CSE508 - Winter 2015
Programming Assignment 1

January 23, 2015

Total marks: 100

Instructions:

1. **Deadline for submission:30 Jan 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.**

1. **Collection Processing**
   For this question, you are expected to use the **CACM** article collection available from `http://www.search-engines-book.com/collections/`.

   Your submissions will be evaluated over a different test collection (HTML tarball similar to the collections above) - so, your programs should not be very specific to the given collection.

   Although efficiency of your program is not the prime concern, the most efficient (as measured by the time taken to process a given collection) and correct submission will receive 5% bonus marks.

   You are free to use any of the following languages to implement these – Java, C/C++. If you want to use another programming language (e.g., Scala, C#, Haskell, etc.), please consult your TA/Instructor to verify if it can be permitted. Use of languages such as Shell-scripting, Python, Ruby, Perl, etc. is not allowed.

   a) Write a program to preprocess the collection, and generate a simple text sequence representation for each document. Specifically:

      i. Remove all the non-content HTML tags (such as HREF, META, etc.). Use your judgement to decide how the removal of tags to be done.

      ii. Tokenize the text - for this write a function called `tokenize (char* document)` which takes a document represented as a character sequence, and returns the list of terms in it.

   30 marks

b) Using the methods developed above, build a method called `printProgressiveWordStats(String filename)` that takes as input a `filename` containing a HTML document collection in a Zipped format, and processes it. It should print (i) the document length in terms of number of terms for each document in the collection, and *in the end*, (ii) the list all unique terms in the collection in their order of frequency.

The output should be in the format:

```
DOCUMENT STATISTICS
<doctitle> : <document_length>
...
TERM DISTRIBUTION
<term> : <frequency>
...
```

Does your observed result from above support Zipf's Law? What is your estimate for the value of c? **30 marks**

c) Now, use the English stopword list available from `http://http://snowball.tartarus.org/algorithms/english/stop.txt` to remove all the stopwords from the collection during its tokenization. *Generate the same output as above*. What are your observations from the outputs? **20 marks**

You are free to use external libraries for HTML parsing (if required). But all other methods should be entirely written by you. Transform the method signature of `tokenize` method appropriately for the language of your choice.

2. **Canonicalizing the Boolean Expressions**
We discussed in the class that any Boolean keyword query can be expressed using only *Intersection*($\cap$), *Union*($\cup$), and *Negation*($\neg$) operators. Rewrite the following Boolean keyword queries accordingly:

a) $K_1 \wedge \neg K_2$

b) $(K_1 \vee K_2) \vee (\neg K_2 \wedge K_3)$

20 marks