College of Computing and Information Science
Northeastern University
Spring 2019

CS6200/IS4200
Information Retrieval
February 24, 2019

Homework Assignment #3
Due Date: Monday, March 11th, 2019 @ 11:59pm

Instructions:

- The assignment is due on the time and date specified.
- This is an individual assignment. You may discuss the problems with your friends, but the code, analysis, interpretation and write-up that you submit for evaluation should be entirely your own.
- You are encouraged to use the Piazza discussion board and seek help from TAs and instructors to get clarifications on the problems posed.
- If you receive 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.

Implement your own inverted indexer

For this assignment, you will use the document collection you extracted for Task 1 of your previous assignment using the URLs provided in BFS.txt. Be sure to use the parsed, tokenized version of the documents, with case-folding, and punctuations removed (except for hyphens).

Task 1 (50 points): **Implement an inverted indexer and create inverted indexes.**

a) Implement a simple inverted indexer that consumes the corpus from HW2: Task 1 as input and produces an inverted index as an output. Your index will contain the document IDs and term frequency within the document.
   Term frequencies (tf) are stored in these inverted lists:
   TERM → *(docID, tf), (docID, tf), …*
   A **TERM** is defined as a *word n-gram*, where n = 1, 2, and 3. Therefore, you will have three inverted indexes, one for each value of n.
b) Store the number of terms in each document in a separate data structure.
c) 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.

d) Generate a positional inverted index for the unigrams. Encode the positions using the gaps between the occurrences.

Task 2 (25 points): **Use Positional Index to Process Conjunctive Proximity Queries**

a) Implement a program that uses a positional inverted index in Task 1-d) above to retrieve the list of documents that contain a pair of terms within a proximity window *k*.

b) Using the positional index created in Task 1-d) above:
   i. Find the list of documents that contain *both* ***space*** and ***mission*** within $k = 6$ and $k = 12$ unigram tokens of each other. [Here, with *k* tokens implies there are at most *k* tokens between the two terms, and the terms may appear in any order.]
   ii. Find the list of documents that contain both ***earth*** and ***orbit*** within $k = 5$ and $k = 10$ unigram tokens of each other.

   Note 1: Terms are not case-sensitive.
   Note 2: You will produce 4 lists of Document IDs in total.

Task 3 (25 points)**: Compute Corpus Statistics; Propose Stop Lists**

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

b) For each inverted index in Task 1-a), generate a document frequency table comprising of three columns: *term*, *docIDs*, and *document frequency*. Sort lexicographically based on term.
   Note: For tasks 3-a) and 3-b), you will generate six tables in total: two tables for word unigrams, two tables for word bigrams, and two tables for word trigrams.

c) Generate three stop lists, one per index from Task 1-a). How would you choose your cutoff values? Briefly justify your choice and comment on the stop lists' contents.

What to hand in?

1- Your source code for solving all parts of 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 4 inverted index files from Tasks 1-a) and 1-d), and the [Document ID, Term] table generated in Task 1-b).
4- The four Document ID lists generated in Task 2-b).
5- The six tables generated in Tasks 3-a) and 3-b).
6- The stop lists and explanations from Task 3-c).