Project 2 – TDT4305

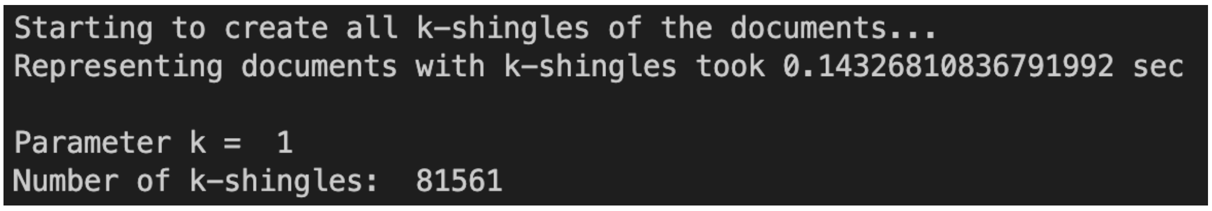
Oda Marie Colquhoun & Emil Bjørlykke Berglund

Task 1

When changing the value of k to be either 1, 5 or 10, it affects the total number of shingles. The bigger k is, the fewer shingles we get. This also means that we get a higher precision of the method when comparing similar documents, but means we get less documents that are categorized as similar. This follows the principles of precision and recall, meaning that the higher precision we choose, it will be at the expense of the recall. We noticed that k = 1 shingles were less, this is probably because there are more combinations of 2 words than single words. Too long shingles would mean fewer pairs because the word order would have to be equal over a longer stretch, while too few would make many too many documents similar as all texts use the same basic words, so somewhere in between, like k = 5 is a nice balance. This also depends on the length of the documents.

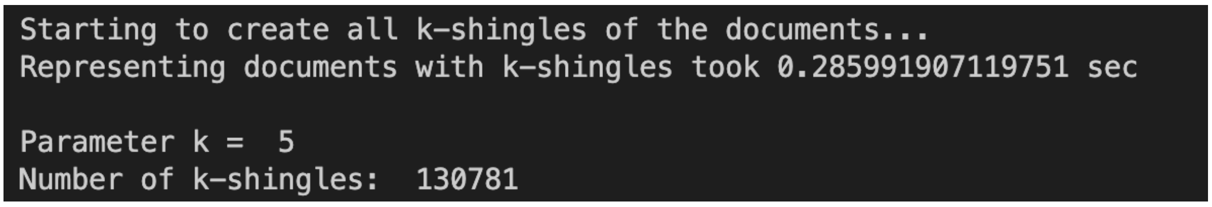
Here we show how the different values of k affect the number of shingles generated:

With k = 1:



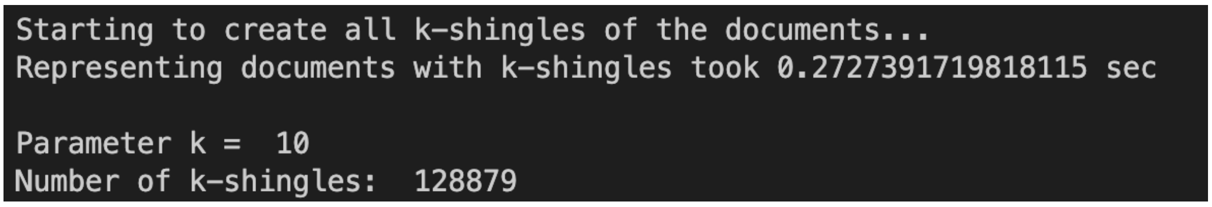
```
Starting to create all k-shingles of the documents...
Representing documents with k-shingles took 0.14326810836791992 sec

Parameter k =  1
Number of k-shingles:  81561
```

With k = 5:



```
Starting to create all k-shingles of the documents...
Representing documents with k-shingles took 0.285991907119751 sec

Parameter k =  5
Number of k-shingles:  130781
```

With k = 10:



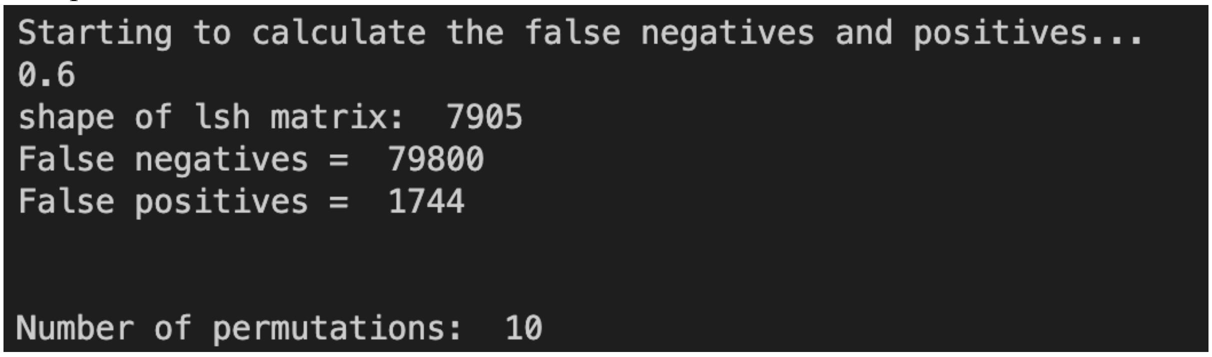
```
Starting to create all k-shingles of the documents...
Representing documents with k-shingles took 0.2727391719818115 sec

Parameter k =  10
Number of k-shingles:  128879
```

Task 2

We see that the number of false negatives are the same, even though the number of permutations have changed. The higher the permutations-number is, we get a higher number of false positives. A higher number of permutations in LSH can generate a higher number of false positives because it increases the likelihood of dissimilar documents being assigned to the same bucket, which can lead to inaccurate results when searching for similar documents. This tradeoff between false positives and false negatives is an important consideration when choosing the number of permutations to use in LSH.

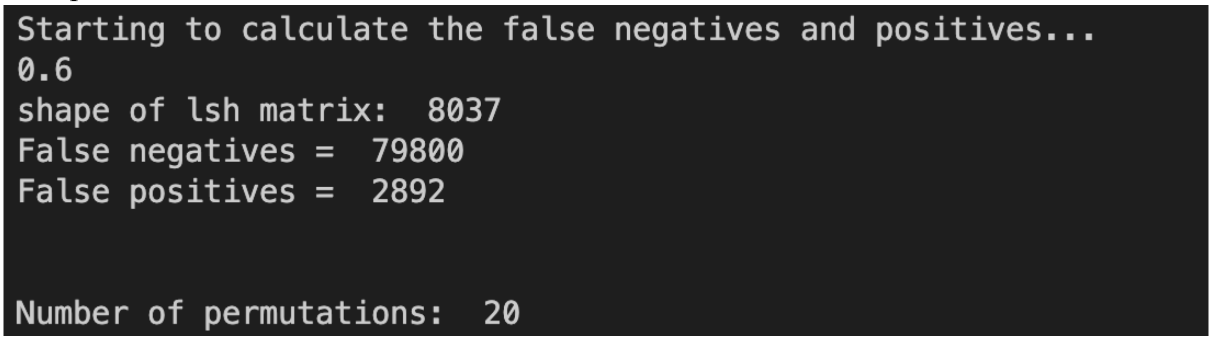
Here we show how the different values of permutations affect the number of shingles generated:

With permutations = 10:



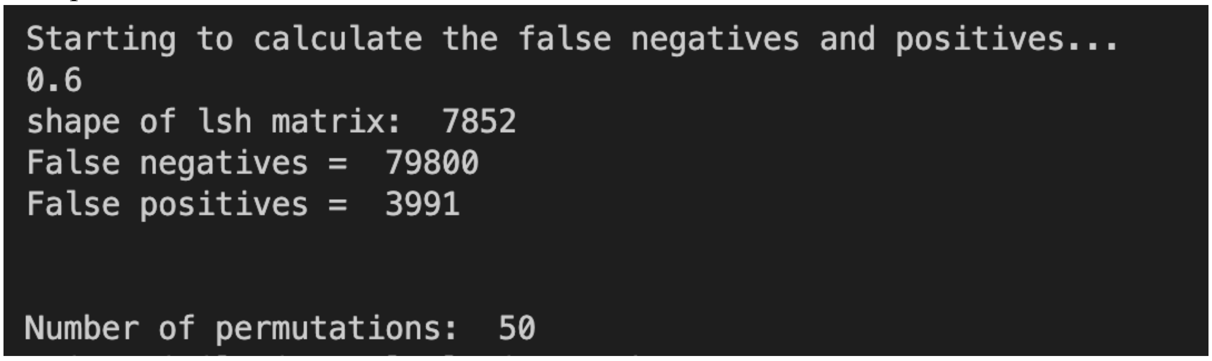
```
Starting to calculate the false negatives and positives...
0.6
shape of lsh matrix:  7905
False negatives =  79800
False positives =  1744


Number of permutations:  10
```

With permutations = 20:



```
Starting to calculate the false negatives and positives...
0.6
shape of lsh matrix:  8037
False negatives =  79800
False positives =  2892


Number of permutations:  20
```

With permutations = 50:

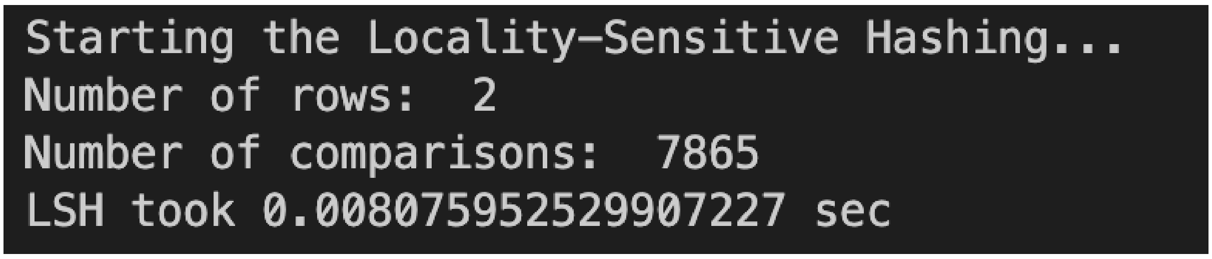


```
Starting to calculate the false negatives and positives...
0.6
shape of lsh matrix:  7852
False negatives =  79800
False positives =  3991


Number of permutations:  50
```

Task 3

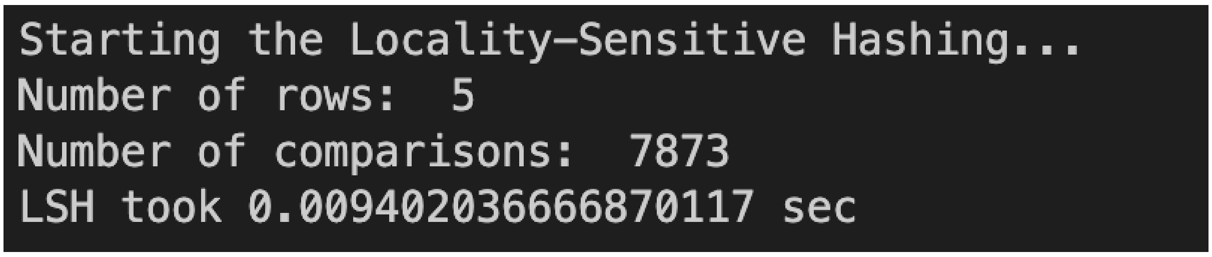
We do the least amount of comparisons when the number of rows is low. A higher number of rows in a band can lead to more comparisons being done in LSH because more hash values need to be computed and compared for each document. This can increase the computational cost of the algorithm, but it can also lead to more accurate results by identifying similar documents that may have been missed with fewer rows. 2 would therefore be the optimal number of rows to get the least amount of comparisons.

With number of rows = 2:



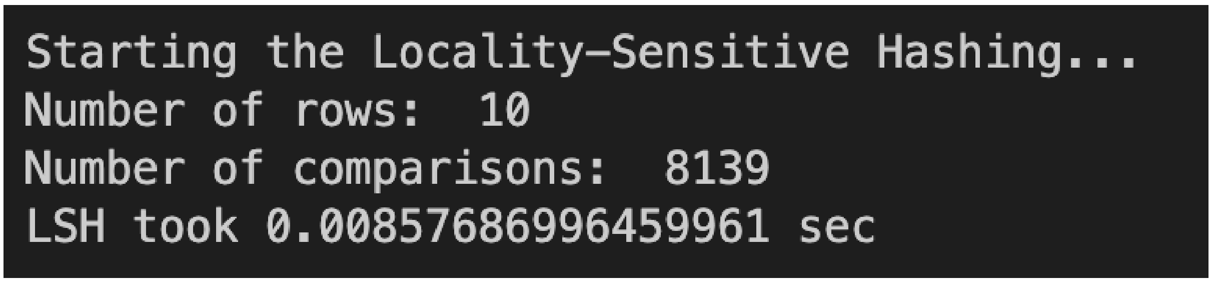
```
Starting the Locality-Sensitive Hashing...
Number of rows:  2
Number of comparisons:  7865
LSH took 0.008075952529907227 sec
```

With number of rows = 5:



```
Starting the Locality-Sensitive Hashing...
Number of rows:  5
Number of comparisons:  7873
LSH took 0.009402036666870117 sec
```

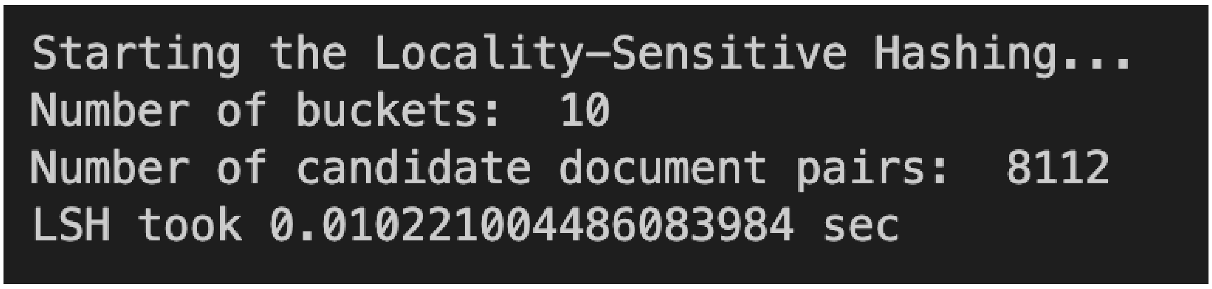
With number of rows = 10:



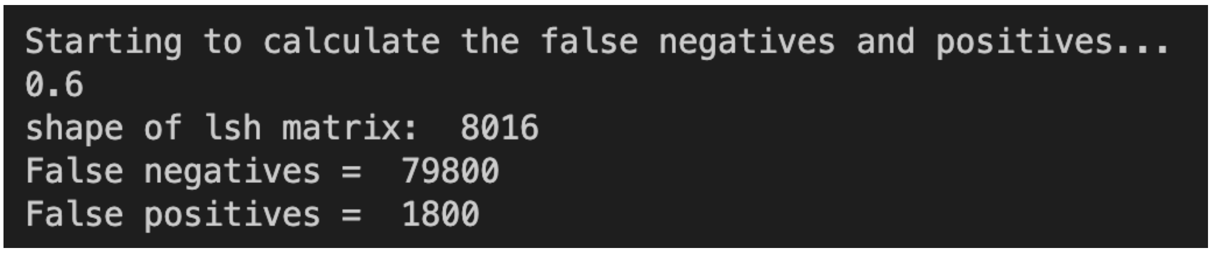
```
Starting the Locality-Sensitive Hashing...
Number of rows:  10
Number of comparisons:  8139
LSH took 0.00857686996459961 sec
```

Task 4
The number of candidate document pairs decreases when increasing the number of buckets. Increasing the number of buckets in LSH can lead to fewer candidate document pairs because the probability of similar documents being hashed into the same bucket decreases. In other words, as the number of buckets increases, the likelihood of two similar documents being grouped into the same bucket decreases. This reduces the number of candidate document pairs that need to be compared to determine their similarity. This is the same reason that more buckets decrease the number of false positives.

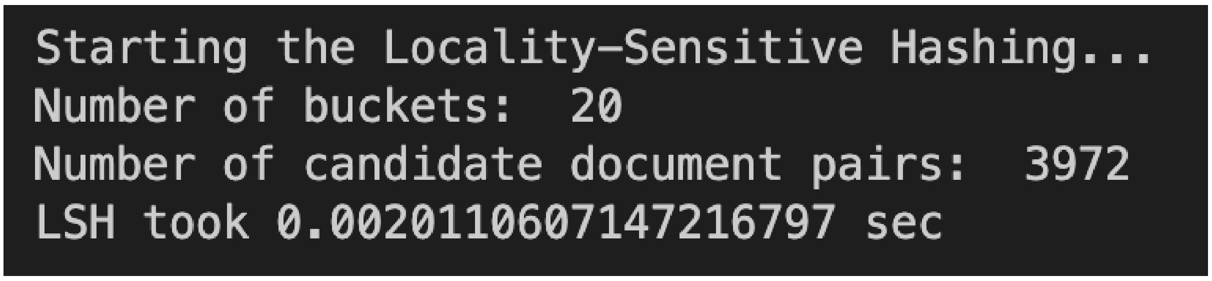
With number of buckets = 10:

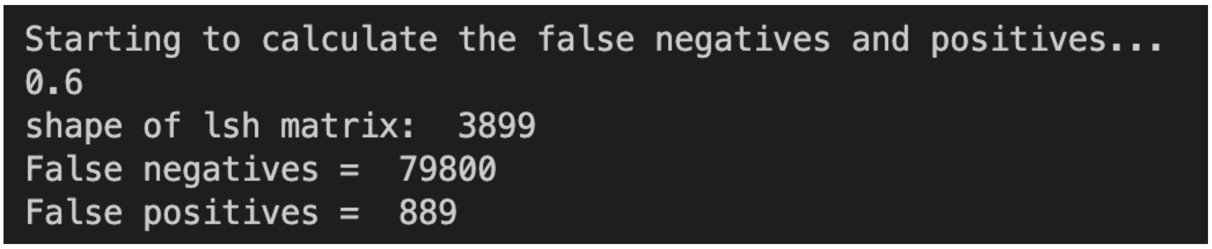


```
Starting the Locality-Sensitive Hashing...
Number of buckets:  10
Number of candidate document pairs:  8112
LSH took 0.010221004486083984 sec
```

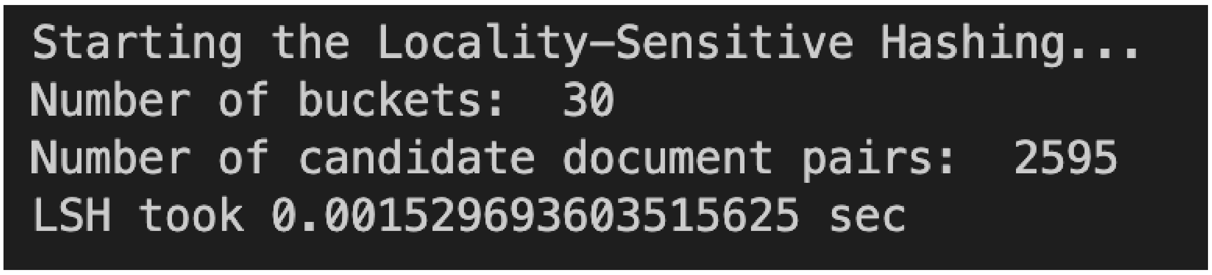

```
Starting to calculate the false negatives and positives...
0.6
shape of lsh matrix:  8016
False negatives =  79800
False positives =  1800
```

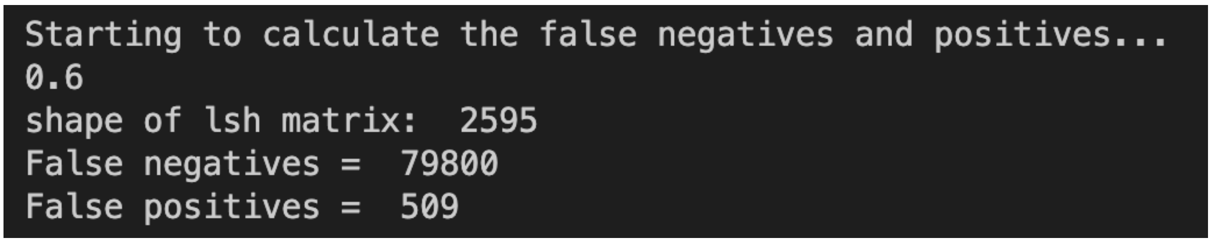
With number of buckets = 20:


```
Starting the Locality-Sensitive Hashing...
Number of buckets:  20
Number of candidate document pairs:  3972
LSH took 0.0020110607147216797 sec
```


```
Starting to calculate the false negatives and positives...
0.6
shape of lsh matrix:  3899
False negatives =  79800
False positives =  889
```

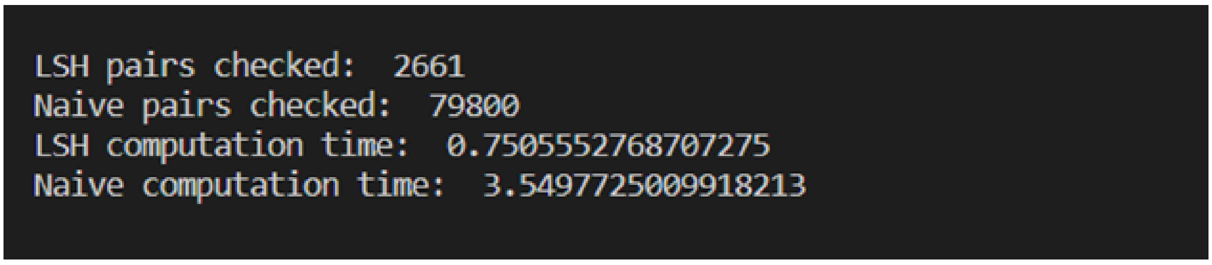
With number of buckets = 30:


```
Starting the Locality-Sensitive Hashing...
Number of buckets:  30
Number of candidate document pairs:  2595
LSH took 0.0015296936035l5625 sec
```


```
Starting to calculate the false negatives and positives...
0.6
shape of lsh matrix:  2595
False negatives =  79800
False positives =  509
```

Task 5

Below we can see the number of pairs checked for both the LSH method as well as the Naive, and the amount of time the checking took for the respective methods.



```
LSH pairs checked:  2661
Naive pairs checked:  79800
LSH computation time:  0.7505552768707275
Naive computation time:  3.5497725009918213
```

The LSH method did significantly less comparisons compared to the naive, this also shows in the run time. The reason for this must be that the naive method naively assumes whether two given documents are similar or not. This means that the “threshold” for two documents being considered similar is very low, and many document pairs will be created, which causes more document pairs to be checked. However, many of these pairs created by the naive method are not necessarily similar, and this costs a lot of computation time.

Task 6

We played around with the parameters, and got some higher positive counts but along with that, we got higher false positive counts, so we chose to keep the parameters as they were given, because we believe those yielded the best results.