COCA Filters: Co-Occurrence Aware Bloom Filters*

Kamran Tirdad, Pedram Ghodsnia, J. Ian Munro, and Alejandro López-Ortiz

Cheriton School of Computer Science
University of Waterloo

Abstract. We propose an indexing data structure based on a novel variation of Bloom filters. Signature files have been proposed in the past as a method to index large text databases though they suffer from a high false positive error problem. In this paper we introduce COCA Filters, a new type of Bloom filters which exploits the co-occurrence probability of words in documents to reduce the false positive error. We show experimentally that by using this technique we can reduce the false positive error by up to 21 times for the same index size. Furthermore Bloom filters can be replaced by COCA filters wherever the co-occurrence of any two members of the universe is identifiable.

Keywords: Information Retrieval, Bloom Filters, Signature Files, Locality Sensitive Hash Functions

1 Introduction

Inverted indices and variants thereof are the preferred data structure currently in use in search engines. However in environments that are very sensitive to index size this method becomes impractical since they require approximately 50% of the size of the corpus for the index file. By compressing the index file and pruning of the less frequent query terms we can reduce the size of inverted indexes down to 10% of the corpus size [30]. In areas where false positive errors are acceptable a more space efficient method called signature files is applicable. With this method it is possible to reduce the size of index file significantly at the cost of precision. Another key advantage of this method is that it can be used in optimizing intersection queries in distributed inverted indices[20, 26]. Parallelizability and the simplicity of the insertion are two other benefits of this method that make it a suitable choice for certain environments.

When using signature files a signature is maintained for each document. A signature is basically a sequence of bits. There are several different methods for computing the signature of a document. One of the most common methods is to use a randomized data structure called Bloom filter. In Bloom filter-based

* This work was supported by NSERC of Canada and the Canada Research Chairs program

signature files it is implicitly assumed that every pair of words is equally likely to appear in the same document while in practice this assumption is not true.

In this paper we introduce a new variant of Bloom filters named co-occurrence aware Bloom filters or COCA Filters for short. COCA Filters utilize the probability of the co-occurrences of the words in documents to improve the false positive error. We show through experiments that COCA Filters can reduce the space by up to 75% for the same false positive error or equivalently reduce the false positive error by up to 21 times for the same index size.

Reducing the size of the signature file index or equivalently its false positive error makes COCA Filters ideally suited for applications which are extremely sensitive to the size of the storage.

The rest of the paper is organized as follows: Section 2 reviews related work and background. Section 3 describes the details of our approach and our proposed methods. Section 4 presents the evaluation and analysis of our proposed methods and finally we conclude our work in section 5.

2 Previous Work and Background

Inverted file indices and signature files are two well established indexing methods which have been proposed for large text databases [11, 15, 30]. Although using the inverted files is more favourable because of its wide range of useful properties in comparison to signature files, Carterette and Can in [11] showed that signature file indexes can be as good as inverted file indexes in special environments where memory is scarce and a given false positive rate is acceptable. Library catalogues, multimedia files with many attributes, medical cross references, and a large lexicon or lists of streets for a GPS system are examples of text databases in which signature file can work faster with less storage. However, the high false positive error rate is one of the critical problems of the signature file method which makes it impractical for many applications.

Signature files are a forward index method which stores a signature for every document. Hashing every single term of a document and concatenating the hash values of the terms can be considered as a simple signature for that document. Alternatively, superimposed coding can be used to create a signature of a document. In this method, hashing every word of the document yields a bit pattern of size m, with k bits set to 1, in which m and k are design parameters. The bit patterns are superimposed (OR-ed) together to form the document signature. Searching for a set of words is handled by creating the signature of each word and OR-ing them together to build the query signature and returning all documents with a matching pattern.

To avoid having document signatures that are flooded with 1s, long documents are divided into smaller blocks, that is, pieces of text that contain a constant number of unique words. Each block of a document gives a block signature and block signatures are concatenated to form the document signature.

Although not explicitly stated in the literature, superimposed coding is a variation of Bloom filters, a well-known randomized data structure first suggested in [5]. A Bloom filter is a probabilistic data structure used to check whether an element belongs to a set with possible false positive error but zero false negative error. It consists of a bit vector of size m and k independent hash functions

$h_1, h_2, ..., h_k$ with ranges of $1, ..., m$. All the bits are initially set to zero. These hash functions can be interpreted as uniform random number generators over the range of $1, ..., m$. For every element of the set, say x, the bits $h_i(x)$ for $(1 \leq i \leq k)$ are set to 1. Some bits of the array might be turned on more than once, but this will not affect the status of the array. To check if an item, say y, is a member of S, the k positions of $h_i(y)$ for $1 \leq i \leq k$ in the array should be checked and if one or more of the k positions are set to 0, it can be assured that y has not been inserted to this array and consequently is not a member of S. If all k positions are set to 1, it is assumed that y is in S. However, there is some probability that this assumption is wrong. Therefore, a Bloom filter may result in a false positive error, also known as false drop error.

Bloom filters have been used in a wide variety of applications in recent years. They are used as spell-checkers [23], as a means of succinctly storing a dictionary of unsuitable passwords for security purposes [28], to speed up semi-join operations [22], for Web cache sharing [16] and in many other areas. In order to support multi-sets, Cohen and Matias introduced spectral Bloom filters [14]. Chazelle et al. in [13] introduce a similar data structure which is called a Bloomier filter in order to approximate functions.

Bloom [5] proved that the false positive probability of the Bloom filter is about $f = (1 - \frac{e^{-kn}}{m})^k$. Recently Bose et al. in [6] showed that the Bloom's formula for false positive is not accurate and gave a proper formula. They also demonstrated that for large enough values of m (size of Bloom filter) with small values of k (number of hash functions), the difference between Bloom's formula and the actual false positive rate is negligible.

To obtain an estimate of the efficiency of Bloom filters, it is good to know the information theoretic lower bound of the size of any data structure that can represent all sets of n elements from a universe with false positive for at most a fraction t of the universe but allows no false negative. Broder et al. [7] showed that to achieve a false positive rate less than t, we must have $m > n\ lg(\frac{1}{t})$ bits. Furthermore, they showed that this lower bound in Bloom filters is $m > n\ lg(e) \cdot lg(\frac{1}{t})$ and consequently argued that space-wise Bloom filters need more than a $lg(e) \simeq 1.44$ factor of the information theoretic lower bound. In [25] Pagh et al. introduced a more complicated data structure that achieved this lower bound.

One of the key observations in both of the aforementioned proofs is the assumption that there is no correlation between any two members of the universe, and any subset of the universe with cardinality of n is equally likely. Now assume that some members of the universe are strongly correlated (i.e. given that one of them belongs to a subset, the probability that the other is also a member of that set is very likely). Intuitively this property of possible subsets can be exploited by using special hash functions which produce more "similar" bit patterns (i.e. with smaller hamming distances) for more correlated members of our set and vice versa. For doing so, we use locality sensitive hash (LSH) functions which are customized to hash similar items to the same hash value with high probability [12]. The LSH algorithm has been used in numerous applied settings from bio-sequence similarity search [10] to audio similarity identification [29] and many other areas [17, 19, 24]. Min-wise independent permutations is a locality sensitive hashing scheme for a collection of subsets with the similarity function defined as follows:

$$Pr_{h\in F}[h(A) = h(B)] = sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

In this setting, hash of a set A is defined as $h(A) = min_{a\in A}\pi(a)$ where π is a permutation which was chosen randomly from a min-wise independent permutation family F. A permutation family F (subset of all n factorial permutations) is *min-wise independent* if for any subset X of $[1...n]$, and any $x \in X$, when π is chosen randomly from F we have $Pr(min\{\pi(X)\} = \pi(x)) = \frac{1}{|X|}$ [9].

One of the applications of min-wise independent hash functions was suggested by Broder in [8] to detect near duplicate documents over a large set of documents. Broder suggested to consider a set of shingles (contiguous subsequences of words) for each document and choose a set of t independent random permutations $\pi_1, \pi_2, \pi_3, ..., \pi_t$. For each document D, calling its set of shingles SD, he defined the sketch of Document D as $(min_{a\in SD}\pi_1(a); min_{a\in SD}\pi_2(a); ...; min_{a\in SD}\pi_t(a))$. Then, he argued that the sketch of two documents can be used to estimate their resemblance by computing how many corresponding elements in their sketches are equal. In the next section a similar approach is taken in order to reduce the false positive errors of signature files.

3 COCA Filters

Considering the concept of signature files over human readable texts, some terms (members of the universe) are more likely to exist in the same document (set).

In order to exploit this non randomness, it is preferable to modify the k hash functions of the Bloom filter such that “similar” words (i.e. with high co-occurrence ratio) have “similar” bit patterns (i.e. with less hamming distance). For example if two words occur in almost the same set of documents their bit patterns can be designed such that they differ in a few places. More importantly by using these bit patterns, after inserting these two words there would be more bit positions available for the rest of the words in the Bloom filter causing reduction in the average false positive error. This observation can be formalized as the following optimization problem.

Consider two keywords of x and y from the universe of all the words W with corresponding posting lists of X and Y. Furthermore assume that the k bit positions of each of these two terms are stored in the sets of $H(x)$ and $H(y)$. Rather than having k random numbers between 1 to m, the proposed objective is to design hash functions such that:

$$\forall x, y \in W, \frac{|H(x) \cap H(y)|}{|H(x) \cup H(y)|} = \frac{|X \cap Y|}{|X \cup Y|}$$

Note that the right hand side is determined by the corpus and so is fixed. So this problem can be characterized as given N^2 rational numbers p_{ij} design a bipartite graph with m vertices on one side and N vertices the other side such that for every two vertices V_i and V_j where $0 < i, j \leqslant N$ the following constraints hold:

$$\frac{|Neighbour(V_i) \cap Neighbour(V_j)|}{|Neighbour(V_i) \cup Neighbour(V_j)|} = p_{ij}$$

We conjecture that this problem is NP-hard when we are given p_{ij} as a pair I_{ij} (intersection size) and U_{ij} (union size). Here we propose the following ad-hoc probabilistic approach. Define k co-occurrence-aware hash functions of x to be k of the min-wise independent permutations over the set of X. So, the probability that each hash of two distinct terms x and y be equal to each other is $\frac{|X \cap Y|}{|X \cup Y|} = sim(x, y)$. This new data structure is named co-occurrence-aware Bloom filters or in short *COCA filter*.

Assuming that the probability that two different hash functions produce the same bit position for two different words is negligible, the expected value of the difference between the left and right hand side of the objective function for every two term can be calculated as follows:

$$E\left[\left|\frac{|H(x) \cap H(y)|}{|H(x) \cup H(y)|} - \frac{|X \cap Y|}{|X \cup Y|}\right|\right] \simeq \tag{1}$$

$$\left|\frac{k \times sim(x,y)}{2k - k \times sim(x,y)} - sim(x,y)\right| = \tag{2}$$

$$\left|\frac{sim(x,y) \times (sim(x,y) - 1)}{2 - sim(x,y)}\right| \tag{3}$$

The approximation from (1) to (2) is based on the assumption that pairs of sets with large intersections on average have large unions but obviously this is not true in general. Since $sim(x, y) = \frac{|X \cap Y|}{|X \cup Y|}$ and is in $[0, 1]$, with simple algebraic calculations it can be shown that this value is less than 0.172 and more importantly for the pairs of x and y such that $sim(x, y)$ is close to 0 or 1 this value is close to 0. So over the sets that most of the members are strongly co-related or are not related to each other at all this approach can perform very well. Note that the reason this formula is not dependant on k, the number of hash functions, is the implicit assumption that the bit vector is large enough such that it is quite unlikely for two different hash functions to produce the same bit position for two different words.

Note that by doing so, the reduction in the false positive probability for all of the terms in documents happens at the cost of increasing the false positive probability over random terms which do not exist in any of the documents. In the next section we describe three experiments over three different English corpora comparing COCA Filters to traditional Bloom Filters.

In order to implement COCA filters, k min-wise independent permutations should be picked randomly from a min-wise independent family. Since min-wise independent families are too big for practical applications (in fact it is known that their size is at least $lcm(1, 2, ..., n)$ [9]), variant notions of min-wise independence have been introduced in the literature [21, 27].

In our experiments in order to keep the implementation relatively simple two-universal hash functions has been employed to replicate the behaviour of k independent permutations. For a large prime value of p, k random pairs of (a_i, b_i) are generated where $1 \leqslant i \leqslant k$. The hash of each document ID, say x can be calculated by $(a_i * x + b_i) mod p$. Each hash of the document IDs corresponds to one permutation over the set of all document IDs. This procedure is repeated for all the k random pairs so that there are k different permutations. Consequently, for each permutation the minimum value of each posting list is the hash of the corresponding keyword.

In algorithm 1, the pseudocode as explained in the last paragraph shows how to calculate the k hash functions of the COCA filter and store them in k hash tables $h_1, h_2, ..., h_k$.

Algorithm 1 Hash Calculator For COCA Filter

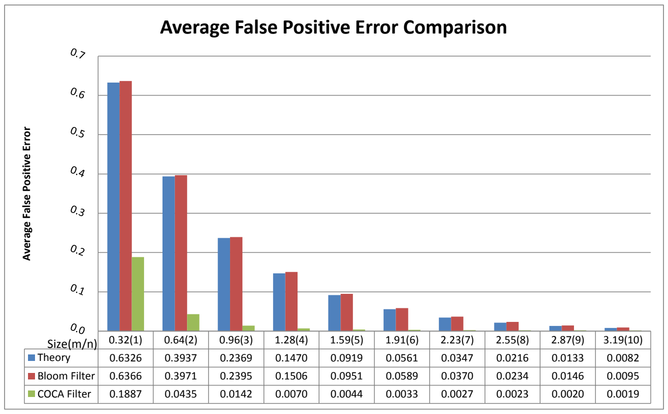
input: Assume documents are numbered from 1 to N and m is the size of the bit vector. The posting list of each term t can be accessed as a set by posting-list(t). k random pairs of (a_i, b_i) where $1 \leqslant i \leqslant k$ are generated.
Output: k hash tables $h_1, h_2, ..., h_k$

```
1: for $i = 1$ to N do
2:     for $j = 1$ to k do
3:         $perm[i][j] \leftarrow (a_j i + b_j) mod P$
4:     end for
5: end for
6: for every term $t$ in the corpus do
7:     for $x = 1$ to k do
8:         $h_x[t] = [min_{num \in posting-list(t)}(perm[num][x])] mod(m)$
9:     end for
10: end for
```

4 Experimental Results

In order to evaluate the effectiveness of COCA Filters in reducing the false positive error, we test them experimentally on three collections. The first corpus is a collection of Wikipedia articles [2]. This collection consists of 2000 high quality pages selected by a team of volunteers. For indexing purposes we stripped all HTML tags, Java scripts, comments and other non-related elements from the html files and removed all numbers and words shorter than 3 characters. The total size of the html files is 244 Megabytes and after cleaning the files and removing the duplicates of the words in each file the total size is reduced to 20.4 Megabytes. According to the statistics provided in [4], in Wikipedia, the average number of words per document is about 400. Based on this assumption, each document of the test collection is divided into partitions of size 400 words. After partitioning each document, the size of its last partition will be less than or equal to 400 words. To address this problem, the number of words in all fragments of each document has been balanced. For example, after partitioning a 700 word document, there will be 2 partitions of size 350 words. The output of this step is $7,500$ partitions with the average size of 350 words per document and 212568 unique terms in total.

The goal of the experiment is to compare the average false positive error of the proposed hash function with that of the theoretic formula and the conventional Bloom filters. Let W be the set of all words. $FP(d)$ is defined as the number of words in W which are not in document d but its corresponding Bloom filter falsely claims that they are. For each document, the false positive error of its



Size(m/n)	0.32(1)	0.64(2)	0.96(3)	1.28(4)	1.59(5)	1.91(6)	2.23(7)	2.55(8)	2.87(9)	3.19(10)
🟦 Theory	0.6326	0.3937	0.2369	0.1470	0.0919	0.0561	0.0347	0.0216	0.0133	0.0082
🟥 Bloom Filter	0.6366	0.3971	0.2395	0.1506	0.0951	0.0589	0.0370	0.0234	0.0146	0.0095
🟩 COCA Filter	0.1887	0.0435	0.0142	0.0070	0.0044	0.0033	0.0027	0.0023	0.0020	0.0019

Fig. 1. The comparison of the average false positive error of the COCA filter method with the conventional Bloom filter and the theoretic formula for the sampled Wikipedia corpus. The x-axis shows the index size in Megabyte and the value in the parentheses indicates the $\frac{m}{n}$ ratio.

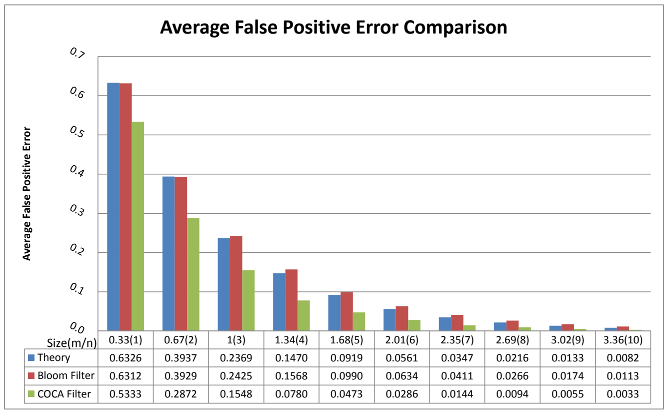
corresponding Bloom filter is defined as $FPE(d) = \frac{FP(d)}{|W|}$. Thus, the average false positive error of a signature file method is $\frac{\sum_{d \in D} FPE(d)}{|D|}$.

In figure 1, the x-axis shows the result of this experiment for different $\frac{m}{n}$ ratios and y-axis shows the corresponding average false positive error. In each method, for each $\frac{m}{n}$ ratio, only the result for the k value which minimizes the false positive error is shown. The total size of the signature file along with the average false positive error is also included in the table below the figure.

From this experiment the following key observations can be derived:

- For all values of $\frac{m}{n}$, the false positive error of conventional Bloom filters is just slightly more than that of the theoretic formula. It confirms the argument of Bose in [6] that Bloom's formula provides only a lower bound for false positive probability. The closeness of the average false positive error of the conventional Bloom filter and the theoretic formula justifies the validity of our implementation of the conventional Bloom filters as well.
- For all values of $\frac{m}{n}$, the false positive error of our proposed methods is significantly better than the false positive ratio predicted by the conventional Bloom filter and the theoretic formula.
- In some cases there is up to a 21 factor improvement in the average false positive error. For example, in $\frac{m}{n} = 2$, the average false positive error of the COCA filter is about 21 times better than the Bloom filter.
- In some cases there is up to a 75% reduction in the size of the index for the same average false positive error. For example if the objective is to achieve an average false positive error less than 0.21 in conventional Bloom filters the $\frac{m}{n}$ ratio should be at least 4 while in COCA filter with the $\frac{m}{n}$ ratio of 1 the average false positive error of 0.20 is achievable. In other words for every bit that is used in the COCA filter, 3 extra bits are required in a

conventional Bloom filter. Note that when $\frac{m}{n} = 1$ the index size is only 1.6% of the polished corpus.



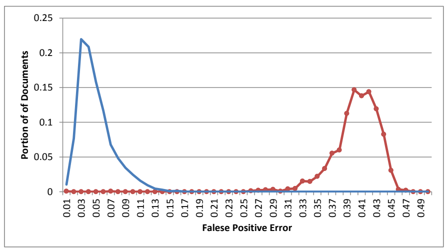
Size(m/n)	0.33(1)	0.67(2)	1(3)	1.34(4)	1.68(5)	2.01(6)	2.35(7)	2.69(8)	3.02(9)	3.36(10)
🟦 Theory	0.6326	0.3937	0.2369	0.1470	0.0919	0.0561	0.0347	0.0216	0.0133	0.0082
🟥 Bloom Filter	0.6312	0.3929	0.2425	0.1568	0.0990	0.0634	0.0411	0.0266	0.0174	0.0113
🟩 COCA Filter	0.5333	0.2872	0.1548	0.0780	0.0473	0.0286	0.0144	0.0094	0.0055	0.0033

Fig. 2. The comparison of the average false positive error of the COCA filter method with the conventional Bloom filter and the theoretic formula for the sampled Google corpus. The x-axis shows the index size in Megabyte and the value in the parentheses indicates the $\frac{m}{n}$ ratio.

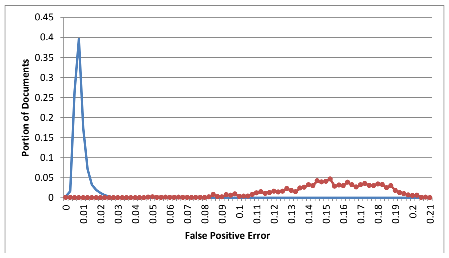
Our proposed approach is based on the co-occurrence of the words in documents and therefore is sensitive to the correlation of documents. In order to investigate the relationship between the degree of correlation among documents and the improvements in average false positive error, the previous experiment was repeated over a collection of weakly-correlated web pages. This collection is a random selection of $900,000$ web pages released by Google in 2002 for a programming contest [1]. We chose about 13500 samples from this collection randomly and performed the previous experiment on the resulting collection. We used the same method as the first experiment for cleaning the documents and fragmenting them. Due to the smaller average size of documents in this collection, we divided the documents into partitions of size 100. After partitioning, the collection had about 35200 documents with the average size of 80 terms and 228715 terms in total.

Figure 2 shows the result of this experiment. Although COCA Filter is still better than conventional Bloom Filter, the improvement in this experiment is not as good as the first experiment. The result of this experiments confirms the sensitivity of our proposed method to the correlation among the terms of the documents.

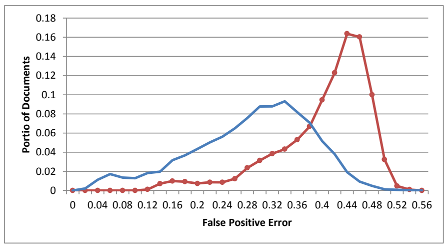
In the first experiment we chose a collection of high quality articles of Wikipedia which are all coherent in writing and have a scientific theme. It is normal to encounter many synonyms of a word instead of a repetition and there is also a somewhat predictable set of antonyms to follow. On the contrary, in the second corpus documents are not coherent neither in meaning nor in the style which results to have lots of random terms from street names and addresses to gene



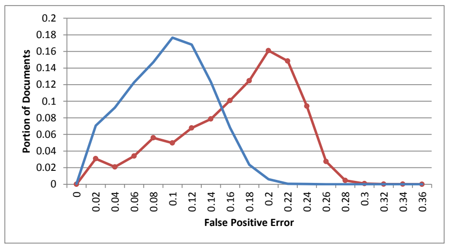
(a) Distribution of false positive error for $\frac{m}{n} = 2$ for the Wikipedia collection



(b) Distribution of false positive error for $\frac{m}{n} = 4$ for the Wikipedia collection



(c) Distribution of false positive error for $\frac{m}{n} = 6$ for the Google collection



(d) Distribution of false positive error for $\frac{m}{n} = 6$ for the Google collection

Fig. 3. Comparison of the distribution of the false positive error of the COCA Filter and the conventional Bloom filter. In each graph the left curve corresponds to the COCA filter and the right curve corresponds to the conventional Bloom filter.

sequences and peoples' names in them. Moreover the diversity of the topics that these terms are covering is higher than the first collection and this diversity reduces the probability of the co-occurrence of the words in documents and consequently reduces the effectiveness of our proposed method.

To ensure that the size of corpus does not have a negative effect on the quality of COCA Filters we repeated the first experiment with a similar but larger collection of Wikipedia documents [3]. This collection is a more comprehensive version of the first collection and consists of 6500 high quality pages selected by a team of volunteers for school students. The size of this collection is more than 3 times the size of the first collection but it is very similar to the first collection in terms of coherency and writing style. We used the same method as the first experiment for cleaning the documents and fragmenting them. After partitioning, the collection had 21543 documents with the average size of 350 words per document and 321500 unique terms in total. Table 1 compares the result of this experiment with the result of the first experiment. It shows that increasing the size of the collection does not increase the average false positive error given that the coherency and style of the corpus remains the same. It confirms that the higher average false positive error of COCA Filters on Google collection is not because of the larger size of this collection and it is only due to the non-coherent, random nature and diversity of that collection.

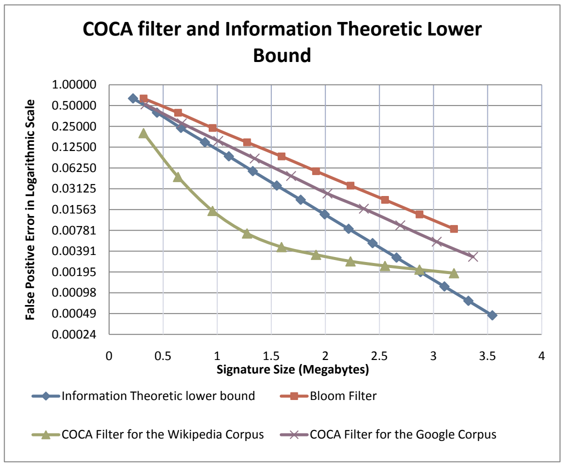


Fig. 4. The Comparison of Information Theoretic lower bound with COCA filters over two different corpora

Table 1. Comparison of the average false positive error of COCA filter over two wikipedia corpora with different sizes

m/n	1	2	3	4	5	6	7	8	9	10
Wikipedia(6500)	0.20075	0.04450	0.01338	0.00509	0.00280	0.00203	0.00159	0.00131	0.00108	0.00096
Wikipedia(2000)	0.20001	0.04643	0.01497	0.00706	0.00448	0.00347	0.00278	0.00239	0.00211	0.00188

One area of concern is whether in COCA Filters the average false positive error decreases at the cost of having many bloom filters with low false positive error and many with high false positive error (i.e. having a bimodal distribution). Figure 3 illustrates a comparison between the distribution of the false positive error in the COCA Filter and the conventional Bloom filter for $\frac{m}{n} = 2$, 4 over the Wikipedia and Google corpora. In each graph the left curve corresponds to the COCA filter and the right curve corresponds to the conventional Bloom filter. In all graphs, in both curves, by increasing the distance from the average, the frequency of documents decreases rapidly. It shows that there are only a few documents with a false positive error significantly greater than (or smaller than) the average false positive error. It can be seen that in the Wikipedia corpus which has higher correlation even the worst false positive error of the COCA filter method is significantly better than the best false positive error of the conventional Bloom filter method while in the Google corpus this property does not hold. Moreover, in the Wikipedia corpus the deviation of the COCA filter curve from the average is much smaller than its corresponding Bloom filter curve while in the other corpus this is not easily observable.

Another interesting comparison is between the COCA filter and the information theoretic lower bound on the three corpora as suggested in [7]. In other words we want to compare our method in terms of space efficiency with the best possible randomized data structure which does not utilize the co-occurrence probability of the words. Figure [4] illustrates this comparison. Note that the y-axis is the average false positive error in logarithmic scale in order to demonstrate the difference more clearly. While the COCA filter for the Google corpus never beats the information theoretic lower bound, the COCA filter for the Wikipedia corpus beats it in most of the cases by a significant margin. Note that as the false positive error gets closer to zero the distance between the curves shows a smaller difference. Interestingly as the correlation among the terms of the corpus gets stronger the rate of the decrease in false positive error tends to be hyper-exponential (as in Wikipedia corpus) rather than exponential (in Google Corpus) but as the index size increases the improvement rate decreases until it becomes very close to Bloom filter. This shows that for these applications where the elements of the corpus are highly correlated, utilizing the extra information about this correlation can be very valuable.

5 Conclusion and Future Work

In this paper the problem of false positive error of Bloom filters has been addressed and a novel technique to reduce the false positive error is proposed. The effectiveness of this approach was evaluated by conducting two experiments and our experimental results showed that up to 21 times improvement in false positive error or equivalently up to 75% reduction in space is achievable. Although this improvement is surprisingly good it is important to note that this technique is very sensitive to the correlation among the terms of the documents in the corpus.

In the current definition of the similarity function the size of each posting list can not affect the similarity of any two words as long as the ratio of the intersection and the union of their corresponding posting list is the same. It would be interesting to investigate similarity functions which are sensitive to the size of the posting lists as well.

Finding the information theoretic lower bound for the minimum number of bits required for a Bloom filter, given the extra information of the co-occurrence probability of each pairs of the members of the universe is another avenue for research.

More recently a particular type of memory called ternary content addressable memory (TCAM) was used to replicate a set of Bloom filters in order to solve the subset query problem for small sets [18]. Another potential opportunity is to explore the possible positive effect of COCA filters in areas where TCAM is used as a group of Bloom filters.

References

1. `http://www.google.com/programming-contest`, 2002. [Accessed January-2011].
2. `http://www.wikipediaondvd.com/site.php`, 2007. [Accessed January-2011].
3. `http://schools-wikipedia.org`, 2008. [Accessed January-2011].
4. `http://en.wikipedia.org/wiki/Wikipedia:Words_per_article`, 2009. [Accessed January-2011].
5. B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13:422–426, July 1970.
6. P. Bose, H. Guo, E. Kranakis, A. Maheshwari, P. Morin, J. Morrison, M. H. M. Smid, and Y. Tang. On the false-positive rate of bloom filters. *Inf. Process. Lett.*, 108(4):210–213, 2008.

7. A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. In *Internet Mathematics*, pages 636–646, 2002.
8. A. Z. Broder. Identifying and filtering near-duplicate documents. In *CPM*, pages 1–10, 2000.
9. A. Z. Broder. Min-wise independent permutations: Theory and practice. In *ICALP*, page 808, 2000.
10. J. Buhler and M. Tompa. Finding motifs using random projections. *Journal of Computational Biology*, 9(2):225–242, 2002.
11. B. Carterette and F. Can. Comparing inverted files and signature files for searching a large lexicon. *Inf. Process. Manage.*, 41(3):613–633, 2005.
12. M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, 2002.
13. B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal. The bloomier filter: an efficient data structure for static support lookup tables. In *SODA*, pages 30–39, 2004.
14. S. Cohen and Y. Matias. Spectral bloom filters. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 241–252, New York, NY, USA, 2003. ACM.
15. C. Faloutsos and S. Christodoulakis. Signature files: an access method for documents and its analytical performance evaluation. *ACM Trans. Inf. Syst.*, 2:267–288, October 1984.
16. L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Trans. Netw.*, 8:281–293, June 2000.
17. B. Georgescu, I. Shimshoni, and P. Meer. Mean shift based clustering in high dimensions: A texture classification example. In *ICCV*, pages 456–463, 2003.
18. A. Goel and P. Gupta. Small subset queries and bloom filters using ternary associative memories, with applications. *SIGMETRICS Perform. Eval. Rev.*, 38:143–154, June 2010.
19. T. H. Haveliwala, A. Gionis, and P. Indyk. Scalable techniques for clustering the web. In *WebDB (Informal Proceedings)*, pages 129–134, 2000.
20. J. Li, B. Loo, J. Hellerstein, M. Kaashoek, D. Karger, and R. Morris. On the feasibility of peer-to-peer web indexing and search. In *Peer-to-Peer Systems II*, volume 2735 of *Lecture Notes in Computer Science*, pages 207–215. Springer Berlin / Heidelberg, 2003.
21. J. Matousek. On restricted min-wise independence of permutations, 2002.
22. J. Mullin. Optimal semijoins for distributed database systems. *Software Engineering, IEEE Transactions on*, 16(5):558–560, May 1990.
23. J. K. Mullin and D. J. Margoliash. A tale of three spelling checkers. *Softw. Pract. Exper.*, 20:625–630, June 1990.
24. Z. Ouyang, N. D. Memon, T. Suel, and D. Trendafilov. Cluster-based delta compression of a collection of files. In *WISE*, pages 257–268, 2002.
25. A. Pagh, R. Pagh, and S. S. Rao. An optimal bloom filter replacement. In *SODA'05*, pages 823–829, 2005.
26. P. Reynolds and A. Vahdat. Efcient peer-to-peer keyword searching. In *Lecture Notes in Computer Science 2672*, page 2140, 2003.
27. M. Saks, A. Srinivasan, S. Zhou, and D. Zuckerman. Low discrepancy sets yield approximate min-wise independent permutation families. *Information Processing Letters*, 73(1-2):29–32, 2000.
28. E. H. Spafford. Opus: Preventing weak password choices. *Computers & Security*, 11(3):273–278, 1992.
29. C. Yang. Macs: music audio characteristic sequence indexing for similarity retrieval. In *Applications of Signal Processing to Audio and Acoustics, 2001 IEEE Workshop on the*, 2001.
30. J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2), 2006.