Tokenization Report

Eric Doughertty

1) The code for both Part A and B is located in the src/* file in the source code Part A is located in the Methods tokenize(), stemmer(), stopRemoval(), isletter(), isSeperator(), and isVowel(). While Part B is located in the Methods findVocab() and wordCounts().

2) The tokenizer works using a Scanner that reads in from a text file provided to the object through the input of string when the object is created in the main method. This was done as it was easier and more efficient to directly input the string in the code rather than have a user input prompt. The scanner reads the file line by line. For each line the tokenizer iterates over each character in the string. For each character, all possibilities are checked. First, it checks if the char is part of an abbreviation. This is done using a series of if statements checking if it follows the pattern letter-period-letter-period-letter-period because of the implementation it will only recognize the strictest of definitions like U.S.A. or A.B.C. but not PH. D. After abbreviations it checks if its a letter and if it is it adds it to the current token. Then if it's a separator it ends the current token adding it to an ArrayList of tokens are resets the current token. Then Stopword removal is done. For this process, the list of stop words is read into a set and then the entire Array of tokenized words is checked as to whether it is in the set, if it does it's removed. The output of that is stemmed using porters algorithm and the tokenization process is complete.

3) The only library used in the code is the built-in java library. The specific class are java.io.File, java.io.FileWriter, java.io.IOException ,java.io.PrintWriter, java.util.ArrayList, java.util.HashMap ,java.util.HashSet ,java.util.Map ,java.util.Scanner; ,java.util.Set. The classes File, FileWriter, Scanner, and PrintWriter were all used to read and write to a file. All other are data structures used to help store and sort data.

4) The first change to the tokenization or stemmer process would be changing the way Apostrophes are handled. In the current algorithm words are split at an apostrophe so Don't becomes Don and t. This creates 2 outputs that will not be removed by stopping as neither are stops words but are meaningless. Combining them into one word Dont would provide a word with more meaning and create fewer one-letter tokens. Second would be a rule change to the stemmer specifically the rule that states if it's short add an e. While the rule helps in some cases the lack of definition for the term short leaves that statement vague and arbitrary.

5) The vast majority of the tops terms do relate to the story. Words like ship, whale, boat, sea, and Ahab are all in the top spots. However, the most notable exception is the term s. As a result of the tokenization process treatment of Apostrophes, it is the top term and has no relation to the text. The only words that are clearly stop words would be the single letter token, however, a few such as two and say have little unique meaning to the

text. I believe that it would be impossible to create one stopword list for all documents and searches as depending on the document the word might have more or less important and there is too much variation in documents to have a single master list.

6) Total Words = 100195 Start Words = 0
   Vocab = 13216 Start Vocab = 0

Vocabulary Growth for the novel Moby Dick