TDT4305 Big Data

Project 2 – Report

Name: Dominik Ucher

Question 1

Q: Using parameter k = [1; 5; 10], and the default values for the rest of the parameters, how does the parameter k affect the total number of shingles? Is having more shingles better for identifying as many as possible document pairs that have similarities over the threshold?

A: The lower number of k, the more shingles there will be. The number k is the number of words that are in each shingle and n are the number of words in the document. Therefore it will have n-k+1 amount of shingles. The lower number of k, the more fine the shingles will be. Therefore, it will be much more accurate to find the document pairs that have the most similarities over the threshold. Although the finer the shingles are, the more time and resources the code will need in order to work, and go through all the documents.

Code:

print("K-Shingles = ", sum(len(s) for s in doc_k_shingles))

For k = 1: Shingles = 505450

For k = 5: Shingles = 844396

For k = 10: Shingles = 834330

Test code:

For k = 1: Shingles = 30

For k = 5: Shingles = 9

For k = 10: Shingles = 0

The reason why k=1 has the least shingles is because we are putting it in sets, and therefore the duplicates are removes. There are much more duplicates of shingles of 1 word than shingles of 5 words. If we were not to remove duplicates then there would be much more shingles for k=1.

Question 2

Q: Using permutations = [10; 20; 50; 100], and the default values for the rest of the parameters, how does the number of permutations affect the number of false positives?

A: The number of permutations effect the accuracy of MinHashing. And the accuracy of MinHashing influences the number of false positives in Local Sensitivity Hashing (LSH). The more permutations, the more accurate the MinHashing will be, because the permutations make the Signature Matrix larger. Which in turn makes the computation more accurate and less false positives.

For permutations = 10: False positives = 2473570

For permutations = 20: False positives = 1870598

For permutations = 50: False positives = 295741

For permutations = 100: False positives = 521

Saw on BB that it should be around 3 false positives for permutations = 100, but I get a few hundred. Used a few days to try and debug and find the error in my code but could sadly not find it in the end. Went through my code and I know that Task 1,2 and 5 is correct. Pretty sure that Task 3 and 4 is correct as well, but because of the large number of false positives then it must be an error in there. Tried several different codes and tried it on the Jupyter Notebook file. Works in the Jupyter Notebook file but not on the python LSH code.

Question 3

Q: Using b = [20; 50; 100], and the default values for the rest of the parameters, what is the optimal number of bands so that we make the least amount of comparisons? also, how does the b value affect the number of false positives?

A: The number of bands determine the number of comparisons to be made and the false positives. Increasing the number b in bands will decrease the number of rows per band. Therefore, the lower number of b the more comparisons there will be. That is because there are fewer signature hash functions to hash through and pair together. That will in turn increase the number of false positives.

And as you can see bellow, with b = 20 gives the fewest comparisons and fewest false positives. Although you do not want b to be too low because then it can negatively affect the outcome of the LSH algorithm

For b = 20: Comparisons = 620, False Positives = 447

For b = 50: Comparisons = 621547, False Positives = 621371

For b = 100: Comparisons = 2462144, False Positives =2491966

Question 4

Q: Using the naive() method, parameter naive = true, count how many document pairs are checked for similarity for the naive() and how many for the LSH. How efficient is the LSH method compared to the simple naive method both in terms of document pairs and running time?

A: By using the naive() method it checks 2474200 document pairs for similarity in 45 seconds, while the LSH uses 4 seconds to check 659 similarity pairs. Although if you take into account the MinHash Signature Matrix, then the LSH method uses 79 seconds. But the goal for the LSH method is to reduce the number of comparisons made by using a hashing and candidate pair technique. Which makes it to compare less document pairs from the beginning and therefore using less time. While the naive technique aims to compare all the documents for eventual pairs. So then the naive technique is more accurate but uses more time, when the LSH method is less accurate but uses much less time. Therefore to determine which algorithm is best, is up to the preferences of the user.