Assignment 3

This is an individual assignment. If you get help from others you must write their names down on your submission and explain how they helped you. If you use external resources you must mention them explicitly. You may use third party libraries but you need to cite them, too.

Date posted: Sunday October 30, 2016

Date Due: Monday November 7, 2016 11:59pm

Goal: Implementing your own inverted indexer. Text processing and corpus statistics.

Description:

Task 1: Generating the corpus: In this task you will be using the raw Wikipedia articles that you downloaded in HW1 to generate the clean corpus following the instructions below:

1- Parse and tokenize each article and generate a text file per article that contains only the title(s) and plain textual content of the article. Ignore/remove ALL markup notation (HTML tags), URLs, references to images, tables, formulas, and navigational components.

2- Each text file will correspond to one Wikipedia article. The file name is the same as the article title, however, without underscores or hyphens, e.g., http://en.wikipedia.org/wiki/Green_Energy → GreenEnergy.txt

3- Use case folding. Remove punctuation from text but preserve hyphens. Retain punctuation within digits (mainly ",", ".", and any other symbols you deem necessary).

Task 2: Implementing an inverted indexer and creating inverted indexes:
Implement a simple inverted indexer that consumes the corpus in Task 1 as input and produces an inverted index as an output.

- Term frequencies (*tf*) are stored in inverted lists:
  WORD → (*docID*, *tf*), (*docID*, *tf*), ...
  <u>Important:</u> See WORD definition below

- For this assignment, you don't need to consider term positions within documents.
- Store the number of tokens in each document in a separate data structure.
- You may employ any concrete data structures convenient for the programming language you are using, as long as you can write them to disk and read them back in when you want to run some queries.

  WORD is defined as a word n-gram, and *n* = 1, 2, and 3. Therefore, you will have **three inverted indexes**, one for each value of *n*.

Task 3: Corpus statistics:

1- For each inverted index in Task 2, generate a term frequency table comprising of two columns: *term* and *tf.* Sort from most to least frequent

2- For each inverted index in Task 2, generate a document frequency table comprising of three columns: *term*, *docID*, and *df.* Sort lexicographically based on term.

   Therefore you will generate **six tables in total**: Two tables for single-word terms (word unigrams), two tables for word bigrams, and two tables for word trigrams.

3- Generate a stoplist using the unigram data. How would you choose your cut-off value? Briefly justify your choice and comment on the stoplist content.

What to hand in?

1- Your source code for solving this assignment

2- A readme text file explaining in detail how to setup, compile, and run your program and what design choices you had to make.

3- The six tables in Task 3

4- The stoplist and explanation from Task 3

Bonus (optional):

This is an all-or-nothing extra credit. If performed correctly, 5 points % will be added to your score of HW3 only.

Plot the Zipfian curves for all three variations (*n*=1, 2, and 3)