7.2: Index to corpus ratio.

Figure 7.3 shows the storage overhead used in the single signature file and that

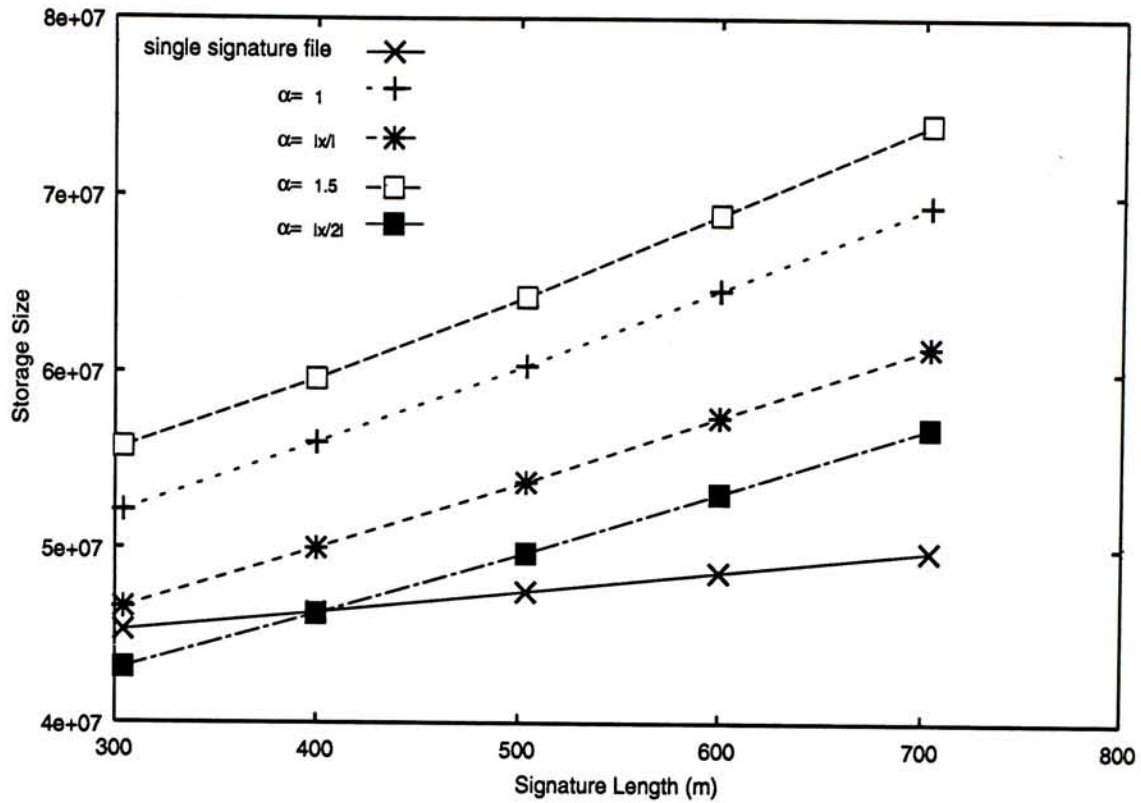


Figure 7.3: Total storage space of the index files.

in PSFC under various  $\alpha$  values. Table 7.2 shows the index-to-corpus ratio of the single signature file, PSFC, and the inverted files. Since the storage overhead of PSFC is  $\sum_{i=1}^n m_i l_i$  and we maintain the number of block signatures to be the same, so the storage overhead requirement will depend on the signature length ( $m_i$ ) in each partition. As the value of  $\alpha$  increases, the signature length ( $m_i$ ) in each partition will increase. Therefore, the storage overhead increases and vice versa. Our experimental results in Figure 7.3 can illustrate this trend.

When  $\alpha = 1$ , the storage used by PSFC is about 28% more than that of the single signature file. But it has about 70% improvement in retrieval precision and about 50% reduction in search space, as shown in Figure 7.1 and Figure 7.2. When



$\alpha$  is  $\frac{1.5}{7}$ , the storage overhead is roughly 3% to 23% more than that of the single signature file but it can still maintain the search space reduction.

The partitioning method using  $\alpha = 1.5$  needs the largest storage overhead among other  $\alpha$  values, but it can achieve the least average number of false drops. Even if its storage overhead used is the largest, its index-to-corpus ratio is still much less than that of the inverted files, about 78% less. The result of the  $\alpha$  value being  $\frac{1.5}{21}$  also matches with our analytical study. It uses less storage overhead and attains less retrieval performance than the other values of  $\alpha$  under the same search space reduction. From the above experiment, it is evident that system designers can determine the  $\alpha$  value for their system based on the availability of the resources.

#### 7.1.4 Discussion

From the experimental results, our partitioning method, PSFC, can offer a faster search speed than that of the single signature file because only one of the partitions is required to be searched given a query term. It results in about 50% average search space reduction. Moreover, it also increases the retrieval precision by reducing the number of block signatures to be checked. The experimental results shows that PSFC can achieve more than 70% improvement in retrieval precision. The drawback of PSFC is that more storage than the single signature file will be used, about 28% when  $\alpha = 1$ . It is because the occurrence of incomplete

block signatures is more than that in single signature files. However, even the storage overhead increases slightly, it is still much less than that of the inverted files approach. The choice of  $\alpha$  value can be determined based on the availability of the system resources.

## 7.2 Performance evaluation of different dynamic signature generation methods

In this section, we evaluate the effectiveness of signature generation methods using different hashing functions, and also compare their performance with that of using lookup table.

We used 300 query words to conduct the retrieval in our experiments. Since the size of address space will affect the number of collisions, we used the size of address space which were 100 and 1,000 times larger than that of the dictionary in the random number assisted division method. For the frequency-based and the Chinese character-based hashing methods, as they transform a word into a  $x$ -bit value for signature generation, we used a 4-byte integer as the  $x$ -bit value for convenience. The address space of them would be  $2^{32} - 1$ . Since both of these hashing methods divide the data set into groups, we set the number of groups used in the frequency-based hashing method to 128 and that used in the Chinese

character-based hashing method to the number of unique characters in GB coding scheme, which is 6,763 groups. Since the dictionary words in the frequency-based hashing method are divided into 128 groups, we used the first 7 bits of the hashed value to represent the group number of each word and the remaining 25 bits were used to store the value after hashing the Chinese word. In the Chinese character-based hashing method, we used the first 13 bits to represent the particular group that the Chinese word belongs after hashing the first character of each word. The remaining 19 bits were then used to store the value after hashing the remaining characters of that word.

In our experiments, we used two measures to evaluate their performance.

1. One measure is the collision of the hashed values and of the word signatures. As some of the hashing methods, e.g. the random number assisted division method, the frequency-based hashing method, and the Chinese character-based hashing method, transform a word into a hashed value before signature generation, any duplicated hashed value will cause the same signature to be generated. It will greatly increase the number of false drops.
2. The other measure is the retrieval performance similar to that in previous experiments. It can be measured by the average number of false drops. Since the retrieval performance of different methods also depends on the distribution of the bit positions generated, the average number of false drops will



increase with an uneven distribution of the bits set within the signature. We used the average number of false drops as a quantitative measure.

### 7.2.1 Collision

In dynamic signature generation, the word signatures are generated by transforming the Chinese words into word signatures using a hashing function. Unlike the lookup table, signature generation using hashing function will cause duplication when two or more different words are mapped into the same word signature. This is called collision. In signature file, the collision of signatures increases the number of false drops and thus reduces the retrieval performance. We used the number of collision as a quantitative measure of performance of each hashing method.

	No. of collided hashed values	No. of collided signatures
Direct division method	—	0
Random number assisted division method (100a.s.)	6,536	6,536
Random number assisted division method (1,000a.s.)	6,536	6,536
Frequency-based hashing method	46	46
Chinese character-based hashing method	21	21

Table 7.3: Statistics of collision using different hashing methods.

In our experiments, the number of collided hashed values and the number of collided signatures have been studied. As the random number assisted division

method, the frequency-based hashing method, and the Chinese character-based hashing method transform the Chinese words into a entry value for the random number generator. Any collision between these entry values will cause the generation of duplicated signature for different words. However, the direct division method transforms the Chinese words into word signatures using the hashing function directly. We can only measure the number of collided signatures of this method.

As shown in Table 7.3, the random number assisted division method gives the largest number of collisions than the others hashing methods. Since the number of collisions in both 100 a.s.<sup>1</sup> and 1,000 a.s. are the same, the collision depends on the data set rather than the address space size. Using the random number assisted division method in Chinese word will hash many words into the same address space and thus increasing the number of duplicated signatures. On the other hand, the frequency-based hashing method and the Chinese character-based hashing method are more effective in the collision reduction as the number of collided hashed value is much less than that of the random number assisted division method (about 99% less). These two methods divide the words set into groups, and the words from different groups will be mapped into different range of hashed values. It reduces the collision of words belonging to different groups. The number of collided hashed values of the Chinese character-based hashing method is even much less than that

---

<sup>1</sup>a.s. - Address Space.



of the frequency-based hashing method because the number of groups used in the Chinese character-based hashing method is more.

As the direct division method generates the bit position directly from the hashing function, we compared the number of collided signatures instead of the number of collided hashed values. From Table 7.3, it shows that this method does not generate duplicated signatures among the dictionary words. However, its performance may be affected by the distribution of 1's within the word signature.

### 7.2.2 Retrieval Performance

As mentioned in Section 7.2.1, the collision of the hashed values causes the duplication of word signatures, which increases the number of false drops. Moreover, the distribution of 1's generated also affect the retrieval performance. We evaluate the retrieval performance of each hashing method using our partitioning method, PSFC. To compare the effectiveness of their performance, we used the average number of false drops as a quantitative measure and used the result of lookup table as a baseline for comparison.

Figures 7.4 - 7.7 show the average number of false drops of each hashing method and of lookup table using various  $\alpha$  values. The plots of the direct division method in all figures show that its retrieval performance is the worst under all  $\alpha$  values because it gives the largest average number of false drops. Although Table 7.3 shows



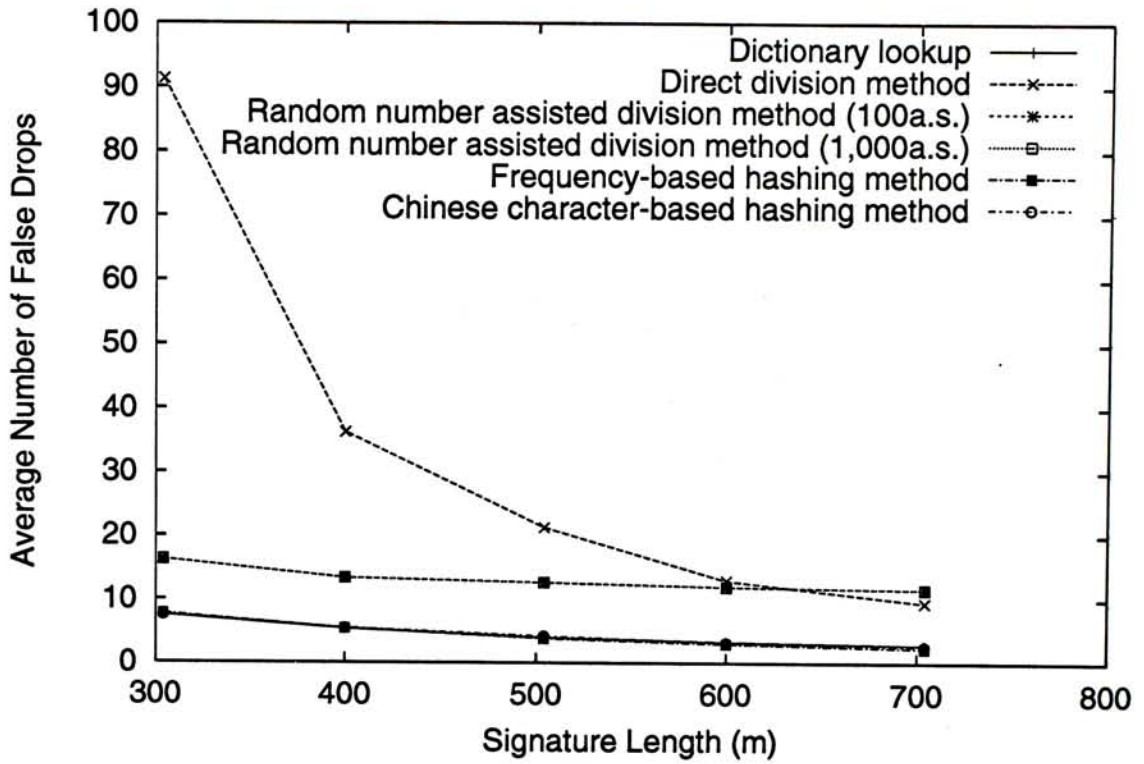


Figure 7.4: Average number of false drop ( $\alpha = 1$ ).

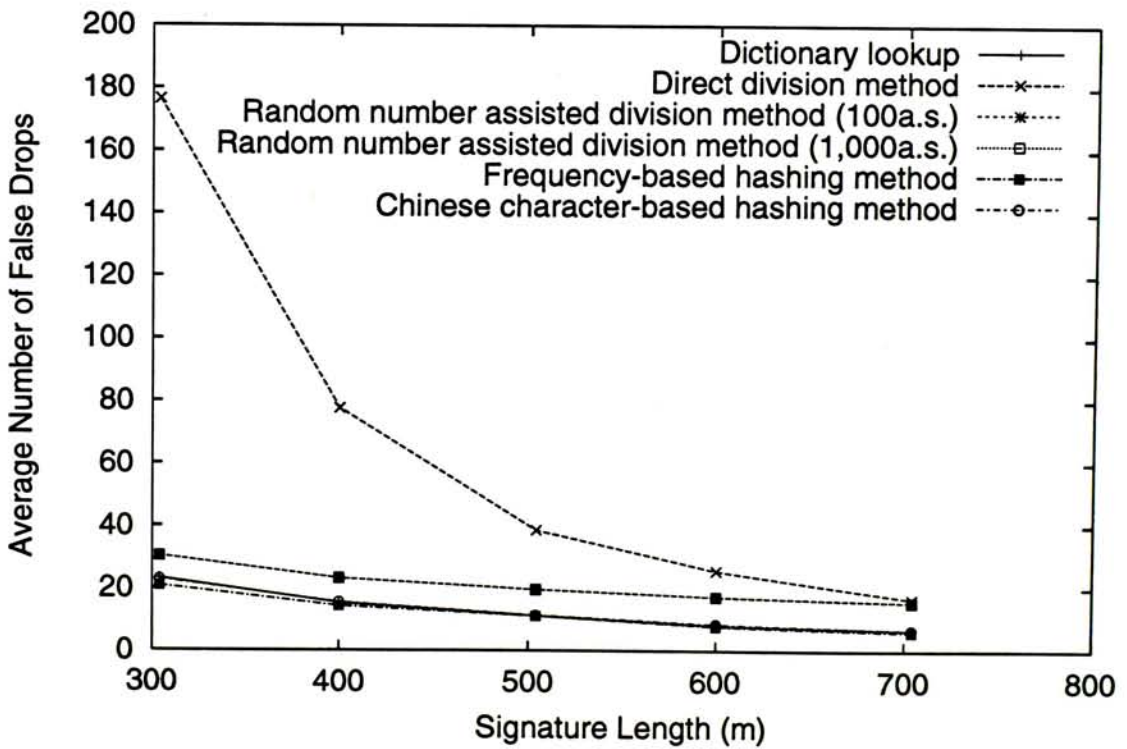


Figure 7.5: Average number of false drop ( $\alpha = \frac{l_x}{l}$ ).

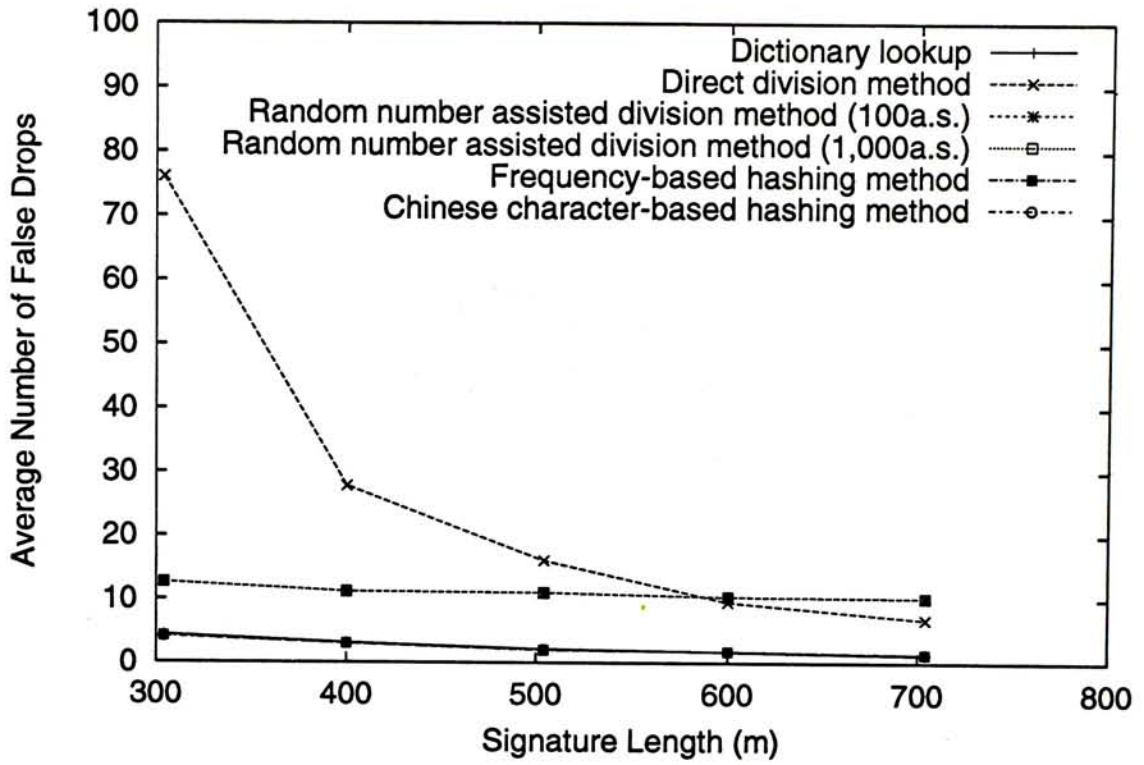


Figure 7.6: Average number of false drop ( $\alpha = 1.5$ ).

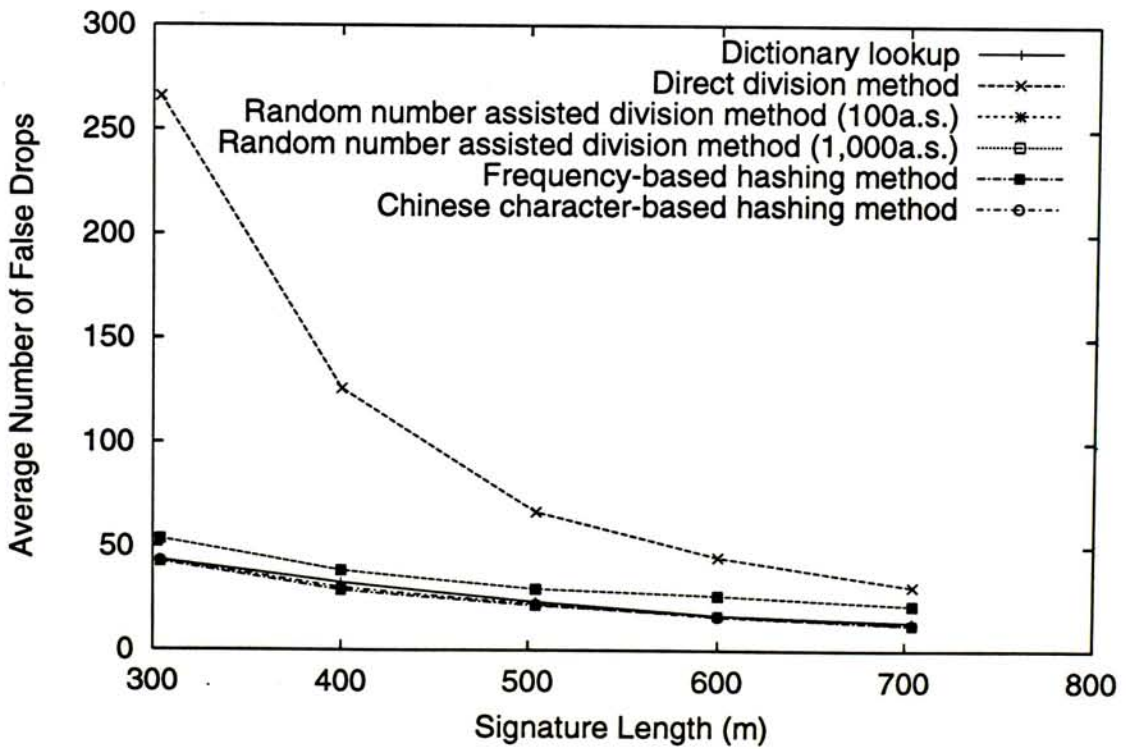


Figure 7.7: Average number of false drop ( $\alpha = \frac{l_x}{2l}$ ).

that the direct division method does not cause the signature duplication, its worst retrieval performance may be due to non-uniform distribution of 1's generated in the signatures.

For the random number assisted division method, its retrieval performance is better than that of direct division method but worse than that of lookup table. Since this method gives the largest amount of collided entry values, as shown in Table 7.3, it increases the number of duplicated signatures which, in turn, increases the average number of false drops. The retrieval performance of the frequency-based and the Chinese character-based hashing methods are very closed to that of the lookup table. This is because these two methods divide the data set into groups for hashing. So the number of collisions of the hashed value can be reduced successfully. As a result, the average number of false drops will also be reduced.

### 7.2.3 Discussion

As mentioned in [27], the signature generation using the traditional hashing methods, such as the direct division method and the random number assisted division method, have worse retrieval performance than that of using the lookup table. This is due to the fact that collision in hashing method will cause duplicated signatures, and the non-uniform bit position assignment increases the chance of matching an un-qualified block. Both of them increase the number of false drops



in the retrieval result and our experiments also show the same result.

Our experimental results show that the frequency-based hashing method gives a much better result than the traditional hashing method. Its retrieval performance is as good as that of lookup table. However, the corpus has to be processed twice. First, it collects the frequency information and performs group assignment. Then it performs transformation using a hashing function. Moreover, extra storage is required for storing the group information.

We propose to divide the data set based on the first character of each Chinese word instead of their frequencies. From the result, it shows that our method has a better collision control than that of the frequency-based hashing method. Its retrieval performance is also as good as that of lookup table. Unlike the frequency-based hashing method, the hashed value can be easily obtained when indexing and no extra storage is required.

# Chapter 8

## Conclusions and Future Work

### 8.1 Conclusions

In this research, we studied different indexing issues in Chinese information retrieval, such as, segmentation, signature file indexing, and signature generation using lookup table and different hashing methods. Owing to the linguistic difference between Chinese and English, many commonly used information retrieval techniques cannot handle Chinese text directly. For instance, there is no word boundaries within a Chinese sentence. Thus it is difficult to determine which part of the sentence can form a term. Extracting indexable terms from a Chinese sentence becomes an important process for Chinese text indexing.

There are three commonly used segmentation approaches, namely, the character-based, the word-based, and the  $N$ -gram segmentation approaches. They extract

indexable terms from a Chinese sentence in different ways. However, there is no indication to show which one is better. In this research, we conducted some experiments to compare the effects of different segmentation approaches on a vector space retrieval model. The measures we studied include the retrieval performance, the time efficiency, and the storage requirement. Our experimental results show that the word-based segmentation approach has a better overall performance than the others. The retrieval performance of this approach is roughly 37% and 18% better than the character-based and the bi-gram approaches respectively. In addition, this approach also consumes the least amount of storage space. However, the word-based segmentation approach uses the longest time for segmentation. Since this approach involves searching the appropriate words of each sentence from a dictionary, this process requires a considerable amount of time.

Although the inverted file approach has a fast retrieval response and high precision, its complex file structure is not suitable for dynamic environment and it also consumes a large amount of space to store its index file. From our experiments, the space occupied by the inverted file is about 100% - 200% of the original corpus size. To alleviate these problems, we investigate another alternative, the signature files, to index Chinese text. Signature files offer the advantages of efficient insertion and deletion under dynamic environment and smaller storage overhead used. However, the drawbacks of this approach are the introduction of false drops and slow search



speed. We address these two drawbacks by introducing a new partitioning method for signature file based on the Chinese word characteristic, called PSFC.

PSFC aims at increasing the search speed and reducing the number of false drops for Chinese text by reducing the number of block signatures to be accessed for each query term based on the Chinese word characteristics. We integrate the word-based segmentation with PSFC because the word-based segmentation has a better retrieval performance than others. Moreover, we use the lookup table for the signature generation based on the dictionary because this signature generation method results in smaller amount of false drops than that of using hashing methods. In PSFC, Chinese words with different length are stored in different partitions. It is because the variation in Chinese word length is much less than in other languages, such as English. In addition, a longer word in Chinese usually has more specific meaning. In this way, for a given query term, only the partition containing the specified Chinese word length will be searched. It dramatically reduces the number of block signatures to be searched. It not only increases the search speed, but also reduces the number of false drops. From our experiments, PSFC offers a faster search speed than that of the single signature file because PSFC has about 50% reduction in average search space. The search space reduction also achieves about 70% improvement in retrieval precision (e.g. when  $\alpha = 1$ ). Moreover, we develop a general scheme for PSFC so that we can control the trade-off between the storage

space and the retrieval precision based on the system resources availability.

Besides the number of block signatures to be searched, different signature generation methods also affect the number of false drops. Although there are two main methods for the signature generation, e.g. lookup table and hashing function, earlier studies showed that lookup table has a better retrieval performance than hashing function. However, lookup table uses extra storage. We investigate the effects of different hashing methods on the retrieval performance and also develop two new hashing methods for improving the retrieval performance. We evaluate the performance of two commonly used hashing methods, namely, the direct division method and the random number assisted division method. Although the direct division method does not result in duplicated signature, it has the worst performance than the other signature generation methods. Our experiments also show that the retrieval performance of the random number assisted division method is also worse than that of lookup table. It is because this method generates a large amount of collided entry values during the signature generation. Therefore, it causes many duplicated signatures for different words. We address this problem by developing two new hashing methods, namely, the frequency-based and the Chinese character-based hashing methods. The aim of these two methods is to reduce the number of collided entry values so as to reduce the number of duplicated signatures. Our experimental results show that they can reduce over 99% collided



entry value compared with the random number assisted division method and their retrieval performance are as good as that of using lookup table.

## 8.2 Future work

In our research, the evaluation of retrieval performance of PSFC only involves comparing the number of false drops of each query term. It is only a Boolean retrieval method in which only the presence of the query term will be checked. In order to process a natural-language query, we can incorporate the information of term frequency and document frequency of each word in PSFC during the indexing process. The retrieval process can make use of the vector space model or probabilistic model to determine the degree of relevance of each document to the query.

Moreover, signature files can support partial matching between a simple word and a compound word by carefully designing the construction of compound word signature. For example, the word “世界貿易組織” will be matched with “貿易” because “貿易” is one of the component of the compound word “世界貿易組織”. This partial matching can be achieved by constructing word signature of the compound word in such a way that it includes the signature of its components. It is different from the inverted file indexing method because the inverted files only support an exact matching between the query word and the document words.



# **Appendix A**

## **Notations of Signature Files**

Notations	Definitions
$b$	bit density
$f_i$	number of block contains the term $v_i$
$f_{ij}$	number of block contains the term $v_i$ in partition $j$
$K(Q_i)$	query probability of term $Q_i$
$l$	number of block signature in the signature file
$l_i$	number of block signature in partition $i$
$m$	signature length
$m_k$	signature length of partition $k$
$n$	number of partitions
$p_i$	number of block that does not contain the term $v_i$
$p_{ij}$	number of block that does not contain the term $v_i$ in partition $j$
$s$	No. of words signature superimposed
$w$	weight of word signature
$w_Q$	weight of query signature
$w_j$	weight of word signature of in partition $j$
$W_B$	weight of block signature
$W_{B_j}$	weight of block signature in partition $j$
$W_B^+$	sum of word's weight of block signature in FWB
$W_{B_{avg}}^+$	average weight of all block signatures
$V$	vocabulary size of dictionary
$V_k$	vocabulary size of words with length $k$

Table A.1: Notations used in signature files.

# Appendix B

## False Drop Probability

The mathematical formulas of the false drop probability, mentioned in Chapter 4, is derived by Roberts [64]. This Appendix presents the derivation. For signature of length  $m$  and weight  $w$ , there will be  $\binom{m}{w}$  possible word signatures that are likely to be chosen when a word signature is generated.

Let  $n_i, i = 1, 2, \dots, t$  be a set of distinct integers in the range  $[1, m]$ . In order to analyze the false drop, we need to determine the probability,  $P(t)$ , that a block signature contains 1's in the  $t$  positions  $n_1, n_2, \dots, n_t$ . In forming the block signature, each of the component words is transformed into a word signature containing  $w$  1's within the  $m$ -width bit string. Then the  $s$  word signatures are superimposed so as to form the block signature. Let  $S(t)$  denotes the set of  $s$ -tuples of word signatures superimposed and results in the the block signature having at least one zero in position  $p_1, p_2, \dots, p_t$ . Similarly,  $S_{p_j}$  denotes the set of  $s$ -tuples



for which the resulting block signature have a zero in position  $p_j$ .

Then the sets  $S(t)$  and  $S_{p_j}$ ,  $j = 1, 2, \dots, t$  are related as follows:

$$S(t) = S_{p_1} \cup S_{p_2} \cup \dots \cup S_{p_t} \quad (\text{B.1})$$

It follows from Equation B.1 that

$$\begin{aligned} \#[S(t)] = & \sum_i \#[S_{p_i}] - \sum_{\substack{i, j \\ i < j}} \#[S_{p_i} \cup S_{p_j}] + \sum_{\substack{i, j, k \\ i < j < k}} \#[S_{p_i} \cup S_{p_j} \cup S_{p_k}] - \dots \end{aligned} \quad (\text{B.2})$$

where  $\#[S]$  denotes the cardinality of set  $S$ , and now

$$\# \underbrace{[S_{p_i} \cup \dots \cup S_{p_m}]}_{j \text{ terms}} = \binom{m-j}{w}^s \quad (\text{B.3})$$

Therefore

$$\#[S(t)] = \sum_{j=1}^t (-1)^{j-1} \binom{t}{j} \binom{m-j}{w}^s \quad (\text{B.4})$$

From Equation B.4, for a block signature composed of the superimposition of  $s$  word signatures, the number of  $s$ -tuple of word signature is  $\binom{m}{w}^s$ . Therefore, the probability that a block signature having at least one zero in the position  $p_1, p_2, \dots, p_t$  is  $\frac{\#[S(t)]}{\binom{m}{w}^s}$ . As a result,

$$P(t) = 1 - \frac{\#[S(t)]}{\binom{m}{w}^s} \quad (\text{B.5})$$

Substituting Equation B.4 into Equation B.5,  $P(t)$  will be

$$P(t) = \sum_{j=0}^t (-1)^j \binom{t}{j} \binom{m-j}{w}^s \binom{m}{w}^{-s} \quad (\text{B.6})$$

When  $t = 1$ , the probability  $P(t)$  means that a particular bit position in the block signature is set to 1.

$$P(1) = 1 - \left(1 - \frac{w}{m}\right)^s \quad (\text{B.7})$$

In the analysis of Roberts [64],  $P(t)$  can be approximated as Equation B.8 which is valid for sufficiently large  $s$  and for  $w \ll m$ .

$$P(1) \approx \left(1 - \left(1 - \frac{w}{m}\right)^s\right)^t \quad (\text{B.8})$$

When given a query signature, a better retrieval performance can be obtained if the false drop probability with the block signature is minimized. For a query signature that is the result of superimposition of  $q$  query terms and has weight  $W$ , where  $W$  bit positions are 1's and  $(m - W)$  bit positions are 0's. In order to reduce the number of false drops, we can minimize the false drop probability such that  $W$  chosen bit positions of the block signature are all 1's.

The expected query weight can be estimated by

$$W = m \left(1 - \left(1 - \frac{w}{m}\right)^q\right) \quad (\text{B.9})$$

For the query signature containing only one query term, the expected weight of the query signature  $W$  is equal to the weight of word signature  $w$ . Then the false drop probability of single-term query signature  $P(w)$  will be

$$P(w) = \left(1 - \left(1 - \frac{w}{m}\right)^s\right)^w \quad (\text{B.10})$$

Thus we can minimize the false drop probability  $P(W)$

$$P(W) = \left(1 - \left(1 - \frac{w}{m}\right)^s\right)^w \quad (\text{B.11})$$

Let

$$u = \left(1 - \frac{w}{m}\right)^s \quad (\text{B.12})$$

Then

$$\begin{aligned} P(W) &= (1 - u)^{b(1 - \frac{uq}{s})} \\ \log P(W) &= m(1 - u^{\frac{q}{s}}) \log(1 - u) \\ &= m(1 - e^{\frac{q}{s} \log u}) \log(1 - u) \\ &\approx -m\left(\frac{q}{s}\right) \log u \log(1 - u) \end{aligned} \quad (\text{B.13})$$

Equation B.13 will have a minimum value when  $u = \frac{1}{2}$  under the assumption that  $\frac{q}{s} \log u$  is small. Since it is usually  $q \ll s$ , this assumption is correct in general. When  $u = \frac{1}{2}$ , then  $P(1) = \frac{1}{2}$ . Thus the false drop probability with a query signature is minimized when the signature length  $m$  and the weight of word signature  $w$  are chosen such that the block signature will have approximately half the number of bits set to one and the other half set to zero. It means the weight of the block signature  $W_B$  is half of the signature length  $m$ . Thus we can derived the suitable value for the signature length  $m$  and the weight of word signature  $w$ .



Substituting  $u = \frac{1}{2}$  into Equation B.11

$$\begin{aligned} P(W) &= (1 - u)^W \\ &= \left(\frac{1}{2}\right)^W \end{aligned} \tag{B.14}$$

When the query signature is composed of one query terms ( $q = 1$ ), then  $W = w$  and we can find the suitable weight of word signature  $w$  and the signature length  $m$  such that the false drop probability  $P(w)$  is minimized.

$$w = \left(\frac{1}{\log 2}\right) \log \left(\frac{1}{P(w)}\right) \tag{B.15}$$

$$m = \left(\frac{1}{\log 2}\right)^2 s \log \left(\frac{1}{P(w)}\right) \tag{B.16}$$

# **Appendix C**

## **Experimental Results**

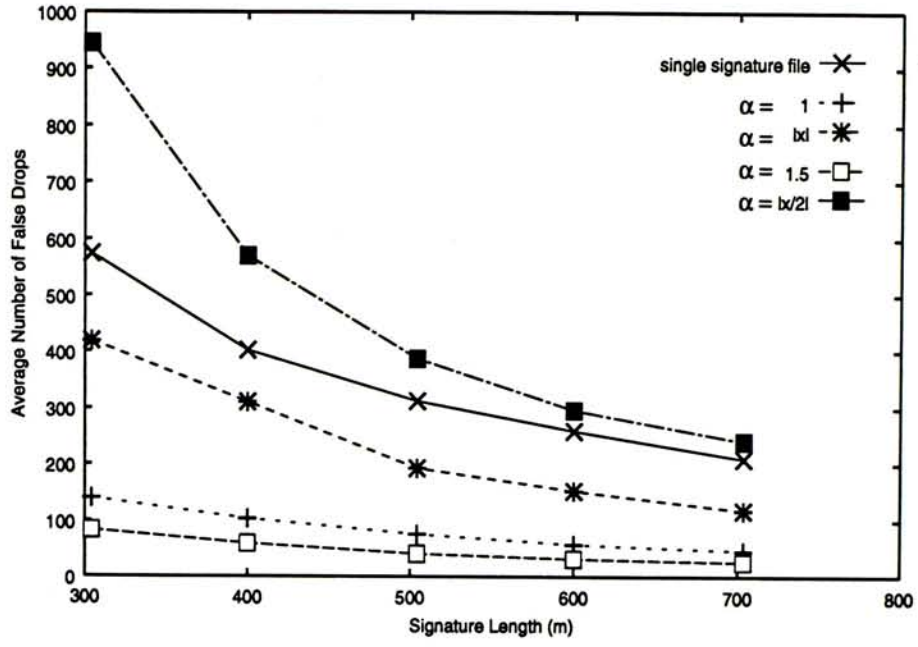


Figure C.1: Average number of false drops of  $w = 10$ .

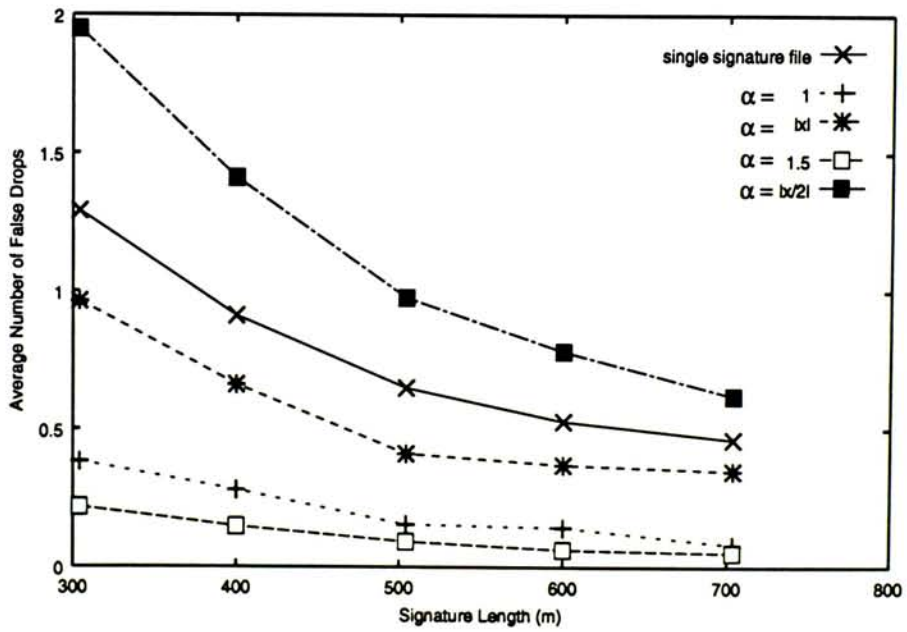


Figure C.2: Average number of false drops of  $w = 20$ .



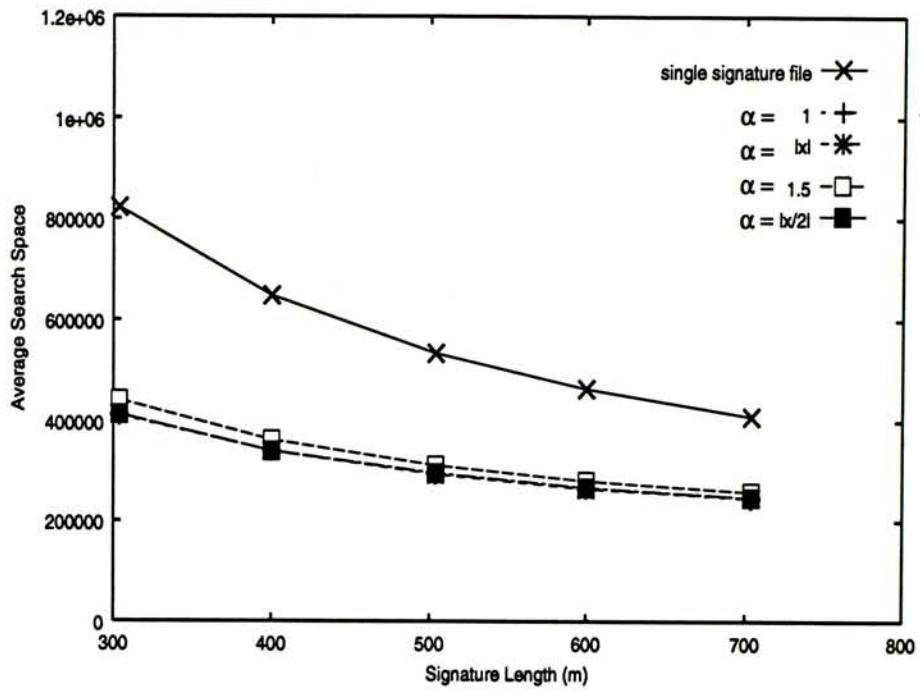


Figure C.3: Average search space  $w = 10$ .

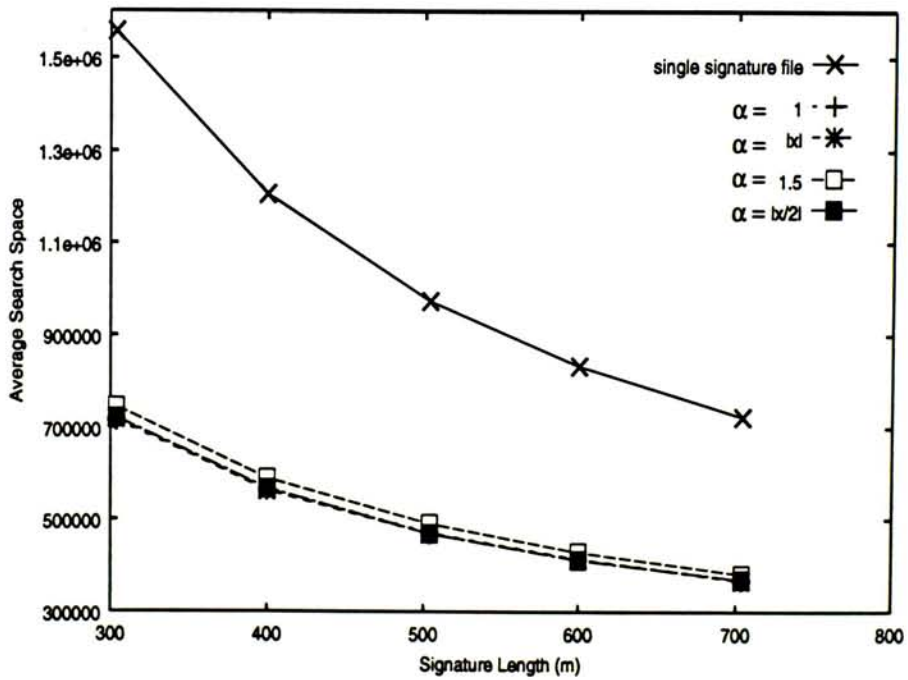


Figure C.4: Average search space  $w = 20$ .

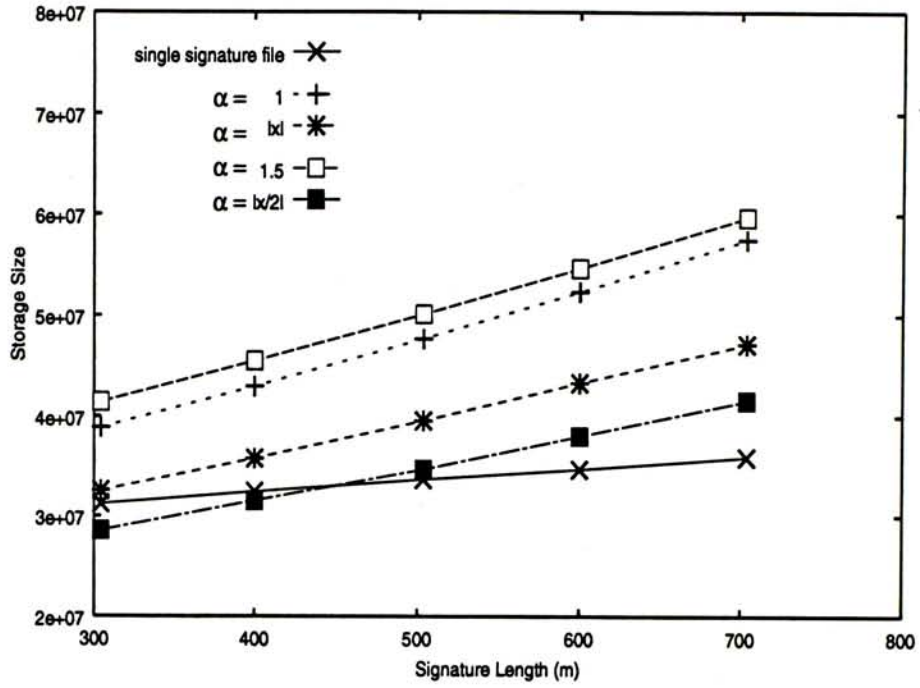


Figure C.5: Total storage space of the index files  $w = 10$ .

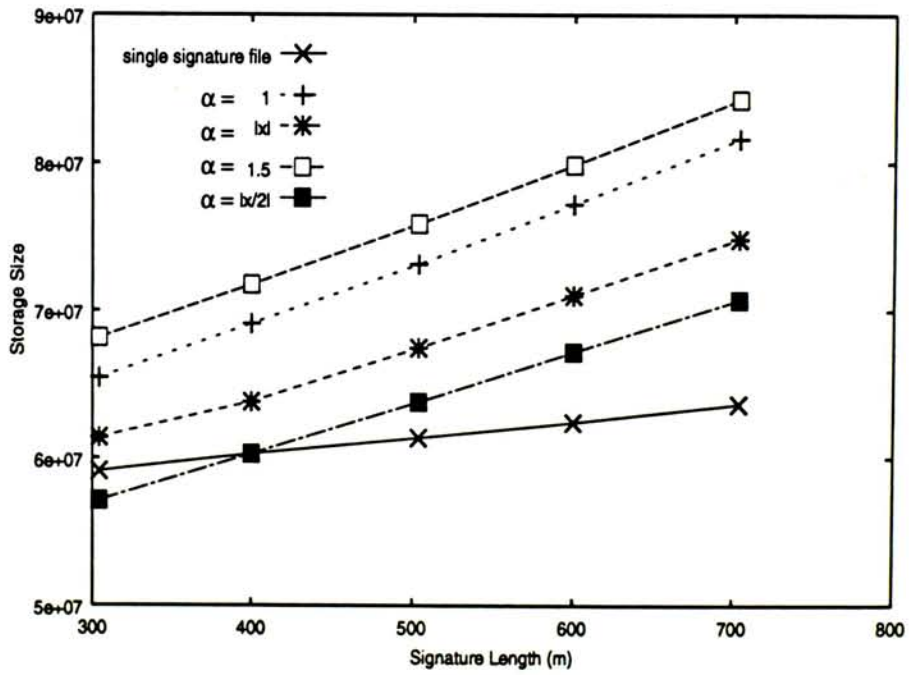


Figure C.6: Total storage space of the index files  $w = 20$ .

# Bibliography

- [1] 劉源、譚強、沈旭昆，“信息處理用現代漢語分詞規範及自動分詞方法”，清華大學出版社，1994.
- [2] A.V. Aho, M.J. Corasick, "Efficient string matching: An aid to bibliographic search", in *Communications of the ACM*, Vol. 18, No. 6, pp. 333-340, Jun. 1975.
- [3] A.V. Aho, J.D. Ullman, "Optimal partial-match retrieval when fields are independently specified", in *ACM Transactions in Database Systems*, Vol. 4, No. 2, pp. 168-179, 1979.
- [4] R.S. Boyer, J.S. Moore, "A fast string searching algorithm", in *Communications of the ACM*, Vol. 20, No. 10, pp. 762-772, Oct. 1977.
- [5] J.P. Callan, W.B. Croft, S.M. Harding, "The INQUERY Retrieval System", in *Proceedings of the 3rd International Conference on Database and Expert Systems Applications*, pp. 78-83, 1992.
- [6] A.F. Cardenas, "Analysis and performance of inverted data base structure", in *Communications of the ACM*, Vol. 18, No. 5, pp. 253-263, 1975.
- [7] W.B. Cavnar, "*n*-Gram-Based Text Filtering for TREC-2", in *Proceedings of the Second Text REtrieval Conference (TREC-2)*, NIST Special Publication 500-215, pp. 171-179, 1994.
- [8] W.B. Cavnar, "Using An N-Gram Based Document Representation With A Vector Processing Retrieval Model", in *Proceedings of the Third Text Retrieval Conference (TREC-3)*, NIST Special Publication 500-225, pp. 269-277, 1995.
- [9] C.C. Chang, "The study of an ordered minimal perfect hashing scheme", in *Communications of the ACM*, Vol. 27, No. 4, pp. 384-387, 1984.



- [10] J.S. Chang, e. al., "Chinese word segmentation through constraint satisfaction and statistical optimization", in *ROCLING-IV*, Taiwan, pp. 147-165, 1991.
- [11] W.W. Chang, H.J. Schek, "A signature access method for the Starburst database system", in *Processings of the Fifteenth International Conference on Very Large DataBases*, Amsterdam, The Netherlands, pp. 145-153, Aug. 1989.
- [12] Y.R. Chao, *A Grammar of Spoken Chinese*, University of California Press, Berkeley, 1968.
- [13] K.J. Chen, S.H. Kiu, "Word identification for Mandarin Chinese sentence", in *5<sup>th</sup> International Conference on Computational Linguistics*, pp. 101-107, 1992.
- [14] T.H. Chiang, e. al., "Statistical models for segmentation and unknown word resolution", in *5<sup>th</sup> R.O.C. Computational Linguistics Conference*, pp. 123-146, 1992.
- [15] L.F. Chien, T.I. Huang, S.P Lee, H.C. Li, " '尋易' (CSmart) – A High Performance Chinese Document Retrieval System", in *Proceedings of the 1995 International Conference on Computer Processing of Oriental Languages*, pp. 176-183, 1995.
- [16] L.F. Chien, "Fast and Quasi-Natural Language Search for Gigabytes of Chinese Texts", in *SIGIR 95*, pp. 112-120, 1985.
- [17] S. Christodoulakis, C. Faloutsos, "Design considerations for a message file server", in *IEEE Trans. Soft. Eng.*, Vol. SE-10, No. 2, pp. 201-210, Mar. 1984.
- [18] S. Christodoulakis, M. Theodoridou, F. Ho, M. Papa, A. Pathria, "Multimedia document presentation, information extraction and document formation in MINOS: A model and a system", in *ACM Transactions on Office Information Systems*, Vol. 4, No. 4, pp. 345-383, Oct. 1986.
- [19] M. Damashek, "Gaushing Similarity with  $n$ -Grams: Language-Independent Categorization of Text", in *Science*, 267(10), pp. 843-848, 1995.
- [20] U. Deppisch, "S-Tree: A dynamic balanced signature index for office retrieval", in *Proceedings of the ACM Conference on Research and Development in Information Retrieval*, Pisa, Italy, pp. 77-87, Sept. 1986.

- [21] T. Dunning, "Accurate Methods for the Statistics of Surprise and Coincidence", in *Computational Linguistics*, Vol. 19, pp. 67-74, 1993.
- [22] J.R. Files, H.D. Huskey, "An information retrieval system based on superimposed coding", in *Proceedings of AFIPS Fall Joint Computer Conference*, Vol. 35, AFIPS Press, Arlington, Va., 1969.
- [23] W.B. Frakes, R. Baeza-Yates, *Information Retrieval: Data Structures and Algorithms*, Englewood Cliffs, N.J., Prentice Hall, 1992.
- [24] C. Faloutsos, S. Christodoulakis, "Signature Files: An Access Method for Documents and Its Analytical Performance Evaluation", in *ACM Transactions on Office Information Systems*, Vol. 2, No. 4, pp. 267-288, Oct. 1984.
- [25] C. Faloutsos, S. Christodoulakis, "Design of a Signature File Method that Accounts for Non-Uniform Occurrence and Query Frequencies", in *Proceedings of the 11<sup>th</sup> VLDB Conference*, Stockholm, Aug. 21-23, pp. 165-170, 1985.
- [26] C. Faloutsos, "Text retrieval methods", in *ACM Computing Survey*, Vol. 17, No. 1, pp. 49-74, 1985.
- [27] C. Faloutsos, S. Christodoulakis, "Optimal Signature Extraction and Information Loss", in *ACM Transactions on Database Systems*, Vol. 12, No. 3, pp. 395-428, Sept. 1987.
- [28] C. Faloutsos, "Description and performance analysis of signature file methods for office filing", in *ACM Transactions on Office Information Systems*, Vol. 5, No. 3, pp. 237-257, 1987.
- [29] C. Faloutsos, "Signature-based text retrieval: a survey", in *IEEE Data Engineering*, Vol. 13, No. 1, pp. 25-32, 1990.
- [30] H. Fuji, W.B. Croft, "A comparison of indexing techniques for Japanese text retrieval", in *Research and Development in Information Retrieval, ACM-SIGIR*, pp. 237-246, 1993.
- [31] K.L. Kwok, "Comparing Representations in Chinese Information Retrieval", in *Proceedings of 20th Ann. Intl. ACM SIGIR Conf. on R&D in IR*, pp. 34-41, 1997.
- [32] R.L. Haskin, "Special-purpose processors for text retrieval", in *Database Engineering*, Vol. 4, No. 1, pp. 16-29, Sept. 1981.



- [33] K.K. He, H. Xu, B. Sun, "The Design Principle for a Written Chinese Automatic Segmentation Expert System", in *Journal of Chinese Information Processing*, Vol. 5, pp. 1-14, 1991.
- [34] L.A. Hollaar, "Text retrieval computers", in *IEEE Computer*, Vol. 12, No. 3, pp. 40-50, Mar. 1979.
- [35] IBM, "STAIRS/VS: Reference Manual", in *IBM System Manual*, 1979.
- [36] G. Jaeschke, "Reciprocal hashing: A method for generating minimal perfect hashing functions", in *Communications of the ACM*, Vol. 24, No. 12, pp. 829-833, 1981.
- [37] C.Y. Jie, Y. Liu, N.Y. Liang, "The design and realization Chinese automatic segmenting system CASS", in *Journal of Chinese Information Processing*, Vol. 5, No. 4, pp. 27-34, 1991.
- [38] W. Jin, "A Case Study: Chinese segmentation and its disambiguation", in *Technical Report MCCS-92-227*, Computing Research Laboratory, New Mexico State University, Las Cruces, 1992.
- [39] D.E. Knuth, *The Art of Computer Programming*, Addison-Wesley, Reading Mass., 1968-1973.
- [40] D.E. Knuth, J.H. Morris, V.R. Pratt, "Fast pattern matching in strings", in *SIAM J. Comput.*, Vol. 6, No. 2, pp. 323-350, Jun. 1977.
- [41] W. Lam, C.Y. Wong, K.F. Wong, "Performance Evaluation of Character-, Word- and N-Gram-Based Indexing for Chinese Text Retrieval", in *Proceedings of the 2<sup>nd</sup> International Workshop on Information Retrieval with Asian Languages - 1997 (IRAL'97)*, pp. 68-80, 1997.
- [42] K.T. Lau, "From Character to Word - an Application of Information Theory", in *Computer Processing of Chinese and Oriental Languages*, Vol.4, No. 4, pp. 304-313, 1990.
- [43] D.L. Lee, C.W. Leng, "Partitioned Signature Files: Design Issues and Performance Evaluation", in *ACM Transactions on Information Systems*, Vol. 7, No. 2, pp. 158-180, 1989.
- [44] D.L. Lee, C.W. Leung, "A partitioned signature file structure for multiattribute and text retrieval", in *Proceedings of the 6<sup>th</sup> International Conference on Data Engineering*, Los Angeles, California, pp. 389-397, 1990.



- [45] D.L. Lee, Y.M. Kim, P. Gaurav, "Efficient Signature File Methods for Text Retrieval," in *IEEE Transactions on Data and Knowledge Engineering*, Vol. 7, No. 3, pp. 423-435, Jun. 1995.
- [46] J.H. Lee, J.S. Ahn, "Using  $n$ -Grams for Korean Text Retrieval", in *SIGIR 96*, pp. 216-224, 1996.
- [47] C.R. Leng, D.L. Lee, "Optimal Weight Assignment for Signature Generation", in *ACM Transactions on Database System*, Vol. 17, No. 2, pp. 346-373, 1992.
- [48] B.I. Li, e. al., "A maximal matching automatic Chinese word segmentation algorithm using corpus tagging for ambiguity resolution", in *R.O.C. Computational Linguistics Conference*, Taiwan, pp. 135-146, 1991.
- [49] Y. Liu, "The rules of modern Chinese segmentation for the purpose of information processing and approaches of automatic Chinese segmentation", in *Beijing: Tsinghua University Press*, pp. 36-63, 1994.
- [50] N.Y. Liang, Y.B. Zhen, "A Chinese word segmentation model and a Chinese word segmentation system PC-CWSS", in *COLIPS*, Vol. 1, pp. 51-55, 1991.
- [51] T. Liang, S.Y. Lee, W.P. Yang, "Approximating false hits of disyllabic terms in Chinese signature file", in *International Journal of Information Science and Engineering*, 1994.
- [52] T. Liang, S.Y. Lee, W.P. Yang, "Optimal Weight Assignment for a Chinese Signature File", in *Information Processing and Management*, Vol. 32, No. 2, pp. 227-237, 1996.
- [53] M.Y. Lin, T.H. Chiang, K.Y. Su, "A preliminary study on unknown word problem in Chinese word segmentation", in *ROCLING V*, pp. 147-176, 1992.
- [54] D.J. Lipman, W.R. Pearson, "Rapid and sensitive protein similarity searches", in *Science*, Vol. 227, pp. 1435-1441, Mar. 1985.
- [55] C. Mooers, *Application of random codes to the gathering of statistical information*, Bull, 21, Zator Co., Cambridge, Mass., 1949.
- [56] J.Y. Nie, Martin Brisebois, "On Chinese Text Retrieval", in *Proceedings of the 19<sup>th</sup> Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 225-233, 1996.
- [57] J.Y. Nie, "An information retrieval model based on modal logic", in *Information Processing & Management*, Vol. 25, No. 5, pp. 447-491, 1989.



- [58] Y. Ogawa, A. Bessho, M. Hirose, "Simple word strings as compound keywords: An indexing and ranking method for Japanese text", in *Research and Development in Information Retrieval, ACM-SIGIR*, pp. 227-236, 1993.
- [59] Y. Ogawa, M. Iwasaki, "A new character-based indexing organization using frequency data for Japanese documents", in *Research and Development in Information Retrieval, ACM-SIGIR*, Seattle, pp. 121-129, 1995
- [60] J.L. Pfaltz, W.J. Berman, E.M. Cagley, "Partial match retrieval using indexed descriptor files", in *Communications of the ACM*, Vol. 23, No. 9, pp. 522-528, 1980.
- [61] C.J. van Rijsbergen, *Information Retrieval 2<sup>nd</sup>*, London:Butterworths, 1979.
- [62] C.J. van Rijsbergen, "A new theoretical framework for information retrieval", in *ACM/SIGIR Conference Proceeding*, 1986.
- [63] C.J. van Rijsbergen, M. Lalmas, "Information Calculus for Information Retrieval", in *Journal of the American Society for Information Science*, Vol. 47, No. 5, pp. 385-398, 1996.
- [64] C.S. Roberts, "Partial-match retrieval via method of superimposed codes", in *Proceedings of IEEE*, Vol. 67, No. 12, pp. 1624-1642, 1979.
- [65] S.E. Robertson, K. Sparck Jones, "Relevance weighting of search terms", in *Journal of the American Society for Information Science*, 27, pp. 129-146, 1976.
- [66] S.E. Robertson, van Rijsbergen, M.F. Porter, "Probabilistic model of indexing and searching", in *Information retrieval research*, London:Butterworths, 1981.
- [67] R. Sacks-Davis, K. Ramamohanarao, "A two level superimposed coding scheme for partial match retrieval", in *Information Systems*, Vol. 8, No. 4, pp. 273-280, 1983.
- [68] R. Sacks-Davis, A. Kent, "Multikey Access Methods Based on Superimposed Coding Techniques", in *ACM Transactions on Database Systems*, Vol. 12, No. 4, pp. 655-696, Dec. 1987.
- [69] G. Salton, A. Wong, C.S. Yang, "A Vector Space Model for Automatic Indexing", in *Communication of the ACM*, Vol. 18, No. 11, pp. 613-620, 1975.
- [70] G. Salton, M.J. McGill, *Introduction to Modern Information Retrieval*, New York, McGraw-Hill, 1983.

- [71] G. Salton, *Automatic Text Processing : The transformation, analysis, and retrieval of information by computer*, Addison-Wesley, 1988.
- [72] M.A. Shepherd, W.J. Phillips, C.K. Chu, "A fixed-size bloom filter for searching textual documents", in *Computer Journal*, 32(3), pp. 212-219, 1989.
- [73] R. Sproat, C. Shih, "A statistical method for finding word boundaries in Chinese text", in *Computer Processing of Chinese and Oriental Languages*, Vol. 4, pp. 336-351, 1991.
- [74] R. Sprugnoli, "Perfect hashing function: A single probe retrieving method for static sets", in *Communications of the ACM*, Vol. 20, No. 11, pp. 841-850, 1977.
- [75] C. Stanfill, B. Kahle, "Parallel free-text search on the connection machine system", in *Communications of the ACM*, Vol. 29, No. 12, pp. 1229-1239, Dec. 1986.
- [76] M.S. Sun, e. al., "Some Issues on the statistical approach to Chinese Word Identification", in *3<sup>rd</sup> International Conference on Chinese Information Processing*, pp. 246-253, 1992.
- [77] O. Vallarino, "On the use of bit maps for multiple key retrieval", in *ACM SIGPLAN Notices 11*, pp. 108-144, Mar. 1976.
- [78] E.M. Voorhess, D. Harman, "Overview of the Fifth Text REtrieval Conference (TREC-5)", in *Information Technology: The Fifth Text REtrieval Conference (TREC-5)*, pp. 1-28, 1996.
- [79] L.J. Wang, T. Pei, W.C. Li, L.C. Huang, "A Parsing method for identifying words in Mandarin Chinese sentences", in *12<sup>th</sup> International Joint Conference on Artificial Intelligence*, Sydney, Australia, pp. 1018-1023, 1991.
- [80] C.Y. Wong, W. Lam, K.F. Wong, "PSFC - A Partitioned Signature File Indexing Approach for Chinese Information Retrieval", *The 3<sup>rd</sup> International Workshop on Information Retrieval with Asian Languages - 1998 (IRAL'98)*, to be appeared in 1998.
- [81] K.F. Wong, V.Y. Lum and C.H. Leung, "Parallel Chinese Word Boundaries Identification in The IPOC Information Retrieval System", in *The International Journal of Information Technology*, World Scientific Pub. Co, Vol.3(1), pp. 63-81, 1997.



- [82] Z. Wu, G. Tseng, "Chinese text segmentation for text retrieval: Achievements and problems", in *Journal of the American Society for Information Science*, Vol. 44, pp. 532-542, 1993.
- [83] Z. Wu, G. Tseng, "ACTS: An automatic Chinese text segmentation system for full text retrieval", in *Journal of the American Society for Information Science*, Vol. 46, pp. 83-96, 1995.
- [84] H. Xu, K.K. He, B. Sun, "The implementation of a written Chinese automatic segmentation expert system", in *Journal of Chinese Information Processing*, Vol. 5, pp. 38-47, 1991.
- [85] T.S. Yao, G.P. Zhang, Y.M. Wu, "A rule-based Chinese automatic segmentation system", in *Journal of Chinese Information Processing*, Vol. 4, pp. 37-43, 1990.
- [86] C.L. Yeh, and e. al, "Rule-based word identification for Mandarin Chinese sentences - A unification approach", in *Computer processing of Chinese and Oriental Languages*, Vol. 5, 1991.
- [87] C.T. Yu, C. Buckley, K. Lam, G. Salton, "A generalized term dependence model in information retrieval", in *Information Technology: Research and Development*, 2, pp. 129-154, 1983.
- [88] P. Zezula, "Dynamic partitioning of signature files", in *ACM Transactions of Information Systems*, Vol. 9, No. 4, pp. 336-339, 1991.
- [89] Y.X. Zhou, W.T. Wu, "A Practical Method of Segmentation of Chinese - A Method Based upon Chain Table", in *Journal of Chinese Information Processing*, Vol. 4, pp. 34-41, 1989.



CUHK Libraries



003704111