Bloom Filters

Khudayar Farmanli
Politecnico Di Torino
Data Science and Engineering
Computer Aided Simulations and Performance Evaluation
Torino, Italy
s276310@studenti.polito.it

***Abstract*—The objective of this homework is the application of Bloom filters technique for storing data and measuring its performance to understand how it generates results empirically and how much it differs from theoritical ones and we compare this technique with others to see its efficiency.**

I. Input Parameters

- Dataset - contains various characteristics related different movies, we just take the rows contain "IT" as region then eliminate all the other things and take the "title" part.
- m - the number of elements in final set which is 176339 in our particular case, there were more but when I create set it directly eliminated the duplicates, probably more pre-processing is needed but for our purpose it is not crucial.
- b - predefined number of bits for further processing, in our case [19, 20, 21, 22, 23]

II. Output parameters

- k - optimal number of hash functions, calculated with below formula:

$$k_{opt} = \frac{n}{m} log(2)$$

- Empirical Probability of False Positive for Bloom Filters, calculated based on below formula:

$$Pr(FP) = (\frac{NumberOf"1"bits}{2^b})^k$$

- Theoritical Probability of False Positive for Bloom filters, calculated based on below formula:

$$Pr(FP) = (1 - e^{\frac{-km}{n}})^k$$

- Probability of False Positive for Fingerprint sets - calculated based on below formula:

$$Pr(FP) = \frac{m}{n}$$

  where $n = 2^b$.
- Probability of False Positive for Bit String Arrays - calculated based on below formula:

$$Pr(FP) = \frac{NumberOf"1"bits}{2^b}$$

III. Main data structures

For developing this simulation I have used python sets, lists and numpy arrays. I stored titles data into python set, then I generated hash functions for Bloom filters and stored them into list. I used numpy arrays in some places for making things easier to work with which were generating arrays based on built-in functions of library.

IV. Results

In this study we observed how Bloom filters performs in terms of false positive probability, we saw how much empirical and theoritical results differ and what is the performance of Bloom filters with respect to Fingerprinting and Bit strings. The results had been demonstrated below:

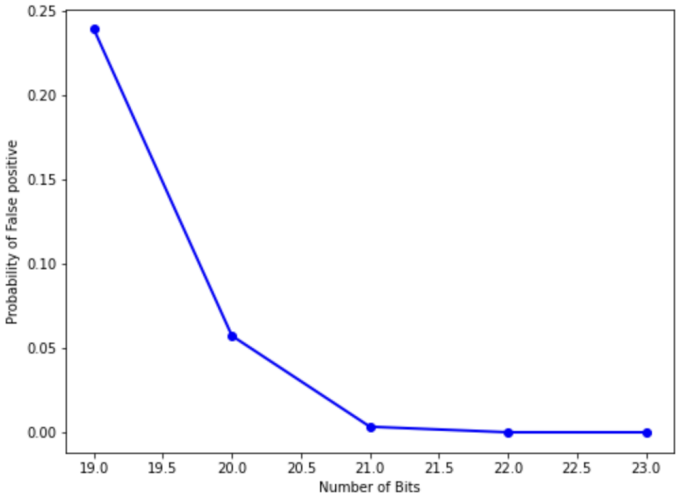

Fig. 1: The graph shows how different number of bits affected performance of Bloom Filters in terms of False Positive probability empirically.

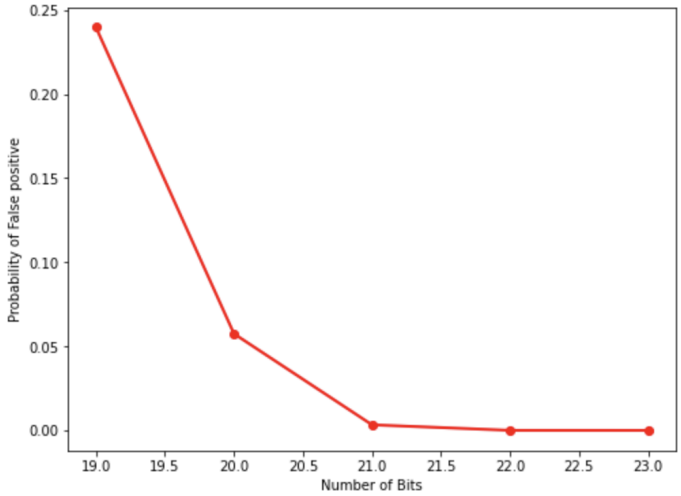

Fig. 2: The graph shows how different number of bits affected performance of Bloom Filters in terms of False Positive probability theoritically.

As it can be easily seen from the both above graphs, the increase in number of bits decrease the probability of false positive and the results provided by both method are very similar, that is why I used different graphs for them, otherwise they overlap.

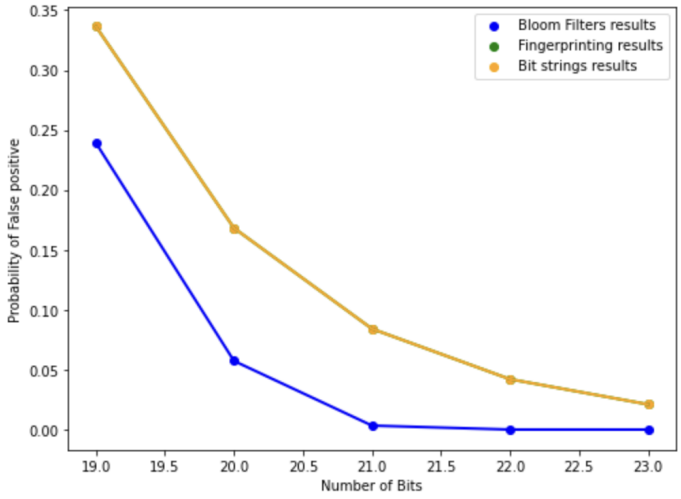

Fig. 3: Comparison of Bloom Filters, Fingerprinting and Bit strings based on pre-defined bits in terms of False Positive probability.

From above graph we see that Bloom Filters provide better results with respect to others, Fingerprinting and Bit strings provide similar results, that is why, they overlap and we cannot observe this phenomenon with human sight from line graph.

V. Conclusion

At the end, I obtained below conclusions based on my experiments:

- Empirical and Theoritical results are similar for Bloom filters based on defined bits, in terms of False Positive probability.
- Bloom filters performs better than Fingerprinting and Bit string array and provide less False Positive probability for any number of previously defined bits, so it is more reliable method.