Nicholas Achin
COMPSCI 446
Tokenization Report

1. For tokenization, I used the FileReader package to parse character by character of the input file. First I lowercased every character and checked if characters were periods so I could remove them and properly tokenize acronyms. As I loop through, I checked for empty spaces and non-alphanumeric characters and identified them as token separators. Using a token String placeholder, I added the character to the token String if the character wasn't a separator. Otherwise, I added the current token String to the token ArrayList and reset the placeholder. After properly tokenizing the input, I converted the stop word input file to an ArrayList using the tokenizer. This may have been inefficient but the stop word list had no token discrepancies. Next I ran the token list and the stop word list through my stop word remover method. I looped through the token list and checked if the stop word list contained the word. It would've been slightly more efficient to do this when I tokenized the word originally so I wouldn't have to add stop word to the original list. After removing all the stop words, I implemented the Porter Stemmer that was heavily outlined in the book and discussed in class in depth. For the Porter Stemmer, I used String methods to check for suffix endings and the substring method to replace the suffixes that matched my logic statements.
2. Software Libraries
    1. java.io.File
        1. I used this library to access the input files.
    2. java.io.FileReader
        1. I used this library to read in input files to my program.
    3. java.util.*
        1. I used the ArrayList to store the tokens in a list. I also used HashMap to keep track of frequencies in MonteCristo.
    4. java.io.IOException

        1. I used this to handle input and output exceptions while accessing files.
    5. java.io.FileWriter
        1. I used this library to write to the output files.
3. Changes to Tokenization or Stemming Rules
    1. I would add a rule that eliminates single characters from the output. Specifically, there was a lot of singular “s”’s in the output because of possessive words. This high frequency token does not really help describe a document.
    2. Another stemming rule I would implement is to remove single digit numbers. The rule would have to ignore multi-digit numbers because dates and years can be very important to a document’s and querying.
4. Count of Monte Cristo
    1. After reviewing the output of the Monte Cristo frequencies, the majority of the words are common English words like door, hand, know, etc. However, there are definitely some words used a lot that are particular to the time and story (i.e. old English words and characters/place of interest names).
    2. I removed the stop word list to review what the list would be without removing them. The list was flooded with stop words that wouldn’t give you an insight into what words are frequent in Monte Cristo. To say a stop word list is complete for all documents is a bit naive. In some cases, the stop word list could remove pertinent words to a document.
5. Graph