CSCI 8970

Advanced Topics in Data Intensive Computing

Bloom Filters - Project Report

10/17/2022

Team Members

Nruthya Kadam
Rutuja Talekar
Swarali Gujarathi

Introduction:

A bloom filter is a space-efficient and time-efficient data structure for testing whether an element is a member of a set. Bloom filter can give False positives (that is, queries might incorrectly recognize an element as a member of the set) but never false negatives. So, it is a probabilistic data structure.

Deleting elements from the filter is not possible because if we delete a single element by clearing bits at indices generated by k hash functions, it might cause the deletion of a few other elements.

Constructing Bloom Filters:

- Consider a set of n elements. Bloom filters describe membership information of the set with array length *m.*
- Each of the *k* hash functions, $h_1, h_2, ..., h_k$ takes in a key (can be a string/url) and generates a value between 0 and m-1, $h_i: X \rightarrow \{0..(m-1)\}$.
- Initially, all the bits of m-bit array are set to zero.
- If a_i is member of the set, in the resulting Bloom filter, all bits obtained corresponding to the hashed values of a_i are set to 1.

*

Testing membership:

- Testing for membership of an element is equivalent to testing that all corresponding bits of the Bloom filter are set.
- Put the element through all the hash functions.
- If all the indices come out as 1 then we can say that the element exists in the set. Every index must be 1.
- A stabilized FP rate is lesser than 0.1%

Probability of False positivity:

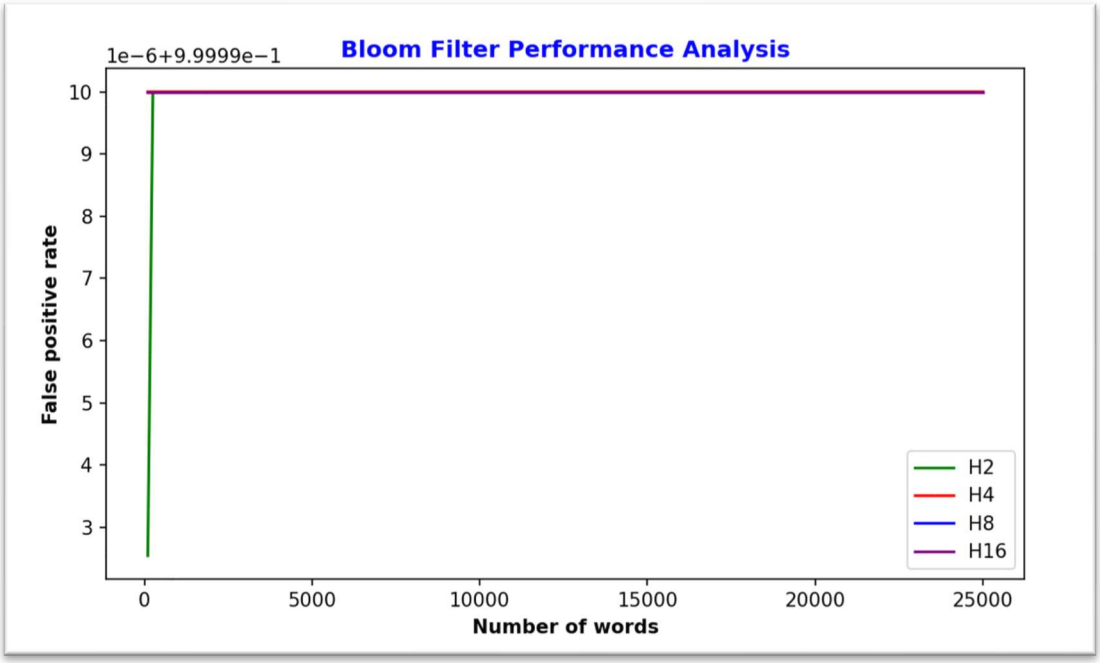
$$\mathbf{P(FP)} = \left(\mathbf{1} - \mathbf{e}^{-\frac{\mathbf{kn}}{\mathbf{m}}}\right)^{\mathbf{k}}$$

where:

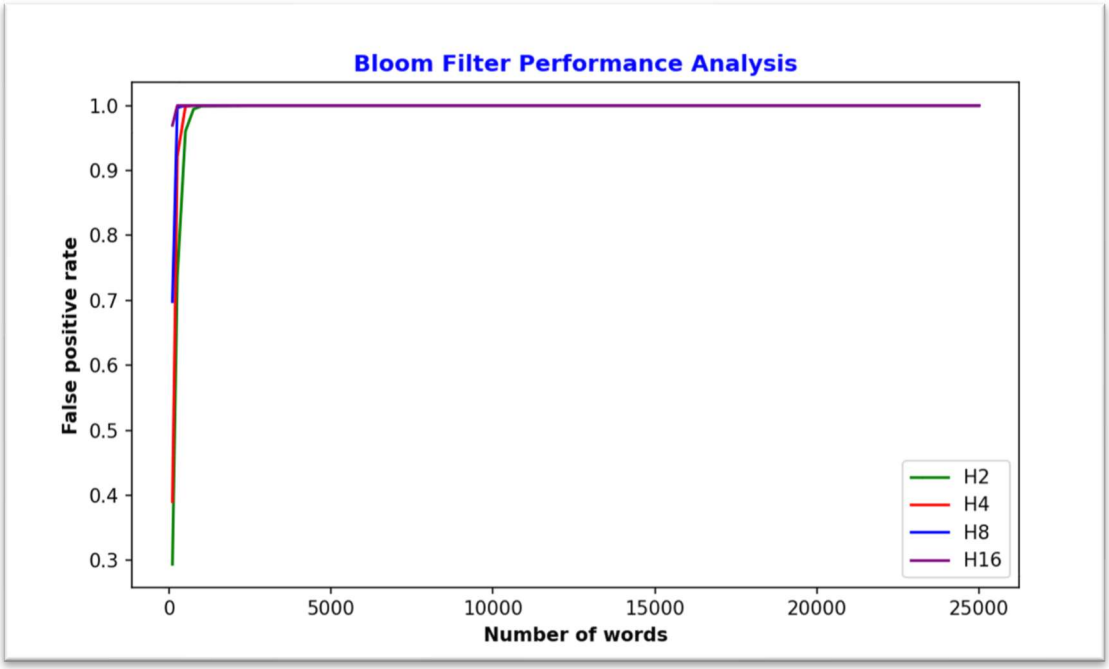
e: Euler's number
n: number of keys/elements inserted
m: filter size
k: number of hash functions
Optimal number of hash functions: $\mathbf{k} = \frac{\mathbf{m}}{\mathbf{n}} \ \mathbf{ln\,2}$

Performance Experiments:

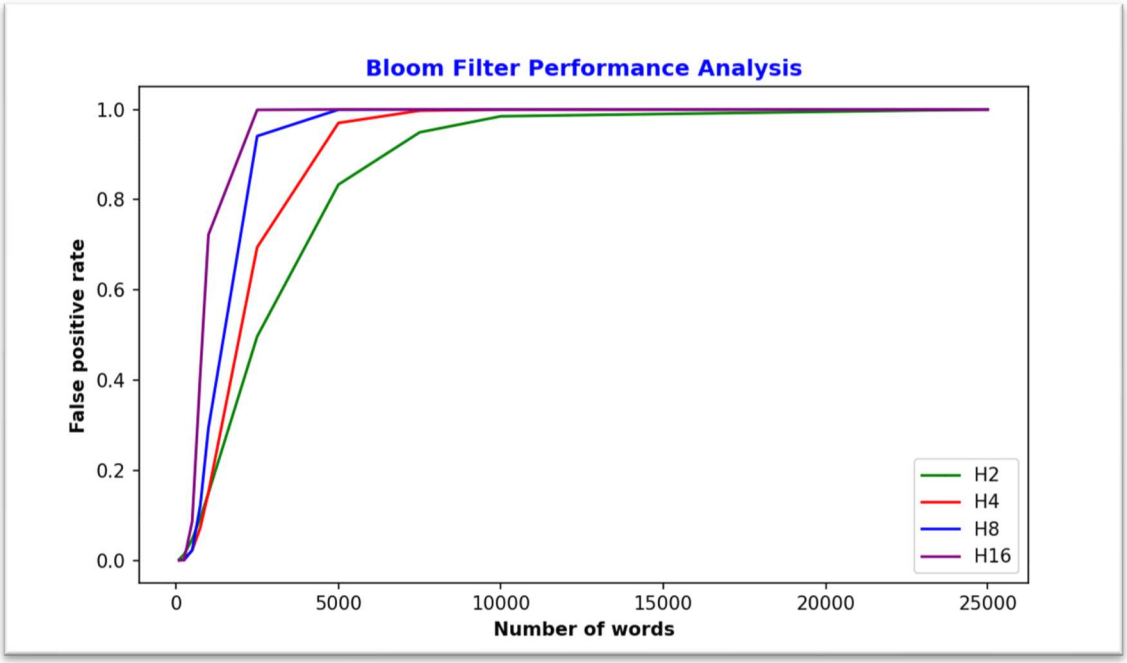
1) **Bloom filter size (m) = 16**



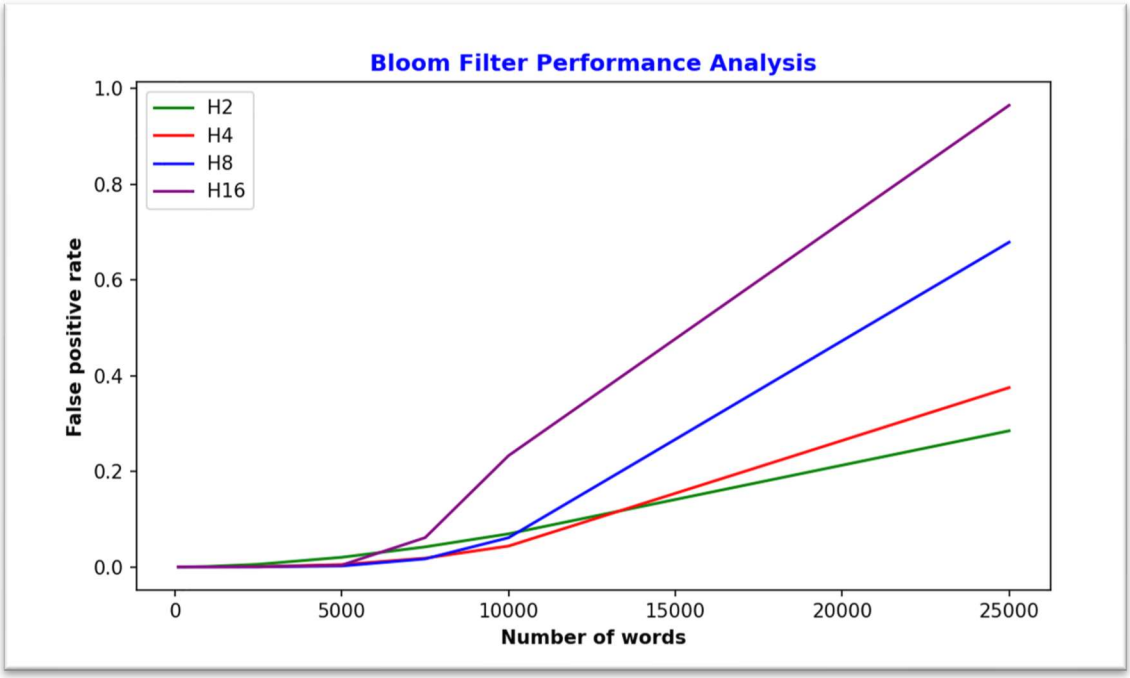
2) **Bloom filter size (m) = 256**



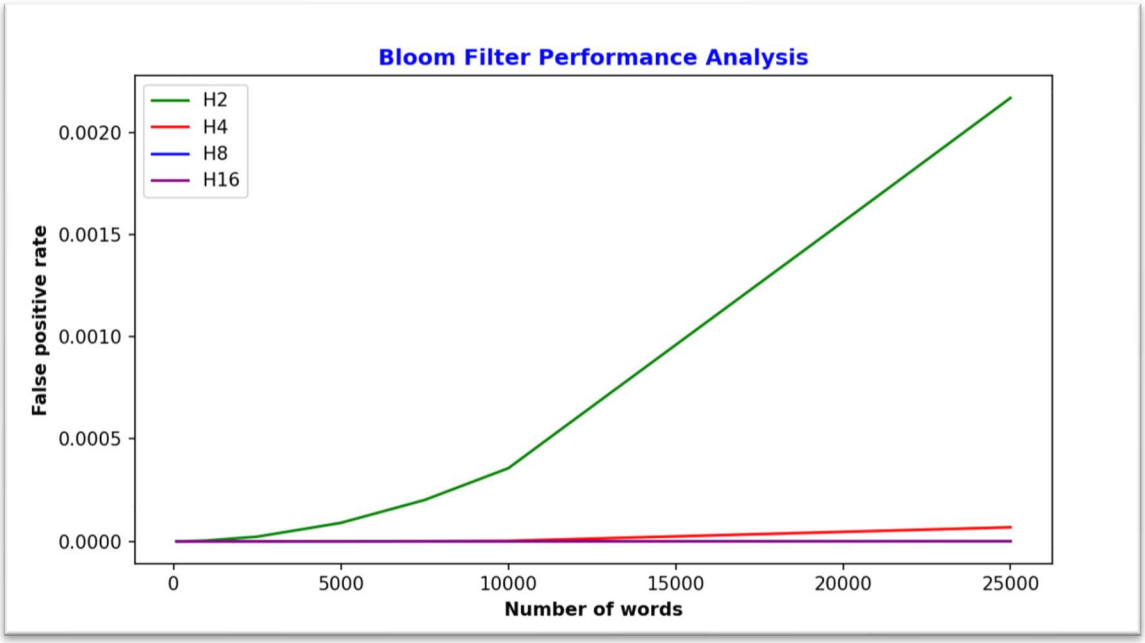
3) **Bloom filter size (m) = 4096**



4) **Bloom filter size (m) = 65536**



5) **Bloom filter size (m) = 1048576**



Experimental Analysis:

According to our study, the false positive rate is higher when the bloom filter size is set to small. Increasing the number of hash functions used can help us achieve better results (which is a lower FP rate), but there is a point at which an increasing number of hash functions worsens the FP rate. This might be because we must set a more significant number of bits to 1 in the bloom filter, and more bits per element results in an overall lower false positive curve.

After achieving the perfect tradeoff between several hash functions and the bloom filter, the false positive rate stabilizes (FP rate less than 0.1%).

Code:

Please find the code at - https://github.com/RutujaTalekar/Bloom-Filter

Output:

After the performance study, we saw that the false positive rate is the most stable for the configurations below.

N = 10,000
m = 1048576
k = 8

Hence, we used 26 words for the membership test on the bloom filter initialized with the above configurations. As per the results below, the words are categorized correctly, and there are no false positives for this testing data.

```
C:\Users\ritu1\Documents\Fall 2022\Data intensive\Bloom-Filter>python bf.py input.txt
Total number of words -  10000
Total number of hash functions used - 8
Size of bit array | bloom filter -  1048576
Projected false positive rate -  8.476704293309944e-10

The following item could be a member - aaron
The following item could be a member - ab
The following item could be a member - aback
The following item could be a member - abacus
The following item could be a member - abba
The following item could be a member - abbas
The following item could be a member - abbot
The following item could be a member - abbott
The following item could be a member - abbreviate
The following item could be a member - abbreviated
The following item could be a member - abbreviation
The following item could be a member - abbreviations
The following item could be a member - abdication
The following item could be a member - abduct
The following item could be a member - abduction
The following item could be a member - abel
The following item could be a member - aberdeen
The following item could be a member - aberrant
The following item could be a member - aberration
The following item could be a member - abetted
The following item is not a member - icecream
The following item could be a member - world
The following item is not a member - cutie
The following item is not a member - Rutuja
The following item is not a member - Swarali
The following item is not a member - Nruthya

The bloom filter bit array has been reset bitarray()
```