Course Assignment 1 – Indexing

CI-6226 Information Retrieval & Analysis

In this course there are a total of three individual assignment. This is assignment 1 of 3. A single PDF file is to be submitted by every student containing the report.

Feel free to use any external materials but don't forget to reference your sources.

Reporting on Completion of Assignment

Read this section very carefully. Failure to adhere to the simple rules of reporting will lead to lowered marks.

The assignment report is due before Sunday March 21st, 2021 @ 23:59. Each group is to submit electronically one PDF file (and nothing else) called '*<MATRIC>*`-1.pdf`' containing the assignments reports, where *<MATRIC>* is your matriculation number. E.g., a student with the matriculation number A123456B should submit a file named '`A123456B-1.pdf`'.

The report should not have a separate title page. At the top of the first page put the following mandatory elements

- Title: **CI6226 Information Retrieval & Analysis / Assignment 1 / AY20-21**
- Full name
- Matric number
- Your NTU email address

The report should not exceed 3 pages excluding references. The report should cover what was done in each step of the assignment, provide reasoning for the chosen course of actions, demonstrate examples (where applicable). The report should be written as a coherent text. You are not required to submit your code but you can showcase portions of your code in the report.

Submission should be done in NTULearn.

The report must be neatly formatted. Reports that are hard to read due to formatting (or any other reason) will be marked low or not marked at all in extreme cases.

Grading

The assignment is overall graded on a 0–100 scale.

Dataset

You are provided a dataset for this assignment, which you are free to use. You can use your own dataset as well.

Assignment

In this assignment you will create the indexing component of an information retrieval system using one of the external sorting algorithms that were discussed in class: BSBI or SPIMI.

The assignment is to write a software system. As an **input** to the system, it should take 1) path to the directory containing the text files to index; 2) block size parameter for BSBI/SPIMI. The **output** of the algorithm should be a single text file containing a sorted list of term-document pairs.

You can pass the inputs to your system using any of the standard approaches: as command-line arguments, via an ini-file, or using environment variables. You may have other inputs and outputs if you prefer (e.g., optional parameter to indicate the desired path for the output file; or output timings and statistics).

The system should contain a linguistic processing module that normalizes tokens and turns them into terms. For this assignment, it is not necessary to create a document-docID mapping and you can use the document path itself as a docID.

In your reports indicate the time it takes to index to whole dataset, to sort a block, to merge blocks, etc. Try to measure how much memory your application requires and whether it is consistent with the block size parameter. Make sure that you never read more at once than the block size stipulated as an input parameter.

Below are some hints that you may consider when doing the assignment.

Toolkit

It might be convenient to start by creating two helper components (functions, methods, classes – depending on which language or programming paradigm you use) that will make doing the rest of the assignment easier and faster.

Directory Listing

Input: string (path to directory)
Output: list of strings (full paths to files in the directory)

This component will list all the files (with full paths) in the directory. This will simplify going through the files at the indexing stage.

File Reading

Input: string (full path to file)
Output: string/text (full contents of a file)

Some programming languages have this function in the standard library, and some don't. It will be very useful to have this functionality easily accessible in this project, so either find it in the standard library of your language or write a small function that does that.

Tokenization

Output: pairs < token , document >

Note that you can't output a full list of tokens at once because it may be larger than the block size. One approach may be to output pairs one by one with each call to the tokenizer component.

For this assignment, split tokens only on whitespace characters (space, newline, tab).

Tip: Take note that there could be several whitespace characters in a row. Do not create empty tokens.

Linguistic Modules

Input: pairs < token , document >
Output: pairs < modified token , document >

This component performs some simple linguistic transformations on tokens, e.g.: removing punctuation symbols (`!@#$%^&*()-_=+'`~ ":;|/.,?[]{}<>`), case folding, stemming, removing numbers. For stemming, you can use existing implementations of Porter Stemmer, such as:

- https://github.com/caarmen/porter-stemmer for Java
- https://github.com/reiver/go-porterstemmer for Go
- https://github.com/nemec/porter2-stemmer for C#
- http://www.nltk.org/howto/stem.html for Python

Also, feel free to find any other implementations. If you want, you can create your own implementation, but there will be **no** additional marks for that.

Good luck, have fun!