Ranking the Validity of Block Candidacies in Signature Files

D. DERVOS*

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

Y. MANOLOPOULOS†

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

and

P. LINARDIS

Department of Informatics, Aristotle University, Thessaloniki 54006, Greece

ABSTRACT

A variation of the superimposed coding version of the signature file method is introduced. The processing of a user query is similar to that of the conventional method with one difference: the output is provided in a ordered (ranked) way. The latter is based on a credibility value reflecting the probability of each candidate block to pass the validation test rather than being a false drop. The principles, an analytical model, and simulation results of the new technique are presented. The model's usefulness is the reduced I/O activity for cases where the user is only sampling the text base and is satisfied by retrieving a limited number of the many documents that qualify.

1. INTRODUCTION

The *signature file* (SF) is an access method suitable for processing large text bases [4]. The method has been studied extensively and a number of variations have been proposed for increasing its efficiency [5]. Compared to alternative information retrieval (IR) methods, such as full text scanning and inverted files, the SF method requires a very low storage overhead and

*One leave from the Department of Informatics, TEI, Thessaloniki 54110, Greece.
†On leave from the Department of Informatics, Aristotle University, Thessaloniki 54006, Greece.

655 Avenue of the Americas, New York, NY 10010 0020-0255/94/$7.00

is second only to the latter in terms of its retrieval efficiency [1, 4].

The *superimposed coding* (SC) variation of the SF method is very attractive because it achieves a high degree of compression. According to this variation, raw text is divided into logical blocks, each one containing a constant number of distinct noncommon words [1, 5]. A *word's signature* is comprised of a number of 1s set in a structure storing binary values. The signatures of all words residing in a logical block are superimposed (ORed) to yield the *block's signature*. The latter becomes the binary representation of the corresponding section of the text base.

A drawback of the SC/SF method is the uncertainty introduced by its intermediate binary representation at query processing time. For example, two distinct words may be mapped onto the same binary word signature pattern and they both appear to be present in the corresponding real text. In addition, a word may appear to be present in the text, whereas in reality it is not, due to another reason: its signature pattern may correspond to 1s set by two or more other words. These cases are called *false drop* (FD) instances and they are filtered out by scanning the text content of each block candidacy with respect to the given search condition. This is an overhead that has a negative influence on the method's efficiency.

In the example shown in Figure 1, a block's signature consists of 12 binary cells and accommodates two word signature patterns. Each pattern is comprised of any four 1s in the [1...12] range. With no reference to a specific signature extraction algorithm, let the word "free" ("text") set to 1 positions 3, 7, 8, and 11 (5, 7, 9, and 12). The signature of the logical block is constructed by superimposing the two word signature patterns. Thus finally, seven positions are set to 1 in the [1...12] range for the specific example. Suppose, that the word "base" has a signature consisting of 1s in position numbers 3, 5, 7, and 8. The 1s set by the word "base" coincide with the 1s of the block's signature, which causes the false impression that the word is present in the block.

The previous example is a case of an FD instance. FDs introduce information loss in the SC/SF method. Thus, the reduction of the FD rate for a given storage overhead becomes a measure of performance for the

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

Fig. 1. Block signature generation in the SC/SF method.

method. The size of the binary structure, F, the number of positions set to 1 in each word's signature, m, and the number of distinct noncommon words present in each logical block of text, D, define a set of design parameters. It has been proved that once the equation

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

holds, the *false drop probability*, *FDP*, is kept to a minimum for a given storage overhead. Such an optimal SC/SF configuration has been proved to contain 0s and 1s with equal probability (50–50%) [1]. Furthermore, it has also been shown that, in the case of the optimal configuration just described, the *FDP* value depends on m only:

$$FDP \approx (1/2)^m. \tag{2}$$

Equation (2) indicates that in the classical SC/SF method each block candidacy has a given *FDP* value.

The present study is an attempt to rank the candidate documents according to their probability of success. The rest of this paper is organized as follows. Section 2 establishes a new approach to the issue of ranking and the principles of the new technique are presented. In Section 3, the file structures of two variations of the new technique are considered. A probabilistic model for the analysis of the new SC/SF variations is introduced in Section 4. Section 5 presents the simulation setup and checks the results obtained against those expected by the probabilistic calculations. Finally, the Epilogue contains concluding remarks plus suggestions for further research.

2. THE RANKING PRINCIPLE

The concept of *ranking* is a well honored issue and has been justified exclusively for the nonsignature IR environment [9]. Its aim is to identify and measure instances of semantic similarities between queries and document candidacies [10]. In this respect, ranking implies some probabilistic processing of text and its semantics in order to produce structures necessary to implement similarity checking at query processing time. Each candidacy in the output of the IR environment is accompanied by a rank value reflecting its probability to be relevant to the user's query. In addition, practice has shown that ranked output is welcomed by the

average IR system user [7]. Croft and Savino have introduced an enhanced variation of the SC/SF method, providing ranked output by coupling the SC/SF configuration to a probabilistic environment similar to the one described above [2]. Their effort focused on making the SC/SF approach effective in addition to being efficient. The Croft–Savino configuration provides ranked output with the aim that, through ranking, the semantic information inherent in both the user query as well as in the block candidacies is used to (a) discriminate the *no false drop* (NFD) instances from the FD ones as well as (b) identify similarities that the SC/SF alone fails to recognize. More precisely, according to their scheme, instances of strong semantic similarity can tolerate some partial matching in the corresponding binary patterns.

However, the concept of ranking should not be restricted to only imply the probability of relevance of a given query to the semantics inherent in the information content of the corresponding text. Ranking can very well indicate the *credibility* possessed by a given block signature to successfully stand in place of the original text. Thus, in the present work, the conventional SC/SF method is modified to support ranking related to the credibility of its binary representation rather than the semantics of the involved text. Equation (1) may be rewritten as

$$\lceil F/m \rceil \times \ln 2 \approx 1 \times D. \tag{3}$$

Equations (1)–(3) indicate that a performance equal to that of the SC/SF method may be achieved in a slightly modified signature file structure. Instead of directing all of the m bits onto a single block of F binary cells, each of the m bits could be directed onto a separate *partition* of size $\lceil F/m \rceil$. Simulation results have confirmed this observation [3]. Equation (3) guarantees that each of the m partitions will be half full with 1s, on the average.

Such a *partitioned* SF configuration hardly introduces any significant increase in the storage, while its performance remains equal to that of the classical SC/SF method. (At this point, it is interesting to note that partitioned SF schemes have been examined in the context of parallel architectures [8, 11, 12].) Figure 2 is an example of the partitioned equivalent to the classical SC/SF method where $F = 28$, $m = 4$, and $D = 5$.

P_1	0	1	0	1	1	0	0
P_2	1	1	1	1	0	0	0
P_3	0	0	0	1	1	1	1
P_4	1	0	1	0	1	1	0

Fig. 2. An example of the partitioned SC/SF method ($F = 28$, $m = 4$, and $D = 5$).

P_1	0	1	0	1	1	0	0
P_2	1	1	1	1	0	0	0
P_3	0	0	0	1	1	1	1
P_4	1	0	1	0	1	1	0
P_5	0	0	1	1	1	1	0
P_6	0	0	1	0	0	1	1

Fig. 3. The partitioned SC/SF configuration when $F = 42$, $m = 6$, and $D = 5$.

The four 7-bit partitions ($P_1 \ldots P_4$) comprise a binary representation for a logical block of text containing five distinct words. This alternative includes positional information regarding the m bits set to 1 by each word. This is so because character triplets of the same order (first, second, third, fourth) direct their 1s to the same respective partition.

Figure 3 shows an attempt to improve the performance of either the conventional or the partitioned version of the SC/SF method. Two more partitions, P_5 and P_6, are constructed in a way similar to that of $P_1 \ldots P_4$. With respect to the example of Figure 2, the following relations show how extra information bits may be produced for each word:

$$m_5 = ((m_1 + m_2 + m_3 + m_4) \bmod 7) + 1,$$

$$m_6 = ((m_1 + m_2 + m_3) \bmod 7) + 1.$$

(m_i's correspond to bits set to 1 by a word in the relevant partition P_i, $1 \leqslant i \leqslant 6$). The configuration shown in Figure 3 is equivalent to the conventional SC/SF, where $F = 42$, $m = 6$, and $D = 5$ in terms of storage overhead and *FDP* rate/performance.

How could one improve the performance with less storage cost than the classical SC/SF storage overhead? Each one of the two extra binary patterns in Figure 4 (P_5 and P_6) is $\lceil F/m \rceil$ bits long just like any one of the $P_1 \ldots P_4$ partitions. In this respect, a certain degree of similarity can be identified between each one of the P_5 and P_6 partitions next to each of

P_1	0	1	0	1	1	0	0
P_2	1	1	1	1	0	0	0
P_3	0	0	0	1	1	1	1
P_4	1	0	1	0	1	1	0
P_5	0	1	1				
P_6	1	0	0				

Fig. 4. A compressed representation for the structure of Figure 3.

the $P_1 \ldots P_4$ partitions. For the example shown in Figure 3, partition P_4 mostly resembles P_5 because three of their 1s are found in the same positions within the $1 \ldots \lceil F/m \rceil$ range. Also, upon initial observation, partitions P_3 and P_4 are equally candidate to mostly resemble P_6. However, once the inverse image of each one of the $P_1 \ldots P_4$ partitions is considered (0s replaced by 1s and vice versa), then all three of P_6's 1s are found to overlap with 1s present in the inverse image of P_1. Therefore, the inverse image of P_1 and not P_4 or P_3 is the partition that mostly resembles P_6.

Figure 4 shows a configuration that stores data relating to P_5 and P_6 in a compressed way as has been described in the previous paragraph. The 3-bit pattern appearing next to P_5 registers that it mostly resembles the direct image ("0") of partition P_4 ("11"), whereas the "100" next to P_6 records its resemblance to the inverse image ("1") of partition P_1 ("00"). This way, the configuration shown in Figure 4 uses just 6 bits to register information requiring 14 bits in Figure 3. Apparently, when the compressed representation of P_5 and P_6 shown in Figure 4 is compared to that of $P_1 \ldots P_6$ shown in Figure 3, the former is seen to be fuzzier than the latter. Information carrying patterns like P_5 and P_6 in Figure 4 (produced by oring the m_5s and m_6s for each word) are referred to as *color* partitions so that they are differentiated from the $P_1 \ldots P_4$ patterns.

The ranked SC/SF output presented in this study has to do with the binary representation of the text base. As shown in Figure 4, information inherent in the P_5 and P_6 partitions is recorded in a compressed way. Thus the resulting representation carries less information than the fully recorded $P_1 \ldots P_4$ partitions. The scheme results in considerable storage savings at the cost of increased fuzziness. The query processing stage is split into two distinct phases: an initial classical phase, where the information inherent in $P_1 \ldots P_4$ produces a number of SC/SF block candidacies, and a second phase, where the P_5 and P_6 representation is decoded and a rank value is calculated for each block candidacy.

It is desirable for the proposed ranking technique to promote the NFD instances into higher places in the list of candidate documents by assigning a rank value to each of them. Such a ranked output is of practical use in cases like the query "I want THAT document," where once a specific document has been retrieved, all the rest of the SC/SF candidacies are dropped. Another example could be an IR environment where the SF method is used as a starter triggering with its output an effective non-SF based backend engine. Such a hybrid system would combine the benefits of both worlds (SC/SF efficiency and non-SC/SF effectiveness) and provide ranked output: selecting the 10 (say) top ranked block candidacies performs better than choosing any 10 at random.

3. TWO PARTITIONED SC/SF VARIATIONS

Two variations of the SC/SF method, each based on the notion of ranking as explained previously, are presented in this section. Section 4 establishes a probabilistic model that allows for the calculation of their performance related metrics. Section 5 considers the values calculated analytically next to those measured during simulation. The results are checked against those for the conventional SC/SF method, which is taken to be one that ranks the candidacies in its output at random.

3.1. VARIATION 1

For the sake of presentation, let there be a conventional SC/SF method where $F = 1008$, $m = 7$, and $D = 100$. In this respect, (3) suggests that the partitioned equivalent configuration consists of seven 144-bit partitions. In accordance with (2), the partitioned SC/SF version is expected to have a *FDP* value that is equal to

$$FDP \approx (1/2)^7. \tag{4}$$

In addition to the seven classical SC/SF positions $(m_1 \ldots m_7)$, each word of text was taken to produce seven more color instances:

$$m_8 = ((m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7) \bmod 144) + 1,$$

$$m_9 = ((m_1 + m_2 + m_3 + m_4 + m_5 + m_6) \bmod 144) + 1,$$

$$m_{10} = ((m_1 + m_2 + m_3 + m_4 + m_5) \bmod 144) + 1,$$

$$m_{11} = ((m_1 + m_2 + m_3 + m_4) \bmod 144) + 1,$$

$$m_{12} = ((m_1 + m_2 + m_3) \bmod 144) + 1,$$

$$m_{13} = ((m_1 + m_2) \bmod 144) + 1,$$

$$m_{14} = ((2 * (m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7)) \bmod 144) + 1.$$

For each color instance $(m_8 \ldots m_{14})$, the patterns of all the words in the $D = 100$ word logical block are superimposed to yield a total of seven 144-bit partitions. Seven color partitions $(c_1 \ldots c_7)$ are constructed for each

logical block of text in a way similar to that of constructing P_5 and P_6 appearing in Figure 3. Instead of being registered as such, each color partition is registered in a compressed form similar to that of recording P_5 and P_6 in Figure 4. Table 1 shows the modified SF structure.

It is obvious that the performance of the proposed technique depends on the number of chances each color is given to identify the partition that is most similar to it. Such an assumption is reasonable because the positions set to 1 in any one of the $P_1 \ldots P_7$ partitions do not relate to the distribution of 1s in the rest. Considering the configuration of Table 1, each one of the $c_1 \ldots c_7$ colors is given $2 \times 7 = 14$ chances as it is checked against the direct and the inverted image of each one of the $P_1 \ldots P_7$ patterns. The storage overhead is $4 \times 7 = 28$ bits, implying a 2.8% increase over that of the conventional SC/SF method.

3.2. *VARIATION 2*

The performance of the ranking technique may be improved by modifying the set of expressions appearing in Section 3.1 as follows:

$$m_8 = ((m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7) \bmod 288) + 1,$$

$$m_9 = ((m_1 + m_2 + m_3 + m_4 + m_5 + m_6) \bmod 288) + 1,$$

$$m_{10} = ((m_1 + m_2 + m_3 + m_4 + m_5) \bmod 288) + 1,$$

$$m_{11} = ((m_1 + m_2 + m_3 + m_4) \bmod 288) + 1,$$

$$m_{12} = ((m_1 + m_2 + m_3) \bmod 288) + 1,$$

$$m_{13} = ((m_1 + m_2) \bmod 288) + 1,$$

$$m_{14} = ((2 * (m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7)) \bmod 288) + 1.$$

TABLE 1
Variation 1: The Modified SC/SF Structure

Block	Partitions				Colors						
	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	10⋯0	01⋯1	⋯	10⋯0	0	011	1	010	⋯	1	010

TABLE 2
Variation 2: The Modified SC/SF Structure

Block	Partitions				Colors								
	P_1	P_2	⋯	P_7	s_{1h}	c_{1h}	s_{1l}	c_{1l}	⋯	s_{7h}	c_{7h}	s_{7l}	c_{7l}
1	01⋯1	11⋯0	⋯	00⋯1	0	110	0	101	⋯	1	001	0	010
2	11⋯1	01⋯1	⋯	10⋯0	1	010	1	001	⋯	1	101	1	101
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
100	10⋯0	01⋯0	⋯	10⋯0	0	101	1	010	⋯	0	010	0	001

Now each color corresponds to a 288-bit pattern and consists of two parts: a "low" and a "high" section (positions 1...144 and 145...288, respectively). Each color section has a size equal to that of any one of the $P_1 \ldots P_7$ partitions and is given $2 \times 7 = 14$ chances to find the one that is most similar to it as explained in Section 3.1. Thus, each 288-bit color pattern is given a total of $14 \times 2 = 28$ chances to find the pair of 144-bit partitions, the 288-bit pattern of which is most similar to the color pattern in question. When considering every possible combination of two out of the seven basic SF partitions, each of the latter is taken to be either the high or the low end of the candidate 288-bit pair plus each basic partition is considered either in its direct or its inverted image. Table 2 shows the modified SF structure in this case. This time the involved storage overhead is equal to $8 \times 7 = 56$ bits or 5.6% more than the classical SC/SF method.

EXAMPLE. To understand the algorithm used to assign a rank value to each SC/SF block candidacy, the seven basic SC/SF partitions are labeled via a three-bit code as 1(000) to 7(110). In the case of color c_7 for block number 1, its 288-bit pattern has been recorded to mostly resemble the inverted ($s_{7h} = 1$) image of P_2(001) in its high end and the direct ($s_{7l} = 0$) image of P_3(010) in its low end.

While processing a specific single word query, let block number 1 be one of the candidacies appearing in the SC/SF output. Also, let the seven color bits set by the word in question be $c_1 = 15$, $c_2 = 253$, $c_3 = 143$, $c_4 = 7$, $c_5 = 299$, $c_6 = 99$, and $c_7 = 76$. The ranking algorithm checks each one of these instances of 1s against the corresponding bit value in the dominant partition for the corresponding (low/high) end of the color/block combination in question. Table 3 shows how the rank value for the specific block candidacy is calculated to be the sum of the "match" = 1 instances, i.e., $1+0+0+0+1+1+1=4$.

TABLE 3
Sample Calculation of the Rank Value in Variation 2

Color	Actual Bit Position	Dominant Partition Number	Sign	Bit Position Considered	Value Read	Match
c_1	15	101	0	15	0	1
c_2	253	000	1	$253-144=109$	0	0
c_3	143	110	0	143	1	0
c_4	7	110	1	7	0	0
c_5	299	100	0	$299-144=155$	0	1
c_6	99	011	1	99	1	1
c_7	76	010	0	76	0	1
Rank value						$\frac{1}{4}$

4. ANALYSIS

In the analysis which follows, only single word queries are considered [1, 4, 5]. In addition, it is assumed that one and only one document qualifies for each user query. Thus, it is possible to categorize SC/SF output to belong to a certain type according to the numbers of the NFD and FD instances involved. In other words, knowing that each word appears in one and only document, the SC/SF output consists of one NFD instance and a variable number (including zero) of FD instances. In this respect, each SC/SF output is labelled to be of a specific "RNG" type. This symbolism indicates the number (N is an integer) of "Gs," ghost words (FDs), accompanying the one and only "R," real word (NFD). For example, R4Gs symbolizes the output consisting of five block candidacies, four of which are FDs; the remaining candidacy is the NFD instance.

The expected number of *FD* instances when Q single word queries are processed is

$$FD = FDP \times Q \times K, \tag{5}$$

where K denotes the number of signature blocks and *FDP* is the value given by (2). In the environment considered, it is known that the number of NFD instances is equal to the number of single word queries issued. The emergence of RNG instances may be viewed as equivalent to the outcome of placing *FD* balls onto NFD cells with replacement [6]. For a

specific RNG type, the expected number of instances relates directly to the probability for any specific cell to accommodate exactly N balls according to the equivalent problem

$$P_{\text{RNG}} = \binom{FD}{N} \times \left(\frac{1}{Q}\right)^{N} \times \left(1 - \frac{1}{Q}\right)^{FD-N}. \tag{6}$$

In the case of a partitioned SC/SF environment with N_c colors, each member of any RNG output is given N_c chances to toss a coin. Assuming that a "heads" outcome of the coin toss is desired, each heads instance increments the rank value of the corresponding block candidacy by 1. The ranking algorithm will successfully manage to distinguish the NFD instances from the FD instances as long as each of the latter tosses an ideal coin, whereas each of the former tosses a biased coin, producing "tails" less often than heads. It is evident that the involved coin bias will directly relate to the ranking performance.

Considering the case of an ideal coin, let P_{i_0} symbolize the probability that after N_c tosses there will be no heads instances, P_{i_1} the probability for just one heads outcome, etc. This way, the $P_{i_0}, P_{i_1}, \ldots P_{i_{N_c}}$ values can be calculated as follows:

$$P_{i0} = \binom{N_c}{0} \times \frac{1}{2^{N_c}},$$

$$P_{i1} = \binom{N_c}{1} \times \frac{1}{2^{N_c}},$$

$$\vdots$$

$$P_{iN_c} = \binom{N_c}{N_c} \times \frac{1}{2^{N_c}}.$$

For example, if $N_c = 7$, then the probability for each rank value to occur may be calculated and appears in the second column of Table 4. In a similar way, the third column of Table 4 shows estimates of the probability for each rank value to occur in the case of a biased coin producing heads more often (60%) than tails (40%). The previous set of equations now

TABLE 4
Probability of Occurrence for Each Rank Value to be Reached After Seven Ideal and Biased (60–40%) Coin Tosses

Rank Value	Ideal Coin	Biased Coin
0	0.0078125	0.163840
1	0.0546875	0.017203
2	0.1640625	0.774144
3	0.2734375	0.193536
4	0.2734375	0.290304
5	0.1640625	0.261274
6	0.0546875	0.130637
7	0.0078125	0.027994

becomes

$$P_{b0} = \binom{N_c}{0} \times (0.4)^{N_c},$$

$$P_{b1} = \binom{N_c}{1} \times (0.4)^{N_c - 1} \times (0.6)^1,$$

$$\vdots$$

$$P_{bN_c} = \binom{N_c}{N_c} \times (0.6)^{N_c}.$$

Table 5 defines the metrics used in order to measure the performance of the proposed ranking technique against that of the classical SC/SF method. It now becomes possible to calculate the expected number of "Hits" for each instance of an RNG type. A Hit is, by definition, a case where the NFD instance achieves the highest rank value in its RNG group of block candidacies. In this respect, the classical SC/SF method may be viewed as a *random select* operation: the Hit values are calculated under the assumption that each of the NFD and FD candidacies tosses an ideal coin for N_c times, the number of heads achieved being assigned as the rank value in each case. On the other hand, in a modified SC/SF structure, which involves a 60–40% biased coin, each FD instance continues to toss an ideal coin for N_c times, whereas each NFD instance tosses a 60–40% biased coin for N_c times.

Let $P_{h\langle \mathrm{RNG} \rangle}$ symbolize the probability that an RNG instance results in a Hit. Also, let P_{g_i} (where i ranges from 0 to N_c) refer to the probability for any FD candidacy to be assigned a rank value equal to i. Similarly, let P_{r_i} (i from 0 to N_c) symbolize the probability for any NFD candidacy to be assigned the rank value i. Evidently, in a random select case where $N_c = 7$, both P_{r_i} and P_{g_i} obtain their values from the second column of Table 4. Similarly, in a $N_c = 7$, 60–40% biased coin equivalent ranking technique case, P_{g_i} takes values from the second column, while P_{r_i} obtains its values from the third column of Table 4. The expected values of $P_{h\langle \mathrm{RNG} \rangle}$ ($N = 1, 2, \ldots$) can now be calculated as follows:

$$P_{h\langle \mathrm{R1G} \rangle} = \sum_{i,j:\,0 \ldots N_c}^{i>j} P_{r_i} P_{g_j} + 0.5 \sum_{i,j:\,0 \ldots N_c}^{i=j} P_{r_i} P_{g_j},$$

$$P_{h\langle \mathrm{R2G} \rangle} = \sum_{i,j,k:\,0 \ldots N_c}^{i>j,k} P_{r_i} P_{g_j} P_{g_k} + 0.5 \sum_{i,j,k:\,0 \ldots N_c}^{i=j,i>k} P_{r_i} P_{g_j} P_{g_k}$$

$$+ 0.33 \sum_{i,j,k:\,0 \ldots N_c}^{i=j=k} P_{r_i} P_{g_j} P_{g_k}.$$

Values produced from the preceding expressions will be presented in the next section together with the corresponding results measured during simulation.

5. THE SIMULATION

A relational DBMS based environment was used to simulate the performance of the proposed ranking technique. The RDBMS seems to be an ideal tool in this case, as it monitors all the intermediate stages via its ad hoc query user interface. The system was implemented on a Data General (UNIX) minicomputer with the C language acting as the host to the embedded SQL (ORACLE) code. A 10,000 distinct word dictionary was created by means of a random number generator. Every 100 consecutive words were seen to comprise a logical block of text with no two groups containing the same word and, therefore, 100 logical blocks were created. Each word produced an $m = 7$ bit signature, the patterns of each $D = 100$

words being superimposed onto an $F = 1008$-bit block signature pattern equivalent. Further details on the specific simulation appear in [3].

Given the specific environment, 10,000 single word queries (one per each word in the dictionary) were processed. Thus, each query was known in advance to involve a single NFD instance regardless of the block candidacies appearing in its SC/SF output. The SF composed of 100 block signatures and a total of 7844 FD instances were measured. Thus, the *FDP* value equals 0.007844, which compares well with the value expected from (4). In the specific environment, only six types of output were observed, i.e., $0 \leqslant N \leqslant 6$, although in theory one expects N to obtain any integer value. At this stage, it is worth noting that all the occurrences of the R0G type of output should be filtered out from the results: an R0G instance (one NFD vs. no FD) is a Hit regardless of the ranking algorithm applied. The number of R0G instances measured (4500) in the specific simulation setup indicates how important it is to exclude them so that the measured performance is of some realistic value.

As stated in Section 3, the coin bias value is a crucial parameter providing better ranking. The C_{avg} metric defined in Table 5 measures the average number of color instances (i.e., 1s in the color patterns) corresponding to 1s in the partition found to be most similar to the color pattern in question. In this respect, for $D = 100$, an ideal case would be one with $C_{avg} = C_{min} = C_{max} = 100$. Values of metrics relating to the achieved coin bias value appear in Table 6. Note that there is room to further increase the performance of the proposed ranking technique. Table 7

TABLE 5

Definition of Quantities Measured/Calculated

$C_{avg}, C_{min}, C_{max}$	Average, minimum, and maximum number of color instances covered by 1s in the dominating partitions, respectively
$\bar{R}_{ALL}, \bar{R}_{NFD}, \bar{R}_{FD}$	Average block rank value measured by considering all the block candidacies, the NFD instances, and the FD instances only, respectively
*M*depth	Number of I/O operations needed to identify all the NFD instances (assuming one operation per block access)
I/O Savings	The FD avoidance ratio calculated as $\frac{\#\,FDs - (M\text{depth} - \#\text{Queries})}{\#\,FDs}$
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 NFD instances measured

TABLE 6
Values of Metrics Relating to the Coin Bias Achieved

Technique	C_{avg}	C_{min}	C_{max}	$\bar{R}_{ALL}$	$\bar{R}_{NFD}$	$\bar{R}_{FD}$
Variation 1	59.40	57.0	62.0	3.95	4.19	3.65
Variation 2	62.73	60.0	65.0	4.08	4.42	3.65

TABLE 7
Performance of Each Variation Next to the Classical Method

Technique	FDs + NFDs	Mdepth	I/O Savings %	Hits	Hit Ratio %
Classical SC/SF	13,344	9508	48.9	2338	42.5
Variation 1	13,344	8587	60.6	3018	54.9
Variation 2	13,344	8215	65.4	3312	60.2

presents the performance of each variation of the proposed technique next to the classical SC/SF case.

For the environment considered, the probability for each RNG instance to occur can be calculated via (6). For example, an R1G instance occurs with probability

$$P_{R1G} = \binom{7800}{1} \times \left(\frac{1}{10{,}000}\right)^{1} \times \left(1 - \frac{1}{10{,}000}\right)^{7799} \approx 0.3576. \qquad (7)$$

Thus for a total of 10,000 single word queries, $0.3576 \times 10{,}000 = 3576$ R1G instances are expected to emerge. Table 8 compares the values calculated with the formula appearing in (6) to the values measured during the simulation. It is obvious that the theoretical calculations compare well to the simulation results.

Table 9 breaks down the SC/SF output into the six RNG types observed during the simulation. For each RNG type, the number of Hits is measured (i.e., instances where the NFD achieves the highest rank value within its RNG group) and the corresponding "Hit Ratio" values are calculated. The results exhibit some consistency for $N < 5$: the R5G and R6G types do not present a large enough statistical sample for the corresponding results to be reliable.

Commenting on the simulation results appearing in Tables 8 and 9, it should be noted that at the cost of a 5.6% increase in storage overhead:

- The true Hit Ratio value is increased by a factor of nearly 18% (for the R3G type of SC/SF output it increases by nearly 23%).

TABLE 8
Expected vs. Measured Values for Each RNG Type in the SC/SF Output

Type	Instances Expected	Instances Measured
R0G	4584	4500
R1G	3576	3709
R2G	1395	1334
R3G	363	371
R4G	71	78
R5G	11	6
R6G	1	2

TABLE 9
Comparison of the Hit and Hit Ratio Values per Each RNG Type in the Classical SC/SF Method and Each of its Two Ranked Variations

Type	Instances	Technique	Hits	Hit Ratio %
R1G	3709	Classical SC/SF	1795	48.4
		Variation 1	2271	61.2
		Variation 2	2448	66.0
R2G	1334	Classical SC/SF	444	33.3
		Variation 1	574	43.0
		Variation 2	676	50.1
R3G	371	Classical SC/SF	77	20.8
		Variation 1	145	39.1
		Variation 2	161	43.4
R4G	78	Classical SC/SF	22	28.2
		Variation 1	25	32.0
		Variation 2	26	33.3
R5G	6	Classical SC/SF	0	0
		Variation 1	3	50.0
		Variation 2	1	16.7
R6G	2	Classical SC/SF	0	0
		Variation 1	0	0
		Variation 2	0	0

- The "I/O Savings" value is increased by a factor of 17% (assuming one I/O operation per logical block of text).

The actual performance values of the new ranking technique may in fact be slightly better than the values appearing in Tables 6 through 9. Tables 1 and 2 show that only seven SC/SF partitions are encoded, although the scheme introduces three bits of storage overhead per color. The 111 code value remains unused, meaning that there is room to encode one more partition. This means that an implementation of the technique that fully

exploits the stated storage overhead would achieve even better performance. However, the objective of the current study is to establish the method together with its performance analysis and simulation results.

At this stage it is worth noting that the RDBMS environment proves to be very useful in calculating values of complex expressions like the ones giving $P_{h\langle \mathrm{R1G}\rangle},\ldots,P_{h\langle \mathrm{R6G}\rangle}$ at the end of Section 3. Two relations are created, one named "real" and one named "ghost". The real relation registers the probability for any NFD instance to achieve a given rank value, whereas the ghost relation does the same for any FD instance. Each relation is comprised of two attributes: "rank" and "prob," which pick up their values from Table 4. The value of $P_{h\langle \mathrm{R1G}\rangle}$ appearing in the previous equations can be calculated now by adding the outcomes of the following two SQL statements:

```
SELECT SUM(Real.Prob*Ghost.Prob)
FROM Real, Ghost
WHERE Real.Rank>Ghost.Rank;

SELECT 0.5*SUM(Real.Prob*Ghost.Prob)
FROM Real,Ghost
WHERE Real.Rank=Ghost.Rank;
```

Things become a bit more complex in more involved cases. For example, the expected value for $P_{h\langle \mathrm{R2G}\rangle}$ is obtained by adding the numeric outcomes of three SQL SELECT statements that follow two SQL CREATE TABLE statements as follows:

```
CREATE TABLE Ghost2(Rank1,Rank2,Prob)
AS SELECT A.Rank, B.Rank, A.Prob*B.Prob
FROM Ghost A, Ghost B;

CREATE TABLE Case2(Rrank,Grank1,Grank2,Prob)
AS SELECT Rank, Rank1, Rank2, Real.Prob*Ghost2.Prob
FROM Real, Ghost2;

SELECT SUM(Prob)
FROM Case 2
WHERE Rrank>Grank1 AND Rrank>Grank2;

SELECT 0.5*SUM(Prob)
FROM Case2
WHERE ((Rrank=Grank1) AND (Rrank>Grank2)) OR
    ((Rrank>Grank1) AND (Rrank=Grank2));
```

TABLE 10
The Hits and Hit Ratio Results per Each RNG Type in the Classical SC/SF and the Proposed Ranking (Variation 2) Configuration

Technique	Type	Instances	Hits Expected	Hits Measured	Hit Ratio Expected (%)	Hit Ratio Measured (%)
Classical SC/SF	R1G	3709	1855	1795	50.0	48.4
Variation 2			2308	2448	64.4	66.0
Classical SC/SF	R2G	1334	455	444	33.3	33.3
Variation 2			652	676	48.9	50.1
Classical SC/SF	R3G	371	93	77	25.0	20.8
Variation 2			148	161	39.9	43.4
Classical SC/SF	R4G	78	16	22	20.0	28.2
Variation 2			26	26	33.6	33.3

```
SELECT 0.33*SUM(Prob)
FROM Case2
WHERE (Rrank=Grank1) AND (Rrank=Grank2);
```

Table 10 compares the analytical results to the simulation measurements for the Hit and Hit Ratio metrics for the SC/SF and ranking variation 2. In the latter case, a coin bias equal to 60–40% is assumed. Results obtained up to the R4G type of SC/SF output are presented because the numbers of R5G and R6G instances measured were very small and thus they are considered as unreliable statistical samples. Note that the theoretical calculations compare well with the values measured during simulation. As a result, some sound confidence is established with regard to the probabilistic model introduced.

6. EPILOGUE

The principles, a theoretical model, and simulation results of a new technique that ranks the output of the classical SC/SF method have been presented. It is emphasized that there are real life applications which would benefit from such a type of ranked SC/SF output. A satellite binary representation is established which calls for minimal storage overhead. This additional representation comes at the cost of an increased CPU overhead, paid mainly during creation time of the modified structure, whereas the query processing stage remains unaffected. The information recorded under the new scheme is of a relatively less deterministic nature when compared to the classical SC/SF method. As a result, at query

processing time, the information content of the new representation is translated into a rank value that is assigned to each block candidacy appearing in the classical SC/SF output.

The performance of the new technique is examined and checked against that of the classical SC/SF both theoretically and by simulation. The flexibility of the specific environment makes it possible to break down the SC/SF output into different types of NFD-to-FD groupings (RNGs). The simulation results as well as the values calculated theoretically are presented and compared to each other at the level of each RNG type in the SC/SF output.

A number of successful partitioned variations of the SC/SF method aimed at increasing its retrieval efficiency in parallel computer architectures have been proposed [5]. The nature of the partitioning scheme introduced in this study does not interfere with any such partitioned variation designed to exploit parallelism. One could thus easily incorporate the methodology proposed in this study by building on top of an existing partitioned SC/SF variation.

It is quite interesting to note that the type of SC/SF output ranking introduced in this study considers only the binary representation part of the SC/SF organization and not the actual text. Croft and Savino [2] have introduced a technique that ranks the SC/SF output by considering the semantics of the text base contents. In essence, the two ranking techniques could coexist and a combined rank value for each SC/SF block candidacy could be produced. Such a combined rank value would reflect both the relevance of the query to the semantics inherent in the corresponding text as well as the credibility of the corresponding portion of the binary SC/SF representation.

The authors thank the Computer Center at the University of Wolverhampton, Max Caines in particular, for the invaluable technical support provided. Special thanks are also due to Prof. R. Moreton for having read and commented on an early version of this effort.

REFERENCES

1. S. Christodoulakis and C. Faloutsos, Design considerations for a message file server, *IEEE Trans. Software Eng.* 10(2):201–210 (1984).
2. W. B. Croft and P. Savino, Implementing ranking strategies using text signatures, *ACM Trans. Office Inform. Syst.* 6(1):46–62 (1988).
3. D. Dervos, P. Linardis, and Y. Manolopoulos, Binary ranking for the signature file method, Information and Software Technology, to appear.
4. C. Faloutsos, Access methods for text, *ACM Comput. Surveys* 17(1):49–74 (1985).
5. C. Faloutsos, Signature based text retrieval methods—a survey, *IEEE Data Eng. Bull.* 13(1):25–32 (1990).

6. W. Feller, *An Introduction to Probability Theory and Its Applications*, 3rd ed., Vol. 1, John Wiley, New York, 1950.
7. D. Harman and G. Candela, A very fast prototype retrieval system using statistical ranking, *ACM SIGIR Forum* 23(3–4):100–110 (1989).
8. D. L. Lee and C. W. Leng, Partitioned signature files—design issues and performance evaluation, *ACM Trans. Office Inform. Syst.* 7(2):158–180 (1989).
9. S. E. Robertson, The probability ranking principle in information retrieval, *J. Document.* 33(4):294–304 (1977).
10. G. Salton, *Automatic Text Processing: The Transformation Analysis and Retrieval of Information by Computers*, Addison–Wesley, Reading, MA, 1989.
11. C. Stanfill and B. Kahle, Parallel free-text search on the connection machine system, *Commun. ACM* 29(12):1229–1239 (1986).
12. C. Stanfill, Information retrieval using parallel signature files, *IEEE Data Eng. Bull.* 13(1):33–40 (1990).

Received March 1992; revised June 1993, August 1993; accepted November 1993