CSE508 - Winter 2015 Assignment 2

February 6, 2015

Total marks: 100

Instructions:

1. **Deadline for submission:14 Feb 2015, 23:55 IST**
2. **All assignments have to be done individually. No collaborations are expected. We are going to verify it by software, as well as manually.**
3. **A complete submission is defined as the one that consists of two files: (i) your code (tarred and gzipped) and (ii) the TXT or PDF document containing instructions to execute your program as well as the summary of the results you obtained. If your code does not compile/run when we follow your instructions, you will be awarded 0 marks.**
4. **Your code will be tested on a Linux workstation with 4GB of RAM, and 500GB of disk. It is expected that your program run on this. Should you require any specific library, you should include it in your submission.**
5. **Each day delay in the complete submission of your assignment will result in a reduction of 10% of total marks. No repeated OR incremental submissions are allowed.**
6. **Please note that developing a fancy user-interface is not going to fetch you any additional points. It is required that you take all arguments from the command-line, or from a configuration file.**

Questions

1. **Inverted Index Construction**
   For this question, you are expected to use a set of files derived from King James' version of Bible, available from
   `http://192.168.1.50/assignments/`
   as a set of 12 files, all of which have to be used. Note that this can be accessed only within the IIITD network.

   You are free to use any of the following languages to implement: **Java, C/C++**. Use of scripting languages such as Shell-scripting, Python, Ruby, Perl, etc. is not allowed.

   Write a program that generates a simple, full-text inverted index from the above text. You can use the following standard punctuation marks for delimiting words in the file: `*,.;:!?'''[]()`. **Do not**: eliminate stop-words and perform stemming.

   The program has to follow the `emit`, `sort`, `invert` steps strictly. At every stage, you have to write the output to a file, and the subsequent step has to read the previously written file to generate the next stage of output. Specifically you have to write the three methods:

   a) **void emit (String directory)**: takes *all* text files (with .txt ending) in the specified **directory**, and generates a file named *emit.out* which has for **each line**, an entry of the form `<term>:<filename>`

   b) **void sort (String emitfilename)**: takes a *single file* which was output in the emit step above, specified as the parameter **emitfilename**, and generates a file named *sort.out* which contains all the `<term>:<filename>` entries from the input file, in a lexicographically sorted order.

   c) **void invert (String sortedfilename)**: takes a *single file* which was output in the sort step above, specified as the parameter **sortedfilename**, and generates the final output file *invert_bible.out* which is the inverted index file consisting of `<term>:<filename1>,<filename2>,<filename3>....`, i.e., a term followed by list of filenames where the term occurs.

   In Java and C++, all the above methods will be enclosed in a class named: `SimpleTextInverter`. In C, all the above methods should be implemented in a single file called `simpletextinverter.c`. In C/C++ the signature of the above three methods will be **void emit (char* directory)**, **void sort (char* emitfilename)**, **void invert (char* sortedfilename)**. **60 marks**

2. **Conjunctive and Disjunctive Keyword Query Processing** In the class we discussed an algorithm for conjunctive query evaluation of two keywords: `k1` and `k2` by processing their inverted lists p_1 and p_2, each containing document ids in sorted order. The algorithm was called **intersect** (p_1,p_2).

   a) Write the algorithm **intersect**, and analyze its computational complexity.

   b) Similarly, develop the algorithm for **union** of two inverted lists, and analyze its computational complexity.

   40 marks