Universidade Federal de Minas Gerais
Departamento de Ciência da Computação

TCC/TSI/TECC: Information Retrieval

Programming Assignment #1
Document Indexer

Deadline: Apr 27th, 2023 23:59 via Moodle

Overview The goal of this assignment is to implement the document indexer module of a web search engine. In addition to the source code of your implementation, your submission must include a characterization of the index built for a mid-sized document corpus provided as part of this assignment.

Implementation You must use Python 3.11 for this assignment. Your code must run in a virtual environment **using only the libraries included** in the provided `requirements.txt` file. Execution errors due to missing libraries or incompatible library versions will result in a zero grade. To make sure you have the correct setup, you can test it in one of the Linux machines provided by the Department of Computer Science[1] using the following commands:

```
$ python3 -m venv pa1
$ source pa1/bin/activate
$ pip3 install -r /path/to/requirements.txt
```

Indexer Your implementation must build upon the provided `indexer.py` script, which will be executed in the same virtual environment described above. The `indexer.py` module will be executed as follows:

```
$ python3 indexer.py -m <MEMORY> -c <CORPUS> -i <INDEX>
```

with the following arguments:

- `-m <MEMORY>`: the memory available to the indexer in megabytes.
- `-c <CORPUS>`: the path to the corpus file to be indexed.

[1] `https://www.crc.dcc.ufmg.br/infraestrutura/laboratorios/linux`

- `-i <INDEX>`: the path to the directory where indexes should be written.

At the end of the execution, your `indexer.py` implementation must print a JSON document to standard output[2] with the following statistics:

- `Index Size`, the index size in megabytes;
- `Elapsed Time`, the time elapsed (in seconds) to produce the index;
- `Number of Lists`, the number of inverted lists in the index;
- `Average List Size`, the average number of postings per inverted list.

The following example illustrates the required output format:

```
{ "Index Size": 2354,
  "Elapsed Time": 45235,
  "Number of Lists": 437,
  "Average List Size": 23.4 }
```

Document Corpus The corpus to be indexed comprises structured representations (with id, title, descriptive text, and keywords) for a total of 4,641,784 named entities present in Wikipedia. These structured representations are encoded as JSON documents in a single JSONL file available for download.[3] To speed up development, you are encouraged to use a smaller portion of the corpus to test your implementation before you try to index the complete version.

Indexing Policies For each document in the corpus (the `-c` argument above), your implementation must parse, tokenize, and index it. Your implementation must operate within the designated memory budget (the `-m` argument) during its entire execution.[4] This emulates the most typical scenario where the target corpus far exceeds the amount of physical memory available to the indexer. At the end of the execution, a final representation of all produced index structures (inverted index, document index, term lexicon) must be stored as three separate files, one for each structure, at the designated directory (the `-i` argument).

In addition to this workflow, **your implementation must abide by the following policies**, which will determine your final grade in this assignment:

1. *Pre-processing Policy.* To reduce the index size, your implementation **must perform stopword removal and stemming**. Additional pre-processing techniques can be implemented at your discretion.

[2] `https://en.wikipedia.org/wiki/Standard_streams#Standard_output_(stdout)`

[3] `https://www.kaggle.com/datasets/rodrygo/entities`

[4] In the provided `indexer.py` file, a RAM limitation mechanism is implemented, which your implementation must obey. Note that the memory budget refers to the total memory available to your implementation, not only to the memory needed to store the index structures. As a reference lower bound, assume your implementation will be tested with `-m 1024`.

2. *Memory Management Policy.* To ensure robustness, your implementation **must execute under limited memory availability**. To this end, it must be able to produce partial indexes in memory (respecting the imposed memory budget) and merge them on disk.[5]

3. *Parallelization Policy.* To ensure maximum efficiency, you **must parallelize the indexing process across multiple threads**. You may experiment to find an optimal number of threads to minimize indexing time while minimizing the incurred parallelization overhead.

4. *Compression Policy (extra).* Optionally, you **may choose to implement a compression scheme** for index entries (e.g. gamma for docids, unary for term frequency) for maximum storage efficiency.

Deliverables Before the deadline (Apr 27th, 2023 23:59), you must submit a package file (`zip`) via Moodle containing the following:

1. Source code of your implementation;

2. Documentation file (`pdf`, max 2 pages);

3. Link to the produced indexed structures (stored on Google Drive).

Your `indexer.py` file must be located at the root of your submitted zip file.

Grading This assignment is worth a total of 15 points distributed as:

- 10 points for your *implementation*, assessed based on the quality of your source code, including its overall organization (modularity, readability, indentation, use of comments) and appropriate use of data structures, as well as on how well it abides by the aforementioned indexing policies.

- 5 points for your *documentation*, assessed based on a short (`pdf`) report[6] describing your implemented data structures and algorithms, their computational complexity, as well as a discussion of their empirical efficiency (e.g. the time elapsed during each step of indexing, the speedup achieved via parallelization). Your documentation should also include a characterization of your produced index, including (but not limited to) the following statistics: number of documents, number of tokens, number of inverted lists, and a distribution of the number of postings per inverted list.

Late Submissions Late submissions will be penalized in $2^{(d-1)} - 0.5$ points, where $d > 0$ is the number of days late. In practice, a submission 5 or more days late will result in a zero grade.

[5] `https://en.wikipedia.org/wiki/External_sorting#External_merge_sort`

[6] Your documentation should be no longer than 2 pages and use the ACM LaTeX template (sample-sigconf.tex): `https://www.acm.org/binaries/content/assets/publications/consolidated-tex-template/acmart-primary.zip`

Teams This assignment must be performed **individually**. Any sign of plagiarism will be investigated and reported to the appropriate authorities.