Assignment 7 - WRITEUP

Kevin Lee

Introduction:

This write-up goes over the differences between the different algorithm types and the size of text files, and finally, the number of noise words that are filtered out.

Different algorithm types:

We used three different algorithms throughout our program.

The first algorithm we used is the Manhattan algorithm. This algorithm uses the frequencies of words in each text file. The frequency as calculated in our function is the number of times that the word shows up in the text file divided by the number of total words that appear in the text file. So for the Manhattan algorithm, we take the frequency of a word, say hello in both text files, and subtract them. Then we take the absolute value of the difference. We do this will all words, naturally, there will be some words in a text file that do not appear in the other, in this case, the frequency would just be the frequency of the word in the text file it appears in. We add all of these frequencies together and ultimately end up with the Manhattan distance.

What I noticed was that the Manhattan distance was for the most part the largest value when compared to the other algorithms.

The second algorithm we used is the Euclidean algorithm, this algorithm is very similar to the Manhattan Algorithm. Rather than simply adding the difference of the frequencies, we add the square of the frequencies. Finally, we take the square root at the very end, and that is the final distance.

This algorithm seemed to be to have the smallest distance. This is because as the word count increases, the frequencies of various words get smaller and smaller. For example, if the word count is 100,000 and there are 10 instances of the word, hello, the frequency would be 0.001. Additionally, you would have to multiply this with the frequency of hello in the other text, further decreasing the frequency value. Naturally, you can see how the distance could be very small. Even after taking the square root, the distance is still very small.

The last algorithm is the cosine algorithm. This algorithm entails simply multiplying the frequencies of the word in both texts and adding that to the distance. Finally, we subtract this value from one.

We can see why this algorithm is probably the second largest distance. This algorithm understandably caps out at 1. This algorithm is very similar to the Euclidean algorithm, however, one difference is that this algorithm only counts the words that are in both algorithms. This is because if one text doesn't have a

word, its frequency would be 0, so when multiplying that to the other frequency, it would still be 0.

Comparing Algorithms:

The Manhattan algorithm is no doubt the most vanilla of the algorithms. It basically simply compares the frequencies of words in texts and adds that together. This works passably, however, I think that the Euclidean algorithm is better. Taking the square root of a number, in this case, numbers smaller than 1, results in magnifying the value. Small values get exponentially smaller, while larger numbers don't get as small. In this case, this algorithm magnifies the differences, so larger differences in frequencies are more pronounced. I think that ultimately this is a better algorithm. Say an author likes to use the word "undoubtedly" a lot. Given that it is an uncommon word, and it is unlikely that another author uses the word a similar amount, it would make sense to magnify that difference in frequency by squaring the value. The cosine algorithm does the same but in the opposite fashion. Since we are subtracting the value from 1, the largest impact would be a frequency product of 0. This makes sense, since if one author uses the word "undoubtedly" a lot and another author doesn't use it at all, the distance would not go down at all.

Size of Text Files

Intuitively, larger text files would lead to more accurate results. However, it would not matter if one text file is larger than another since we are simply using frequencies, derived by dividing the word count by the total word count. However, a larger text file is helpful in capturing words that are part of an author's diction, even if he or she may not use that word in every paragraph. With one paragraph or less, we get misrepresented data.

Number of Words Filtered out

I personally think that the number of words filtered out has little effect on the distance. It most definitely has an impact on the numerical value of the distances, however, I am skeptical if they make the program any more accurate. My train of thought is that while it makes sense to censor out words that are used by almost all authors, to really get at the specific diction of an author, we are looking at the frequencies of words. Even though the word he might be used by all authors numerous times during their work, one might use the word exponentially more than another. Since we are looking at frequency, even though two authors use a common word a lot, one has used it more frequently, which will still give us data. This difference in frequency of widely used words is something I think can

improve the effectiveness of the program and the accuracy of the algorithms. However, I do agree that some words such as The, and whatnot should be removed.

Conclusion:

All in all, I had a great time doing this assignment. I learned about hash tables and their remarkable speed in comparison just linear or even binary search methods. Additionally, while testing, the program led me to wonder if there wer any other algorithms that may be more accurate, along with a possible optimal noise word limit.