Review of a technical paper

A Fixed-Size Bloom Filter for Searching Textual Documents

Written by:

Naveen Venkat	2015A7PS078P	nav.naveenvenkat@gmail.com
Harsh Bansal	2015A7PS075P	harshbansal97@gmail.com

Abstract

The paper takes an experimental approach to quantitatively prove that fixed length bloom filters are “better” than other established methods used in searching textual documents. It also describes a method to find the filter size for a database of textual documents given a fixed false drop rate.

The research paper discussed in this document, is based on developing a modified fixed-length bloom filter. Contrary to the usual approach, the filter discussed here is constructed on the basis of a given false drop rate and on the distribution of the number of different words per document over the database.

The method is based on an estimation technique so that the actual distribution need not be known and the method is independent of the form of the distribution. Fixed-length filters based on this approach produce empirical false drop rates equivalent to the desired theoretical false drop rates.

To check the practical validity of the method discussed, it has been applied on two independent databases. The theoretical and practical accuracy of the method has been compared and based on the results mathematically valid conclusions have been drawn.

Summary

1. Introduction

This exposition is a review of the paper written by-

M. A. Shepherd, W. J. Phillips and C. K. Chu

Searching textual documents for keywords in a given database is a time consuming process, especially if the documents are large in size and number. Several methods have been developed to overcome this problem. The method used in this paper is the signature file method which

permits boolean queries on a signature file which requires significantly less storage overhead. One such approach is using bloom filters.

2. Bloom Filters

Bloom filters reduce query execution time, by providing an additional layer (a 'filter') which filters out query depending upon likeness of the existence of the key (a word, in this case) in the database. A simple bloom filter is implemented as an array of bits as follows:

0	1	1	0	0	0	1	0	0	1	1	0	1	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Querying is with the help of transformation functions which map each word to a bit in the filter. If any of the bit corresponding to the transformation is 0, the word doesn't exist in the document. This improves query execution speed because it filters out all those queries whose words don't exist in the database and searches for only those which can exist.

However, bloom filters are prone to false positive results, because multiple insertions can set certain bits which might evaluate a query to positive, hence indicating its existence in the database. To investigate further into this the authors have considered two different textual databases (CACM & OON). Each has several documents from which certain keywords are extracted and inserted into the corresponding filters.

The theoretical probability of a false positive is calculated as follows:
Let the size of the bloom filter be **b** bits. Let **w** be the number of different words per document inserted in the bloom filter and **t** transformations be used per insertion. Assuming that each of the t transformations are uniformly random (the probability of yielding any of the b indices is same) we can draw the following observations:

- Probability of a bit being set by one transformation = 1/b
- Probability of a bit not being set by one transformation = 1 - 1/b
- Probability of a bit not being set by t transformations = $(1 - 1/b)^t$ **(1)**
  This is the probability that a bit will not be set due to insertion of a word into the filter.
- Hence, for w words per document, the probability is = $(1 - 1/b)^{tw} \cong \mathbf{e^{(-tw/b)}}$
- Probability of a bit being set after w insertions = **Pset** = $1 - e^{(-tw/b)}$ **(2)**

A false positive can occur for a query when all the corresponding t bits yielded by the transformations are set. Thus, after inserting w keys, we get
$P_e = (Pset)^t$, where P_e is the probability of a false positive (or false drop rate).
If we have **'a'** keys in an ANDed query then the false positive rate = $(Pset)^{ta}$ **(3)**
It has been shown that the optimum number of transformations is when Pset = ½ (half of the bits are set to 1). Thus the theoretical false drop rate (or probability of a false positive) is
$P_e = (Pset)^t = (1/2)^t$ for a single query. (4)

3. Filter Size

The appropriate theoretical filter size can be calculated from the equations above. Approximating $(1-1/b)^{tw}$ to $e^{-tw/b}$ under the assumption that b >> 1 and tw >> 1 we get:

$$b = tw/\ln 2 \quad (5)$$

This theoretical value assumes that there are w words per document. However, practically each document would have different number of words. This leads us to the next observation.

4. False drop rate

The optimum false drop rate is rarely obtained in practice, because of the assumption that there are w words per document. To overcome this issue, we consider each document to have different number of words (say the i^{th} document has $\mathbf{w_i}$ number of words), and find an expected value of false the drop rate:

$$Pe = E(1-K^w)^t$$

where

$$K = (1-1/b)^t$$

Thus the expected value is:

$$Pe = \left[\sum_{i=1}^{n} (1-K^{w_i})^t\right] \Big/ n \quad (6)$$

For optimum case, $P_e = (1/2)^t$. This results in:

$$(1/2)^t = \left[\sum_{i-1}^{n} (1-K^{w_i})^t\right] \Big/ n \quad (7)$$

Now a better estimate can be given for the filter size as follows:

$$b = 1/[1-e^{(\ln K)/t}] \quad (8)$$

Experimentally, false drop rate for a query has been calculated as:

Fd/(D-Dq) (9)

where, Fd is the number of false drops for a query, D is the total number of documents and Dq is the number of documents in which the query term actually occurred.

Using the equations above, the size of the bloom filter can be estimated and hence can be constructed on the basis of a given false drop rate and on the distribution of the number of different words per document over the database.

5. Experiments Conducted

Three experiments were conducted:

a. to determine if desired theoretical false drop rates can be obtained experimentally using distribution-based fixed length Bloom filters.

b. to compare the storage requirements and false drop rates of distribution-based fixed-length filters with those that assume a uniform distribution of the number of different words per document, and with those of variable-length filters.
c. to compare the speed of searching a database using variable-length filters with that of using the distribution-based fixed-length filter.

The filter was constructed by inserting certain keywords from each document into the respective bloom filters, introducing certain noise words as well. Each query given to the filter was a non-noise word selected randomly from the appropriate database.

6. Mathematical Observations

- Fixed-width retrieval method with mean number of words, while having a similar space complexity, has very high false drop rate (20 to 30 times of the theoretical value).
- Fixed-width retrieval method with maximum number of words, while having very low false drop rate (0.001 times of the theoretical value), has a high space complexity (5 to 6 times of proposed method).

Therefore based on tables 3 and 4 it is evident that for modelling the solution with space limitations and a bound on the false drop rate, the proposed method will surely perform better than the others.

- Variable sized filters might seem like a valid option but with their associated time overhead (approximately 1.5 times the proposed method) as seen in tables 5 and 6 they perform poorly for a fast query execution.

Critique

The abstract of the paper was quite clear and precise about the practical problems and the methodology adopted to minimize them.

The introduction was well structured and paved a proper path for the reader to understand why the proposed structure of the bloom filter was necessary. The huge number of references implies that an extensive reading has been done by the authors. The introduction also gave an abstract of the mathematical approach used in the paper.
The mathematical formulas used/derived require only basic understanding of probability.
Since a comparison based approach was taken the data provided eliminated the need for the reader to understand the intricate working of other approaches used like variable sized bloom filters.

There were a few shortcomings in the mathematical analysis :-

1. The definition of “non-noise keywords” is a bit unclear.

2. The authors failed to mention how the ANDed query is executing. If each query in the ANDed request is executing sequentially, then this could affect the probability of false positives, due to the influence of one query over another.

   For instance, consider two queries having p bits common in the filter. If the first query yields a false positive, then we know that those p bits were also set to 1. Thus, the calculation of the probability of false positives of the second query should not include these p bits. Thus it would be $(Pset)^{(t-p)}$.

3. In this paper $(1— 1/b)^{tw}$ has been approximated to $e^{(-tw/b)}$. This is only true for large b and w. Thus, the mathematics used in this paper is not valid for small databases.

4. Both the databases used were of similar nature. A better experiment could have used a variety of documents over the

5. The false drop rate was calculated as Fd/(D-Dq), however the authors failed to mention the approximation used in boundary case scenario (D=Dq).

6. The only method which was better than distribution based fixed size bloom filters is variable sized bloom filter which the authors have pointed out have a large time complexity. But they have said so in a very abstract way using the concept of associative memory.
   A proper explanation/reference is essential to understand their claim.

7. The authors haven't mentioned anything about the dynamic updation of the filter which is a major issue in any practical application.

Conclusion

As we can see from the mathematical observations, if we take the cumulative effect of all the factors like time overhead, space overhead and a false drop rate close to the desired value we can surely say that the paper has proved that distribution based fixed width bloom filter can be surely used for searching textual documents with a given false drop rate.

References

1. Shepherd et. al. A Fixed-size Bloom Filter for Searching Textual Documents
   Link- comjnl.oxfordjournals.org/content/32/3/212.full.pdf