Information Retrieval

Assignment 1 - Text Processing Functions

Due date: 1/21 at 11:59pm

This assignment is to be done individually. You cannot use code written by your classmates. Use code found over the Internet at your own peril -- it may not do exactly what the assignment requests. If you do end up using code you find on the Internet, you must disclose the origin of the code. **Concealing the origin of a piece of code is plagiarism**. Use the Message Board for general questions whose answers can benefit you and everyone.

Project skeleton:
http://www.ics.uci.edu/~lopes/teaching/cs221W14/projects/project1.zip

General Specifications

1. You can use any programming language, but Java is strongly encouraged. This spec has all sorts of helpers for Java. Also, the next homework will use a Java crawler, so you may want to use this homework to brush up your knowledge of the language.
2. If you use Java, your solution must fill out the program skeleton provided.
    a. Fill in each method according to its Javadoc specification.
    b. Feel free to create additional methods / classes where necessary
3. If you don't use Java, you should produce a similar skeleton to start with and fill it out.
4. At points, the assignment may be underspecified or slightly wrong. In those cases, make your own assumptions and be prepared to defend them.

Part A: Utilities (20 points)

Write a method that reads in a text file and returns a list of the tokens in that file. Write a method to print out frequency results.

Package: `ir.assignments.two.a`
File: `Utilities.java`
Method: `tokenizeFile(File)`
Method: `printFrequencies(List<Frequency>)`

Part B: Word Frequencies (20 points)

Count the total number of words and their frequencies in a token list.

Package: `ir.assignments.two.b`
File: `WordFrequencyCounter.java`
Method: `computeWordFrequencies(ArrayList<String>)`

Part C: 2-grams (30 points)

A 2-gram is two words that occur consecutively in a file. For example, *two words, words that* and *that occur* are all 2-grams from the previous sentence.

Count the total number of 2-grams and their frequencies in a token list.

Package: `ir.assignments.two.c`
File: `TwoGramFrequencyCounter.java`
Method: `computeTwoGramFrequencies(ArrayList<String>)`

Part D: Palindromes (30 points)

A palindrome is a words or phrase that read the same in both directions. For example: *eye* is a palindrome and so is *Do geese see god*.

Count the total number of palindromes and their frequencies in a text file.

Package: `ir.assignments.two.d`
File: `PalindromeFrequencyCounter.java`
Method: `computePalindromeFrequencies(ArrayList<String>)`

Once you have implemented your palindrome counting algorithm, please perform a short analysis of its runtime complexity (does it run in linear time relative to the size of the input? Polynomial time? Exponential time?) This analysis should go in the **analysis.txt** file in this package.

Submitting Your Assignment

Your submission should be a single zip file submitted to EEE. This zip file should match the skeleton zip file provided with the assignment, with the addition of your implementations of the four sections.

Late Submission Policy

-1% for every hour past the deadline for the first 24 hours. Flat -25% until 1/28 at 11:59pm. No submission accepted past that date/time.

Grading Process

You will meet with the grader for about 10 minutes during the week starting on 1/22. During that meeting, be ready for the following:

1. Show your program running on your own computer, on an input file provided by the grader on a USB stick
2. Answer questions about your program (as submitted to EEE) and its behavior (during the meeting)

You should **not** continue to work on your program once you submit it to EEE.

Evaluation Criteria

Your assignment will be graded on the following four criteria.

1. Correctness (40%)
   a) How well does the behavior of the program match the specification?
   b) How does your program handle bad input?
2. Understanding (40%)
   a) Do you demonstrate understanding of the code?
3. Efficiency (20%)
   a) How quickly does the program work on large inputs?