Lab 17

Ali Ghasemi - S289223

Introduction

This simulation lab consists of different sections, each section will be analyzed and explained in order. The algorithms that are used will be explained and the outputs will be shown here as well.

Preliminary study and set of sentences

In this part of the simulation, the words, distinct words, verses, and the set of verses will be created. To make the verses, a sliding window method has been obtained. The inputs of the code are the Commedia text file and the size of the verses which will be the size of the sliding window. The output of the in this part will be the number of words, the number of distinct words and the number of verses and distinct verses. The size of the set of sentences is available in the code (in bytes). Also, the number of collisions are checked using the fingerprints for the set of sentences.

Fingerprint set

In this part of the simulation, first the sentences are created based on the sliding windows method with length of 6 and then for each word a fingerprint is created using the hash function provided in the description of the previous labs. Then after creating the list of fingerprints, Python's keyword, set() is used. The lengths of these two items (the list and the set) are subtracted from each other. If this value is more than zero, it means that a collision has happened and if not, it means that there's no collision.

In the first question, we go through the different values for b (which is the minimum number of bits) and we find the smallest value for b that does not lead to a collision, and we call it Bexp.

In my simulation, this value was equal to 34.

In the second part we find the theoretical value for b and we call it Bteo. We use the formula below to find this value:

$$m = 1.17 * \sqrt{n} \text{ and since } n = 2^b \text{ we can say that } b = Log_2^{m/1.17^2}$$

And the obtained value is equal to 33.

In the third question of this part, we must find the value of the false positive based on the value of Bexp which in my case is equal to 34.

Several formulas could be used, the below formula is one of them:

$$\varepsilon = \frac{m}{n} = m/2^b$$

The answer to the fourth question is, yes, Bteo is a good approximation of Bexp since the they are equal to 33 and 34 (respectively).

Bit string array

In the first question, we create bit arrays with the different lengths. The lengths of the values that are given in the question for the memory. We create the hash values for each verse and we the set the value of that bit equal to one and at the end we count the number of ones and divide it by the number inputs which will be the number of verses to calculate the values for probabilities of false positives for different values for memory. You can see the output plot below in figure 1.
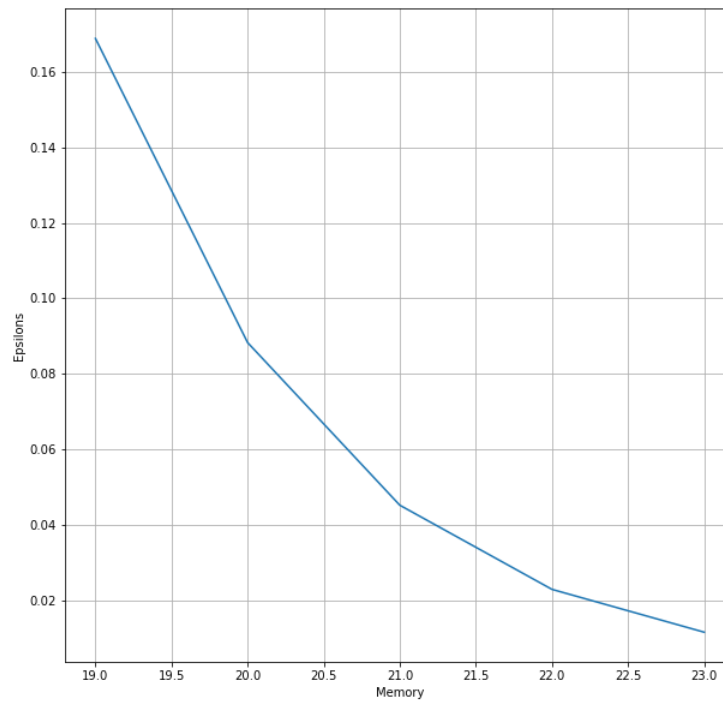
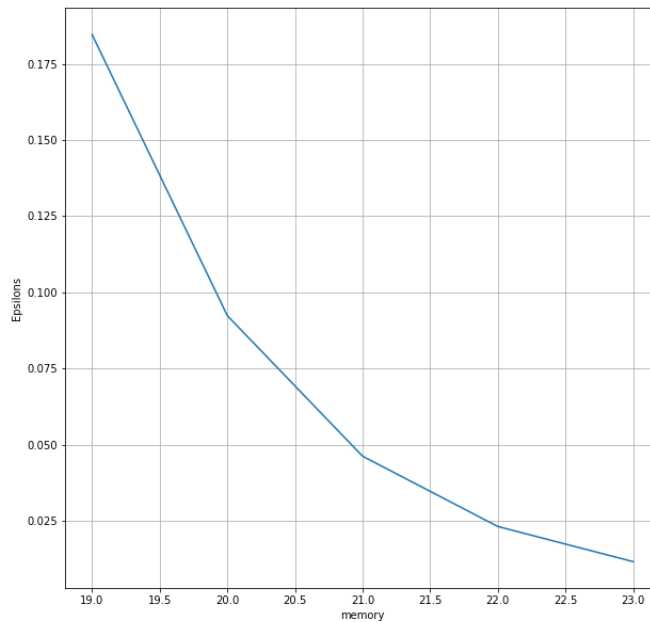For memory, it's 2 to the power of what you see on the plot for the X axis

Figure 1

In the second part, we find the values for the probabilities of false positives for each value for the memory based on the formula below:

$$\varepsilon = \frac{m}{n} = m/2^b$$

You can see the output plot below in figure 2.

For the third question, I plotted the theoretical values and also the values obtained from simulation in a graph in order to compare them and as you can see in figure 3, they are close to each other, so we can assume that the theoretical values are good approximations of the values obtained from simulation.

For memory, it's 2 to the power of what you see on the plot for the X axis

Figure 2

Bloom filters

In the first question, we need to find the right values for the number of hashes for different values for the memory. For that matter, we first need to find the optimal number of values for the number of hashes using the formula below:

$$\frac{n}{m} * \log(2)$$

This formula gives us real numbers, but we want integers, for that matter need to plot the performance plot and find the right number of integer values for the number of hashes.

You can see the performance plot in figure 5 (in figure 4, you can see how the value for the number of hashes change as we increase the memory.

Based on the values obtained from the graph and the values for the probability of false positive, we can find the right values for the number of hashes. The values are as shown in the table below:

Number of hashes	Memory
4	2 ** 19
8	2 ** 20
15	2 ** 21
30	2 ** 22
60	2 ** 23
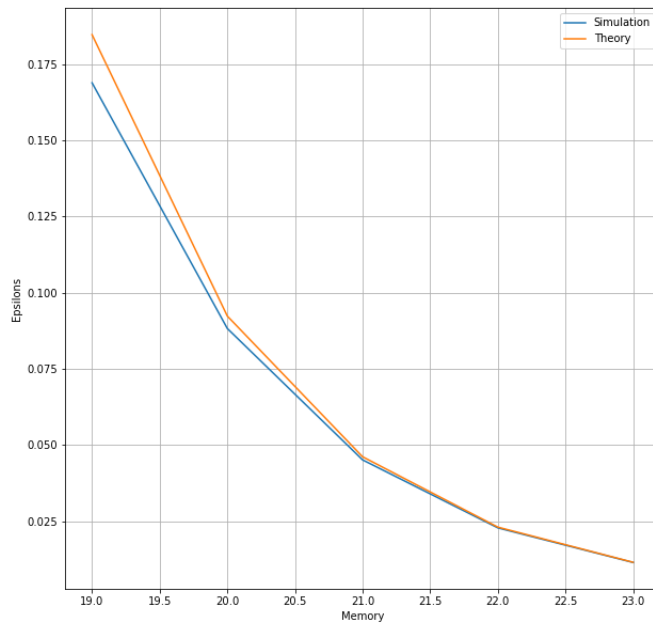
For memory, it's 2 to the power of what you see on the plot for the X axis

Figure 3

In The second part of the questions, we calculate the values for epsilon based on the formula below:

$$(1 - e^{-k*m/n})^k$$

The answer to the third question is that each time we add a number to the string to have different hash values.

In the fourth question we go through all the values for memory and calculate the values for false positive and we have figure 6 as the output:

In the last part we compare the outputs of the simulation and theory, and we can see that these values overlap. You can see the output in figure 7
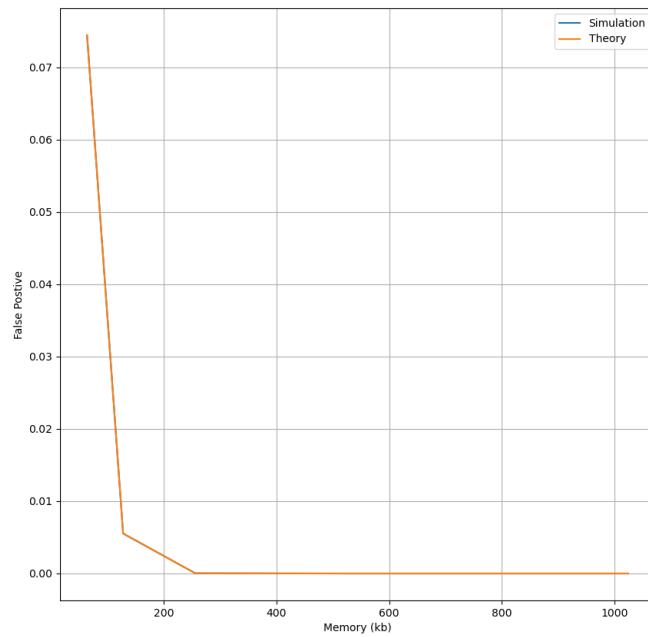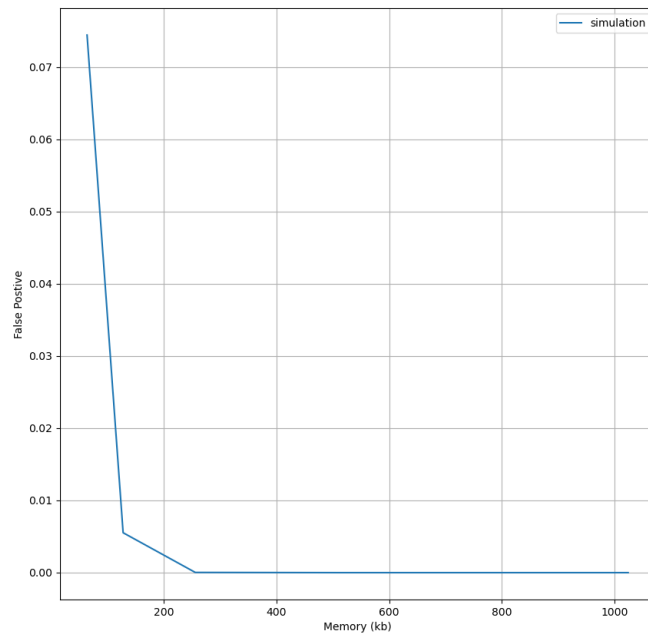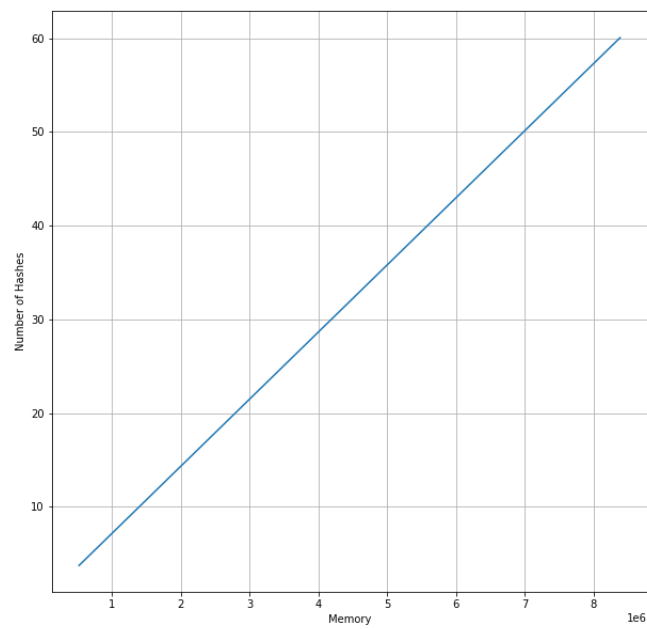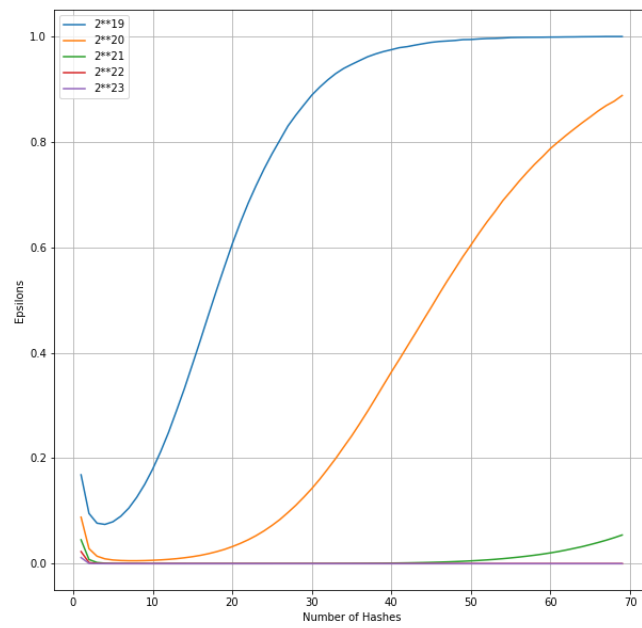

Figure 7

Figure 6

Figure 4

Figure 5