Perfect Encoding: a Signature Method 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

A new methodology is introduced, where blocks of text are replaced by a compressed, fully reversible, signature pattern. Full reversibility implies zero information loss, thus the new method is termed Perfect Encoding. The method's analytical model is produced and, where applicable, contrasted with the current practice in signature file organizations. Analysis results indicate that it comprises a potential candidacy for information retrieval implementations. In particular, perfect encoding has the potential to develop into an alternative or complementary scheme to inverted or signature file based systems.

1 Introduction

Free text indexing methodologies, like the inverted file and signature file approaches, enjoy applicability in the modern Information Retrieval (*IR*) environment [6, 4]. The inverted file approach is characterized by its efficiency in text retrieval operations whereas the Signature File (SF) involves a simple structure and requires significantly less storage overhead. The Superimposed Coding Signature File (SC-SF) comprises the most widely used signature file variation. SC-SF is applied in indexing objects for a variety of text-based applications [13, 7, 11].

SC-SF considers the textbase to consist of a number of logical blocks, each block involving a constant, pre-specified, number of distinct, non-common words (D). An F-bit *signature* or *descriptor*, consisting of m ones (1s) set in the [1...F] range, is associated with each word in the logical block. For each block of text, its D word signature patterns are bit-ORed and produce an F-bit *block signature* pattern. Block signature patterns are used as an intermediary, compressed representation for text indexing purposes.

The SC-SF intermediary representation utilizes nearly 10% of the storage utilized by the corresponding text base. This is a significant improvement over the inverted file environment which calls for a storage overhead of 100% or larger [3, 10]. In this respect, SC-SF comprises a compromise between inverted file and full-text scanning methods. Its structure is highly modular and allows for efficient query processing in parallel architectures [8, 14, 9].

A desirable development would be to combine the speed and the exactness of the inverted file organization with the low storage overhead, modularity and simplicity of the signature file. The current study comprises an effort in the direction of improving the signature file organization by achieving a higher text compression rate as well as by eliminating the information loss involved.

Proceedings of the International Workshop on Advances in Databases and Information Systems (ADBIS'96). Moscow, September 10–13, 1996.
Moscow: MEPhI, 1996.

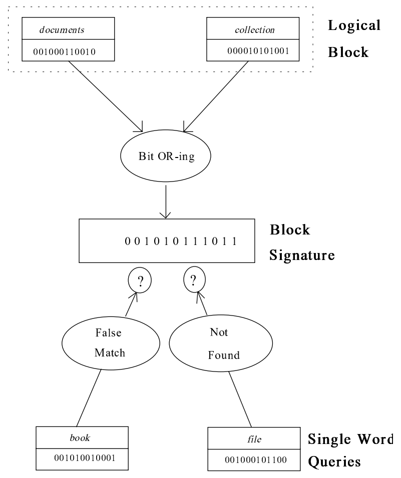

Figure 1: SC-SF example with F=12, D=2, m=4

Figure 1 comprises a simplified (F=12, D=2, m=4) configuration. A two word (*documents*, *collection*) logical block of text is considered, together with its 12 bit long block signature pattern. When four single word queries (*documents*, *collection*, *book* and *text*) are processed against the intermediary binary representation of text, the latter is successful in predicting the presence of *documents* and *collection*: every single 1 in each word signature pattern has the corresponding bit position in the block signature pattern set to 1, too. The scheme is also successful in predicting the absence of the word *file*: bit position number 10 is set to 1 in the word signature pattern but registers a 0 in the block signature.

In the case of the single word query *book*, the SC-SF configuration in Figure 1 is seen to introduce information loss. The word is erroneously taken to be present in the block, due to the 1s of its signature corresponding to 1s set by either one of the two other words present. Searching for the word *book* is said to result into a *False Match* or a *False Drop*, thus being indicative of the information loss introduced by the SC-SF text compression stage. The rate at which false matches occur is reflected by the *False Drop Probability* (*FDP*) metric [4]:

$$FDP = \text{Prob}\{\text{signature qualifies where block does not}\}$$

Alongside with the storage utilized by the signature file and the method's query processing efficiency, FDP comprises an important performance relating parameter for SC-SF. An increase in storage utilization decreases FDP and vice-versa.

SC-SF performs optimally when F, m and D resume values which make Relation (1) hold true [3].

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

When this happens, half of the binary positions in the average block signature pattern register an "1" value, the other half register a "0" value, and FDP is given by Relation (2) [12].

$$FDP \;\approx\; 2^{-m} \tag{2}$$

2 SC-SF Vocabulary

Under the SC-SF configuration, Relation (1) holds true and the scheme appears to able to "accommodate" a *potential* vocabulary of up to $V_0 = \binom{F}{m}$ distinct word signature patterns. The latter may safely be considered to be infinitely large in practice.

Let the average SC-SF block signature pattern involve x ones and $F-x$ zeroes. Figure 2 shows the distribution of 1s for a typical (F=600, m=7, D=60) SC-SF configuration [2]. Analysis shows x to be very near the $F/2$ value [10].

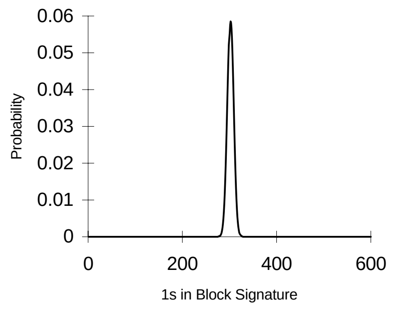

Figure 2: Distribution of 1s in the (F=600, m=7, D=60) block signature pattern

It is noted that $F/2$ is usually much larger than m. This implies a very large number of potential distinct word signatures "present" in the average block signature pattern: $D_0 = \binom{F/2}{m}$. In practice, the number D of text words placed in each SC-SF block is much smaller than the number of detected word signatures D_0, i.e. $D << D_0$. As a consequence, SC-SF introduces information loss in the form of false matches/drops. For example, for the SC-SF configuration in Figure 2 where D=60, $D_0 \approx \binom{F/2}{m} = \binom{300}{7} = 4.04E + 13$.

Let w_0 be a word, randomly chosen from the V_0 terms in the potential vocabulary. Relation (3) calculates the probability P_{w_0} for w_0 to have its 1s match with 1s in the average block signature pattern.

$$P_{w_0} \;=\; \frac{D_0}{V_0} \tag{3}$$

As stated previously, the average SC-SF block signature pattern is one which is half-full with 1s. Each one of the 1s present in the word's signature patterns thus has a probability of 0.5 to match a 1 in the block signature. P_{w_0} is thus given by:

$$P_{w_0} \;\approx\; \frac{1}{2^m} \tag{4}$$

D.L. Lee, Y.M. Kim and G.Patel in [10] note that the number of blocks containing the query term (true drops) is a parameter which may safely be ignored in single-level SC-SF configurations. This being the case in the current study, plus by having $D << D_0$, Relations (2) and (4) suggest that P_{w_0} is the well known false drop probability metric (FDP).

Let us assume that the actual vocabulary, herewith referred as the practical vocabulary, encountered in real textbases is of size V. The practical vocabulary will be a subset of the potential vocabulary defined by the $\binom{F}{m}$ patterns, so $V \leq V_0$. Assuming a uniform distribution of the V real words over the V_0 potential words as well as of the D terms amongst the D_0 patterns in the average block signature:

$$\frac{D}{D_0} \;=\; \frac{V}{V_0} \tag{5}$$

Relation (5) suggests that when $D << D_0$ then $V << V_0$. Thus, one may safely assume that SC-SF deals with a practical vocabulary which is finite rather than infinite in size.

Combining Relations (1), (3), (4) and (5):

$$V \;=\; D \times 2^m \;=\; D \times 2^{\frac{F \times \ln 2}{D}} \tag{6}$$

Relation (6) allows for the calculation of V given the values for D and m.

Table 1 considers a number of SC-SF variations whereby F=60 and lists the values of V calculated by Relation (6) next to $V_0 = \binom{F}{m}$. Without compromising on the efficiency of SC-SF, it is seen that as D becomes larger, the size of the real word vocabulary becomes smaller. Clearly, V_0, an infinitely large number for most practical applications, should not always be considered as the vocabulary size supported by SC-SF.

		Vocabulary Size	
m	D	V	V_0
21	2	4194304	7.98E+15
14	3	49152	1.73E+13
7	6	768	3.86E+08
6	7	448	50063860
3	14	112	34220
2	21	84	1770

Table 1: SC-SF vocabulary size dependency on D and m when F=60

An SC-SF configuration often used in real life applications is one where F=1000, D=100 and m=7. Utilizing Relation (6), the size of the corresponding practical vocabulary is of the order of 13,000. This value is much smaller than the potential word vocabulary for the configuration in question, i.e. $\binom{F}{m} = \binom{1000}{7} = 1.94E + 17$. However, a 13,000 word vocabulary suffices for most practical applications: the number of attributes is usually less than a few hundred in a formatted database [4] and a vocabulary of 10,000 distinct terms is reported to correspond to a textbase of about 100,000 words [1].

3 Perfect Encoding

A new methodology is introduced, termed Perfect Encoding (*PE*), whereby blocks of text are replaced by a compressed, signature like pattern. The method's analytical model is established in the sequel and, where applicable, contrasted with the current practice.

Unlike the classical superimposed coding approach (SC-SF), *PE* establishes a 100% accurate, fully reversible intermediary representation of the textual information. Text words are assumed to be taken from a vocabulary of large but finite size V (e.g. V = 30-100,000 words). Each word is mapped on a unique integer w_i in the [1...V] range, e.g. by using a perfect hash function [5]. In accordance with the analysis presented in section 2, the finite vocabulary assumption should not be considered as a drawback for *PE*.

Similarly to SC-SF, each *PE* logical block of text involves a fixed, predetermined number (D) of distinct, non-common words drawn from the vocabulary in question. The block is said to comprise a *message*, labelled by a unique *message number* M, where $M \in [1, 2, ..., M_{max}]$. Two messages differ for as long as there exists at least one word which belongs to only one of the corresponding blocks. The maximum number of messages encountered for a given (V, D) configuration equals the maximum message number M_{max}. Thus, the latter equals the number of times D items may be drawn from a population of V:

$$M_{max} = \binom{V}{D} \tag{7}$$

The minimum space required to encode each one of the M_{max} messages is F bits, such that:

$$M_{max} = 2^F = \binom{F}{1} + \binom{F}{2} + ... + \binom{F}{F} \tag{8}$$

Section 4 which follows shows perfect encoding to fully utilize the encoding capacity of the F bits long block signature pattern. Assuming a typical F=1000, D=100 configuration for *PE*, the size of the vocabulary supported is calculated to be of the order of 38,900 words. The latter is nearly 3 times larger than the one calculated for the corresponding SC-SF organization in section 2.

4 PE Signature Creation and Query Processing

Given the vocabulary size V and the blocking factor D, Relation (7) calculates the total number of distinct messages produced. In addition, each word is hashed on a unique word number $w_k \in [1...V]$, where k=1,...D marks the position of the word in the block of text considered.

To construct the *PE* block signature pattern, the block is mapped on a message number M which is calculated by considering the D word numbers present. More specifically, the D word numbers are first arranged in descending order: $w_D < w_{D-1} < ... < w_1$. The subscripts are rearranged so that w_1 and w_D correspond to the largest and the smallest word numbers present, respectively. The $w_D w_{D-1} w_{D-2} ... w_1$ sequence of word numbers is taken to temporarily identify the corresponding message. Temporary message identifiers of this type are sorted in descending lexicographic order. Once sorted, the message with the "largest" temporary identifier is assigned message code number M=0, followed by message code number M=1, etc. The message which corresponds to the "smallest" identifier is assigned message code number M= $\binom{V}{D} - 1$. Each message code number M is used to uniquely identify the message in question. The scheme produces the most compact form of block (message) signature.

For the *PE* method to efficiently construct the block signatures, message number M must be computable by means of an expression which involves the block's $w_1, w_2, ..., w_D$ numbers. Relation (9) is the expression used for the message encoding scheme in question:

$$M = \sum_{k=1}^{D} \binom{V - w_k}{k} = \sum_{k=1}^{D} \prod_{n=1}^{k} \frac{V - w_k + 1 - n}{n} \tag{9}$$

For example, Table 2 lists *PE* message code numbers assigned to blocks in the case of a V=9, D=4 environment.

Message Number	Words in Message			
(M)	w_4	w_3	w_2	w_1
0	6	7	8	9
1	5	7	8	9
2	5	6	8	9
3	5	6	7	9
4	5	6	7	8
5	4	7	8	9
6	4	6	8	9
7	4	6	7	9
8	4	6	7	8
...	...	...	...	...
125	1	2	3	4

Table 2: *PE* message number assignment for V=9, D=4

Considering the seventh row in Table 2, one has w_4=4, w_3=6, w_2=8 and w_1=9. Relation (9) calculates the corresponding M (message code number) value as follows:

$$M = \binom{9-9}{1} + \binom{9-8}{2} + \binom{9-6}{3} + \binom{9-4}{4} = 6$$

Following the construction of the *PE* signature file, information is retrieved by considering a single word with word number q and by asking whether q is present in message M. The pseudocode in Figure 3 processes q against M and an $exists$=true outcome implies that q is one of the D words present in M. Appendix I refers to the principle behind the just introduced *PE* block signature construction algorithm.

The k_c term in Figure 3 is calculated (recursively) by:

```
$k_0 = 1$;
$k_1 = 2$;
$c' = c - 2^{U(c)-1}$;
if $c - 2^{U(c)} \leq 2^{U(c)-1} - 1$ then $k_c = k_{c'}$
else $k_c = k_{c'} + 1$;
```

where $U(x)$ symbolizes the floor of $\log_2 x$, i.e. $U(x) = \lfloor \log_2 x \rfloor$.

At this point, it should be noted that the present study does not emphasize on establishing the most efficient query processing algorithm for *PE*. Rather, it aims at outlining a new framework which achieves the highest possible compression of information without introducing information loss. In this respect, the relative performance of various signature file variations may be checked against

```
$exists$=false;
$current = \binom{V-q}{D}$;
while ($current < M$) and ($current < \binom{V}{D}$)
    {$a = 0$;
    while ($a < q$) and ($current < M$)
        {$c = 1$;
        repeat until ($c = 2^a$) or ($M < current$)
            {calculate the $k_c$ number;
            $current = current + \binom{V-q}{D-k_c}$;
            $exists$ = NOT $exists$;
            $c = c + 1$;
            }
        $a = a + 1$;
        }
    }
exit.
```

Figure 3: Query processing pseudocode for *PE*

that of *PE*. The *PE* framework may also be used as a guide for creating other, less efficient in storage utilization but faster than *PE*, information encoding/decoding methodologies of the signature file type.

5 Comparison: PE vs. SC-SF

In the lines which follow, *PE* is considered next to SC-SF with regard to the storage overhead introduced as well as the efficiency in processing user queries. A plus for perfect encoding is that it introduces zero information loss. As a consequence, no full text scanning operation needs to be conducted during the query processing stage. Apart from the CPU overhead involved, this fact also implies additional storage savings as the textbase does not need to be present at the end user end in a network based realization.

5.1 Storage Overhead

Let us refer to a message by the term *Perfect Encoding Block Signature* (*PE* signature) to differentiate it from the classical block signature which is termed *SC-SF Block Signature* (SC-SF signature). Let also S be the size, in bits, of the signature file for both the *PE* and SC-SF configurations. The signature file, together with the file of pointers marking the beginning of each logical block of text, comprise the total storage overhead. Let S_{Ovh} stand for the percentage of increase in storage introduced by the intermediary representation. Symbolizing by F the size of the block signature, by l the size (in bits) of the average word and by N the number of words in the document, S_{Ovh} is given by:

$$S_{Ovh} = \frac{S}{N \times l} = \frac{\frac{N}{D} \times F}{N \times l} = \frac{F}{D \times l} \tag{10}$$

Relation (10) suggests that the F/D ratio may be interpreted as a quality measuring factor: for as long as F/D remains fixed, S_{Ovh} remains constant.

Perfect encoding starts differing from the classical signature file when one considers the dependency of S_{Ovh} on D. For the SC-SF method, F, D and m comprise a set of design parameters. Furthermore, Relation (1) has been found to optimize S_{Ovh} and the false drop probability rate for SC-SF. Thus for SC-SF:

$$S = \frac{N}{D} \times F \;\Rightarrow\; S = \frac{m}{\ln 2} \times N \tag{11}$$

The right hand side of Relation (11) suggests that in the case of SC-SF, S is independent of D.

In the case of *PE*, full utilization is made of the encoding capability inherent to the F bits long block signature pattern:

$$F = \log_2 M_{max} \tag{12}$$

Combining Relations (7) and (12):

$$\begin{aligned} S &= \frac{N}{D} \times F = \frac{N}{D} \times \log_2 \frac{V!}{D!\,(V-D)!} \\ &= \frac{N}{D} \times \sum_{k=1}^{D} \log_2 \frac{V+1-k}{k} \end{aligned} \tag{13}$$

As it is derived in Appendix II, a $D \rightarrow D+1$ change results into a negative variation for S, namely $\Delta S < 0$. In other words, the larger the block size D becomes, the smaller is the *PE* introduced storage overhead.

From Relation (1) for SC-SF and Relations (7) and (12) for *PE*, F is plotted against D in Figure 4. The text base environment considered in this example consists of N=1000 words, taken from a vocabulary of size V=400. Two curves are plotted for the SC-SF scheme, one which involves word signatures with m=1 and a second with m=7. The two SC-SF configurations are labelled $SC - SF(m = 1)$ and $SC - SF(m = 7)$, respectively.

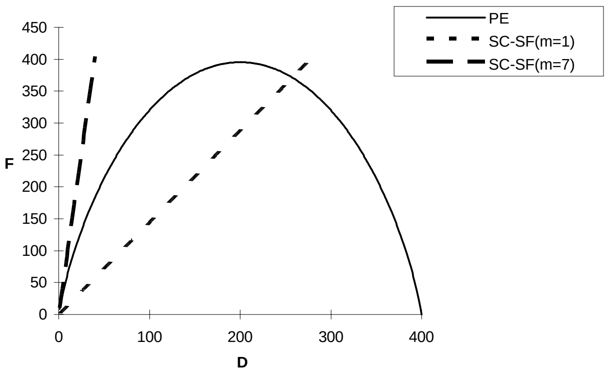

Figure 4: The V=400, N=1000 environment: block signature size dependency on block size for PE and SC-SF

It is noted that the two SC-SF lines in Figure 4 do not extend past the F=V=400 value. When F=V, it is simpler plus more efficient to map each word on a separate bit position in the block signature. Simpler because the number of bit positions in the block signature pattern equals the number of words in the vocabulary: one may thus establish a simple one-to-one correspondence between words in the vocabulary and numbers in the [1,...,F] range. This special case in considered in section 6 and results into a simple information encoding/decoding organization labelled *Exactly Reversible Signature File* (*erSF*.

The *PE* structure in Figure 4 is seen to utilize less and less storage (F) as D approaches the vocabulary size (V). As expected, SC-SF(m=7) utilizes more storage than SC-SF(m=1). The latter achieves a text compression rate higher than that of the *PE* structure up to nearly D=250. However, both the SC-SF(m=1) and SC-SF(m=7) schemes introduce information loss (i.e. false drops), whereas *PE* does not. Quite rightly, the *PE* storage overhead drops to zero when D=V. When the block size equals that of the vocabulary, each block is sure to contain all the words: a fact which needs zero bits to be encoded.

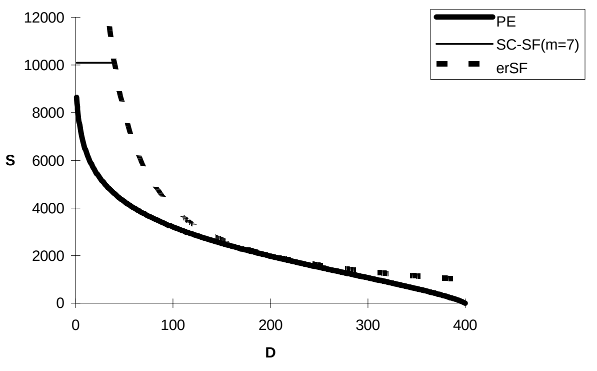

Figure 5: The V=400, N=1000 environment: signature file size dependency on block size for PE, *erSF* and SC-SF

Figure 5 considers the environment used in the example of Figure 4. This time, the size of the signature file (S) is plotted against D. In the classical SC-SF configuration, Relation (11) suggests that S depends only on m and remains constant as D varies. The *PE* curve was produced by utilizing Relation (13). In accordance with the analysis, S decreases monotonously as D increases. Section 6 comments on the *erSF* curve shown in Figure 5.

5.2 Query Processing Efficiency

In the case of the classical SC-SF, the complexity of the query processing algorithm is linear with m, the number of bit positions set to 1 in the average word signature pattern. Moreover, one needs to take into consideration the extra processing overhead associated to full text scanning which is necessary in order to resolve the false drops present in the SC-SF output.

Perfect encoding involves no information loss but its query processing algorithm (Figure 3) is less efficient than that of the SC-SF structure. It is for this reason that perfect encoding is herewith presented as a framework for signature file organization rather than a methodology directly applicable in real life, as such. Section 5.1 presents a case where this approach pays off: *erSF* is seen to relate to zero information loss plus involve a simple signature file structure leading to SC-SF comparable query processing performance.

6 Exactly Reversible Signature File

It is obvious that PE falls behind SC-SF in query processing efficiency. However, the exactly reversible signature file variation introduced in section 5.1 is indicative of the way PE may be considered as a framework for improving the performance of signature file organizations. The *erSF* structure efficiently encodes/decodes textual blocks into block signature patterns in a way which introduces zero information loss.

More specifically, it is more efficient to replace SC-SF by *erSF* for D values which are greater than or equal to the size of the vocabulary used. Let $D_{F=V}$ be the value of D for which F=V. For the curves in Figure 4, $D_{F=V}$=278 for SC-SF(m=1), and $D_{F=V}$=40 for SC-SF(m=7). Thus, the *erSF* structure encodes text with a direct word-to-bit position mapping mechanism when D=$D_{F=V}$. The storage overhead associated with the lookup table for the word-to-bit position mapping may be avoided by utilizing a perfect hash function.

The *erSF* storage utilization curve is plotted in Figure 5. Perfect encoding is seen to achieve a higher text compression rate when compared to *erSF*. However, it is worth noting that when D approaches the $V/2$ value, the two curves converge to each other with *PE* always being better (lower S values) than *erSF*. Consequently, the *erSF* structure achieves a text compression rate which is nearly as good as that of the *PE* scheme when $D \approx V/2$. In contrast to the SC-SF structure, *erSF* involves zero information loss. This fact, together with the simplicity of its structure which implies efficient query processing, makes *erSF* a good choice for the $D \approx V/2$ region. A scheme which would approach perfect encoding in other regions of D values, much like *erSF* does in the $D \approx V/2$ region, comprises a desirable objective and a subject of further research.

7 Conclusion

A new approach to information encoding and retrieval is introduced. Perfect Encoding (*PE*) is characterized by the following:

- A finite sized vocabulary of words is considered, this being the case in most practical applications.
- The method does better than the classical Superimposed Signature File (SC-SF) by (a) involving a fully reversible signature pattern, (b) achieving a higher degree of information compression and (c) supporting a larger vocabulary of distinct terms in practice.
- The scheme comprises a framework for measuring the performance of signature file based information encoding structures. For when $D \approx V/2$, a simple *PE* variation is introduced which is shown to achieve better performance than that of SC-SF.

Provided that its efficiency at the query processing stage is improved, perfect encoding has the potential to evolve into an alternative or complementary scheme next to the currently used inverted or signature file based information retrieval realizations.

Appendix I

Upon close inspection, the example of the PE message encoding scheme in Table 2 involves M value regions which alternate into containing and not containing a given word. For example, considering the single word query q=4: M numbers in [0,4], [15,24], [35,44], etc. correspond to blocks which do not contain q, whereas the blocks encoded by M numbers in [5,14], [25,34], etc. contain the word in question.

Table I-1 considers the case where q=6, V=9 and D=4. It presents the k_c, $C_{k_c} = \begin{pmatrix} V-q \\ D-k_c \end{pmatrix}$ and *exists* values calculated and used by the PE query processing algorithm in Figure 3. Also listed is information with regard to the w_4, w_3, w_2 and w_1 (word) values present in the corresponding M range numbers.

Table I-1. q=6 presence/absence for the PE encoding scheme in Table 2.

Appendix II

From Relation (15), when D increases to $D+1$ the change of S is:

$$\Delta S = \frac{N}{D+1}\sum_{k=1}^{D+1}\log_2\frac{V+1-k}{k} - \frac{N}{D}\sum_{k=1}^{D}\log_2\frac{V+1-k}{k} =$$

k_c	C_{k_c}	M range	w_4	w_3	w_2	w_1	*exists*
0	0	none	> 6	> 6	> 6	> 6	False
1	1	[0,0]	6	> 6	> 6	> 6	True
1	1	[1,1]	5	> 6	> 6	> 6	False
2	3	[2,4]	5	6	> 6	> 6	True
1	1	[5,5]	4	> 6	> 6	> 6	False
2	3	[6,8]	4	6	> 6	> 6	True
2	3	[9,11]	4	5	> 6	> 6	False
3	3	[12,14]	4	5	6	> 6	True
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...

$$\frac{N}{D+1}\log_2\frac{V-D}{D+1}+\left(\frac{N}{D+1}-\frac{N}{D}\right)\sum_{k=1}^{D}\log_2\frac{V+1-k}{k}=$$

$$\frac{N\times D}{D\times(D+1)}\log_2\frac{V-D}{D+1}-\frac{N}{D\times(D+1)}\sum_{k=1}^{D}\log_2\frac{V+1-k}{k}=$$

$$\frac{N}{D\times(D+1)}\sum_{k=1}^{D}\left(\log_2\frac{V-D}{D+1}-\log_2\frac{V+1-k}{k}\right)$$

Thus, we have:

$$\Delta S \;=\; \frac{N}{D\times(D+1)}\sum_{k=1}^{D}\log_2\left(\frac{V-D}{V+1-k}\times\frac{k}{D+1}\right)$$

Since k varies in the [1,..,D] range it follows that:

$$\frac{1}{V+1-k}\leq\frac{1}{V+1-D}\Rightarrow\frac{k}{D+1}\times\frac{V-D}{V+1-k}<1$$

Combining the last two equations, one gets:

$$\log_2\left(\frac{V-D}{V+1-k}\times\frac{k}{D+1}\right)<0\Rightarrow\Delta S<0$$

Thus, for the *PE* envinronment, the larger the block size D becomes, the smaller is the introduced storage overhead.

References

[1] Dewey C.: "Relative Frequency of English Speech Sounds", *Harvard University Press*, Cambridge, MA, 1950.

[2] Dervos D., Manolopoulos Y and Linardis P.: "Comparison of Signature File Models with Superimposed Coding", *submitted*.

[3] Christodoulakis S. and Faloutsos C.: "Design Considerations for a Message File Server", *IEEE Transaction on Software Engineering*, Vol.10, No.2, pp.201-210, 1984.

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

[5] 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, Copenhagen, Denmark, 1992.

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

[7] Ishikawa Y., Kitigawa H. and Ohbo N.: "Evaluation of Signature Files as Set Access Facilities in OODBs", *Proceedings of the 1993 ACM SIGMOD Conference*, pp. 247-256, 1993.

[8] Kim J.K. and Chang J.W.: "HPSF: A Horizontally-divided Parallel Signature File Method", *Proceedings of the IEEE 1st ICA^3PP Conference*, pp.559-562, April 1995.

[9] Lee D.L.: "Massive Parallelism on a Hybrid Text-Retrieval Machine", *Information Processing & Management*, Vol.31, No.6, pp.815-830, 1995.

[10] Lee D.L., Kim Y.M. and Patel G.: "Efficient Signature File Methods for Text Retrieval", *IEEE Transactions on Knowledge and Data Engineering*, Vol.7, No.3, pp.423-435, June 1995.

[11] Sacks-Davis R., Kent A., Ramamohanarao K., Thom J. and Zobel J.: "Atlas: A Nested Relational Database System for Text Applications", *IEEE Transactions on Knowledge and Data Engineering*, Vol.7, No.3, pp.454-470, June 1995.

[12] Sacks-Davis R. and Ramamohanarao K.: "A Two Level Superimposed Coding Scheme for Partial Match Retrieval", *Information Systems*, Vol.8, No.4, pp.273-280, 1983.

[13] Tsichritzis D. and Christodoulakis S.: "Message Files", *ACM Transactions on Office Information Systems*, Vol.1, pp.88-98, 1983.

[14] Zezula P., Ciaccia P. and Tiberio P.: "Hamming Filter: a Dynamic Signature File Organization for Parallel Stores", *19th International Conference on Very Large Data Bases*, pp.314-327, March 1993.