CS168 Project 2

Henry Lin, Kaylee Bement

1. A. See code.py

B. See code.py

C. Based on the heat maps, the Jaccard and cosine similarity metrics seem to both perform reasonably well for this dataset. Since any bag of words model is bound to be sparse by design, it is expected that Jaccard would perform well in comparing the articles of each newsgroup.

The atheism newsgroup seems to share a noticeable similarity with the religion Christian newsgroup. While the two groups may have differing opinions, it is expected that they are rated similar since they discuss the same topics. There is a surprising amount of similarity between science-related and religion-related newsgroups such as religion Christian and medicine, but the two groups definitely have an intersection where there is a significant level of debate, both in the dataset and real life.

1. A. See code.py

Classification Error: 0.544

B. For Part 1, the plots dealt with distances between points, and distance is symmetric. The distance from A to B is equal to the distance from B to A. For this matrix, we are dealing with closest neighbors, which is not necessarily a symmetric property. For example, AB = 4, BC = 3, AC= 10. In this case, B would be A’s closest neighbor, but B’s closest neighbor is actually C. This causes the asymmetric matrix.

C.

Classification Error, d = 10: 0.88

Classification Error, d = 25: 0.771

Classification Error, d = 50: 0.702

Classification Error, d = 100: 0.645

Note: Due to the randomness in the model, these values change slightly with each run of the program.

It is clear that as the target dimension increases, the classification error decreases. For the given values of d, 100 gives the most comparable results to the original dataset.

D.

The runtime in big O for reducing the dimension of the data will be O(dk + dnk), which reduces to O(dnk). dk is required to set up the matrix and the dnk is the runtime of the matrix multiplication.

The runtime of classifying a new article is O(dk + nd) since we already have the d by n matrix we can now use and we also need to do dimensionality reduction.

Since the data is sparse with each tweet containing a relatively small number of words, we can use a dictionary or some other efficient data structure to improve the runtime to O(50n) or O(n).

Comparing this to the dimension reduction nearest-neighbor system, the two have comparable search time with the dimension reduction system being more costly when including the original matrix multiplication.

1. A.

From lecture, we know that choosing each coordinate of the vector from a normal distribution creates a vector in a uniformly random direction. For one bit/dimension, the normal vector must go through the angle theta to hash to different buckets, so we can find the probability of it hashing to the same bucket. We then raise this to the power of d for the d bits that must match.

B.

Following from the previous question, we can simply extend this to consider the new range of angles.

C.

To solve this problem, we have to consider equations that model the requirements. We first consider than when the angle between the query and a datapoint is less than 0.1 radians, we want the probability that the query hashes to the same bucket in at least one of the tables to be at least 0.9

Secondly, we want to make sure that for the query and any datapoint with angle greater than 0.2, we want there to be very few collisions. In the equation below, we want there to be at most 60,000 such collisions.

Solving this system of inequalities, we get a range of possible values for l and d that satisfy our threshold, and we decided to use d = 114 and l = 98. These values can change depending on where importance is placed. For instance, decreasing l will decrease greatly the computational cost and expected number of collisions overall, but it will also decrease the probability of finding the nearest neighbor (angle-wise) in the resulting set of points. For d, decreasing it will increase the chances of the nearest neighbor appearing in the set we consider, but it will also noticeably increase the number of collisions and consequently the resulting set.

D. See code.py

E.