# Assignment 7 Write Up

Nicholas Chu

March 2022

## 1 Results

The program, identify, first takes a specified noise file which will contain words that will be filtered out from the texts when scanning them in and using them to compute distance. This is to avoid unnecessarily comparing very commonly used word in order to make the program be more efficient and focus more on the linguistic styles present through the text's diction. It will create a text based off a text specified by the command line input, or from stdin, with the words from noise being filtered out and for the purposes of the explanation will be referred to as text1. Then in a loop, it goes through each file in a specified data base, creating a text, text2, from each file with a specified author (making sure to again filter out the words from noise), then computing the distance between text2 and text1 based off given a specific metric. Then take the calculated distance and enqueue it with its given author to a priority queue. When the program has iterated over the entire database then it will dequeue the from the priority queue that authors with the lowest distances, up to the number specified by to command line or five if nothing was specified.

As for the results of running the program, it did not produce the exact distances that were produced by the sample identify that was provided in resources. This caused the rank the authors be very similar, only occasionally accidentally swapping some ranks. My assumption to why this issue is caused is most likely due to precision, or possibly an issue with the regular expression that was used to match the words of from the texts.I arrived to this conclusion because the calculated distances were close to an extent to the ones in the sample so perhaps identify was just not to properly match all the words. The regular expression that was used to match the word from the document is: [A-Za-z](['-]?[A-Za-z])* which theoretically should have been able to match all the words in the file when based off the assignment specifications. Perhaps the program was not able to match specific cases, though I through testing I was not able to pinpoint any.

### 1.1 Metrics

Since my calculated distances had some differences with the provided program so I have included the results from testing my program and the one provided from resources in order to compare results and more accurate observations. Below are tables that hold the results of the distances calculated when using the file texts/william-shakespeare.txt and using lib.db. The authors of the tables are ordered from least to greatest in terms of their distance.

| Author | Provided Identify | Identify |
|---|---|---|
| William Shakespeare | [0.000000000000000] | [0.000000000000000] |
| Christopher Marlowe | [1.063170916906052] | [1.058281996434170] |
| John Webster | [1.121722393777185] | [1.117818051222378] |
| Dante Alighieri | [1.174548673999550] | [1.161933773598890] |
| Alexander Whyte | [1.177841832164966] | [1.169725833659008] |
| Chretien DeTroyes | [1.179479376748648] | [1.174361153716954] |
| Johann Goethe | [1.184382981775651] | [1.179396629709519] |
| John Dryden | [1.192816248413113] | [1.188191156023469] |
| R. D. Blackmore | [1.197748883230065] | [1.189945722680511] |
| Charles Dickens | [1.203770283122245] | [1.189392876951548] |

Figure 1: Calculated when using Manhattan metric.

While the authors are still all there in the rankings, there is a discrepancy between the two identify programs which is that the rankings of Blackmore and Dickens in which the sample ranks Blackmore higher. Otherwise the ranking for all the other authors (including the ones in later tables) are in the correct order. Again might be an issue with the regex where it was not matching properly which caused this difference in distance. Otherwise from this we see that identify was able to clearly identify Shakespeare as the correct author when using calculating Manhattan distance.

| Author | Provided Identify | Identify |
|---|---|---|
| William Shakespeare | [0.000000000000000] | [0.000000000000000] |
| William McClintock | [0.026516404305561] | [0.026479145848064] |
| Dante Alighieri | [0.026865145178097] | [0.026822873715543] |
| Edgar Allan Poe | [0.027829100437714] | [0.027841325226450] |
| Johann Goethe | [0.027873601844574] | [0.027892918353402] |
| Henry Timrod | [0.028370953256889] | [0.028365949790793] |
| Lew Wallace | [0.028546232963706] | [0.028525766848149] |
| Saxo Grammaticus | [0.029282174356059] | [0.029247701592499] |
| Rudyard Kipling | [0.029385152618790] | [0.029347359606063] |
| Various | [0.029410943073263] | [0.029384625799055] |

Figure 2: Calculated when using Euclidean metric.

The program was able to properly rank the top ten authors and identify the author of the text, Notice that the authors that the programs ranks as the top ten most likely authors is different to those in the Manhattan table. This is due to the differences in the calculation of distances, where Manhattan simply just takes the absolute value of the difference of the two texts while Euclidean takes the differences between two texts and then squares it and takes its square root. Naturally these two different calculation methods would lead to different authors being more likely to have written a text. The exception to this is the distance for Shakespeare which is 0 and is because the two texts being compared are the same so it accurately meaning that without doubt that Shakespeare is the author. The difference between the two texts would always be zero and its square root and absolute value both being 0. We also can observe that the distances of the ranked

authors have a smaller difference between each other when compared to distances produced by the Manhattan metric, where the difference of the distances for the Manhattan table range from 0.01 to 0.06 while on the Euclidean table these differences are around 0,0001 to 0.001 between each of the distances of the authors. This could have implications on how accurate each metric is at finding the most likely authors, as the larger difference in distances between authors would make it better to identify which distance is smaller when comparing, especially the closer the diction from these texts are.

| Author | Provided Identify | Identify |
|---|---|---|
| Elizabeth Browning | [0.998922776695346] | [0.998920445737951] |
| William Shakespeare | [0.998928467671442] | [0.998926969468923] |
| John Webster | [0.999151810753424] | [0.999150570699283] |
| Richard Sheridan | [0.999209862746223] | [0.999208924301898] |
| Christopher Marlowe | [0.999217109132166] | [0.999215949658970] |
| John Dryden | [0.999282258234486] | [0.999280959771734] |
| Johann Goethe | [0.999321061927905] | [0.999319822263981] |
| Mary Rowlandson | [0.999353291314523] | [0.999351881037342] |
| Howard Pyle | [0.999357128425179] | [0.999355583308841] |
| Algernon Swinburne | [0.999358562358709] | [0.999357315837064] |

Figure 3: Calculated when using Cosine metric.

When using the cosine metric something interesting can be observed, even though the text being used as input is by Shakespeare, it does not rank him as the most likely author with a distance of 0, even though they are the same text. The reason behind this discrepancy is most likely due to the way the Cosine metric calculates distance, as it takes the products of the two texts' frequencies instead of the difference. For example when getting the distance of a word unique specific to only one text, that only appears in one of the text would always lead to 0. This is because if it only appears in one text then that word's frequency in the other text would be zero. This in essence would lead to the frequencies of unique words would be ignored.

### 1.2 Noise

The table below shows the top five matches of running identify similarly to the way in Figure 2, except the nose limit was set to 50, instead of left to the default of 100.

| Author | Provided Identify | Identify |
|---|---|---|
| William Shakespeare | [0.000000000000000] | [0.000000000000000] |
| William McClintock | [0.027940911142240] | [0.027912915949830] |
| Johann Goethe | [0.027976252345310] | [0.027987617514893] |
| Lew Wallace | [0.029925600231610] | [0.029908475300693] |
| Rudyard Kipling | [0.030087173060618] | [0.030056457284669] |

Figure 4: Calculated when using Euclidean metric with noise of 50.

## 2 Conclusion

Through observing the results of the graphs and what they imply about the metrics used to calculate the distances of each text it can be concluded that cosine is the most inaccurate out of the metrics used in this context when computing the distances between texts. This is obvious when presented with the fact that the metric was not able to match the most likely author even when comparing two identical texts, which is caused by how the metric uses to products of the frequencies of words in the text to compute distance. Based of the information and graphs from the results I assume that the Manhattan and Euclidean metric are more accurate, with Manhattan potentially being the best metric. The reasoning behind this conclusion is due to the larger differences between the distances of each matched author when using the Manhattan metric. This greater distance between distances would imply that it would be better and more accurate when comparing, especially when involving very similar texts which would produce more similar distances. Another thing to consider is precision, as a double in c can store up to 15 decimal places. While 15 decimal points is very accurate for most purposes, this could be a slight issue when using the Euclidean metric, as since doubles are not exact you would end up losing some precision when working with longer or floating decimals. Taking the products and square rooting when using the Euclidean metric would cause more of a loss of precision when contrasted to simply just subtracting when using the Manhattan metric. Some of the personal takeaways I had from this assignment was learning how to implement hash tables, bloom filters, and regular expressions. It also was an introduction into how to use databases and analyzing large sets of texts.