Krishna Ammini

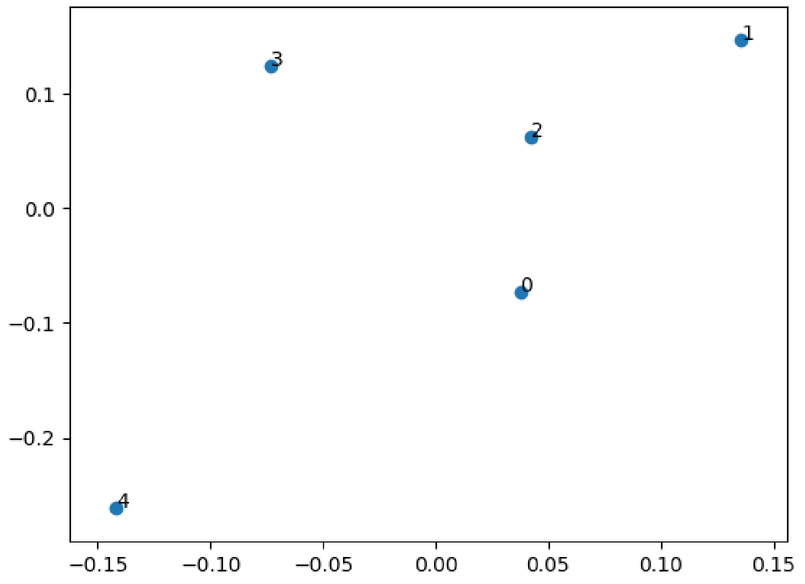
Text Mining and Analysis- Write-up and Reflection

I chose to analyze historical literature from Project Gutenberg because I liked that the website contains such a variety of texts from so many different periods. I also chose Gutenberg so that I would not have to worry about the texts' copyright protections, which expired a long time ago. This may not be the case for other types of text mined online. I used several analysis techniques- determining the most common words in each text, performing sentiment analysis, computing matrices showing the pairwise similarity, and generating a chart to show this visually. I was interested to see how similar or different these old works of literature are and to spot interesting patterns.

To begin implementing the code, I went systematically, starting with determining the most frequently occurring words. The first challenge I needed to face was how to break a text down into individual words and store them in a dictionary. This was harder than it looked because I learned that the decoding function converted the entire text to a string. Reading items from this string would only return individual letters, not words. As a result, I had to break the "text" string into words using the split () function. I then realized that the program was counting words differently if they were capitalized or punctuated, so I stripped away all the whitespace and punctuation before counting the word frequencies. One place where I had to choose how to implement specific concepts was stop words. I realized that many of the most frequent words in the texts were common "stop words" such as a, and, the. I downloaded the list of stop words from the class GitHub and then decided to modify the list to add in stop words frequently used in the Gutenberg header and footer that accompanies each text. By doing so, I avoided the need to code instructions for Python to remove the header and footer, which it could not do with individual words from a text. I used another dictionary named e to store all of the stop words and instructed the program to skip any stop words that it encountered in each text. I also decided to allow the user to customize the number of most frequent words that the program would return, to add an extra layer of complexity to the program.

Once I could determine word frequencies, I decided to do some sentiment analysis and text similarity analysis. I needed to first install the nltk Python library and then import a Sentiment Intensity Analyzer software from the vader_lexicon. The program takes a while to perform sentiment analysis on long texts, but it outputs the proportion of positive, neutral and negative words in each text. I wondered how the program classifies most words in the text as neutral, and what criteria it is using to determine if a word is positive or negative rather than neutral. I then moved on to text similarity. After doing some online research, I learned how to perform this analysis using the sklearn library, which I had to pip install first. I then selected five works of fiction from Gutenberg- A Modest Proposal, Beowulf, Frankenstein, works of poet Edgar Allan Poe, and Pride and Prejudice. I downloaded them and stored them as .txt files in VSC. Then I instructed the program to decode all of the texts and calculate tf-idf vectors that would allow it to determine the similarity between each pair of texts. This information could be stored in a matrix, which would make it easier to read and process. Finally, I used the matrix to generate a plot displaying text similarities. Texts that are more similar tend to cluster together, while those that are different appear as outliers.

From the text similarity analysis, I noticed that three of them seem to show high degrees of similarity in language and vocabulary used, while one is slightly different and one is an outlier. Here is a graph of similarities:



The highest pairwise similarity value was 0.849 (85%), while the lowest was still 0.505 (51%). I think that the most similar texts are Frankenstein, Poe, Pride and Prejudice and A Modest Proposal, which were written in the 18th and 19th centuries, while Beowulf is a much older text, so it is probably the outlier near the bottom of the graph. The chart was much easier to understand and interpret than the similarity matrix, so it would be a much better choice, especially if you were working with a large number of texts and wanted to compare their pairwise similarities.

I originally wanted to work with a partner on this project, but I ultimately decided to work by myself to get more practice writing my own code and learn more in-depth. This assignment was useful because I learned more about basic topics in text analysis, which is useful for business analytics. I think that my approach of coding each individual part of the assignment was effective because I was able to focus on one task at a time and leave enough time for fixing errors in the code. I tried to keep the project scope manageable enough to complete by myself. I may have had to do slightly different steps if I was working with another data source, for example social media, but the basic text analysis principles remain the same. One additional feature I could have added in was a function to remove the Gutenberg header and footer, but I found it easier to add those words to the stop words list.