Cmpe 493 Introduction to Information Retrieval, Spring 2023
Assignment 1 - A Simple Search System for Phrase and Proximity Queries
Due: 03/04/2023, 23:59

In this assignment you will implement a document retrieval system for phrase and proximity queries using the **positional** inverted indexing scheme. You will use the Reuters-21578 data set (provided at Moodle). Reuters-21578 contains 21578 news stories from Reuters newswire. There are 22 SGML files, each containing 1000 news articles, except the last file, which contains 578 articles. You should perform the following steps:

1. Pre-processing the Data Set: The text of a news story is enclosed under the <TEXT> tag. You should use the <TITLE> and the <BODY> fields to extract the text of a news story. Implement your own tokenizer to get the tokens from the news texts and perform normalization operations including case-folding and punctuation removal. You can use the list in "string.punctuation" for punctuation removal for Python. Please use "latin-1" encoding while you read the .sgm files due to the corruption in one file.

2. Building the Inverted Index: Instead of taking each SGML file as a document unit, you should index each news article as a separate document and use the $NEWID$ field as document IDs. Then, you need to create an inverted index consisting of the dictionary and the postings lists. You should store your inverted index as a file(s) and during query processing you should only use the inverted index, not the original Reuters-21578 dataset. We will delete the "reuters21578" data set while testing the query processor. Note that the inverted index construction and query processor should be designed as two separate modules. That is, the query processor should NOT construct the inverted index each time it is run. It should just use the inverted index file(s) generated by the indexing module.

3. Implementing a query processor: You should implement a query processor for phrase and proximity queries. That is, the queries will be of the following two types (here w_i is a single-word keyword):

   (i) Phrase query: "$w_1\ w_2 ... w_n$"
   (ii) Proximity query: $w_1\ k\ w_2$

   Note that phrase queries will be expressed with double quotation marks. In the proximity queries k is a constant number which implies that w_1 is within k words of w_2. That is, a matching document will have w_1 in the text followed by at most k words, followed by w_2. Note that the order of the query terms is not important for the proximity queries: $w_1\ k\ w_2$ is the same query as $w_2\ k\ w_1$. For example, w1 0 w2 is equivalent to the disjunctive phrase query "w1 w2" OR "w2 w1".

   The query processor should return the IDs of the matching documents sorted in **ascending order**.

Here are some example input queries:

- "old crop cocoa"
- old 1 cocoa

You should use Python to implement your search system. We should be able to run your program by following the instructions in your readme file. You have to state the exact commands to run the indexing module and to run the query processing module. You should NOT use any third party libraries, except the ones available in the Python Standard Library.

Submission: You should submit a *".zip"* file named as YourNameSurname.zip containing the following files using the Moodle system:

1. Report:
   (i) Describe the steps you have performed for data preprocessing.

   - How many tokens does the corpus contain before and after case-folding?
   - How many terms (unique tokens) are there before and after case-folding?
   - List the top 100 most frequent terms after case-folding.

   (ii) Describe the data structures (hash, b-tree, trie, linked list etc.) that you used for representing the inverted index (i.e., the dictionary and the postings lists).

   (iii) Provide a screenshot of running the indexing module of your system.

   (iv) Provide two screenshots of running your system for each of the two types of queries.

2. Source code: Commented source code of your document retrieval system. Please do not submit the Reuters dataset.

3. Readme: Detailed readme describing how to run your program including the Python version.

Honor Code: You should work individually on this assignment and all the source code should be written by you. You are NOT allowed to use any available libraries or any code written by other people. Violation of the Honor Code will be strictly penalised.

Late Submission: You are allowed 3 late days (until 06 April 23:59 o'clock) for this assignment with no late penalty. After that, 1 point will be deducted for each late hour (unless you have a serious excuse).