Project title: Text Analytics

Project members: Chaitanya Kovuri

Accomplished: Most of the goals mentioned in the project proposal were accomplished. In addition to those goal, I have added 6 more insights into the text analysis. To be more specific about my goals and process, I used '.txt' e-books from Project Gutenberg to parse and analyze. I did so, by using an external Haskell parsing library: *Attoparsec*. The text is then divided into lists of words, sentences and paragraphs. And the following are the insights I was able to analyze based on the text: **1)** Number of words in the text **2)** Number of sentences **3)** Average Sentence length **4)** Number of paragraphs **5)** Longest word in the text **6)** Top 10 words with highest frequencies **7)** Number of word occurrences from a user requested words list **8)** Comparing two e-books for similar words **9)** Gather the 'unique' words of the text, by comparing it with 3000 commonly used words **10)** Average word length **11)** Displaying words of length 'n' **12)** Word Similarity index between two longest words from 2 different e-books, on a scale of 0 to 1 (Based on Jaro Distance[1]) **13)** Lexical Density of the text.

Not completed: There are particularly 2 goals that I couldn't achieve. These are: Plotting a pie chart and replacing a word. A pie chart couldn't be accomplished due to the lack of reliable Haskell chart packages. Most 'popular' packages such as chart-diagram, google chart are currently broken and the next available package *Matplotlib* has issues and lacks documentation for Haskell. And as for the word replacement function, I could have accomplished it if I did not make too many last minute changes.

Code accounting: All of the project code is present in one file: *Main.hs.* It contains all the non-trivial functions and some print statements intended to display the analysis. All the non-trivial code accounts to 122 lines as of submission. There are several functions that help in analyzing the text and one function I would like to highlight is: *"wordSimilarity".* This function is based on calculating the *Jaro Distance between two strings.* It ranks the similarity on a scale of 0 to 1; O being the least similar strings and 1 indicating identical strings. I think this is interesting, because most of the other functions just involve in handling the lists.

Sample Inputs and Outputs: The repo contains two e-books of huge amount of text, named *sample1.txt* and *sample2.txt*. And another important input file is the *common*.txt, that contains most common words. The rest of the parameters depending on the insight, are explicitly written in *Main.hs*. The result of the e-book analysis is found in the *output.txt* with clear descriptions.

Instructions for running the code: The program requires *Attoparsec* library, which can be fetched using cabal install. To run the code, just compile it and type "main" in the GHCi. If you want to change the parameters such as the name of the text file, it needs to be edited under the main function in Main.hs. The comments on the code, regarding this are clear.