Information and Software Technology 1994 **36** (3) 131–139

Binary ranking for the signature file method

D Dervos*

School of Computing and Information Technology, University of Wolverhampton, Wolverhampton WV1 1SB, UK

P Linardis

Department of Informatics, Aristotle University, Thessaloniki 54006, Greece

Y Manolopoulos†

Department of Computer Science, University of Maryland, College Park, MD 20742 USA

A new method for ranking the output of the superimposed variation of the signature file method (SC/SF) is presented. The method is termed 'Binary Ranking' and assigns a value ('B-rank value') to each block candidacy of the signature file output in a way which reflects the *credibility* of the signature to stand in place of the corresponding real text. In order to measure the performance of the proposed technique, a simulation environment based on a Relational Database Management System is established. Binary Ranking is found to be promising for a category of real-life applications and allows for future enhancements.

The signature file method

The signature file is an information retrieval access method suitable for processing large text databases[1]. The method has been studied extensively and a number of variations have been proposed[2]. It exhibits many advantages over alternative methods (full text scanning and inversion) and, therefore, it has been applied in numerous systems and applications.

The superimposed coding variation of the signature file method (SC/SF) uses a compressed binary representation of textual data to efficiently process simple one-word or even more sophisticated Boolean type queries. Raw text (i.e. documents like letters, newspaper articles, etc.) is logically broken down into blocks containing a fixed number of distinct non-common words. Each non-common word is then mapped on to a uniquely specified binary pattern by using a signature pattern extraction algorithm. As a result, a number of 1s are set in an ordered, finite sized population of binary cells/slots. The latter is the block's signature pattern which is initialized to contain all 0s prior to the activation of the signature extraction algorithm. The patterns of all the words in each block are superimposed (OR-ed) to yield a compressed binary representation corresponding to the real text in question.

Word	Signature			
Free	001	000	110	010
Text	000	010	101	001
Block signature	001	010	111	011

Figure 1 Superimposed signature file method: block signature generation and usage.

Figure 1 (adapted from Lee and Leng[3]) shows the basic principles of the technique. Each block is taken to comprise two distinct words. Each word sets four bit positions to 1 in the [1...12] range. Ignoring the details of the word signature pattern extraction algorithm, say that the word *text* sets bit positions 5,7,9 and 12 whereas *free* sets bit positions 3,7,8 and 11. The two-word signature patterns are superimposed (OR-ed) to yield the shown block signature pattern which comprises seven 1-valued bit positions in the [1...12] range.

User queries are efficiently processed when checked against the binary representation instead of having to perform CPU and I/O intensive full text scanning operations. More specifically, each query is processed by scanning the binary representation plus a small portion of

* On sabbatical leave from the Department of Informatics, TEI, Thessaloniki, 54101, Greece.
† On sabbatical leave from the Department of Informatics, Aristotle University, Thessaloniki 54006, Greece.

0950-5849/94/030131–09

the large text database. For example, given a query searching for the word 'free' (or the word 'text'), the block in Figure 1 is candidate to contain this word, as the relevant bit positions are found to be set to 1 in its signature pattern. In an analogous manner, by processing the query for the word 'data', which (say) sets to 1 the bit positions 2,6,11 and 12, one easily concludes that the block in question does not contain the specific word. However, when searching for the word 'base', which (say) sets to 1 the bit positions 3,7,8 and 9, a false result (called 'False Drop', FD) is obtained, indicating that the word is present in the block of Figure 1.

It is evident that the SC/SF method is deterministic in saying '*No, it is not there*' but contains some '*noise*' or fuzziness in indicating the presence of a word in a given logical block of textual data. Two factors contribute to causing the above stated fuzziness:

- the signature extraction algorithm, whereby two distinct words may set to 1 (possibly in a different order) the same m bits;
- the superimposition process, whereby m cells/slots may be set to 1 not only by a single word but by OR-ing the patterns of more than one word.

The latter gives the false impression that a word instance is present in the logical block, whereas in reality it is not (i.e. it is an FD).

Successfully tackling the issue of false drops plays a key role in improving the performance of the method. The number of distinct non-common words per block (D), the number of bits set to 1 in each word's pattern (m) and the size of the block's signature pattern in bits (F) comprise a set of design parameters (see Table 1). It has been proven that once the equation:

$$F \times \ln 2 = m \times D \tag{1}$$

holds, the technique is guaranteed to provide optimal results[4]. This implies maximum information content for the intermediary binary representation next to the original text. As a direct consequence, one then measures a minimum false drop rate.

To realize the quantitative (as opposed to qualitative) nature of the FD issue, one could consider the following example: let $F = 1000$, $m = 7$ and $D = 100$. Say that the first word in the block has just set its seven (distinct) bit positions to 1. The most likely thing to happen next is that the signature pattern of the second word will not have any one of its own 1s match with a 1 set by the first word. The resulting pattern, representing just two words for the time being, appears to accommodate

$$\binom{14}{7} = 3432 \text{ '}words\text{'}$$

Table 1. Definition of symbols

Symbol	Definition
F	Block signature size (in bits)
m	Number of bits set to 1 by each word
D	Number of distinct words in block
F'	Partition size (in bits)

as there exist that many ways of choosing groups of seven out of a given set of membership of 14. It is clear that 3430 of the above (possible) pattern-match instances are nothing but potential FD instances.

The SC/SF method is widely accepted for implementing text retrieval systems. When compared to inversion, SC/SF is found to be efficient in many respects:

- One need not worry about filtering out multiple occurrences of the same non-common word in a block of text. The issue is automatically taken care of by the signature extraction algorithm.
- It does not involve any type of re-organization of the index structure during subsequent insert operations.
- It calls for a very small storage overhead (e.g. 10–15%[1]).
- Because of the compressed nature of the intermediary binary representation, query processing is carried out in a most efficient manner.
- The method easily adapts itself to handling involved queries of a Boolean structure. For example, a query like: *Retrieve all the documents that involve the words 'Physics' AND 'Science' AND 'Equation' AND 'Energy' AND 'Field'* can easily be handled by a signature file configuration, whereas it would be a tough task to be undertaken by a Sciences database that utilizes inverted files.

However, the SC/SF method is mostly applicable in environments calling for insert/append (as opposed to update) operations, plus it is not exact in predicting the existence of a word within a block of text[1,4].

Assuming that the 1s in each block signature pattern follow the binomial distribution[3], the False Drop Probability (FDP) value for the system can be approximated by[4]

$$FDP \approx \left[\frac{M'}{F}\right]^m \tag{2}$$

where M' is the average number of bit positions set to 1 in the $[1 \ldots F]$ range of the block's signature pattern. The FDP value is by definition the rate at which a (any) single-word query produces false drops when its signature is checked against any one of the block signature patterns of the binary image taken to represent the text database.

The entropy/information content maximization rule states that the maximum amount of information is conveyed by an information carrying bit when it has equal probability to be 0 or 1[5]. Such a result is obtained by maximizing the value of the entropy function:

$$H(p) = -p \times \ln(p) - (1-p) \times \ln(1-p) \tag{3}$$

One can generalize the above and say that a block signature pattern conveys maximum information when half of its bit positions are set to 1, the other half being set to 0. This means that any bit position chosen at random is either a 1 or a 0 with probability equal to 0.5. A rigorous proof of the above stated intuitive approach can be found in[4]. Equation (2) now reads as:

$$FDP \approx \left[\frac{1}{2}\right]^m \tag{4}$$

Equation (4) implies that once the F, D and m design parameters are set to values satisfying Equation (1), then the (approximate) *FDP* value depends on m only.

Assigning B-rank values to the SC/SF output

The SC/SF output comprises a number of block candidacies. Each one of the latter is equally likely to satisfy the condition imposed by the query. A deterministic result is reached only after one scans through the text of all the candidate documents.

Equation (4) indicates that a limit is reached with regard to the maximum possible information content of the signature pattern. To increase the information content of the SC/SF configuration, one has to decrease the *FDP* value appearing in Equation (4), and this can only be achieved by increasing m (the number of bits set to 1 by each word). However, upon increasing m, F (the storage overhead) increases so that Equation (1) continues to hold.

There exists a number of reasons why a text database user would prefer to receive the system's response ordered according to the degree of relevance or usefulness with respect to his/her queries. The issue of ordering the system's response has been tackled in the context of traditional Information Retrieval (IR). Robertson[6] justifies the probability ranking principle in IR. More recently, the ranking principle has been investigated in the context of signature files[7,8]. There exist real-life applications that would benefit from having the output of the SC/SF method presented in a form where each candidate block is assigned a rank value which reflects its probability of being relevant to the user query. Croft and Savino[7] have introduced ranking to the signature file method by coupling it to a probabilistic study of the corresponding textual data contents.

It is possible for the user to have a specific document in mind during a document retrieval session, a document the title of which he is unable to recall. However, he is sure to recognize the document upon inspecting part of its contents. This means that the user may not be interested in retrieving every document that meets the conditions of his query. During such a special (yet, possible) SC/SF session, the full text scanning operation on the document candidacies will be abandoned once the specific document has been retrieved. A ranked SC/SF output would be particularly useful in serving such a type of query, especially when the user accesses a remote (i.e. costly to file transfer) environment. Another, more involved, example, could be a case where the SC/SF environment might be used (because of its retrieval efficiency) as a front-end processing subsystem to an effective (yet inefficient) back-end information retrieval (IR) environment. Allowing for the *Recall* metric to achieve a less than 1 value[9], the SC/SF output could thus comprise a representative *sample* set of documents which is used by the back-end system at an early stage of its feedback circle. Selecting the (say) 10 top ranked candidacies in such an environment would perform better than selecting any 10 from the classical SC/SF output.

With the above, it is understood that ranking increases the information content of the SC/SF output. Croft and Savino[7] have achieved this at the cost of some additional probabilistic calculations on the corresponding text base. The present study introduces a technique which calculates the 'B-rank value' (to differentiate it from what ranking came to imply up until now) at the cost of removing part of the deterministic nature of the SC/SF method. As has been explained in the first paragraph of this section, the SC/SF is deterministic in rejecting blocks from being candidates to satisfy a query by considering their signature patterns, only.

The B-rank values calculated in the following do not reflect any degree of relevance of each block candidacy to the given query as done by Croft and Savino[7]. Instead, the B-rank value is calculated by decoding information registered in addition to the binary representation of the classical SC/SF environment. In order to encode additional information with minimal storage overhead, a new representation is introduced which reuses the pattern of the first/classical one. This new binary representation can be decoded at query processing time to calculate a B-rank value which is then assigned next to each block candidacy.

The technique

A special case of Equation (1) is when $m = 1$:

$$F' \times \ln 2 = 1 \times D \tag{5}$$

Combining Equations (1) and (5) and solving for F' one has:

$$\frac{F}{F'} = \frac{m}{1} \Rightarrow F' = \left\lceil \frac{F}{m} \right\rceil \tag{6}$$

Equation (6) suggests that rather than having each word produce m bits, each in the $[1 \ldots F]$ range (as in Figure 2), one could have an equivalent partitioned configuration like the one shown in Figure 3.

Figure 3 considers the m bits corresponding to a word signature as being an ordered set. Within a block of text, the first bits of the D word signature patterns are directed to and are superimposed on to partition number one of the block signature pattern. Similarly, the second bits (D again, in all) are directed to partition number two, etc. It is worth noting that each partition accommodates bit positions/cells in

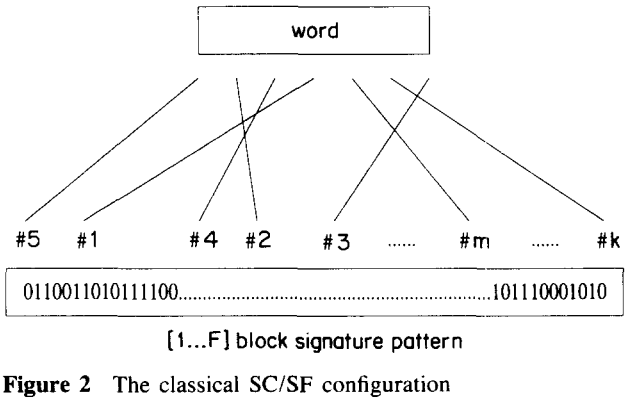

Figure 2 The classical SC/SF configuration

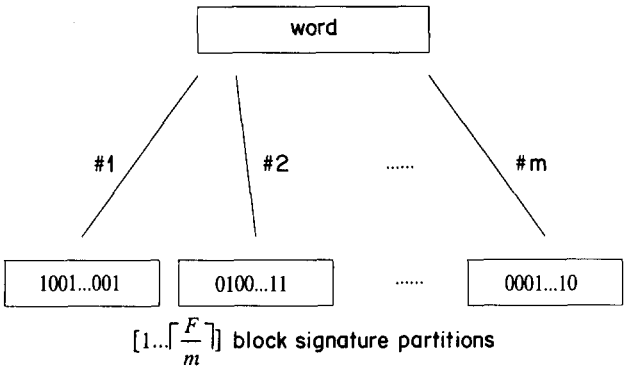

Figure 3 The partitioned SC/SF configuration

the $[1..\lceil \frac{F}{m} \rceil]$ range. The latter implies the existence of an appropriately modified word signature pattern extraction algorithm which is different from the one of the classical SC/SF. The use of the word *partition* in this study should not be confused with other partitioning techniques which have been found to improve the efficiency of the SC/SF method[2,3].

Equation (5) holds true for each one of the m partitions in Figure 3. Because of the entropy/information content maximization rule, each one of the m partitions will thus have half of its bit positions set to 1, on average. Assuming that each one of the m bit positions set to 1 by a (any) word's pattern is independent (orthogonal) of the rest m-1 bit positions set by the same word, a word chosen at random will have a

$$\frac{1}{2} \times \frac{1}{2} \times \ldots \frac{1}{2} = \left[\frac{1}{2}\right]^m \tag{7}$$

probability of producing a 'block is candidate to contain' result when its signature pattern is checked against that of a (any) block. It is assumed that the membership of the dictionary of words used is much larger than D, the blocking factor of the specific SC/SF configuration. Thus, the value calculated in Equation (7) is a realistic approximation of the FDP value. The FDP result just obtained for the partitioned variation of the SC/SF is equal to that found in Equation (4) which applies to the classical SC/SF. In agreement with the simulation results that follow, this proves the partitioned SC/SF variation shown in Figure 3 to be equivalent to the classical one appearing in Figure 2. The partitioned SC/SF calls for a very small amount of additional storage overhead which is considered to be negligible.

Proceeding with the modified SC/SF configuration shown in Figure 3, each logical block of textual data corresponds to a row in each one of the m partitions of the signature file. A (any) row of a (any) partition in the signature file is a 0/1 pattern in the $[1..\lceil \frac{F}{m} \rceil]$ range. A horizontal series of m row patterns (called *'partitions'* in the following) are taken together to logically comprise the compressed binary image of a D-word block of textual data.

Figure 4 illustrates the structure of the proposed enhanced SC/SF configuration with Binary Ranking. A sample case where $m = 7, F' = 144$ and $D = 100$ of the partitioned SC/SF environment is considered. The B-ranking configuration is established in addition to the partitioned SC/SF one: each word produces its $m = 7$ SC/SF bits, each one of which is directed to the corresponding block signature partition as shown in Figure 3. In addition to this, each word produces some extra bits, each in the [1...144] range. Labelling the bit positions set to 1 in the SC/SF method as m_1, m_2, ..., m_7, seven (say) extra bit positions that characterize the same word could be produced by the following expressions:

$$c_1 = ((m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7) \; mod \; 144) + 1$$
$$c_2 = ((m_1 + m_2 + m_3 + m_4 + m_5 + m_6) \; mod \; 144) + 1$$
$$c_3 = ((m_1 + m_2 + m_3 + m_4 + m_5) \; mod \; 144) + 1$$
$$c_4 = ((m_1 + m_2 + m_3 + m_4) \; mod \; 144) + 1$$
$$c_5 = ((m_1 + m_2 + m_3) \; mod \; 144) + 1$$
$$c_6 = ((m_1 + m_2) \; mod \; 144) + 1$$
$$c_7 = ((2*(m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7)) \; mod \; 144) + 1$$

One notes that each one of the $c_1, c_2, \ldots, c_7$ bits lies in the [1...144] range just like each one of the $m_1, m_2, \ldots, m_7$ does. The $c_1, c_2, \ldots, c_7$ bits may be taken to comprise a set of *'colour'* bits in order to differentiate them from the $m_1, m_2, \ldots, m_7$ of the SC/SF method. By superimposing the patterns of all the c_1 colour bits for the $D = 100$ words in a block, one obtains yet one more (colour) signature pattern which is 144 bits wide. In a similar way, six more (colour-2 to colour-7) colour signature patterns are produced. The seven colours represent additional information for the textual data contents of each logical block. However, each colour pattern never gets registered as such in the extended signature file structure. If this were the case, one would simply obtain an SC/SF configuration where the entropy/information content maximization norm would have been violated. An $F = 1008$, $m = 14$ and $D = 100$ configuration would no longer satisfy Equation (1). Instead, for each logical row of

	Partitions				Colours						
Block	P_1	P_2	...	P_7	s_1	c_1	s_2	c_2	...	s_7	c_7
1	01...1	11...0	...	00...1	0	110	0	101	...	1	001
2	11...1	01...1	...	10...0	1	010	1	001	...	1	101
...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...
100	00...0	01...1	...	10...0	0	111	1	010	...	0	010

Figure 4 Structure of the signature file in the proposed technique

the partitioned signature file, each of the block's colour patterns is checked against the existing partitions and what gets registered is the identifier of the partition which most closely resembles the colour pattern in question.

Two partitions (an SC/SF one and a *colour* one in this case) are identified to mostly resemble to one another by counting the number of 1-to-1 matches as well as the number of 1-to-0 mismatches in the corresponding bit positions of their patterns. Each partition is 144-bits long in the configuration considered here. For example, say that colour pattern C is found to have 74 *match* instances with partition P1 and 68 *match* instances with partition P2. Then, C is said to resemble to P1 more than P2. It is worth noting that a 74 *matches* degree of resemblance thus defined would be equivalent to a 74 *mismatches* one. The latter is so because 74 mismatches to a given binary pattern imply 74 matches to its inverted (0's replaced by 1's and 1's by 0's) image. The sign bit registered in the s_1, s_2,... s_7 columns of the structure shown in Figure 4 encodes the extra information needed to cater for in this case.

The scheme described in the above gives each colour partition a total of 2*7 = 14 chances to identify one of the existing SC/SF partitions it most closely resembles to. This is done by checking the image of the colour partition in question next to the direct as well as to the inverse images of each SC/SF partition registered to represent the textual block in question.

Considering Figure 4, one reads the information off the specific example as follows: for block number 2 and colour number 1 the colour pattern produced was found to mostly resemble the direct ($s_1 = 1$) image of partition number 2 ($c_1 = 010$) whereas for block number 100 and colour number 7, the colour pattern is registered to mostly resemble the inverted ($s_7 = 0$) image of partition number 2 ($c_7 = 010$), etc. This information is used at query processing time to calculate a B-rank value for each one of the SC/SF block candidacies. The B-rank value is an integer equal to the number of times a colour bit (a total of seven to consider for each word in this case) is found to comply with the binary value stored in the corresponding binary position in the SC/SF partition registered to mostly resemble the colour pattern in question and for the specific word-block pair.

Referring to the example in Figure 4 again, say that while processing a single word query, block number 2 appears to be one of the candidates to contain the word in accordance with the classical SC/SF method. Let the seven colour bits set by the word in question be (each in the [1...144] range): $c_1 = 15$, $c_2 = 3$, $c_3 = 143$, $c_4 = 14$, $c_5 = 3$, $c_6 = 99$ and $c_7 = 76$. This means that for the specific colour pattern, bit positions 15, 3, 143, 14, 3, 99 and 76 are meant to be set to 1. The B-ranking algorithm is now to check the binary values stored in the seven bit positions in the corresponding *dominant* partition (i.e. the one registered to mostly resemble the colour pattern in question). Table 2 shows the values read off the dominant partition this way. The B-rank value for the block candidacy in question is then easily calculated to be the sum of the match instances appearing in Table 2 (i.e. 1 + 0 + 0 + 0 + 1 + 1 + 1 = 4). It is noted that the *sign* value in Table 2 indicates whether the direct (sign = 1) or the inverse (sign = 0) image of the dominant partition should be considered in each case.

Table 2. Calculation of the B-rank value

	Bit position	Dominant partition	Bit value		
			Sign	Read	Match
Colour-1	15	2	1	1	1
Colour-2	3	1	1	0	0
Colour-3	143	6	0	1	0
Colour-4	14	6	1	0	0
Colour-5	3	4	0	0	1
Colour-6	99	3	0	0	1
Colour-7	76	5	1	1	1
Rank value					4

The scheme is equivalent to having each block candidacy toss a coin seven times. The number of '*head*' instances achieved is assigned to be its B-rank value. The important thing to note is that the coin tossed by any NFD (no false drop, one where the word is present in the block) instance is biased to produce '*heads*' more often than the one tossed by any false drop (FD) instance. The latter may be considered to be tossing an '*ideal*' coin: i.e. one which has a 50% probability to produce either '*heads*' or '*tails*' result in accordance with the entropy/information content maximization rule (Equation (4)). The degree of the '*coin bias*' value achieved depends on the performance of the ranking technique used and has to do with the number of chances each colour pattern is given to maximize its degree of resemblance to one of the existing partitions (14 chances is the number for the case considered in this paper). In this respect, the proposed technique allows for improvement by increasing the number of patterns each colour pattern is checked against.

The storage overhead introduced by the $m = 7$, $F' = 144$, $D = 100$ configuration shown in Figure 4 is: $7 \times 4 = 28$ extra bits for each SC/SF row instance, i.e. nearly 3% when compared to the classical method. Considering that the classical signature file method calls for only 10–15% storage overhead and comparing it to the 100–300% figure of the inverted file model[1], one can say that the storage overhead introduced by the proposed technique is a negligible cost to pay provided that reasonable performance improvement is achieved.

The simulation environment

For the purpose of measuring the performance of the proposed technique, a simulation environment based on a Relational Database Management System (RDBMS) was set up. Until now, a DBMS environment has been used only in information retrieval related research as part of the '*retrieval engine*' : the '*inexact match*' retrieval model is mapped on to the '*exact match*' DBMS environment[10]. In the current study, advantage has been taken of the RDBMS's flexibility while setting up a testbed in order to measure the performance of the proposed technique. The simulation environment was implemented on a Data General (UNIX) minicomputer by embedding SQL (ORACLE) command syntax in C.

A '*vocabulary*' equivalent was established, making use of a random number generator. It consisted of 10 000 distinct '*words*', a number which in accordance with Dewey[11] corresponds to a 100 000 word textual database. The overall setup was controlled in the sense that it was known in advance which '*word*' was contained in which '*document*' or, rather, logical block of textual data. The scheme allowed for monitoring the behaviour of a number of performance metrics defined in the next section. Care was taken to ensure that no two words shared the same word signature pattern, i.e. the signature extraction phase introduced no fuzziness/information loss in this respect.

Every single word of the vocabulary was placed in just one block of text (document equivalent). This means that at query processing time it was known in advance that there was one and only one NFD instance in the corresponding block candidacies. This is definitely not the case in a real-life environment. However, with respect to the B-ranking technique considered, such an assumption considers a 'worse-case' equivalent: the more NFD instances present in the ranked output of a query processing session, the more likely it becomes for the top ranked one to result into a 'Hit' (the numbers of hits being one of the performance metrics introduced in the next section).

Each logical block of data consisted of 100 distinct words (100 logical blocks in total). The SC/SF structure was partitioned in accordance to the configuration shown in Figure 3 with design parameter values: $m = 7$, $D = 100$ and $F' = 144$. The m, D and F' parameters (shown in Table 1) were chosen so that they comply with Equation (1). As a result, an average number of 71.72 1s was measured in each block signature partition. This compares well to the value expected ($144/2 = 72$). Each block signature partition was thus nearly half-full with 1s, conveying maximum entropy/information content.

A total of 10 000 single word queries (one for each word in the vocabulary) was processed against the binary representation of the text database equivalent. As a result, 17 844 logical block candidacies were measured. In the controlled environment of the specific simulation this means that 7844 false drop (FD) instances were observed. Dividing this value by the number of queries as well as by the number of logical block patterns considered, one calculates the false drop probability value:

$$\frac{7844}{100 \times 10\,000} = 0.007844 \approx \left[\frac{1}{2}\right]^7 \qquad (8)$$

In this respect, the number of FD instances measured during the simulation compares well with the value expected. The latter is calculated by substituting $m = 7$ into Equation (4). One may use this fact to establish confidence for the specific (controlled) simulation setup: the 'worst case' assumption that has been made (i.e. only one NFD instance per each set of block candidacies) does not affect the validity of the results obtained with regard to the general (real-life) case. Plus, it was the same false drop value appearing in Equation (8) that was measured both for the classical as well as for the partitioned SC/SF configurations. As expected, the partitioned (Figure 3) scheme introduced in this paper is thus proven to be equivalent to the SC/SF setup in terms of the false drop rate it produces.

Performance metrics

In the simulation environment considered, it is known in advance that each set of block candidacies produced by any of the 10 000 queries, comprises only one NFD (no false drop) and zero or more FD (false drop) instances. It has been shown already that this is of no harm to the general case where the SC/SF output involves more than one NFD instance.

It was measured that the $F' = 144$, $m = 7$, $D = 100$ (partitioned) SC/SF configuration produces false drops at a rate of 0.78 when any one of the single word queries is processed against all of the 100 block signature patterns. This means that the one and only NFD instance associated to each query is accompanied by 0.78 FD instances, on average. In this respect, it is possible for the NFD instance not to be accompanied by any FD instances. The latter (called *no-conflict*) case introduces some type of '*noise*' when it comes to measuring the performance of the B-ranking technique. This is due to the fact that the NFD instance '*wins*' regardless of the B-rank value it is assigned: no FD instances exist for the NFD.

In the specific simulation environment, the number of block candidacies in the SC/SF output varies from one single-word query to another. For each query processed, its type is labelled by '*rng*', where n is an integer, r stands for '*real*' and g stands for '*ghost*'. Thus $r1g$ means that one NFD instance competes against one FD, $r2g$ implies one NFD against two FDs, etc. A no-conflict case, as it is defined in the previous paragraph, is labelled by $r0g$. The simulation considers all the *rng* types of NFD-to-FD conflicts measured for the 10 000 single-word queries issued. In theory, n could take very large values but the results revealed the highest value of n to be equal to 6 for the specific environment.

The proposed ranking scheme helps the NFD instances '*float*' by achieving higher rank values when compared to the FD instances which thus '*sink deeper*' by achieving lower rank values, on average. A satisfactory result would be to measure the B-ranking technique as performing better than a '*Random Select*' case. The latter models a classical SC/SF configuration. A metric called '*depth*' is thus established, measuring the order in which an NFD instance is retrieved next to its FD companions in the corresponding query/*rng* instance. For example: during a $r4g$ type of query processing result, say that the NFD instance is assigned a B-rank value that ranks it as number three in its group. This means that the NFD block will be the third one to be retrieved in accordance with the order implied by the ranking scheme. For the specific case, one then speaks of a '*depth*' value which is equal to 3.

To be more precise, it is not the FD or the NFD instance that is retrieved in each case but the corresponding logical block containing textual data. Such a block retrieval is a necessary step to be taken prior to having a subsequent full

text scanning operation categorize the block candidacy to being a successful (NFD) or a false drop (FD) instance. A reasonable assumption to make is that each logical block retrieval comes at the cost of an I/O operation for the system. In this respect, the '*depth*' metric is really a value that needs to be measured in order to calculate the I/O savings involved.

Considering the configuration described in the previous section, one deals with a total of 10 000 NFD and 7844 FD instances. An ideal ranking scheme would force all the FD instances to obtain rank values, each one of which would lie below the lowest rank value assigned to any one of the NFD instances. Having a total of 10 000 *rng* types of query processing results, each one of them would ideally rank its NFD instance higher than any one (if any) of its FD companions. The $depth_i$ value (where $1 \leq i \leq 10\,000$) for each one of the *rng* instances would thus be equal to unity. The '*sum of depths*' (labelled as: *M depth*) metric value would then be:

$$Mdepth = \sum_{i=1}^{10\,000} depth_i = 10\,000 \qquad (9)$$

In the above described (ideal) SC/SF output ranking scheme, one avoids 7844 out of a total of 7844 FD instances, thus achieving a 100% '*FD avoidance*' or *I/O savings* value.

At this point it is worth noting that the absence of an output ranking mechanism in the classical CS/SF method is equivalent to assigning a rank value to each block candidacy at random. The latter may be simulated by means of a random number generator. Even the scheme where block candidacies are picked up at random manages to '*avoid*' some I/O operations along the lines of the '*I/O avoidance*' and '*depth*' metrics defined in the previous paragraph. The performance of the proposed B-ranking technique is checked against such a 'select at random' equivalent to the classical SC/SF configuration which is simulated by having a random number generator assign 'B-rank' values to block candidacies.

Table 3 defines the metrics used in the next section. C_{avg}, C_{min} and C_{max} measure the degree of resemblance between any colour partition and the corresponding SC/SF partition which is found to be the one that mostly resembles it (dominant). They are established during signature file creation time by measuring the number of 1-to-1 *match* instances present in such colour-dominant pairs of partitions. It is noted that the C_{avg}, C_{min} and C_{max} metrics comprise a measure of the '*coin-bias*' achieved by the simulation setup along the lines of the model presented in the section above headed 'The technique'.

The '*Hit ratio*' metric in Table 3 is the percentage of '*Hits*' measured during the simulation. A '*Hit*' is the case where an NFD instance is assigned the highest B-rank value within its *rng* group. This means that the corresponding block candidacy is the first to undergo a full text scan operation at query processing time. Once the word being sought is found to be present in the block, all the other (FD) candidacies are dropped at no further I/O cost.

Table 3. Definition of quantities measured/calculated

C_{avg}	Average number of 1-to-1 matches between any colour partition and the corresponding dominant SC/SF partition
C_{min}	Minimum number of 1-to-1 matches between any colour partition and the corresponding dominant SC/SF partition
C_{max}	Maximum number of 1-to-1 matches between any colour partition and the corresponding dominant SC/SF partition
$R_{avg}(ALL)$	Average B-rank value (all word/block candidacies)
$R_{avg}(NFD)$	Average B-rank value (NFD instances only)
$R_{avg}(FD)$	Average B-rank value (FD instances only)
Mdepth	Number of I/O operations to retrieve all the NFD instances
I/O savings	Calculated as: $\frac{NumberOfFDs - (Mdepth - NumberOfQueries)}{NumberOfFDs}$
Hits	Number of instances where the NFD word/block candidacy achieves the highest rank value within its *rng* group
Hit ratio	Percentage of *Hit* instances over the total number of single word queries considered

At this stage, one can easily realize why an *r*0*g* type of query processing output measures a misleading number of '*Hit*' instances: no FD instances exist for the NFD to compete against, so the latter will '*win*' no matter what B-rank value is assigned to it. In the course of the simulation, a total of 4500 *r*0*g* instances were measured. This is a little less than half the number of single word queries processed (10 000). It is very important that all this *r*0*g* introduced '*noise*' is filtered out while measuring the '*Hit ratio*' value for the proposed method. Otherwise, the performance of the proposed technique would be measured to be of a much higher (however, misleading) value.

Results and discussion

As already mentioned, the B-ranking method involves a classical SC/SF part which is '*deterministic*' in its output. The membership of each *rng* type of SC/SF output is independent of the B-ranking technique used. In this respect, the values appearing in Table 4 only relate to the SC/SF and not to the (satellite) B-ranking configuration of the simulation environment. Table 4 shows that *rng* types up to $n = 6$ were observed in the specific simulation. As a check, the sum of all the *rng* ($n = 0 \ldots 6$) instances in the table is equal to the (expected) value 10 000: the total number of single-word queries generated and tested.

Tables 5 and 6 contain simulation results that directly relate to the performance improvement introduced by the proposed ranking technique when checked against the '*Random Select*' (classical SC/SF equivalent) one. One notes that the lower part of Table 5 involves only the *rng* instances where $n > 0$, i.e. the 4500 *r*0*g* instances have been filtered out. For the SC/SF setup, the 68.4% *Hit ratio* value appearing in the upper part of Table 5 is reduced to 42.5% in the lower part of the same table. The former is higher due to the presence of the *r*0*g* noise.

Table 4. Number of instances measured for each type of an *rng* output

*r*0*g*	*r*1*g*	*r*2*g*	*r*3*g*	*r*4*g*	*r*5*g*	*r*6*g*	*r*7*g*	$\sum_{n=1}^{7} rng$
4500	3709	1334	371	78	6	2	0	10 000

Table 5. Performance of the B-ranking technique next to that of the classical SC/SF (with *r0g* instances included or filtered out)

Technique	*r0g* included	FDs+ NFDs	*Mdepth*	*I/O savings*	*Hits*	*Hit ratio*
SC/SF	Yes	17 844	14 008	48.9%	6838	68.4%
B-ranking	Yes	17 844	13 087	60.6%	7518	75.2%
SC/SF	No	13 344	9508	48.9%	2338	42.5%
B-ranking	No	13 344	8587	60.6%	3018	54.9%

Table 6. *Hit ratio* values for the B-ranking technique next to those of the classical SC/SF configuration

Type	Instances	Technique	*Hits*	*Hit ratio*
r1g	3709	SC/SF	1795	48.4%
		B-ranking	2271	61.2%
r2g	1334	SC/SF	444	33.3%
		B-ranking	574	43.0%
r3g	371	SC/SF	77	20.8%
		B-ranking	145	39.1%
r4g	78	SC/SF	22	28.2%
		B-ranking	25	32.0%
r5g	6	SC/SF	0	0%
		B-ranking	3	50.0%
r6g	2	SC/SF	0	0%
		B-ranking	0	0%

For the *r1g* instances in the SC/SF configuration (Table 6), the *Hit ratio* value measured (48.4%) is close to the expected 50.0% value. This shows that the simulation environment behaved reasonably well in assigning random B-rank values to the SC/SF output thus managing to simulate the latter successfully. One more observation to be made is that the *'I/O savings'* value is independent of the presence or absence of the *r0g* type queries in Table 5. By comparing the proposed B-ranking configuration with the classical SC/SF, one notes an improved *Hit ratio* (54.9% vs. 42.5%) as well as considerable *I/O savings* (60.6% vs. 48.9%).

The proposed B-ranking technique calls for just a 3% additional storage overhead when compared to the classical SC/SF method. However, this comes at the cost of an increase in the CPU overhead involved. Fortunately, it is noted that most of this CPU overhead cost is paid only once, during creation time of the combined (SC/SF and B-ranking) binary pattern as well as only during any subsequent document insert/append operation in the text database. The technique introduces minimal CPU overhead at query processing time. Such an increased *'one time only'* CPU overhead cost appears to be acceptable when dealing with textual databases updated mainly in the *'append new text'* as opposed to the *'modify existing document'* mode: office automation environments and library/bibliographic databases are typical examples. One last thing to note is that the proposed technique achieves the stated performance improvement for the SC/SF method with the measured *'coin bias'* and average B-rank values appearing in Table 7. It is clear that there is room to further increase the performance of the B-ranking technique in the future by making C_{avg} get closer to the $D = 100$ value.

Table 7. Coin bias and B-rank values measured during the simulation

C_{avg}	C_{min}	C_{max}	$R_{avg}(ALL)$	$R_{avg}(NFD)$	$R_{avg}(FD)$
59.4	57.0	62.0	3.95	4.19	3.65

Epilogue

The technique presented in this paper provides a means for ranking the output of the superimposed variation of the classical signature file method. It establishes a new view to recording additional information into an environment already optimized to provide maximum entropy/information content. B-ranking manages to increase the information content of a given binary representation by introducing an extension to it which reuses part of the existing pattern. Such a *'satellite'* representation calls for negligible additional storage overhead at the cost of an increased CPU overhead incurred only during creation time for the combined representation. An additional cost incurred is that the new technique encodes information which is of a non-deterministic nature, i.e. it can only be used for assigning a B-rank value to each block candidacy in the classical SC/SF output.

For the purpose of measuring the performance of the proposed technique, a simulation environment based on a relational database system (RDBMS) was introduced. The flexibility inherent to the relational DBMS allows for the creation of a controlled simulation environment which was used to break down the SC/SF output into a number of *rng* instances by considering the number of FDs that accompany the one NFD instance in each case.

It is interesting to note that the technique provides a means to rank the candidacies in the output of the classical SC/SF by considering only the binary representation part of the method. Croft and Savino[7] introduce ranking by considering the textual part of the environment, their ranking mechanism being tailored to function in parallel to the SC/SF method. In principle, one could calculate a combined rank value for each SC/SF block candidacy by considering both the text present in each logical block as well as the corresponding binary representation. The B-ranking configuration could thus co-exist with one which assigns rank values by considering the textual part of the text database. It would then be possible to assign a combined rank value to each block candidacy; one that would reflect the credibility of both the textual and the binary representations to successfully represent the information that is meant to be communicated in each logical block of text.

A number of partioned variations[2,3] of the SC/SF method exist and increase its retrieval efficiency. The partitioned scheme introduced in this study does not interfere with any of those variations. One may thus introduce B-ranking, by building on top of any classical (partitioned) signature file, in a modular fashion.

Acknowledgements

The authors wish to thank Rob Moreton, Mike Jackson and Graham Bosworth for the useful comments they have made on earlier versions of this paper.

References

1 Faloutsos, C 'Access methods for text' *ACM Comput. Surveys* Vol 17 No 1 (1985) pp 49–74
2 Faloutsos, C 'Signature based text retrieval methods—a survey' *IEEE Data Eng. Bulletin* Vol 13 No 1 (1990) pp 25–32
3 Lee, D L and Leng, C W 'Partitioned signature files—design issues and performance evaluation' *ACM Trans. on Office Inf. Systems* Vol 7 No 2 (1989) pp 158–180
4 Christodoulakis, S and Faloutsos, C 'Design considerations for a message file server' *IEEE Trans. on Soft. Eng.* Vol 10 No 2 (1984) pp 201–210
5 Reingold, E and Hansen, W *Data structures in Pascal* Little, Brown, (1986) pp 410–413.
6 Robertson, S E 'The probability ranking principle in information retrieval' *J. of Documentation* Vol 33 No 4 (1977) pp 294–304
7 Croft, W B and Savino, P 'Implementing ranking strategies using text signatures' *ACM Trans. on Office Inf. Systems* Vol 6 No 1 (1988) pp 46–62
8 Wong, W Y P and Lee, D L 'Signature file methods for implementing a ranking strategy' *Inf. Proc. and Management* Vol 26 No 5 (October 1990) pp 641–653
9 Salton, G *Automatic text processing: the transformation, analysis and retrieval of information by computer* Addison-Wesley (1989) pp 277–278
10 Belkin, N and Croft, W B 'Retrieval techniques' *Annual Review of Information Science and Technology (ARIST)*, Williams, M E (ed), Elsevier, No 22 (1987) pp 110–145
11 Dewey, C *Relative frequency of English speech sounds* Harvard University Press (1950)