S-Index: a Hybrid Structure for Text Retrieval

D. Dervos[1,2] P. Linardis[1] Y. Manolopoulos[1]

[1] Dept. of Informatics, Aristotle University, 540 06 Thessaloniki, Greece
e-mail: {ddervos,manolopo}@athena.auth.gr

[2] Dept. of Informatics, Technology Educational Institute, 541 01 Thessaloniki, Greece

Abstract

Today, two classes of indexing methods enjoying wide applicability are the Inverted Index and the Superimposed Coding based Signature File (SC-SF). The former is most efficient in query processing but utilizes extra storage of size comparable to that of the textbase, whereas the latter is most efficient in storage utilization. The present study builds upon the results obtained in previous research [2], and proposes a hybrid structure for text retrieval. The new structure is labelled S-Index and is shown to be of a tunable performance which ranges between two extreme ends. At the one extreme end, S-Index turns into a Signature File, which involves zero information loss and, in this respect, it is faster than the ordinary SC-SF method. At the other extreme end, S-Index becomes an Inverted Index. The advantage of the proposed access method is that frequently queried sections of text are indexed via an Inverted Index, whereas the bulk of the textbase, which is not frequently targeted by user queries, is stored in the form of a Signature File.

1 Introduction

Two standard methods for processing Boolean type queries by an Information Retrieval (IR) system are the Inverted Index [5] and the Superimposed Coding based Signature File (SC-SF) [3]. In either case, an intermediary index structure is utilized. The Inverted Index excels in query processing efficiency, whereas the Signature File involves a simpler structure and utilizes significantly less secondary storage. The Inverted Index is implemented as a B+tree variant. By construction, the scheme registers multiple copies of each real-text block address, i.e. once for each key-word present in the block. The index needs to frequently undergo re-organization under intensive information insertion/updating procedures. Also, the method is reported to perform poorly for multiple term user queries [6].

The SC-SF intermediary index is a sequential structure with records consisting of a real text block address and a fixed size binary signature. By construction, the scheme does not register search key values and each real text block address is stored only once. Compared to the Inverted Index, SC-SF is more efficient in handling new document insertions. However, it introduces information loss, i.e. the search engine output involves a number of false matches. The latter may be identified only by applying a full text scan operation on every real text block short-listed in the search engine output. For each single word query processed, the entire SC-SF structure needs to be scanned. This implies increased processing cost, which is further overloaded by the full text scanning stage mentioned earlier. In terms of query processing efficiency, SC-SF is better only next to full text scanning.

A desirable development would be to combine the best of the two worlds and establish a scheme, which encodes frequently addressed sections of the textbase via an Inverted Index, while the bulk of the remaining text is compressed in the form of a fully reversible, i.e. lossless, signature file pattern.

In a previous paper, we have investigated the limits of SC-SF, by introducing two new information encoding schemes: Perfect Encoding (PE) and the Exactly Reversible Signature File (ERSF) [2]. Both approaches introduce zero information loss, i.e. no false matches occur at the query processing stage. The textbase is assumed to utilize a vocabulary which may be of a large, yet finite, size V. As in the case with SC-SF, text is seen to be divided into logical blocks, each involving D distinct vocabulary terms. PE and ERSF differ from SC-SF in assuming each vocabulary term (i.e. key-word) to be mapped on a distinct integer number in the [0,...,$V-1$] range, an integer which becomes the ID number of the word in question. This word-to-integer mapping scheme may be implemented by means of a perfect hash function [4].

Perfect encoding considers the maximum number of messages B_{max} a given (V,D) configuration may support:

$$B_{max} = \binom{V}{D}$$

Under PE, block signatures are encoded via a binary number in the [0,...,$\lceil log_2 B_{max} \rceil - 1$] range. Evidently, PE comprises an upper limit for SC-SF with regard to information compression. However, PE involves an increased CPU overhead when compared to SC-SF. This due to block signature encoding at signature file creation, and due to block signature decoding at query processing.

Proceedings of the First East-European Symposium on Advances in Databases and Information Systems (ADBIS'97). St.-Petersburg, September 2–5, 1997.

St.-Petersburg: University of St.-Petersburg, 1997.

In the case of ERSF, each block signature pattern is V bits long and registers 1s in bit positions which correspond to the D numbers of the vocabulary terms present in the block. The remaining bit positions register zeroes.

The moment when PE comprises an upper limit for SC-SF with regard to information compression, ERSF comprises an upper limit with regard to processing efficiency. This is because ERSF signatures involve zero information loss. As stated in the previous, each SC-SF block candidacy short-listed in the search engine output needs to undergo a full text scan operation. The latter determines the false matches and the qualifiers for the given search criterion. However, this is not the case for ERSF: its search engine output contains no false matches. In all other aspects, the ERSF structure is of an SC-SF type, implying analogous CPU overhead at signature file creation as well as at the query signature processing stage.

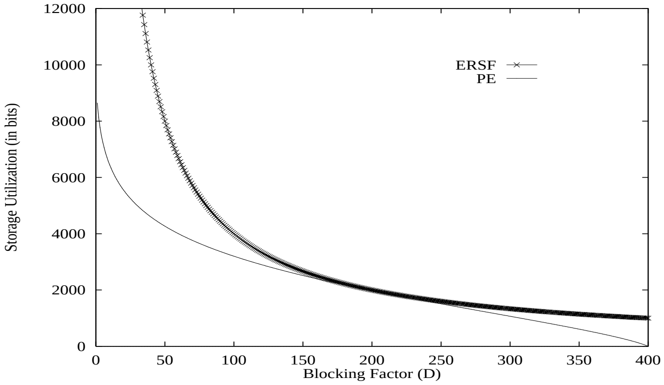

Figure 1: Signature file size dependency on blocking factor for PE and ERSF (assuming V=400 and N=1000).

Figure 1 is taken from [2]. It is representative of a number of calculation results obtained for PE and ERSF storage utilization. It considers an example textbase of N=1000 words. The textbase consists of words which are drawn at random from a vocabulary of size V=400. Apparently, a vocabulary term may be selected more than once. The curves reflect the upper limit values of storage each method's signature file utilizes, for a range of blocking factor (D) values.

The PE and ERSF curves approach each other, with ERSF staying at higher storage values, in the region around $D \approx \frac{V}{2}$. For ERSF, the latter translates into having half of the block signature pattern register 1s. Summarizing, when the blocking factor is approximately equal half the vocabulary size, ERSF (which excels SC-SF with regard to query processing efficiency) achieves nearly optimal storage utilization.

The benefits of PE, ERSF and the Inverted Index may be combined in a new hybrid text retrieval scheme which is introduced in the sequel. The new method is labelled *S-Index*, where *S-* stands for "Signature" and *Index* implies the Inverted Index.

2 Description of S-Index

Hybrid structures which combine SC-SF with the Inverted Index are reported to improve the performance of the Signature File method [1, 6]. However, the information loss problem remains. Consequently, significant processing and I/O overhead is introduced during the full text scanning stage.

In our case, the aim is to combine the query processing efficiency of ERSF with the information compression rate achieved by PE. In accordance with Section 1, the two co-occur when the blocking factor D is approximately half the size of the textbase vocabulary V. Since D is usually set to a value which is much smaller than $\frac{V}{2}$, a divide-and-conquer strategy may be applied. In order to take advantage of the maximum information compression rate of ERSF, the vocabulary V is partitioned into two subvocabularies which are equal in size. Each subvocabulary is then further partitioned into smaller subvocabularies, and so on. Next, the subvocabularies are mapped on nodes in a binary tree structure, each node corresponding to one specific subvocabulary. The root which is at level 0, represents the whole vocabulary, whereas the leaves correspond to single vocabulary terms. This scheme is generalized so that each node has M children, i.e. M is the tree fanout.

One begins with an ERSF-like block signature at level zero. However, it does not get appended to the signature file as such. Sub-sections of the block signature are considered in a top-down traversal of the tree structure. The block signature sub-section which is considered for a given node relates directly to the node's local textbase subvocabulary.

A block signature sub-section is appended to the signature file as soon as the number of 1s contained is larger than half the size of the local subvocabulary. For as long as the latter is not true, block signature sub-sections continue being divided into smaller partitions each one of which is considered against the corresponding node at the lower tree level. As soon as a signature sub-section is appended to the signature file, neither it nor any one of its partitions are considered any further in this top-down tree traversal process.

Evidently, S-Index consists of two parts. The *upper* part is a tree with nodes relating to specific subvocabularies. The latter are not registered since each one of them is implied by the corresponding node's positioning in the tree. Each tree node points to a linked list of records. Linked records comprise the *lower* part of the S-Index.

Linked records pointed to by nodes in the same level of the upper S-Index are stored in the same physical file. For a node at level i, each record registers three fields: a *pointer* to the next record in the list, a *block number* which is the address of the corresponding real text block, and a *signature section* which is a $\lceil V/M^i \rceil$ bits long binary string. The latter registers the block signature sub-pattern, which corresponds to the vocabulary which is local to the node in question. For the special case of a leaf node, the local vocabulary is of size one, i.e. the corresponding linked record registers only two fields: a *pointer*, and a *block number*.

Furthermore, the pointer value may be dispensed with by allocating a separate physical file to each node of the S-Index structure. The linked records (if any) of the node would then reside in a single file, all by themselves. As a result, each one of them would need to register only *block number* and *signature section* values. Even for the case when a separate file is allocated to each tree level rather than to each tree node, this is the case for level–0, since there is only one node to it.

Example

Let V=8, D=3, M=2.The nodes of the binary tree in Figure 2 relate to specific subvocabularies: e.g. node B accommodates words 0 through 3, node G has a local vocabulary of two words (6 and 7), etc. In the beginning, at signature creation time, the typical block of text is mapped on an 8-bit signature of the ERSF type; a signature which registers three 1s and five 0s. A perfect hashing algorithm may be utilized to map each vocabulary term on a distinct number in the [0,..,7] range.

The original (full size) signature is split into sub-sections which are in direct relationship with the subvocabularies of the S-Index tree structure. Sub-sections of each block signature are then considered to traverse the S-Index from the root to the leaves. Considering a typical node at level i, the block signature sub-section and the block's address value are appended to the node's linked records for as long as the signature sub-section registers more 1s than 0s.

Figure 3 presents linked records pointed to by node "C" in Figure 2. The node has associated with it a local vocabulary of size four. Consequently, each record in Figure 3 registers a block signature sub-section with three 1s and one 0. □

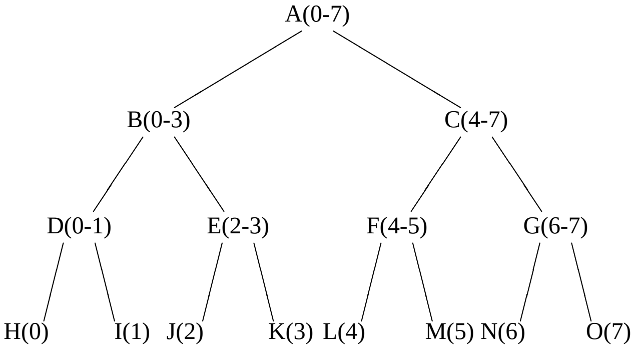

Figure 2: Binary S-Index for V=8.

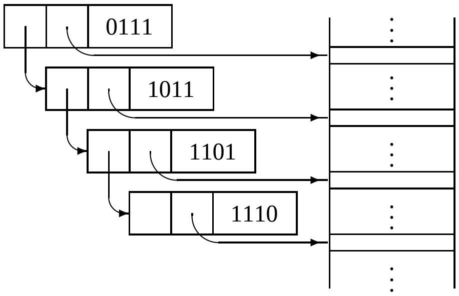

Figure 3: Linked records and textbase.

One more special case should be considered for S-Index. It is by construction that nodes in the level which is just above the leaves will have their linked records register signature sections of the "11" type. The latter implies that both words in the local vocabulary are present in the corresponding logical block of text. For this special case, linked records will have a structure similar to that of the leaves, i.e. the *signature section* is not registered.

3 Analysis

Quite similarly to SC-SF, the textbase is divided into logical blocks, each containing D distinct vocabulary terms. Evidently, the given (V, D) configuration may record up to $\left(\begin{array}{c} V \\ D \end{array}\right)$ distinct logical blocks, or "messages". For simplicity, let the textbase comprise of all messages encoded by the (V,D) scheme, each message taken once. One more assumption may now be made which does not affect the general case for S-Index; V is taken to be an integer power of M, the fanout of the tree structure:

$$V = M^n$$

One may now proceed and calculate the size of the population of linked records under each and every tree node. The result comprises a first indication on the performance of S-Index. Evidently, the first assumption in the previous paragraph is made in order to simplify this calculation. For the special case considered, the S-Index nodes are populated with linked records in a symmetric fashion, i.e. the number of records registered under each node depends only on the corresponding tree level. In other words, each S-Index level–i is characterized by N_i, the number of linked records registered under a typical node at the specific level of the S-Index structure. The issue is better explained by means of an example.

Example

Consider the binary tree structure shown in Figure 2. The root is at level i=0, whereas the leaves are at level i=3. The structure is populated with linked records registering block signature sub-sections from a textbase utilizing a vocabulary of V=8 terms. Each logical block of text contains D=3 vocabulary terms. In accordance with the previous, the textbase consists of $\left(\begin{array}{c} 8 \\ 3 \end{array}\right) = 56$ messages, each message taken once. Each block signature involves three 1s set in the [0,..,7] range of binary bit positions.

By construction, there will be no records appended at level–0. This is because the one and only node at level–0 has a local vocabulary of size 8. Signature (sub)sections are appended to a node for as long as the number of 1s contained exceeds half the size of the local vocabulary:

$$N_0 = 0$$

Each of the two nodes at level–1 has a local vocabulary of size 4. Apparently, records are appended to nodes B and C for signature sub-sections which register three 1s. Thus, for each node at level–1:

$$N_1 = \left(\begin{array}{c} 4 \\ 3 \end{array}\right) = 4$$

For the typical node at level–2, say D: the (local) vocabulary being of size 2, it registers signature sub-sections with two 1s. For the case considered, there exists a third 1 which may lie anywhere in the subtree originating at node E, or anywhere in the subtree originating at node C. Let the latter be labelled as the E- and C-subtree sets of registered records. Each linked record registered under node D is meant to be combined with the members of the E- and C- sets of registered records to produce valid block signature patterns:

$$\left(\begin{array}{c} 2 \\ 2 \end{array}\right) \left\{ \left(\begin{array}{c} 2 \\ 1 \end{array}\right) \left(\begin{array}{c} 4 \\ 0 \end{array}\right) + \left(\begin{array}{c} 2 \\ 0 \end{array}\right) \left(\begin{array}{c} 4 \\ 1 \end{array}\right) \right\}$$

However, the first product term of the inner summation in the above should be ruled out as it never occurs in the specific S-Index signature sub-sections placement algorithm. Had the third 1 been placed anywhere in the vocabulary which is local to node E, then there would be three 1s in the vocabulary which is local to node B. The latter implies that the signature sub-section would have been appended to node B and would not have stepped down one level to node D. Thus, for the typical node at level–2:

$$N_2 = \left(\begin{array}{c} 2 \\ 2 \end{array}\right) \left(\begin{array}{c} 2 \\ 0 \end{array}\right) \left(\begin{array}{c} 4 \\ 1 \end{array}\right) = 4$$

Quite analogously, the vocabulary local to a typical node at level 3 (say H) is of size 1. The one (trivial) bit signature sub-section combines itself with all possible ways of allocating the remaining two 1s anywhere within the I-, E- and C- subtrees. However, subject to the constraint that valid block signature instances are only the ones which have not had their signature sub-sections be appended at higher tree levels, the number of linked records appended to the typical level–3 node will be:

$$\begin{aligned} N_3 &= \binom{1}{1}\left\{\binom{1}{0}\binom{2}{1}\binom{4}{1}+\binom{1}{0}\binom{2}{0}\binom{4}{2}\right\} \\ &= 14 \end{aligned}$$

□

Summarizing, for the special type of textbase considered and for a typical node at level–i in the (binary) S-Index structure:

- Signature sub-sections appended to the node's linked list of records involve numbers of 1s exceeding half the size of the local vocabulary (V_i) up to and including $min(D, V_i)$, where D is the blocking factor.

- Symbolizing by q the number of 1s in the signature sub-section of a record appended to the linked list of records belonging to the typical node at level–i: when q is less than D, the remaining $D-q$ 1s distribute themselves in subtrees originating at level–i, level–(i–1),..., up to and including level–1. This way, valid block signatures are produced, subject to one constraint: for each level–s which lies higher than level–i, the sum of all the numbers of 1s distributed in nodes at level–s and lower in the tree structure does not exceed half the size of the vocabulary local to the typical node at level–(s–1).

For the typical node at level–i and the linked record which registers q 1s in its signature sub-section, the two constraints mentioned in the previous are stated as follows:

$$\begin{array}{llcl} C_1 & : (k_1 + ... + k_i & = & D-q) \\ C_2 & : (q + k_i + ... + k_{i-\lambda} & < & M \cdot \frac{V_{i-\lambda}}{2} + 1) \end{array}$$

where $\lambda \in [0, .., i-1]$, and the k_j (for $1 \leq j \leq i$) parameters relate to numbers of 1s present in sub-trees originating at the j-th level.

The previous are integrated into the expression which calculates N_i for the general (V,D,M) case:

$$N_i = \sum_{q=\lfloor \frac{V_i}{2}+1 \rfloor}^{min(D,V_i)} \binom{V_i}{q} \times$$

$$\times \sum_{\substack{k_j=0,\\ j\in[1,...,i],\\ C_1,\\ C_2}}^{min(D-q,(M-1)V_j-q)} \binom{(M-1)V_1}{k_1} \cdots \binom{(M-1)V_i}{k_i} \quad (1)$$

Having calculated N_i, it becomes possible to proceed and calculate:

1. the storage (in bits), S_{SI} utilized by the lower S-Index structure, and

2. the number, R_{SI}, of lower S-Index records accessed during the processing of a single word query.

By comparing S_{SI} to the corresponding ERSF and PE values, a first indication on the secondary storage utilized by S-Index may be obtained. Similarly, R_{SI} comprises a first indication on the query processing efficiency of the S-Index, assuming one I/O operation per record retrieved.

The next two sections focus on the details of calculating S_{SI} and R_{SI} for the example textbase considered, i.e. the one outlined in the beginning of the current section.

4 Storage Utilization

The upper part of S-Index, i.e. its backbone tree structure, introduces storage overhead similar to the one introduced by the backbone tree structure of the Inverted Index. Both are much smaller in size than the storage utilized by the lower S-Index and Inverted Index structures. In this respect, the calculations which follow consider the storage utilized by the linked records of the S-Index structure and by the leaf nodes of the Inverted Index structure.

For the example textbase considered, $\binom{V}{D}$ distinct messages imply a $\left\lceil log_2 \binom{V}{D} \right\rceil$ bits long block signature pattern, under perfect encoding. Of the same size will be the block's binary address. Thus, the storage (in bits) utilized by PE will be:

$$S_{PE} = \binom{V}{D} \times 2 \times \left\lceil log_2 \binom{V}{D} \right\rceil$$

ERSF differs from PE in that each block signature pattern is V bits long. The corresponding expression for storage utilization (in bits) will be:

$$S_{ERSF} = \binom{V}{D} \times \left(V + \left\lceil log_2 \binom{V}{D} \right\rceil \right)$$

In the fully Inverted Index environment, the probability for a vocabulary term to be contained in a D-term (logical) block of text equals:

$$P_w = \frac{D}{V}$$

As a result, $\frac{D}{V} \times \binom{V}{D}$ records will populate the linked list of each and every one of the V vocabulary terms used. Each linked record will register a $\left\lceil log_2 \binom{V}{D} \right\rceil$ block address, plus a 4-byte pointer. The total storage (in bits) utilized will be:

$$S_{II} = D \times \binom{V}{D} \times \left(32 + \left\lceil log_2 \binom{V}{D} \right\rceil \right)$$

In the case of S-Index, one needs to first calculate the height of the tree structure. Assuming a fanout value M, for the (V,D) configuration, the leaf nodes will be at tree level $k = \lceil log_M V \rceil$. For example, for a binary tree structure involving a V=16 terms vocabulary, the leaves of the S-Index structure are at level $k = \lceil log_2 16 \rceil = 4$. The root of the tree is assumed to always be at level zero.

For the intermediate level, m, of the S-Index structure, there will be a total of M^m nodes to it. Utilizing Equation (1), for the example textbase considered, each node involves N_m linked records. With the exception of the root and the two lower levels, a typical linked record at level m registers a $\lceil \frac{V}{M^m} \rceil$ bits long signature pattern, a 4-byte pointer and a $log_2 \left\lceil \binom{V}{D} \right\rceil$ bits long address.

In accordance with the assumption made earlier, the lower part of each S-Index level is stored into a separate physical file. This implies that the 4-byte pointer overhead may be saved for the root level. Also, it is by construction that signature sub-sections appended to nodes in the lower two levels contain all 1s. They may also be skipped, i.e. they need not be registered. Thus, the total storage overhead (in bits), S_{SI}, for S-Index is given by:

$$S_{SI} = N_0 \times \left(V + \left\lceil log_2 \begin{pmatrix} V \\ D \end{pmatrix} \right\rceil \right) +$$

$$+ \sum_{i=1}^{k-2} N_i \times \left(\left\lceil \frac{V}{M^i} \right\rceil + \left\lceil log_2 \begin{pmatrix} V \\ D \end{pmatrix} \right\rceil + 32 \right)$$

$$+ \left(\frac{N_{k-1}}{2} + N_k \right) \times V \times \left(\left\lceil log_2 \begin{pmatrix} V \\ D \end{pmatrix} \right\rceil + 32 \right)$$

5 Cardinality of Inspected Index Records Set

An indication on the query processing efficiency, under each one of the indexing schemes considered, may be obtained by considering the total number of index records inspected for the typical query. It is noted that none of structures involves information loss. This means that each index record selected points to a block of text known to qualify for the query in question.

For simplicity, single term queries are considered. The more general case involves Boolean constructs which break down into single term query sub-tasks.

ERSF and PE are both sequential index structures, i.e. the total number of block signatures need to be retrieved and inspected for each single term query. The total number of index records inspected under either method, $R_{ERSF/PE}$, is given by:

$$R_{ERSF/PE} = \begin{pmatrix} V \\ D \end{pmatrix}$$

In the Inverted Index scheme, the entire population of linked records, registered under the vocabulary term in question, is inspected:

$$R_{II} = \frac{D}{V} \begin{pmatrix} V \\ D \end{pmatrix}$$

In the case of S-Index, for each single term query processed, one path of nodes is traversed from the root to the leaves of the tree structure. Each linked record which resides along this path is retrieved and inspected for possible qualification of the corresponding textual block. Thus:

$$R_{SI} = \sum_{m=0}^{k-2} N_m$$

where N_m is calculated from Equation (1) and $k = \lceil log_M V \rceil$.

In the calculation of R_{SI}, it is noted that linked records at levels (k–1) and k need not be inspected. Their signature sub-sections are known to contain all 1s, i.e. the corresponding blocks of text are known to contain the vocabulary term in question.

6 Calculation Results

Figures 4 and 5 present calculation results obtained for the storage utilization and the number of index records inspected under each one of the indexing schemes considered. The testbed used was a textbase with vocabulary size V=64 and variable blocking factor D. In each case, the textbase was taken to consist of all possible messages encoded by the (V, D) configuration, each message taken once. Three types of S-Index fanout values were considered: M=2, 4 and 8.

As stated previously, PE is the most efficient signature file scheme with regard to storage utilization. So, the Inverted Index and S-Index curves in Figure 4 are normalized to the corresponding PE value. The PE line intersects the vertical axis at the (normalized) "1" value. For example, for D=20 the Inverted Index was measured to introduce a storage overhead which is nearly 15 times larger than the storage utilized by the PE index.

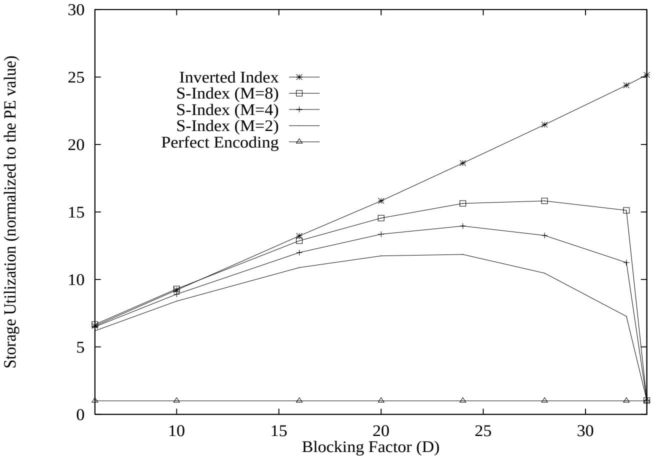

Figure 4: Storage utilization efficiency curves for the indexing structures considered (V=64).

The S-Index variation with M=2 (i.e. a binary tree) is seen to outperform the M=4 and M=8 ones. As expected, S-Index turns into an Inverted Index for small D values.

Considering what has been stated in Section 2, for the case where $D > \frac{V}{2}$ (i.e. $D > 32$ for Figure 4), S-Index turns into ERSF: the only node which points to linked records in the lower S-Index part is the root. This explains the sudden "drop" of the S-Index curves in Figure 4: all the linked records are stored in a single sequential file, leading to considerable storage savings. This fact may also be taken to suggest that the storage utilization efficiency of S-Index will be improved by allocating a separate file to each tree node.

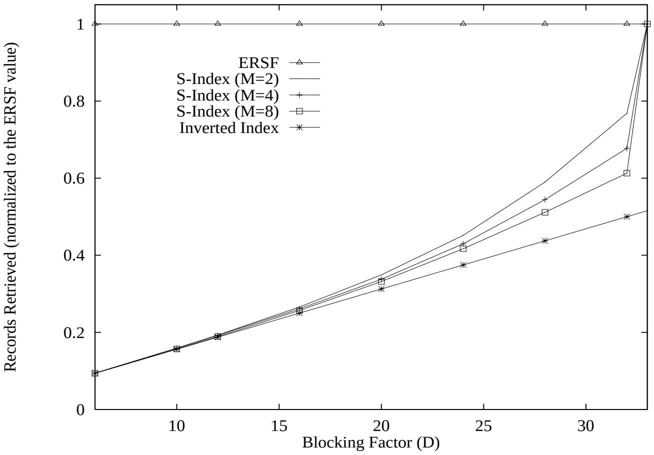

Figure 5: Index records retrieved for the structures considered (V=64).

For the example textbase considered, the upper S-Index part may safely be assumed to always reside in main memory. Similarly, the backbone tree structure of the Inverted Index is also assumed to reside in main memory.

As stated in the previous, it is possible to calculate the number of (lower part) S-Index records retrieved for inspection by a single word query. Quite similarly to Figure 4, Figure 5 plots the number of records inspected under the S-Index and the Inverted Index schemes, normalized to the corresponding ERSF value.

Commenting on Figure 5, S-Index is seen once more to turn into an Inverted Index for small D values and into ERSF for $D > \frac{V}{2}$. Quite notably, S-Index achieves retrieval efficiency values which, for $D < 32$, lie close to the Inverted Index ones. For example, S-Index and the Inverted Index retrieve nearly one third of the records retrieved by ERSF when D=20 (Figure 5). The Inverted Index is guaranteed to retrieve only records corresponding to real text blocks containing the query term in question. Thus, for D=20, S-Index avoids retrieving two thirds of the records retrieved by ERSF. For M=2, such a gain comes at the cost of having S-Index utilize nearly two thirds of the storage utilized by the Inverted Index (Figure 4).

Summarizing on Figures 4 and 5, for as long as the upper S-Index part resides in main memory, the binary tree (M=2) variation is seen to perform better than the M=4 and M=8 ones.

7 Conclusion

A hybrid indexing scheme (S-Index) has been introduced which, for certain (V, D) textbase configurations, combines the advantages of the Inverted Index and the Signature File structures. The scheme has been shown to turn into an Inverted Index when $D \ll V$, and into an ERSF structure for $D > \frac{V}{2}$.

Once a variable blocking factor D scheme is adopted, S-Index may be tuned into indexing selected textbase sections (i.e. sections containing words appearing frequently in user queries) in the form of an Inverted Index, while the bulk of the remaining text is encoded in the form of a Signature File.

In the immediate future, we intend to

- consider S-Index with a variable blocking factor (D), and
- take into consideration practical aspects of real textbases, like the Zipfian distribution of words, and incorporate this into the mapping of words on numbers in the [1,...,V] range. By this, it is expected that the storage utilization efficiency of the method will be improved.

Last but not least, it is noted that during the single word query processing stage, only one node is considered at each level of the S-Index structure. Also, nodes at different levels may be processed independently from each other. In this respect, S-Index allows for further investigation in the context of distributed processing.

References

[1] Deppisch U.: “S-Tree: A Dynamic Balanced Signature Index for Office Retrieval”, *Proceedings of the ACM SIGIR Conference*, pp.77-87, 1986.

[2] Dervos D., Linardis P. and Manolopoulos Y.: “Perfect Encoding: a Signature Method for Text Retrieval”, *Proceedings, International Workshop on Advances in Databases and Information Systems (ADBIS)*, pp.176-182, September 10-13, 1996.

[3] Faloutsos C.: “Access Methods for Text”, *ACM Computing Surveys*, Vol.17, No.1, pp.49-74, 1985.

[4] Fox E.A., Chen Q.F. and Heath L.S.: “A Faster Algorithm for Constructing Minimal Perfect Hash Functions”, *15th ACM SIGIR Conference Proceedings*, pp. 266-273, 1992.

[5] Harman D., Fox E., Baeza-Yates R.A. and Lee W.: “Inverted Files”, in *Information Retrieval: Data Structures and Algorithms*, Frakes W. and Baeza-Yates R. (Eds.), Prentice Hall, pp.28-43, 1992.

[6] Jagadish H., Faloutsos C.: “Hybrid Index Organizations for Text Databases”, *Proceedings of the Extending Database Technology Conference (EDBT)*, pp. 310-327, March 1992.