# Lab 4: MinHash Similarity

# Introduction

The purpose of this lab was to develop and test the MinHash algorithm for comparing documents. First the baseline Jaccard similarity measure between all documents was taken directly from the raw feature vector. Next, different numbers of hash functions (K values) were to be used to estimate Jaccard similarity between hashed documents. The MinHash estimated similarities were compared with the actual Jaccard similarities for efficacy and efficiency.

# Preprocessing

In order to make the feature vectors more suitable for MinHash, the preprocessing was completely redone from previous labs. All documents were shingled into 3-gram feature vectors, whereupon the 3 words were separated by a space, and hashed using the binascii library’s CR32 hash function to create a 32bit string. The code to processes the Reuters files into the shingled feature vectors can be seen in preprocess4.py.

# MinHashing

The implementation of MinHash is in utilities.py. No external libraries for similarity measures or MinHashing were used, thus the performance and optimization is likely lower than could have been achieved. This will affect the efficiency of the MinHashing, and it should be noted that a more in depth implementation could achieve better performance.

Efficiency

The testing for efficiency was done by tracking the total time it took to create the K hash signatures and then to transform them into estimated Jaccard similarity arrays.

Efficacy

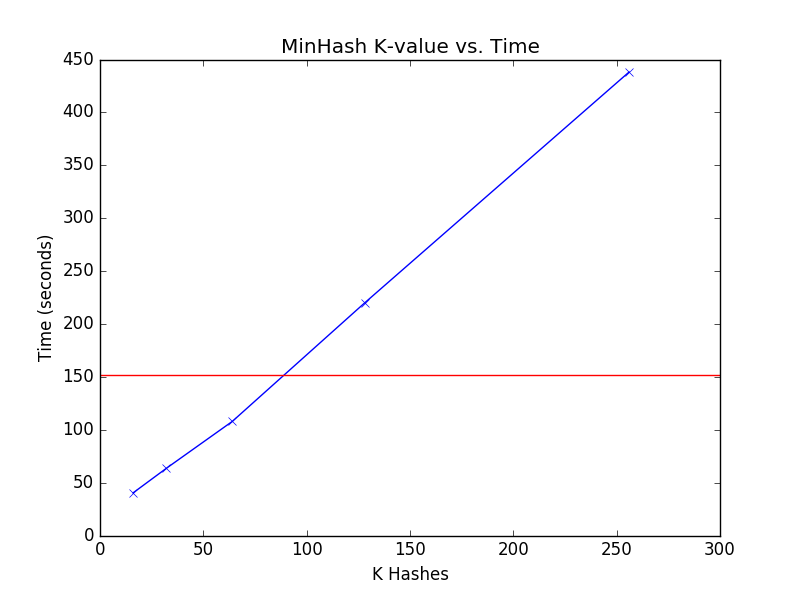
The efficacy of the MinHash algorithm in estimating Jaccard similarities was measured by taking the mean squared error (MSE) between the estimated and actual Jaccard similarities.

# Results

When originally testing on my laptop with 8GB of RAM, memory would sometimes run out and cause erratic differences in times when running multiple tests in a row. Because of this the data for efficiency when running all 10,500 documents was untrustworthy. To alleviate the problem I reduced the number of documents examined from 21 files to 8, making for much more consistent data.

Efficiency

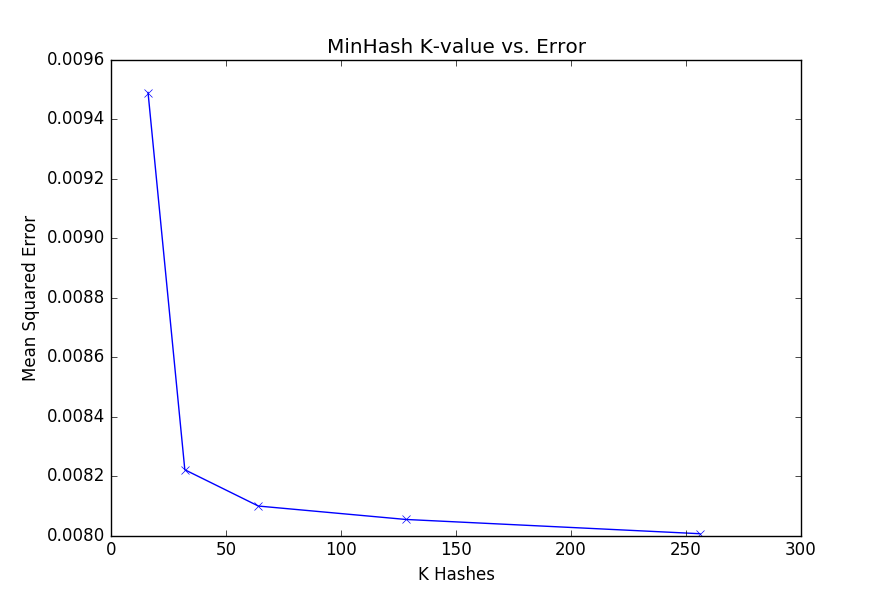
Figure 1 on the following page shows a linear relationship between the number of hash signatures used and the time in seconds it takes to calculate the estimated Jaccard similarities. The red horizontal line on the plot is the time it took to calculate the actual Jaccard similarities. The raw feature vector took 151.54 seconds to compare all = 10,481,331 combinations. The average length of a feature vector was 130.7 shingles, and the median was 86. On the plot, it can be seen that it begins taking more time to generate the estimated Jaccard similarities when the number of different hash signatures approaches the median number of shingles in the list. This makes intuitive sense, as using more hash signatures than there were original shingles completely defeats the purpose of the MinHash algorithm, and thus all values of K above the median document feature vector length should not ever be considered for actual implementation.

 **Figure 1**: This plot shows the amount of time in seconds to create the estimated similarity matrix

given K hash signatures. It can be seen that the time is linearly dependent upon the number of hashes. K values tested were 16, 32, 64, 128, 256. The red line denotes the baseline time to compute actual Jaccard similarities.

Efficacy

The results for the efficacy of the algorithm were examined in Figure 2 seen on the following page. The efficacy of the algorithm followed an exponential trend. The MSE drops rapidly as K increases from 0 to 1/4th of the average length of document feature vector, whereupon it begins to level off. This indicates that K values above 1/4th the length of the average document feature vector have very low marginal utility, and thus for efficacy purposes it is reasonable to use K near this value. For this dataset, K=32 produced a MSE of 0.008221 and only took 41.82% of the time it took to compute the actual Jaccard similarities.

. **Figure 2**: This plot shows the number of hash signatures versus the average mean squared

error between actual and estimated Jaccard similarities. K values tested were 16, 32, 64, 128, 256.

# Conclusion

The results for both the efficacy and efficiency align with the idea that any value of K above the average length of a document feature vector ought not to be considered in MinHashing. Furthermore, time to estimate the Jaccard similarity has positive linear correlation with K, and thus to reduce time, K should be as low as possible. For this dataset, the cutoff point where increasing K does not improve MSE drastically is around 32. This would be the recommended value when MinHashing Reuters news articles.