A Fixed-Size Bloom Filter for Searching Textual Documents

M. A. SHEPHERD,* W. J. PHILLIPS† AND C.-K. CHU†

* *Computing Science Division, Dalhousie University, Halifax, Nova Scotia, Canada B3H 3J5*

† *Technical University of Nova Scotia, Halifax, Nova Scotia, Canada B3J 2X4*

The empirical false drop rate associated with a fixed-size Bloom filter used to represent textual documents may be quite different than the theoretical rate. This problem arises when the filter size is based on the expectation of a uniform distribution of the number of different terms per document. The distribution is, in fact, not uniform. This paper describes a method to determine the filter size for a database of textual documents, based on the desired false drop rate and the actual distribution of different words over the documents for that database. Theoretical and experimental results are reported and indicate that a filter size based on this method produces empirical false drop rates equivalent to the theoretical rates. The filter was also compared to variable-length filters with respect to storage requirements and search times.

Received June 1988, revised October 1988

1. INTRODUCTION

In recent years, the signature file approach to the retrieval of textual material has received increasing attention.[1–4] The approach has been applied to document retrieval, office information systems, and text editing. In the application of this approach to retrieval systems, documents are stored in a 'text' file and their representations or 'signatures' are stored in a signature file for searching. Each document is represented in the signature file by a binary vector which may be generated by various methods.[1,5,6]

Retrieval of a document in the signature file approach occurs when all the bits set in the query signature are also set in a document signature. This retrieval method, however, is associated with an inherent false drop rate. A false drop occurs when a document is retrieved by a query term even though the query term is not present in the document. This happens when bits in the document signature corresponding to the bits in the query signature are set by a combination of other words occurring in the document.

Document retrieval systems typically provide access through Boolean queries using the 'AND', 'OR', and 'NOT' operators. This access is usually implemented through the use of inverted files.[7] Although the inverted file approach provides fast retrieval speed, it requires an index overhead of as much as 50–300% of the original file[8] and poses significant update problems. The signature file approach is attractive as it permits Boolean queries on a signature file that requires significantly less storage overhead,[9] poses minimal update problems, and lends itself to hardware searching.[4] This approach is sufficiently flexible to extend the Boolean model to include the 'ADJ' (adjacency) operator by basing the signatures on triplets as opposed to whole words.[10] In addition, experiments indicate that the signature approach provides acceptable response times of only a few seconds when implemented on microcomputers for moderate-size databases.[11]

The signature file approach can also be used in the implementation of document retrieval systems not based on the Boolean model. It has been used in the retrieval of documents in a ranked order based on a simple query-document matching algorithm[12] and based on a probabilistic retrieval model.[13]

The document signature investigated in this research is based on Bloom filters.[14] This approach to signatures[15] adapts a technique originally intended to reduce the amount of space required to store hash-coded information. This method is very similar to the superimposed coding method for signature extraction.[1,6] The superimposed coding method divides each document into logical blocks containing a constant number of different words, creates a signature for each block, and concatenates these block signatures to form a document signature. The Bloom filter method differs from the superimposed coding method in that it does not break the document into logical blocks. Each different word in the document is passed through a series of hash transformations to set a number of bits in the filter (binary vector). The resulting filter is the document signature.

The size or number of bits in a Bloom filter is determined by the desired theoretical false drop rate and the number of different words expected in each document. This length is usually fixed for all documents of the database. Fixed-length filters that are based on the assumption of a uniform distribution of the number of different words per document exhibit poor performance with respect to the false drop rates.[6,15]

The desired false drop rate can be obtained in practice through the use of a set of variable-length filters developed for the database.[15] The variable-length approach partitions the database into a number of categories, depending on document length. The appropriate filter size is determined for each category and is used for each document in that category. In addition to giving the desired false drop rate, this approach also provides a fixed upper bound for storage loss.

The approach presented in this paper has been to develop a fixed-length Bloom filter based on the desired false drop rate and on the distribution of the number of different words per document over the database. The method is based on an estimation technique so that the actual distribution need not be known and the method is independent of the form of the distribution. Fixed-length filters based on this approach produce empirical false drop rates equivalent to the desired theoretical false drop rates. While this approach requires more storage than does the variable-length filter approach it does provide

the desired false drop rate without the inherent overhead associated with processing a set of variable-length filters.

2. THEORETICAL FALSE DROP RATE

Mullin's[15,16] analytical approach has provided the theoretical false drop rate for Bloom filters, as follows. Given that t transformations are used on w different words, the probability that a particular bit in the b bits of the filter is not set (given that all bits are equally likely to be selected) is

$$Pset' = (1-1/b)^{tw}.$$

This is the probability that each of the tw transforms set some other of the b bits. The probability that a bit is set is

$$Pset = 1-(1-1/b)^{tw}.$$

The probability that all bits are set by a random word and hence of a false drop is

$$Pallset = Pset^{ta},$$

where a is the number of words in the ANDed request. In order to keep the analysis tractable, the parameter a can be set to 1.

It has been shown that the optimum number of transformations is that where half of the number of bits in the filter are set to 1;[6,14] i.e. $Pset = 1/2$. Thus, at optimum, the probability of a false drop on a single word query would be

$$Pallset = Pset^t = (1/2)^t.$$

3. FILTER SIZE

Again from Mullin[15], the size of the filter should be proportional to the number of words per document, and the number of transforms per word for a desired false drop rate for single word queries, giving

$$Pe = [1-(1-1/b)^{tw}]^t. \quad (1)$$

As $Pe = Pallset = Pset^t = (1/2)^t$ for half the filter bits to be set, then the number of transformations t can be calculated to provide the desired false drop rate, Pe. For example, if the desired false drop rate is $Pe = 1/1024$, then $Pe = 1/1024 = (1/2)^t$ and $t = 10$.

If the filter size $b \gg 1$ and $tw \gg 1$ then $(1-1/b)^{tw}$ can be approximated by $e^{(-tw/b)}$. Thus $Pset = 1-e^{(-tw/b)}$. With $Pset$ set to its optimal value of 1/2, taking the natural logarithm of both sides gives

$$b = tw/\ln 2.$$

Thus the filter width b can be determined directly from the number of transformations t and the number of different words expected in the document. For example, for a desired false drop rate of 1/1024 the value of t would be 10. If the number of different words per document is uniform and $w = 5$, then the filter size b would be 73 bits.

This analysis assumes a uniform distribution of the number of different words per document, giving a constant w for a given database. As Mullin[15] has shown, filter widths based on this assumption lead to empirically poor performance with respect to the false drop rate because the distribution is, in fact, not uniform.

4. DISTRIBUTION-BASED FILTER SIZE

The approach in this paper is to determine the size for a filter based on the actual distribution of the number of different words per document over the database and the desired false drop rate.

If we consider a database in which the number of words w per document is not constant but has a certain distribution, then we can revise equation (1) using expected values such that:

$$Pe = E(1-K^w)^t \quad (2)$$

where

$$K = (1-1/b)^t \quad (3)$$

The probability of a false drop can be estimated for any given database by taking a random sample of documents of size n, determining the number of different words, w_i, $i = 1, 2, \ldots, n$, in each document of the sample, setting the number of transformations t, and then calculating the average false drop rate:

$$Pe = \left[\sum_{i=1}^{n} (1-K^{w_i})^t\right] \Big/ n$$

At optimum, $Pe = (1/2)^t$. Therefore,

$$(1/2)^t = \left[\sum_{i=1}^{n} (1-K^{w_i})^t\right] \Big/ n \quad (4)$$

Equation (4) can now be solved for K, using the bisection method. K will be in the range of $K = 1$ and $K = 0$, for which the false drop rate Pe will be 0 and 1, respectively.

The filter size b can now be determined by taking the natural logarithm of both sides of equation 3 to give:

$$b = 1/[1-e^{(\ln K)/t}] \quad (5)$$

and substituting in the value of t and the value of K, found by solving (4).

5. DATABASES

Two different textual databases were used in these investigations. The filters were created by transformations of the non-noise keywords extracted from selected fields. The transformation functions were developed and tested to ensure a uniform distribution of bits over the length of the filter.

Although a list of noise words[17] was used, it should be noted that the approach taken in this paper is independent of any such list. The filter size, b, is dependent on the desired false drop rate and the distribution of the number of different words per document over the database. If no list of noise words is used there will be more words per document, which will result in a larger filter size in order to achieve the desired false drop rate.

The first database contained 1,235 document surrogates of the articles appearing in the *Communications of the Association for Computing Machinery* (CACM) from 1970 to 1979. The keywords were extracted from the title and abstract fields. The average number of non-noise keywords was 36.7 per document and the maximum was 211.

A histogram illustrating the distribution of the number of different non-noise words per document surrogate in the CACM database is given in Fig. 1. The two peaks in the histogram indicate the presence of surrogates

No. of words (w)	No. of documents	Each * represents max 10 documents
$0 \leqslant w \leqslant 2$	11	**
$2 < w \leqslant 7$	326	**********************************
$7 < w \leqslant 12$	71	********
$12 < w \leqslant 17$	11	**
$17 < w \leqslant 22$	31	****
$22 < w \leqslant 27$	43	*****
$27 < w \leqslant 32$	53	******
$32 < w \leqslant 37$	64	*******
$37 < w \leqslant 42$	94	**********
$42 < w \leqslant 47$	93	**********
$47 < w \leqslant 52$	83	*********
$52 < w \leqslant 57$	59	******
$57 < w \leqslant 62$	73	********
$62 < w \leqslant 67$	56	******
$67 < w \leqslant 72$	48	*****
$72 < w \leqslant 77$	32	****
$77 < w \leqslant 82$	27	***
$82 < w \leqslant 87$	11	**
$87 < w \leqslant 92$	12	**
$92 < w \leqslant 97$	18	**
$97 < w \leqslant 102$	8	*
$102 < w \leqslant 107$	3	*
$107 < w \leqslant 112$	2	*
$112 < w \leqslant 117$	0	
$117 < w \leqslant 122$	1	*
$122 < w \leqslant 127$	2	*
$127 < w \leqslant 132$	0	
$132 < w \leqslant 137$	0	
$137 < w \leqslant 142$	0	
$142 < w \leqslant 147$	1	*
$147 < w \leqslant 152$	0	
$152 < w \leqslant 157$	0	
$157 < w \leqslant 162$	0	
$162 < w \leqslant 167$	0	
$167 < w \leqslant 172$	0	
$172 < w \leqslant 177$	1	*
$177 < w \leqslant 182$	0	
$182 < w \leqslant 187$	0	
$187 < w \leqslant 192$	0	
$192 < w \leqslant 197$	0	
$197 < w \leqslant 202$	0	
$202 < w \leqslant 207$	0	
$207 < w \leqslant 212$	1	*

Figure 1. Histogram of CACM database showing distribution of different words per document.

No. of words	No. of documents	Each * represents max 5 documents
0	1	*
1	16	****
2	89	******************
3	134	***************************
4	98	********************
5	65	*************
6	26	*******
7	19	****
8	10	**
9	7	**
10	1	*
11	1	*
12	0	
13	2	*
14	0	
15	0	
16	0	
17	1	*

Figure 2. Histogram of OON database showing distribution of different words per document.

consisting of titles only and of surrogates consisting of both titles and abstracts. The heterogeneity of the surrogates presents no problem for the approach taken in this paper as the method is independent of the form of the distribution.

The second database contained 470 document surrogates from the OON database of the Canada Institute for Scientific and Technical Information (CISTI). The OON database contains conference reports, monographs, and technical reports in science, technology and medicine held by CISTI and its branch facilities. The keywords were extracted from the title field of each surrogate. The average number of non-noise keywords was 3.8 per document and the maximum was 17. A histogram illustrating the distribution of the number of different non-noise words per document is given in Fig. 2.

6. EXPERIMENTS

Three experiments were conducted over the two databases. The first experiment was conducted to determine if desired theoretical false drop rates can be obtained experimentally using distribution-based fixed-length Bloom filters.

The second experiment was to compare the storage requirements and false drop rates of distribution-based fixed-length filters with those of fixed-length filters that assume a uniform distribution of the number of different words per document, and with those of variable-length filters. The storage requirements were also compared to those of the inverted file approach.

In order to determine false drop rates for these first two experiments, 3000 single-word queries were processed against the first database and 2000 single-word queries were processed against the second database. Each query was a non-noise word selected randomly from the appropriate database. Therefore, each query retrieved at least one document in which the term occurred. Single word queries were used to be consistent with the theoretical analysis of the false drop rate presented in Section 2, above.

The false drop rate for a specific query was calculated as $Fd/(D-Dq)$, where Fd is the number of false drops, D is the number of documents in the database, and Dq is the number of documents in which the query term actually occurred. Averaging over all of the queries gave the overall false drop rate.

The third experiment was to compare the speed of searching a database using variable-length filters with that of using the distribution-based fixed-length filter.

In order to determine the length of the distribution-based fixed-length filters, it was necessary to solve Equation (4). This required knowing the distribution of the number of different words per document. Although for a large database the distribution can be determind by sampling, the entire database was used for this experiment.

All the experiments were conducted on a VAX-11/785 and all programs were written in Pascal.

6.1 Theoretical *vs* empirical false drop rates

An experiment was conducted to determine if desired theoretical false drop rates can be obtained experimentally using distribution-based fixed-length Bloom filters.

Thirty different theoretical false drop rates were generated by varying the number of transforms from 1 to 30 for $(1/2)^t$. The corresponding filter size b was calculated for each value of t, using equations (4) and (5). Filters for each document and query were created for each of the 30 $t-b$ combinations. The queries were processed against the databases and the resulting false drop rates determined. A linear regression was performed on the experimental data to determine if the experimental false drop rates were equivalent to the theoretical false drop rates.

The algorithm for calculating the experimental false drop rates is:

```
for t:= 1 to 30 do
  begin
    find K from equation (4);
    find b by bisection from equation (5);
    run a number of queries;
    compute false drop rate = Pe;
    write (t, b, Pe)
  end;
```

At optimum, the desired false drop rate is $Pe = (1/2)^t$. Taking natural logarithms of both sides to linearize the equation gives, $\ln(Pe) = t * \ln(1/2)$.

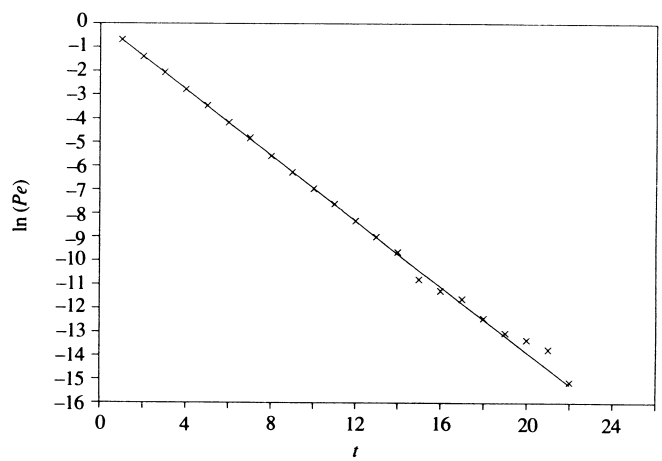

Figure 3. Linear regression of natural logarithms of experimental data for CACM database and the theoretical regression. —, Theoretical; ×, experimental.

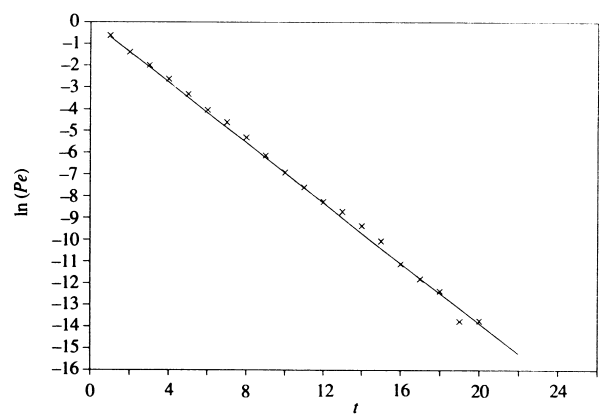

Figure 4. Linear regression of natural logarithms of experimental data for OON database and the theoretical regression. —, Theoretical; ×, experimental.

Table 1(*a*). Regression results for CACM database

Regression equation	$\ln(Pe) = -0.677 * t$
R^2	99.8%
Estimated slope	$-0.677 \pm (0.00709 * 2)$
Standard deviation of slope	0.00709

Table 1(*b*). Regression results for OON database

Regression equation	$\ln(Pe) = -0.692 * t$
R^2	99.9%
Estimated slope	$-0.692 \pm (0.00441 * 2)$
Standard deviation of slope	0.00441

Table 2(*a*). Proportion of filter bits set for CACM database

Transformed regression equation	$Pe = e^{-0.677*t} = 0.50814^t$
Estimated proportion of bits set	0.50814
95% confidence interval for bits set	0.4933 to 0.5219

Table 2(*b*). Proportion of filter bits set for OON database

Transformed regression equation	$Pe = e^{-0.692*t} = 0.50057^t$
Estimated proportion of bits set	0.50057
95% confidence interval for bits set	0.4913 to 0.5098

Figs 3 and 4 illustrate the results of the linear regressions of the natural logarithms of the experimental data for the CACM and the OON databases, respectively. The actual regression results are presented in Tables 1(*a*) and 1(*b*).

The proportions of bits set in the filters follow from the transformations of the regression results, as presented in Tables 2(*a*) and 2(*b*). As can be seen from these tables, the 95% confidence intervals for the proportion of bits set in the filters for the CACM and the OON databases includes the value 0.5. Therefore, the experimental false drop rates of the distribution-based fixed-length filters found for the CACM and OON databases is equivalent to the desired theoretical false drop rate, $Pe = (1/2)^t$, with 95% confidence.

Figs 3 and 4 do not show values of t for the full range of 1–30, for as t increases the false drop rate approaches 0. A preliminary analysis was performed to determine the maximum value of t for which a non-zero false drop rate can be expected. The minimum number of false drops for q queries over m documents is 1, giving a false drop rate of $1/mq$. Therefore the equation

$$(1/2)^t = 1/mq$$

can be rewritten as

$$t = \ln(mq)/\ln(2) \qquad (6)$$

and solved for t, the number of transformations required to give the minimum non-zero false drop rate. As any values of t greater than this should produce a false drop rate of 0, t can be treated as the maximum that will produce non-zero false drop rates.

Three thousand queries were run against the 1,235 documents of the CACM database, while 2,000 queries were run against the 470 documents of the OON database. Substituting these values into (6) gives value of $t = 22$ and $t = 20$, respectively. Therefore, the algorithm given above should produce non-zero false drop rates for values of t from 1 to 22 for the CACM database and of t from 1 to 20 for the OON database. This analysis was verified as correct by the experimental data.

6.2 Comparison with other filters

The storage requirements and experimental false drop rates of four different Bloom filters were compared. Theoretical false drop rates were selected arbitrarily to be 1/1024, 1/2048, 1/4096. The corresponding number of transforms, t, for each of these is 10, 11, and 12. The number of transforms, t, was used to calculate the size of the Bloom filters being compared.

The first two Bloom filters were the distribution-based fixed-length filter and the variable-length filter.[15] The third and fourth filters were fixed-length filters based on a uniform distribution of the number of different words per document. The third filter assumed that the number of different words per document, w, was the mean for that database. The fourth filter assumed that w was the maximum number of different words per document.

The storage requirements of these filters were compared also with the storage requirements of a simple inverted file. The file structure of the index is shown in Fig. 5. Each entry in the keyword file is a variable-length record consisting of a variable-length-term field and fixed-length fields of 4 bytes each for the number of postings and the pointer into the posting file. The postings file consists of 4-byte fixed-length entries containing the identifiers of the documents in which the keyterms occur.

Tables 3 and 4 present the storage requirements and experimental false drop rates for the CACM and OON databases, respectively. Each table contains the results for four different Bloom filters and the inverted-file approach. The storage requirements are presented as the total number of bits to store the index file. This does not include the storage required for the document database itself, only for the Bloom filters and, in the case of the inverted-file approach, only for the inverted file and the postings file. The data are presented in terms of the total

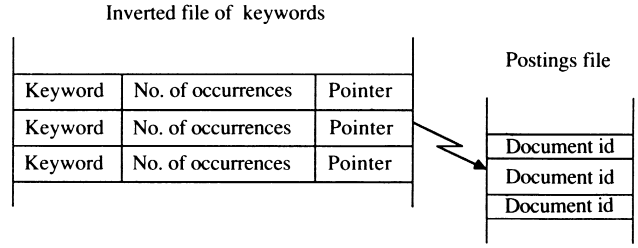

Figure 5. File structure of inverted-index approach.

Table 3. Experimental false drop rates and storage requirements for various retrieval methods on the CACM database

Theoretical false drop rate	Retrieval method	Number of different words	Experimental false drop rate	Index file size (in bits)
1/1024	Fixed	Mean	1/54	654550
	Fixed	Maximum	1/924752	3759340
	Variable	Variable	1/1328	691585
	Fixed	Distribution	1/1043	1242410
n/a	Inverted file	n/a	n/a	2448832
1/2048	Fixed	Mean	1/65	720005
	Fixed	Maximum	1/1804878	4134780
	Variable	Variable	1/2529	760471
	Fixed	Distribution	1/1975	1452360
n/a	Inverted file	n/a	n/a	2448832
1/4096	Fixed	Mean	1/83	785460
	Fixed	Maximum	1/3702003	4511455
	Variable	Variable	1/5856	829667
	Fixed	Distribution	1/4030	1689480
n/a	Inverted file	n/a	n/a	2448832

Table 4. Experimental false drop rates and storage requirements for various retrieval methods on the OON database

Theoretical false drop rate	Retrieval method	Number of different words	Experimental false drop rate	Index file size (in bits)
1/1024	Fixed	Mean	1/88	26320
	Fixed	Maximum	1/234625	115150
	Variable	Variable	1/1233	29785
	Fixed	Distribution	1/999	44650
n/a	Inverted file	n/a	n/a	227272
1/2048	Fixed	Mean	1/112	28670
	Fixed	Maximum	1/940000	126900
	Variable	Variable	1/2522	32543
	Fixed	Distribution	1/1967	52170
n/a	Inverted file	n/a	n/a	227272
1/4096	Fixed	Mean	1/152	31490
	Fixed	Maximum	1/937998	138180
	Variable	Variable	1/4045	35588
	Fixed	Distribution	1/3862	61100
n/a	Inverted file	n/a	n/a	227272

number of bits for the index file in order to be able to compare the storage requirements of the variable-length Bloom-filter approach to the other approaches.

In all instances, the false drop rate for the fixed-size filter based on the average number of words per document is unacceptably high. The filter based on the maximum number of words in a document gave far better false drop rates than required, but at a high cost in storage.

The distribution-based fixed-length filter gives experimental false drop rates essentially the same as the theoretical rate, whereas the variable-length filter gives false drop rates that appear to be better than the theoretical rates of 1/1024, 1/2048 and 1/4096. The rates of the variable-length filter averaged 24% lower than the theoretical rates for the CACM database and 13.3% lower for the OON database.

In addition, the variable-length filter requires an average of 47.6% less storage than the distribution-based fixed-length filter for the CACM database and 29.2% less storage for the OON database.

6.3 Search timings for fixed and variable-sized filters

The speed of searching a database using variable-length filters was compared against the speed of searching the same database using the distribution-based fixed-length filter. For each database, 100 non-noise words were selected randomly as single-word queries. The timings were taken for filter sizes based on false drop rates of 1/1024, 1/2048, and 1/4096.

The filter for each document was stored as an array of binary bits. Rather than generating a query filter, the query was represented by a list of bit positions that would be set in a filter. These query bit positions were then used, one at a time, to index into the document filter to determine if the corresponding document filter bit was 0 or 1. If the document filter bit was set, the next query-bit position was checked. If the document filter bit was not set, the query term did not occur in the document and the search progressed immediately to the next document filter.

Table 5. Mean search times in seconds and 95% confidence intervals for fixed and variable-length filters on the CACM database

Type of filter	Number of transforms (t)	Mean	Standard deviation	95% confidence intervals
Fixed	10	5.1138	0.1013	5.0937–5.1339
Fixed	11	5.1502	0.0945	5.1314–5.1689
Fixed	12	5.7147	0.0975	5.6953–5.7340
Variable	10	8.0941	0.1542	8.0635–8.1247
Variable	11	8.3533	0.1544	8.3227–8.3839
Variable	12	8.5958	0.1911	8.5579–8.6337

Table 6. Mean search times in seconds and 95% confidence intervals for fixed and variable-length filters on the OON database

Type of filter	Number of transforms (t)	Mean	Standard deviation	95% confidence intervals
Fixed	10	2.8272	0.0696	2.8133–2.8410
Fixed	11	2.8477	0.0586	2.8360–2.8593
Fixed	12	2.8343	0.0644	2.8215–2.8470
Variable	10	3.8513	0.0934	3.8327–3.8698
Variable	11	3.8567	0.0768	3.8414–3.8719
Variable	12	3.9096	0.0910	3.8915–3.9276

The signature file for the variable-length filters was not kept as a single contiguous file. The file was partitioned into a number of separate subfiles, each subfile containing all the filters of a particular length. Thus, if a database required n different-sized filters, the signature file consisted of n separate files. This would be more amenable to bit-slice processing in associative memory than would a single file of unordered filter sizes.

Tables 5 and 6 give the times to search the CACM and the OON databases, respectively. Each table shows the mean time in seconds, along with the 95% confidence interval, required to search the entire index file for 100 single-word queries at a desired false drop rate.

The variable-length filter method generated 28 different filter lengths for the CACM database and 8 for the OON database. As filters were kept in separate files by length, this resulted in opening and closing 28 files to search the CACM database and 8 files for the OON database.

The mean time is 'very well known' in each case since the confidence intervals are so narrow. The mean time to search the CACM database using the distribution-based fixed-length filter is definitely less than the mean time to search using the variable-length filter. This can be seen from the fact that the corresponding confidence intervals are separated by 3 seconds. Similarly, searching the OON database is definitely faster using the distribution-based fixed-length filter than using the variable-length filter, as the corresponding confidence intervals are separated by one second.

7. CONCLUSIONS

The results indicate that a distribution-based fixed-length Bloom filter is superior to fixed-length filters based on a uniform distribution of different words over the documents of a textual database. It is superior in that it produces false drop rates equivalent to the desired theoretical rates at a reasonable cost in storage. This is because the filter is based on the actual distribution of different words over the documents, which is not uniform.

The distribution-based fixed-length filter did not, however, perform as well as the variable-length filter with respect to false drop rates and storage requirements. This approach was superior with respect to the time required to search the entire signature file. However, this is because of the overhead of opening and closing a large number of files for the variable-length filters. It should be noted that Mullin[15] did not keep the filters in separate files but rather in a single, unordered file with a flag bit to indicate the start of the next filter. While Mullin's approach avoids the overhead of multiple files, it may not be as efficient for searching in associative memory.

In conclusion, the approach presented in the paper can be used when the application calls for fixed-length Bloom filters with a particular false drop rate.

REFERENCES

1. C. Faloutsos and S. Christodoulakis, Signature files: an access method for documents and its analytical performance evaluation. *ACM Transactions on Office Information Systems* **2** (4), 267–288 (1984).
2. F. Rabitti and J. Zizka, Evaluation of access methods to text documents in office systems. *Proceedings 3rd Joint ACM–BCS Symposium on Research and Development in Information Retrieval, Cambridge*, pp. 21–40 (1984).
3. C. Faloutsos, Access methods for text. *ACM Computer Surveys* **17** (1), 49–74 (1985).
4. E. Shuegraph, Signature searching: a review of theory and application. *Canadian Journal of Information Science* **12** (2), 22–35 (1987).
5. C. Faloutsos, Signature files: design and performance comparison of some signature extraction methods. *ACM SIGMOD*, pp. 63–82 (1985).
6. C. Faloutsos and S. Christodoulakis, Design of a signature file method that accounts for non-uniform occurrence and query frequencies. *Proceedings, VLDB, Stockholm* pp. 165–170 (1985).
7. G. Salton and M. McGill, *Introduction to Modern Information Retrieval.* McGraw-Hill, New York (1983).
8. R. Haskin, Special-purpose processors for text retrieval. *Database Engineering* **4** (1), 16–29 (1981).
9. C. Faloutsos and S. Christodoulakis, Design considerations for a message file server. *IEEE Transactions on Software Engineering* pp. 201–210 (1984).
10. C. Damier and B. Defude, The document management component of a multimedia data model. *Proceedings 11th International Conference on Research and Development in Information Retrieval, Grenoble* (1988).
11. M. Nelson, Comparison of signature and inverted files. Paper presented at the 16th Annual Conference of the Canadian Association for Information Science, Ottawa (1988).
12. C. Stanfill and B. Kahle, Parallel free-text search on the connection machine system. *Comm. ACM* **29** (12), 1229–1239 (1986).
13. W. B. Croft and P. Savino, Implementing ranking

strategies using text signatures. *ACM Transactions on Office Information Systems* **6** (1), 42–62 (1988).
14. B. Bloom, Space time tradeoffs in hash coding with allowable errors. *Comm. ACM* **13** (7), 422–426 (1970).
15. J. Mullin, Accessing textual documents using compressed indexes of small Bloom filters. *The Computer Journal* **30** (4), 343–348 (1987).
16. J. Mullin, A second look at Bloom filters. *Comm. ACM* **26** (8), 570–571 (1983).
17. B. Hunt, M. Snyderman and W. Payne, Machine-assisted indexing of scientific research summaries. *Journal of the American Society for Information Science* **26**, pp. 230–236 (1975).

Announcements

26–30 MARCH 1990

International Conference on Extending Database Technology, Fondazione Cini, Venice, Italy.

Promoted by the EDBT Foundation, sponsored by AFCET, AICA, BCS and GI, in cooperation with (requested) ACM.

Under the patronage of the Italian Ministry of Research and Technology, the Italian National Research Council and the Commission of European Communities.

The Conference

EDBT 90 will be a forum for presentation of new results in research, development, and applications of database technology. The conference will favour the sharing of information between researchers and practitioners and outline the future developments of database systems and applications.

Tutorials will be offered in the first two days of the Conference, and keynote speakers will be invited. The Conference and tutorials will be held at the Fondazione Cini on San Giorgio Island in Venice.

Topics of interest

EDBT 90 will accept scientific and technical papers on all areas related to database technology. The following list of topics should not be considered exclusive.

- Deductive databases
- Knowledge bases
- Multimedia databases, hypermedia systems
- Object-oriented database systems, Object managers
- Environment and technology to support data-base design and programming
- Database programming languages and persistent programming
- New applications of databases
- Active databases and real-time databases
- Performance issues and implementation techniques
- Extensible systems
- Database machines
- User interfaces

Information for authors

Five copies of a manuscript limited to 25 double-spaced pages (5000 words) should be submitted before **30 June 1989** to:
François Bancilhon, EDBT 90, Altaïr, Bp 105, 78153 Le Chesnay Cédex, France. e-mail: François @ bdblues.altair.fr

The Conference Proceedings will be edited and published by a major publishing house and will be distributed to the conference participants.

Important dates

30 June 1989 – submission deadline
30 September 1989 – acceptance notification
30 November 1989 – camera-ready copies due

Venice is a very busy and popular tourist centre. Thus, in order to facilitate arrangements and to ensure the reservation of hotel accommodation for all participants, you are kindly requested to contact quickly the Conference Secretariat:

Manuela Mennucci, IEI-CNR, Via S. Maria, 46, 56126 Pisa, Italy. Tel. 39-50-500159. Telex: 39-50-590305. Fax: 39-50-500342. e-mail: CASTELLI @ ICNUCEVM.BITNET.