CSI4107, Winter 2025

Assignment 1
Due Feb 9, 10pm

Information retrieval system [50 points]

Note: **The work should be done in groups of students and submited via BrightSpace. Once a group member submits, all group members can see the submission.**

You will implement an Information Retrieval (IR) system based on the vector space model, for a collection of documents. You will submit the results of your system on a set of test queries. You also have their ideal answers, the relevance judgments so that you can evaluate the performance of your system before submission (for computing evaluation measures you can use the trec_eval script, the latest version).

We will use the **Scifact dataset** available here. It is part of the BEIR collection. Here is the link to the whole collection and associated code. Please use only the Scifact dataset (not any of the other datasets). The corpus/collection of the Scifact dataset consists of scientific claims and abstracts. Read more about the format of the collection, queries, and relevance judgements in this paper. Use only test queries (the queries with odd numbers 1.3.5, …), not any training queries. Size of the dataset:

- corpus.jsonl: 7,917 KB
- queries.jsonl: 205 KB
- test.tsv: 6 KB

Implement an indexing scheme based on the vector space model, as discussed in class. Alternatively, you can use an existing IR system from the Internet and adapt it to work on this collection and queries (this system can use the vector space model or a more advanced model). The steps pointed out in class can be used as guidelines for the implementation. For weighting, you can use the tf-idf weighting scheme ($w_{ij} = tf_{ij} \cdot idf_i$). **Alternatively, you can use a similar formula for that is known to work well and lead to higher retrieval performance, called BM25.** For each query, your system will produce a ranked list of documents, starting with the most similar to the query and ending with the least similar.

Step1. [5 points] **Preprocessing**: Implement preprocessing functions for tokenization and stopword removal. The index terms will be all the words left after filtering out markup that is not part of the text, punctuation tokens, numbers, stopwords, etc. Optionally, you can use the Porter stemmer to stem the index words.

- Input: Documents that are read one by one from the collection
- Output: Tokens to be added to the index (vocabulary)

Step 2. [10 points] **Indexing**: Build an inverted index, with an entry for each word in the vocabulary. You can use any appropriate data structure (hash table, linked lists,

Access database, etc.). An example of possible index is presented below. Note: if you use an existing IR system, use its indexing mechanism.

- Input: Tokens obtained from the preprocessing module
- Output: An inverted index for fast access

Step 3. [10 points] **Retrieval and Ranking**: Use the inverted index (from step 2) to find the limited set of documents that contain at least one of the query words. Compute the cosine similarity scores between a query and each document.

- Input: One query and the Inverted Index (from Step2)
- Output: Similarity values between the query and each of the documents. Rank the documents in decreasing order of similarity scores.

Run your system on the set of test queries. Include the output in your submission as a file named Results. [10 points]
The file should have the following format, for the top-1000 results for each query/topic (the queries should be ordered in ascending order):

```
query_id Q0 doc_id rank score tag
```

where: query_id is the topic/query number, Q0 is an unused field (the literal 'Q0'), docno is the document id taken from the doc_id field of the segment, rank is the rank assigned by your system to the segment (1 is the highest rank), score is the computed degree of match between the segment and the topic, and tag is a unique identifier you chose for this run (same for every topic and segment). Example:

```
1 Q0 doc_id1 1 0.8032 run_name
1 Q0 doc_id2 0.7586 run_name
1 Q0 doc_id3 3 0.6517 run_name
…
```

The relevance feedback file (expected solution) contains one or more relevant cdocuments for each query (any other documents are considered non-relevant). Example:

query-id	doc-id	score
1	31715818	1
3	14717500	1
5	13734012	1

…

Resources: Here is a list of stopwords you could use in Step 1. In Step1 you could use a regular expressions library to help with the filtering. If you want to use stemming, you can use the Porter stemmer. Stemming might increase the performance a bit.
One way to increase the performance is to add a pseudo-relevance feedback loop.

Note: Feel free to add any optimizations or components that could improve the evaluation scores. Please explain them in your report and submit the Results file only for your best run. But **please do not use neural information retrieval methods** (based on deep learning, transformers, BERT or GPT models). We will use

them in Assignment 2.

Optionally, you can use some code from a former student that ran experiments on the BEIR collection (ignore the deep learning models for Assignment 1, you can use them in Assignment 2). Read his report here.
You can use any resources from the Internet, as long as you explain in your report how you used them (compilation, installation, adaptation, etc.). You can use any IR system available on the Internet (ElasticSearch, Lucene, Lemur, Terrier, Inquire, etc.).

Submission instructions:
- write a README file (plain text, Word format, or pdf) [10 points for this **report**] including:
  * your names and student numbers **Specify how the tasks were divided between the team members [5 points] (if this info is not provided, the penalty is 5 points).**

  * a detailed note about the functionality of your programs,
  * complete instructions on how to run them
  * explain the algorithms, data structures, and optimizations that you used in each of the three steps. How big was the vocabulary? Include a sample of 100 tokens from your vocabulary. Include the first 10 answers to the first 2 queries. Discuss your results.
  * Include the Mean Average Precision (MAP) score computed with trec_eval for the results on the test queries.
- produce a file named Results with the results for all the test queries for your best run, in the required format.
- for the queries/topic, use only the titles for one run and titles and full text for another run. Discuss which gives better results.
- submit your assignment, including programs, README file, and Results file, as a zip file in BrightSpace (only one team member needs to submit).
- don't include the initial text collection.

Have fun!!!