Search Technologies for Data Science: Lab 2

due **11:55 pm on Thursday, October 12 2023**.

The goal of this lab is to develop a search engine for scientific documents.

Dataset

You are dealing with a collection of news documents that are formatted using XML mark-up. Each document consists of a document id, profile, date, headline, and text sections, all identified by xml tags (see sample.xml). Some elements may be missing in some of the documents. You've been given sample.xml containing 5 1-line documents for developing. trec.sample.xml contains 1000 news articles for testing.

Tasks

Your goal is to develop a search engine that allows you to retrieve relevant documents in response to Boolean queries. Your tokenizer should index the <headline> and <text> sections of each document. In order to do this, you need to complete the following:

1. Modify your code for lab 1 so that it:
    a. parses only the headline and text sections of each document.
    b. can run with stopping and/or stemming either on or off
2. Implement an inverted index for the documents. You need to save the following information for each term:
    a. term (pre-processed) and it's document frequency
    b. list of documents in which this term occurred. I suggest you use a linked list to store this data as it will be easier to merge postings for query evaluation later.
    c. for each document, the list of positions where the term occurred in the document (extra credit)
3. Print out for visualization, the output of your inverted index in a text file. Example output can be found in trec.index.txt. Note that this output file was generated without applying stopping or stemming. Also notice that the output is sorted by term. Check your output inverted index when you enable/disable the following:
    a. stopping
    b. stemming
4. Build components that allow you to do each of the following:
    a. Read an index file and load it into memory. If you have difficulty running the search in Jupyter Notebooks, you can just perform text processing in

the notebook and write the inverted file to disk, then run your search code on a lab machine or your own machine.

b. Implement a simple **word overlap** retrieval algorithm using the similarity function below. Run the queries in the file queries.lab2.txt

$$s(Q, D) = \sum_{w} 1_{w \in Q} * 1_{w \in D}$$

c. Implement Boolean search using AND, OR, and NOT. Run the queries in the file queries.boolean.lab2.txt.

d. Report the list of documents retrieved for each query.
   i. How do the word overlap results compare to the Boolean results?
   ii. Where it makes sense, compare and contrast the results for different versions of the same query between the Boolean retrieval and word overlap retrieval (e.g. "middle AND east AND peace" versus "middle east peace")

e. These collections are quite small. What do you expect would happen if you run your system as implemented on a very large collection?

Restrictions

Continue to use **Python** as the programming language for this assignment. You can use other libraries and packages subject to the following conditions:

1. Do not use any software packages or libraries that provide out-of-the-box indexing, or retrieval/search functionality.
2. Do not use any network services. This includes online APIs, spell-checkers, etc.
3. The core logic of the retrieval algorithm (matching queries to documents) should be clearly and identifiably your own work.
4. You are free to use any libraries or external programs that perform generic tasks such as reading / writing of files, manipulation of strings, lists, vectors, matrices, hash-tables and priority queues, as well as sorting and merging methods. Please cite the libraries you use.
5. If you are uncertain as to whether a package qualifies as "generic" or not -- please ask.
6. If you consult external sources, please cite them (this includes personal communication).