Name: Kiley Fischer, Yuxuan Wang
Prof. Zhi Li
MIS 3640
Assignment 3

Project Writeup

Project Overview:

Our group chose two books from the Project Gutenberg to be our data source. The two books are “Flatland”, and “Treasure Island”. We decided to use the following four techniques to analyze the books: Characterizing by Word Frequency, Computing Summary Statistics, Doing Natural Language Processing, and Text Similarity. We hope to learn how to use python to dig out valuable information from texts, and also understand the newly introduced package, NLTK. In addition, because we worked in a group, we also hope to learn how to do group coding and utilize each other as resources throughout the assignment.

Implementation:

Our code can be divided into four components. The first component includes the first two functions: “process_file” and “skip_gutenberg_header”. The second component includes the following four functions: “total_words”, “total_different_words”, “most_common_ten”, and “different_word”. The third component includes functions “process_word” and “similarity_test”. The first component is used to extract every word in the texts, without the header and the end. We created a histogram to store every different words in its lower case, and the frequency of each word. The second component is used to compute some basic summary statistics for the texts.

For the third component, we use it to test two books’ similarity. Because our function, similarity_test, can only test two strings’ similarity, we firstly use a function, process_word, to

convert the histogram of different words to a string, which included every words in a text. To compare the similarity between two text, we use the module “TfidfVectorizer” to convert the two strings, which include total words of two texts, to a matrix. The matrix has two columns. Each columns represents a book, and each row represents a word in the two text. For every row, if the book has that words, there would be a “1” in its column. If the book does not contain that word, there would be a “0” in its column. We then multiply the matrix by its matrix, and we can get a score, which represents the similarity between the two texts.

One of the design decisions we made is that we include every word in texts to test similarity instead of just different words. Although only testing the different word can increase efficiency, it actually hurts the accuracy. Frequency of each words in a text is also important when considering similarity between two texts. For example , if “breakfast” appears 11 times in a text, but only 1 times in another text. Although both of them have this word, there is still a lot of difference.

In addition to the implemented codes mentioned, we used NLTK to run a sentiment analysis. We chose to do this solely on one text which was “Treasure Island”. Another decision we had to make when creating this code was to use a portion of the text instead of the whole book. We did this for testing purposes. We ran the code on a test file of the original text made up of 800 lines which was able to run a lot quicker than if we had decided to do this analysis on the whole book. This code includes the steps of “process file”, “skip gutenberg header”, “process word” and “sentiment analysis”. The most important part of this code was preparing the text file to be understood and able to be analyzed. This code then analyzed the text file and categorized the words as being “positive”, “negative”, “compound” or “neutral”. These scores determine if the sentiment of the overall text is ultimately positive or negative.

Result:

In the summary statistics, we find out the 10 most frequent words in each text, and also discover the difference between each book's 10 most frequent words. The two books we used evidently very different in regards to content and word choice. "Flatland" is a science fiction book, which was published in 1884. On the other hand, "Treasure Island" is an adventure novel written in 1882. However, the 10 most frequent words in two books are highly similar. Only three words, "that", "is", "my", appear frequently in one book, but do not appear in another one.

One of the interesting program we have deeveloped tests the similarity of the two texts. The result we get is 0.2694 out of 1, which indicates that the two books are highly different. The result meets our expectation to the test, hence the two book are telling totally different stories.

The NLTK Sentiment Analysis program yielded interesting results in regards to the "Treasure Island" test file. The polarity scores that we focused on the most for the text analysis were "positive", "negative" and "neutral". The book text was analyzed as (pos: 0.072, neu: 0.851, neg: 0.077), shown below. These values tell us that when compared to dictionary words that fall within these three categories, a majority of the words in "Treasure Island" are seen as neutral. Additionally, there is almost an equal amount of positive words as there are negative. Given the results, it is hard to determine the polarity of the text since there is little differentiation between positive and negative. Thus, we found that it could be more effective to use this method of text analysis on social media platforms where there are trends to be found, instead of a book.

Reflection:

Throughout the assignment process, we did a good job of preparing the data. For each code written we were able to prepare the data in a way that it could be understood by the system. Additionally, we were able to learn from mistakes and problem solve to fix bugs in our code while getting help from each other while doing so. Something that we could have done better would be to create a visual output of some sort to make the presentation of data more interesting and appealing. This wasn't done because it took a while for us to get the codes to run "bug free". We appropriately planned our project by deciding to use two texts from the Gutenberg site. We decided to analyze them separately while also comparing the texts to each other. Throughout the project, we have learned that when coding it is important to try and fail multiple times while fixing the bugs that appear in order to develop a code that will run. We also realized that having someone else to help code is important as you can combine what you know or do not know. The team dynamic worked well throughout the project. We each picked a text file to use for the project. We then each did a portion of the coding. Additionally, we both worked equally on the write up portion. When one person did not understand the code, the other willingly compensated and offered help which was a great group member quality. Overall, the project and group dynamic went well throughout the process.