COMS 435 PA2 Report

Brad Warren 184030907

MinHash:

The procedure I used to collect all the terms of within every single one of the documents started with constructing a HashMap with strings as the keys and integers as the values. I made an array of files and with this array of files I made a for loop that goes through every file and runs a method called getTerms(). The method directly adds a term to my HashMap if containsKey(term) returns false and the length of the term is greater than 2 and not equal to "the" as well as does not have random punctuations like (">", "<"," *", etc. to decrease number of terms). The integer assigned to each term was done in order of which the terms were found. Thus, if the first word in the first document was "hello" then hello would be at index 1 on our HashMap. Then I added all the terms to an Array list in order and the order they came in was their number. In order to construct random permutations, I created a 2d integer array called permutations the size of the [number of permutations] [number of terms]. I then constructed a for loop over the number of permutations and used a method called fisherYatesShuffle where I implemented fisher Yates Shuffle Algorithm.

MinHashAccuracy:

Running the program and results:

	Permutations: 400	Permutations: 600	Permutations: 800
Threshold: 0.04	12765	5302	4490
Threshold: 0.07	528	515	500
Threshold: 0.09	422	400	270

From this data above we can conclude that out of 1004004 pairs from the space.zip getting this small amount of difference between approximate and exact jaccard is great. We can also conclude that the more permutations we use, the more closely related approximate and exact jaccard are. Also, we can determine that not many pairs have substantial differences in approximate and exact jaccard (seen as threshold decreases). The higher the threshold the smaller the count of pairs which is good.

MinHashTime:

With 600 permutations on the files from space.zip, the runtimes are as follows:

```
Time taken to compute exact Jaccard between each document = 31631.0 ms
Time taken to compute approximate Jaccard between each document = 1298.0 ms
```

We can see that the total time taken to compute the exact Jaccard is 30,333 ms slower than the time it takes to compute approximate Jaccard between each document.

NearDuplicates:

For this I used subset of PA2.zip of 1,610 items, permutations of 600 for every test a test for .9 threshold and .7 threshold for each file tested. The outputs are below:

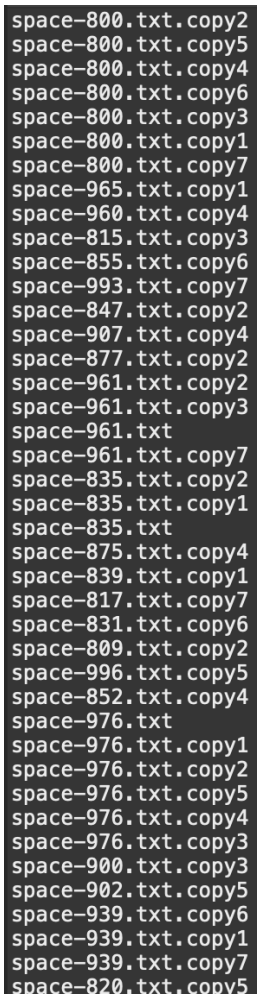
Space-1000.txt - .7 threshold:

```
space-1000.txt.copy4
space-1000.txt.copy1
space-1000.txt.copy6
space-1000.txt.copy5
space-1000.txt.copy2
space-1000.txt.copy3
space-918.txt.copy3
space-1000.txt.copy7
space-894.txt.copy7
space-805.txt.copy2
space-838.txt.copy2
space-840.txt.copy4
space-983.txt.copy5
space-937.txt.copy6
space-877.txt.copy7
space-897.txt.copy4
space-975.txt.copy4
space-926.txt.copy2
space-855.txt.copy1
space-951.txt.copy3
space-988.txt.copy6
space-874.txt.copy5
space-949.txt.copy6
space-943.txt.copy2
space-983.txt.copy2
space-937.txt.copy7
space-845.txt.copy3
space-872.txt.copy7
space-918.txt.copy1
space-821.txt.copy3
```

- .9 threshold:

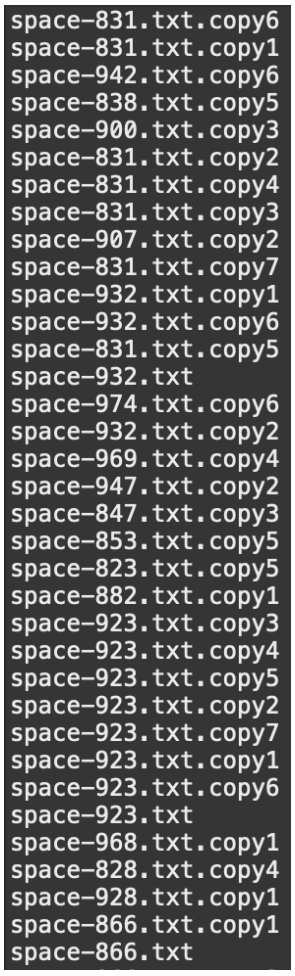
```
space-1000.txt.copy5
space-1000.txt.copy2
space-1000.txt.copy3
space-1000.txt.copy1
space-1000.txt.copy6
space-1000.txt.copy7
space-1000.txt.copy4
space-880.txt.copy1
space-964.txt.copy5
space-806.txt.copy1
space-839.txt
space-839.txt.copy5
space-952.txt.copy7
space-802.txt.copy2
space-823.txt.copy6
space-823.txt.copy2
space-997.txt.copy7
space-967.txt.copy4
space-967.txt.copy1
space-901.txt.copy7
space-824.txt.copy1
space-824.txt.copy7
space-824.txt
space-808.txt.copy7
```

Space-800.txt – threshold .7:

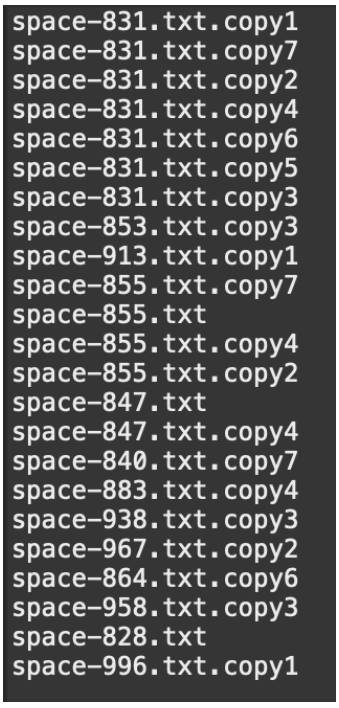


threshold .9:

Space-831.txt – threshold .7:



- threshold .9:



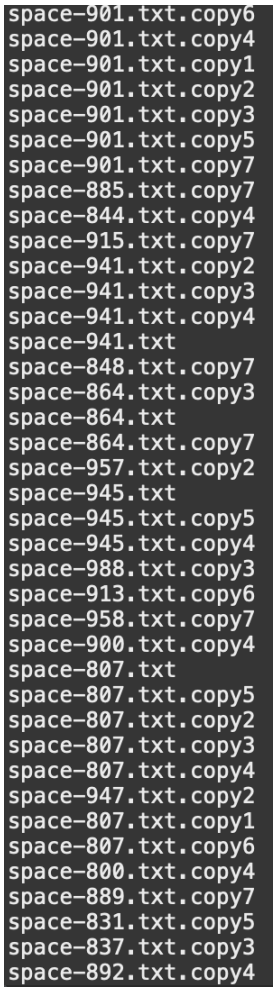
Space-866.txt – threshold .7:

```
space-866.txt.copy6
space-846.txt.copy3
space-866.txt.copy5
space-866.txt.copy7
space-866.txt.copy1
space-866.txt.copy3
space-866.txt.copy4
space-929.txt.copy1
space-866.txt.copy2
space-990.txt.copy6
space-844.txt.copy4
space-993.txt.copy7
space-993.txt.copy6
space-993.txt.copy3
space-993.txt
space-851.txt.copy6
space-970.txt.copy7
space-947.txt.copy4
space-867.txt.copy7
space-867.txt.copy1
space-867.txt
space-867.txt.copy4
space-813.txt.copy5
space-954.txt.copy2
space-881.txt.copy6
space-801.txt.copy6
space-863.txt.copy3
space-863.txt.copy2
space-863.txt.copy5
space-863.txt
space-863.txt.copy7
space-863.txt.copy1
```

- threshold .9:

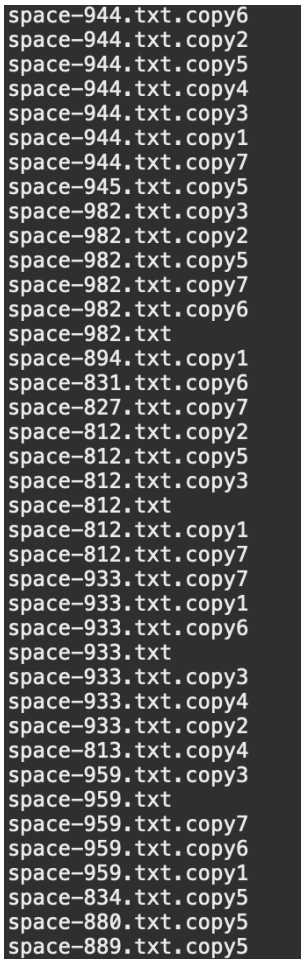
```
space-866.txt.copy7
space-866.txt.copy6
space-866.txt.copy3
space-866.txt.copy1
space-916.txt.copy1
space-866.txt.copy4
space-866.txt.copy2
space-866.txt.copy5
space-923.txt.copy4
space-831.txt.copy6
space-816.txt.copy2
space-981.txt.copy1
space-800.txt
space-800.txt.copy5
space-800.txt.copy6
space-896.txt.copy7
space-916.txt
space-916.txt.copy6
space-945.txt.copy6
space-985.txt.copy7
space-940.txt.copy7
```

Space-901.txt – threshold .7:

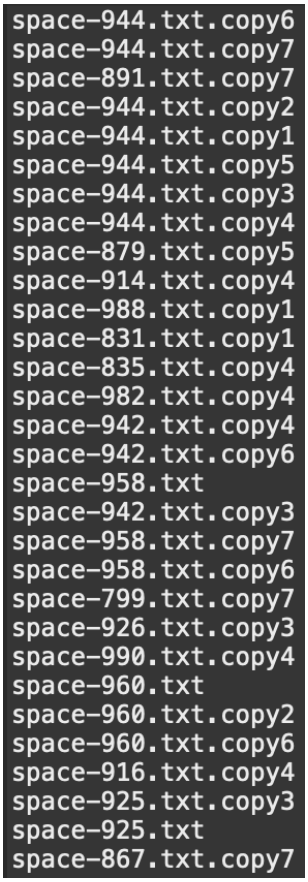


- threshold .9:

Space-944.txt – threshold .7:

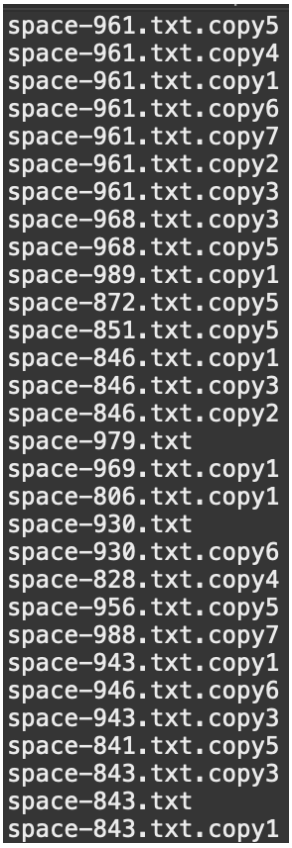


- threshold .9:

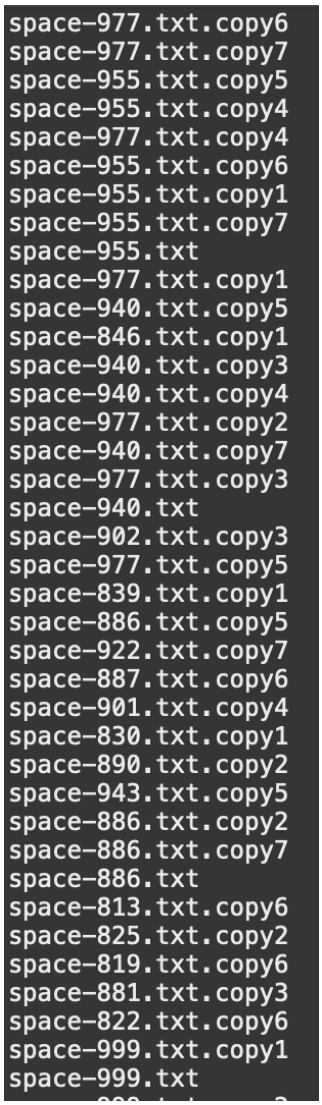


Space-961.txt – threshold .7:

- threshold .9:



Space-977.txt – threshold .7:



- threshold .9:

Space-985.txt – threshold .7:

- threshold .9:

Space-993.txt – threshold .7:

- threshold .9:

From these outputs I can conclude that somewhere along in the process of getting the nearDuplicate documents my data is inconsistent. This is because when I run a threshold of .9 we should only get the outputs of the copies, however this is not the case. I believe it may have something to do with how I get the permutations. But I am unsure. I have looked into this for the last week of the project, but I have yet to come to any conclusions. This data is without removing false positives. Also, each output does have all of the copies of the given file.