Antiplagiarism Poem System(lab 17)

[redacted]

Abstract— This report is related to the simulation to compare different methods: fingerprint, bit string array and bloom filter in terms of storage and false positive probability.

I. STRUTURE OF SIMULATOR

In the following, we address different parts of the simulation.

A. Assumption:

For simplicity, we assume that the software is able to detect sentences taken from 'La Divina Commedia' by Dante Alighieri. The text with size about 0.5 MB is downloaded and used for the simulation. The number of total words is roughly 96000 and distinct words is about 14377 with sizes about 5.9 MB and 1.4 MB, respectively. We produce sentences with certain length of words from these words which is 6.

B. Preliminary study:

*Inputs: each method uses its own inputs.

- Length of each sentence
- Number of bits for storage

*Outputs:

1. For fingerprint, there are sentences with length 6 words. We compute their fingerprint with nonsufficient bits to find Bexp as follows:

$$b_{nonsufficient} = log_2\, m$$

$$b_{sufficient} = log_2\, \frac{m}{\varepsilon}$$

$$Total\ storage\ (bit) = m \times log_2(\frac{m}{\epsilon})$$

Where b is number of bits and m is number of elements to insert. Outputs are:

- Number of distinct sentences created by poem's words and its size
- Number of integers created by fingerprint method and its size
- Bexp
- Bteo

2. For bit string array, there are two scenarios, experimental by simulation and theorical by formula as follows:

$$Total\ storage\ (bit) = (\frac{m}{\epsilon})$$

So, outputs are:

- Total storage of bit array
- False positive probability
- Related graphs

3. For bloom filter, again two scenarios exist, experimental by simulation and theorical by formula as follows:

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

$$\epsilon_{opt} = 0.5^{\text{kopt}}$$

$$Total\ storage\ (bit) = m \times 1.44 \times log_2(\frac{1}{\epsilon})$$

So, outputs are:

- Performance graph
- k_{opt} for inputs storage
- ϵ_{opt} based on k_{opt}
- Total storage of bloom filter
- False positive probability
- Related graphs

*Sentences specification: we have set of 95866 distinct 6-word sentences with average size 133.4 bytes and total size 12.84 MB.

C. Fingerprint set:

In order to find Bexp, we use nonsufficient bits 17 as our starting point and then adding to this number to reach no collision condition in storing. To compute Bteo, we use the formula $m = 1.17\sqrt{n}$ when $p = 0.5$. Finally, using $b_{sufficient} = log_2\, \frac{m}{\varepsilon}$ to calculate probability of false positive for Bexp and the results are 34, 33 and 5.58e-6, respectively. The results show that 33 bits can generate a collision with p = 0.5 while 34 bits produces no collision at all and Bteo is a good approximation of Bexp.

D. Bit string array:

To evaluate false positive probability by simulation, first we create bit string array by computing hash number of each sentence and set its corresponding index to 1. After obtaining that, we calculate the false positive probability as total number of one of the array divided by total array size. The idea is that when we pick one sentence outside our set, how much it is possible, its index (computed by hash) is already 1 so the false positive event is generated.

From figure below, it can be seen that, results of simulation and theory get closer as the number of bits increases and both have a similar trend. Comparing to Bexp with 5.5e-6 for probability of false positive and size of 7200 KB, we reach about 5e-3 for probability of false positive which is acceptable by 23 bits and size of 1000 KB which 7 times less than that of Bexp. So, there is a trade-off between size and epsilon.

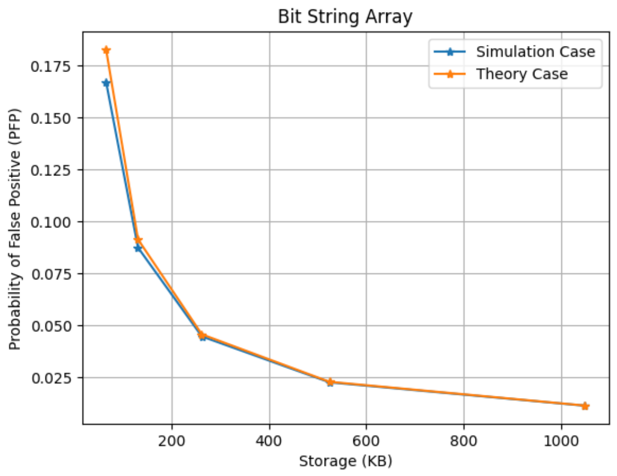

Fig1. Bit string array results comparison

E. Bloom filter:

To find optimal number of hash function for each size, we first compute the value of $k_{opt} = \frac{n}{m}\log(2)$ which is not an integer number. Afterward, we use performance plot (Fig.2) to decide which neighboring k produces less false positive probability.

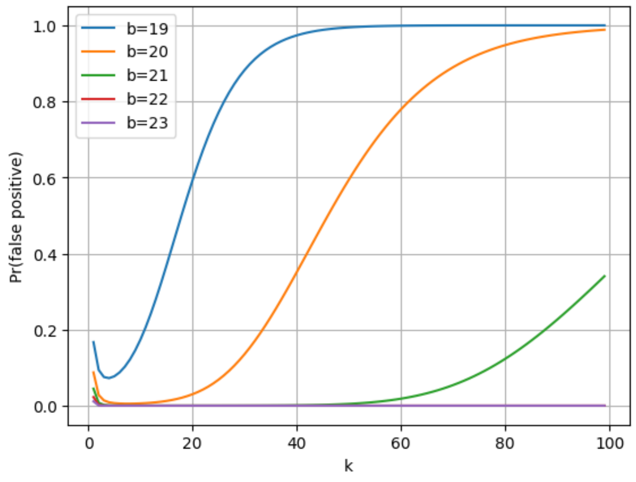

Fig2. Performance

Optimal Ks and their corresponding false positive probabilities are as follows:

X	2^19	2^20	2^21	2^22	2^23
k	4	8	15	30	61

Table1. Optimal Ks w.r.t. number of bits

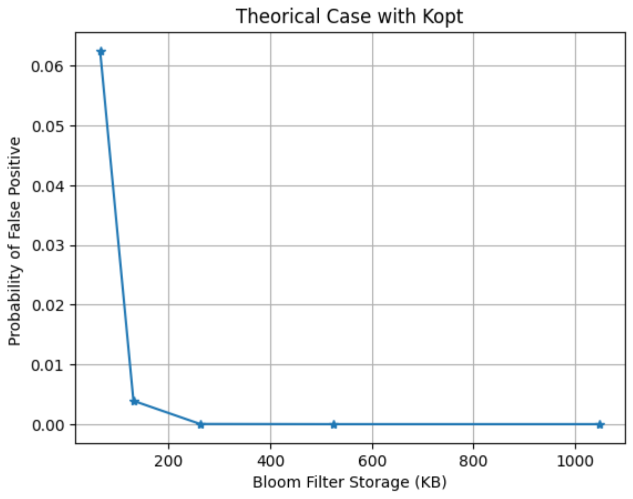

Fig3. Theorical false positive probability by kopt

To implement different hash functions, we generate k different numbers then add its string to each sentence to have k different sentences for every single sentence, find their hash values and set their corresponding index to 1.

To evaluate false positive probability by simulation, after generating hash values, we perform the same procedure as bit string array but, since we generate k hash numbers (i.i.d) and set their corresponding bits to 1, it is needed to multiply the probability k times to obtain the false positive probability.

Again, we see that both simulation and theory plots have a similar monotonicity with a small difference which vanishes as the number of bits increases. It shows that our simulation results are reasonable as fig4.

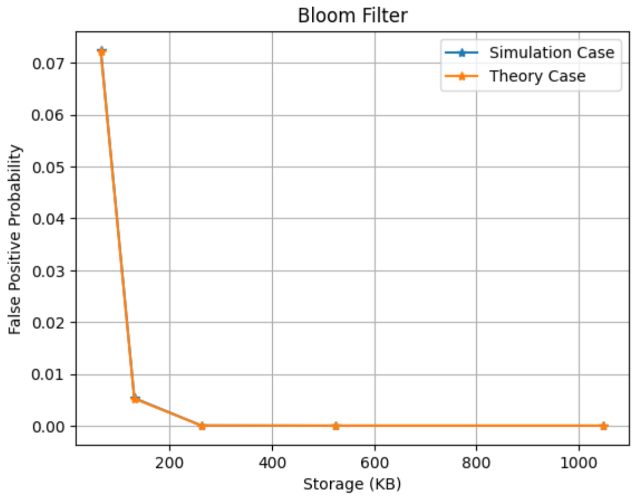

Fig4. Comparison between simulation and theorical results of bloom filter

In terms of optional case, if we are inserting one sentence at a time into the bloom filter, the accuracy of the formula will depend on the length and uniqueness of each sentence. If the sentences are very long and unique, the accuracy of the formula will be higher, as there will be fewer false positives. On the other hand, if the sentences are short and similar to one another, the accuracy of the formula will be lower, as there will be more false positives.

Overall, the accuracy of the formula will tend to improve as the number of elements inserted into the bloom filter increases, as the probability of false positives decreases with more elements in the filter. However, the formula may not be very accurate for small numbers of elements, especially if the number of bits and hash functions used is small.

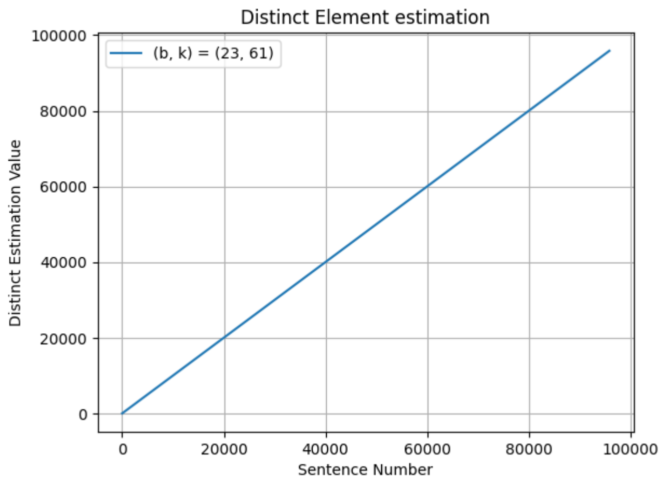

Fig5. Relationship between distinct estimation value and number of sentences

In figure below, there exists a tolerance in difference between actual number of stored sentences and estimated values in which this difference tends to decrease and be stationary when more sentences are inserted.

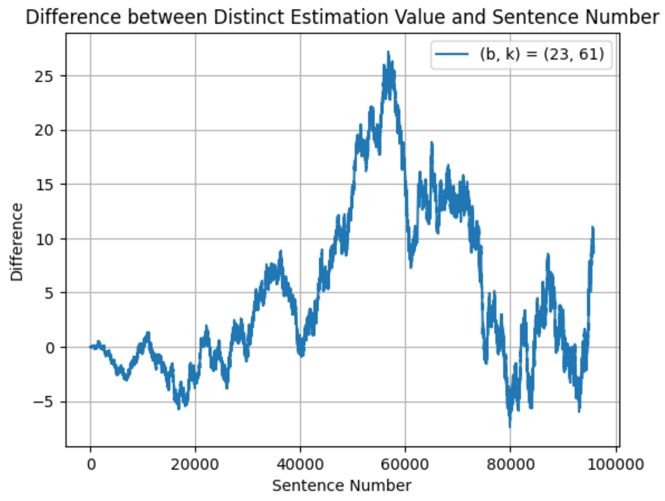

Fig6. Related differences when inserting one sentence at a time

F. Discussion:

Table2 illustrates that, on the one hand, we can obtain good false positive probability by fingerprint but it has highest size. On the other hand, bit string array has less size w.r.t. fingerprint but the probability is higher so there is a trade-off for choosing between them. Considering bloom filter, the sizes are less than which of formers and it also has smaller probabilities compering the two others. Consequently, bloom filter overcomes both fingerprint and bit string array in terms of size and false positive probability.

Data Structure	Memory (kB)	Prob. false positive
Set of sentences	12843	….
Fingerprint set with Bexp	7262	5.5 e-6
Bit string array, 2^19	65.6	0.1670
Bit string array, 2^20	131	0.0874
Bit string array, 2^21	262	0.0447
Bit string array, 2^22	524	0.0226
Bit string array, 2^23	1048	0.0113
Bloom filter, 2^19	65.6	0.0724
Bloom filter, 2^20	131	0.0053
Bloom filter, 2^21	262	2.7e-5
Bloom filter, 2^22	524	7.4e-10
Bloom filter, 2^23	1048	5.5e-19

Table2. Final results for different methods