Journal of Discrete Algorithms 18 (2013) 63–74

Contents lists available at SciVerse ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

# A novel approach for leveraging co-occurrence to improve the false positive error in signature files [☆]

Pedram Ghodsnia [*], Kamran Tirdad, J. Ian Munro, Alejandro López-Ortiz

*Cheriton School of Computer Science, University of Waterloo, Canada*

ARTICLE INFO

*Article history:*
Available online 6 November 2012

*Keywords:*
Information retrieval
Bloom filter
Signature files
Locality sensitive hash functions

ABSTRACT

Signature file is a well-studied method in information retrieval for indexing large text databases. Because of the small index size in this method, it is a good candidate for environments where memory is scarce. This small index size, however, comes at the cost of high false positive error rate. In this paper we address the problem of high false positive error rate of signature files by introducing COCA filters, a new variation 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 in real document collections we can reduce the false positive error by up to 21 times, for the same index size. It is also shown that in some extreme cases this technique is even able to completely eliminate the false positive error. COCA filters can be considered as a good replacement for Bloom filters wherever the co-occurrence of any two members of the universe is identifiable.

## 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 [36]. 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 [24,30]. 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 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.

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

* Corresponding author.
*E-mail address:* pghodsnia@uwaterloo.ca (P. Ghodsnia).

1570-8667/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jda.2012.10.006

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 Section 5 discusses alternative techniques to solve the problem. Finally we conclude our work in Section 6.

## 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 [13,17,36]. 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 [13] 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 [7]. 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 $h_1, h_2, \dots, h_k$ with ranges of $1, \dots, m$. All the bits are initially set to zero. These hash functions can be interpreted as uniform random number generators over the range of $1, \dots, m$. For every element of the set, say $x$, the bits $h_i(x)$ for $(1 \leqslant i \leqslant k)$ 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 $1 \leqslant i \leqslant k$ 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 [27], as a means of succinctly storing a dictionary of unsuitable passwords for security purposes [32], to speed up semi-join operations [26], for Web cache sharing [18] and in many other areas. In order to support multi-sets, Cohen and Matias introduced spectral Bloom filters [16]. Chazelle et al. in [15] introduce a similar data structure which is called a Bloomier filter in order to approximate functions.

Bloom [7] proved that the false positive probability of the Bloom filter is about $f = (1 - \frac{e^{-kn}}{m})^k$. Recently Bose et al. in [8] 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. [11] 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 [29] 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 [14]. The LSH algorithm has been used in numerous applied settings from bio-sequence similarity search [12] to audio similarity identification [35] and many other areas [19,22,28]. Min-wise independent permutations is a locality sensitive hashing scheme for a collection of subsets with the similarity function defined as follows: