Programming Assignment #8
CS253, Fall 2017
Due: Thursday, Nov. 16th

"*Document Retrieval*"

Motivation & Description

Finally! The final project (minus the efficiency assignment). You are writing a system that takes in a query document and a set of background documents. The goal is the find the background documents that is (1) on a similar subject to the query document and (2) at a desired reading level. For example, imagine that you have a document on a subject that you have to talk to your child about. Then you might want to find a similar document, but at a lower reading level. Alternatively, you might have a simple, introductory document, and you want to find a more sophisticated text on the same document. This is what the tool you are building is for. (Granted, it would be nicer with a web interface….)

Task

Your program will take two filenames and two numbers as arguments, and write out a single filename to std::cout. The first input file is the document you are comparing to. It is a standard text document. The second file is more complicated. It contains a series of filenames separated by whitespace, so that they can be read using the >> operator. (We will probably put one file name per line, but your program shouldn't count on that.) The first filename in the second file is the name of the exceptions file. All other filenames in the second file are the names of background documents. The third and fourth arguments are numbers, specifically the lowest reading level (as defined by the Coleman-Liau Index) allowed and the highest reading level. It is an error if the third value is greater than the fourth.

Your program should analyze the reading level of every background document, as in PA7. Then, for documents whose reading level is within the target range (as determined by the 3rd and 4th arguments), compare them to the target document using TFIDF (as in PA8). Print to std::cout the filename of the most similar document to the target document that is within the target reading level range.

Hints

- Suddenly, there are lots of error cases again. The reading level range could be empty (if the 3rd argument is bigger than the 4th). There could be no background documents within the given reading level. The exceptions file could be mis-formatted. A background file might not exist. A background document could have no words….
- PA9 is the efficiency assignment. *It will have the exact same task description at PA8.* However, it will be measured on speed as well as correctness. In particular, we will grade it on several test cases. For each case, you will go no points if your program produces the wrong filename. If you program produces correct output for a case, you will get at least 50% of the points for that case. The other 50% will determined by your program's speed. (No, I won't tell you what the mapping is from time to points.)

Submitting Your Work

To submit your program, first create a single tar file containing all of your source files (i.e. your .cpp and .h files) and your makefile, but not your compiled files (no executable or .o files, please). Then submit the tar file as PA8 on Canvas.

Grading Your Homework

To grade your assignment, the GTAs will unpack your archive in an empty directory. They will compile your code by typing 'make'. This command must compile your code from scratch, and it must produce an executable called PA8 in a directory called PA8. If your code does not compile using make, you will receive a zero for the assignment. Compiling your code should not produce any warning messages. Although we will not deduct points for warning messages, we may in future assignments so get in the habit of making sure you code compiles cleanly. Assuming your program compiles, it will be graded by how it performs on test cases, including test cases with errors in one or both of the input files. If the input file contains errors, you will receive full credit if your program returns -1 to main. (In case of an error, it should also print a meaningful error message to std::cerr.) If the input does not contain errors, your program should return 0 to main.

Policies

All work you submit must be your own. You may not submit code written by a peer, a former student, or anyone else. You may not copy or buy code from the web. The department academic integrity policies apply.

You may not submit your program late. To receive credit, it must be submitted on Tuesday, Nov. 9th. There is no late period. The exception is an unforeseeable emergency, for example a medical crisis or a death in the immediate family. If an unforeseeable emergency arises, talk to the instructor.