Information Retrieval

2022/2023

Assignment 1

Submission deadline: **21 October 2022**

For this assignment you will implement a text indexer following the SPIMI approach.

The corpus for this assignment was compiled from the 'BioASQ Task B Dataset', described here http://bioasq.org/participate/challenges

1. Create a corpus reader that iterates over the documents and returns, in turn, the contents of each document.
   For this assignment consider only the `title` and `abstract` as content (indexed) fields and use the `pmid` as the document identifier.

2. Create a tokenizer based on tokenizing rules specified by you. It should also include different settings to allow adjusting its behaviour. As minimal requirements, your tokenizer should allow the user to apply:
   i. A minimum length filter that removes tokens with less than the specified number of characters;
   ii. Normalization to lowercase;
   iii. A stopword filter, allowing the user to specify a file listing the stopwords to use;
   iv. Porter stemmer (from Snowball, using the PyStemmer wrapper, or from NLTK).

3. Implement an indexer that follows the SPIMI approach. To signal the dumping of indexing blocks to disk you may consider either: 1) the amount (or %) of available memory; 2) the memory required by the index, estimated by the number of posting and the size of the posting in bytes.
   Note: You should experiment with different values for strategy 1) or 2), to simulate a system with lower memory resources.

4. Index, separately, each of the files listed below and gather the following statistics:
   a) Total indexing time
   b) Merging time (last SPIMI step)
   c) Number of temporary index segments written to disk (before merging)
   d) Total index size on disk
   e) Vocabulary size (number of terms)

   Files to use:
   pubmed_2022_tiny.jsonl.gz (128 MB)
   pubmed_2022_small.jsonl.gz (1.3 GB)
   pubmed_2022_medium.jsonl.gz (4.1 GB)
   pubmed_2022_large.jsonl.gz (8.8 GB)

See further instructions in the next page

Instructions:

- **Modelling**, code **structure**, **organization** and **readability** will be considered when grading your project
- **Comment** your code; and make sure you include your name and student number
- Write **modular** code
- Favour **efficient** data structures
- Use **parameters**, preferably through the command line
- Make sure all your programs run correctly
- Submit your assignment by the due date using Moodle