CSE13s Winter 2022
Assignment 7: Author Identification
Theodore Ikehara, tikehara@ucsc.edu
Due: March 13th at 11:59 pm
Writeup

What I learned:
In this assignment I learned much about hash tables, bloom filters, and AIs attempting to detect text similarities. In this assignment we focused mainly on three metrics, these being: Manhattan distance, Euclidean distance, and cosine distance. All of these methods make use of a calculation regarding the count of the words inside the text to create this distance that we can then use to judge the similarity index of the text. They all make use of their own methods, ultimately doing similar things. These methods were implemented in the text module as we chose the metric with enumeration char*.

Current bugs that are in the module:
Text_delete - not operating properly still creating leaks
Identify.c - fgets sometimes goes off if db was not properly written (this means all the text files need to be openable for this to work properly)
Identify.c - enqueues wrong items because of the previous bug

These bugs do not apply when I use my small test case scenario…

All the observations made below are made using my small test case scenario.

Observation when noise level is turned down:
When the noise level is turned down we can generally observe that the similarity level of the texts goes up. This makes a lot of sense as there are more common words that people use that the algorithm will start adding to the total similarity. Words like: it, and, or, but, can, the etc. all have a high use rate and scenario. Due to this nature the texts are going to seem more and more similar to the algorithm as the algorithm will start treating all the words as the same. The noise level is the only item inplace to stop this occurrence. And the opposite phenomenon occurs when you turn the noise level up. The texts will seem less and less similar.

However this becomes less and less useful as then we see texts are either super similar or not similar at all and when using this algorithm we want to see similar texts and not texts that are the same or not. Thus the similarity starts to be less range-like and more of a yes it's similar or no its not similar. And this is not what we are looking for in this algorithm and thus we need to be careful when setting the noise level as we want a space in between when it does not do any of these situations given above.

Large vs small texts:
When this program is being fed a small text it generally thinks that the texts are less similar to the other one when the text being compared is larger. This is due to the fact that the small text does not have a lot of the words that the large text has and thus we will end up adding a lot of 0's for the normalized frequency as that word does not exist in the small text. As the variance of number between the text gets larger we see that the algorithm thinks the similarity is getting smaller as well. When one text has less of the words than the other text we see that the similarity also goes down.

How are do the different metrics compare:
The three metrics we considered when doing this assignment: Manhattan, Euclidean, and Cosine distances. These three distances all have unique ways to numerate the value of similarity but ultimately all do the same thing. All three of these distances are calculated with the normalized frequency then gets the distance using either the square root, the absolute value or having 1 subtracting the value. These are all positive numbers and give relative amounts of appearance of the words for each text. The differences between the three however are quite clear when you look at the numbers that are outputted. Generally if the texts are similar Manhattan will calculate a number larger than 1. This does not happen often with the other two methods given the nature they calculate it. The other two methods will give you more of a percentage like number as a total when you add the number together.

Conclusion:

This was a very interesting assignment as we got to experiment with the different ways that algorithms predict the similarities and differences of peoples text. This algorithm can also be used to detect plagiarism and other related offenses. In the future these implementations of priorityqueue and hash tables and bloom filters can also come in great use.