LAB 8: Apache Lucene

Exercises

Your task is to build a simple Lucene application. This exercise is divided into two sub-tasks:

- Firstly, you will build a simple index using Apache Lucene, based on the provided HTML documents.
- Then, you will make several queries and search the collection (index) for the most relevant documents.

Before going further:

- Download Indexer.java, Searcher.java, Constant.java, and pages.zip from http://www.cs.put.poznan.pl/mtomczyk/ir/lab8/. Create java project using any IDE and these files to the project (unzip pages folder!).
- From http://www.cs.put.poznan.pl/mtomczyk/ir/lab7/, download **lucene-7.2.1.zip** and unzip it in the project's directory. Add the following jars to your project: **lucene-core-7.2.1.jar**, **lucene-queryparser-7.2.1.jar** and **lucene-backward-codecs-7.2.1.jar**.
- From http://www.cs.put.poznan.pl/mtomczyk/ir/lab2/, download tika-app-1.17.jar and add it to your project

Exercise 1. Firstly, use Apache Lucene to build construct an index from the given collection:

1) Go to indexer.java. This class is responsible for generating a Lucene index.
2) Seek for **indexDocuments()** method. There are several TODOs that must be completed.
3) Firstly, consider the method **getHTMLDocuments()**. It should load the files from the collection and construct Document (Apache Lucene class) objects. You may go to **getHTMLDocuments()** method and see that it iterates over the files in "pages" folder. This method is completed. You should go to **getHTMLDocument()** method that construct a Document object for a single File.
4) Read carefully the provided comments. Focus on the Document <-> Field relation. What is the Field? What does STORED and INDEXED mean? Complete this method (as you may notice, apache Tika is involved in content extraction ☺).
5) Go back to **indexDocuments()** method. Now, having a list of Documents, you should create an index using IndexWriter. Follow the provided comments.
6) If you completed all the TODOs, run Indexer.java. You should see that "index" folder is generated and it contains an index.

Exercise 2. Let's search for some documents.

7) Go to Searcher.java. It is suggested to start with **printResultsForQuery()** method.
8) This method is invoked after the Query object is constructed. As you may notice, IndexSearcherobject is passed as an argument. This object provides a method for seeking through the index for the first N documents that are the most relevant

according to the query. Follow the provided comments and complete this method. The documents should be printed in the following form (each in a separate line):

SCORE: FILENAME (Id=…ID) (Content=CONTENT) (Size=SIZE**b**).

Why CONTENT will be null☺?

9) Now, go to **main()**. Follow the provided comments to generate several queries:
   a. TermQuery – seeking for documents that contain some term,
   b. BooleanQuery – boolean query based on terms, i.e., "medicine AND drug",
   c. RangeQuery – seeking for documents which values (according to some field) are in the provided range, e.g., size of the document,
   d. PrefixQuery – like TermQuery, but the prefix of a term is provided instead of a whole string (i.e., "antelope" and "ant" match "**ant"** prefix),
   e. WildcardQuery – You can use ?to indicate any single letter (e.g., "cats" matches "cat?") or use * to indicate multiple letters (e.g., "antelope" matches "ant*"),
   f. FuzzyQuery – Seeking for terms that are similar to the provided term, e.g., "mammal" matches "mamml".
   g. **Use** QueryParserto parse human-entered query strings.

Programming assignment – Report (deadline: 13th lab)

Your task is to use information retrieval techniques that you have learnt during this course to gather, process, and analyse some useful data. You may focus on one of the following:

- gather some data and build a search engine (rather easy option),
- gather some data and extract some useful information from it (it might be more complicated since it involves more advanced data exploration techniques and/or machine learning).

But these options are just suggestions and you may propose some new interesting topics. The most important thing is the **data** you want to explore and analyse. Working with data that is irrelevant to you may be very boring. So you should think about the things you like, about something that may be interesting to you. For instance: sports, medicine, biology, music, movies, art, geography, sciences, politics, computer games, etc. Depending one topic, you may explore and analyse the data differently. For instance:

- **Sports:** You may build a search engine that is dedicated to football. Your program may aid the user in seeking for information about a particular player, a football match, a team, an event, etc. So your program may suggests some keywords ("Manchester", "World Cup"), the user may emphasize importance of some keywords (weight of "goal" > weight of "penalty"), or provide some relevance feedback techniques.
- **Geography:** You may seek for relevant data in, e.g., Web, and explore & analyse the downloaded information. These two processes may be performed simultaneously (if the program needs some additional information for analysis, it downloads it). For instance, you may collect many pages that describe some geographical concepts as

well as some places and their locations. You may analyse the content and derive some interesting relations, e.g., “deserts are common in Africa, the related words are: dry, hot, sand”. You may also download the coordinates of some places and put them on the map (see, e.g., geographic information system).

As you may notice, independently to which topic you will choose, you have to collect the data. You are not allowed to do this manually (ok, you are allowed to download, e.g., 1000 HTML files, but it is recommended not to do it ☺). You can use Apache Nutch but you can write a simple script (e.g., in Python) that will do the job. You may consider some API that may be helpful, e.g., Wikipedia, GeoHack, or Google Map. After you get the data that you need, it must be processed. For this reason, you may use the Apache libraries that you have become acquainted with:

- Apache Tika: to extract textual content from files (e.g., from HTML),
- Apache Open NLP: to process natural language (sentence segmentation, POS tagging, etc.).

You may work in groups that contain up to 3 members. The important part of this assignment is the report. It should be written in *LaTeX* and should be nicely formatted. It is suggested to include vector illustrations rather than raster. You can write the report in Polish ☺. The deadline is 13th lab (one week before the last lab). The report should be saved in PDF file and sent via e-mail. If your project is more application-oriented, then you are expected to present it during the last lab. Some hints/requirement about the aspects the report should cover:

- Motivation (short): what is the main idea? What data you want to collect and process?
- Data collection:
  - How did you collect the data? Please describe this process in details.
  - How many files were downloaded?
  - How long did it take?
  - Did the crawler download some irrelevant pages?
  - You may present the above results on a plot (as a function of iteration).
- Data processing & analysis: It is recommended to present data processing process step-by-step:
  - How do you extract the textual (or other) data?
  - Do you extract some metadata?
  - Do you perform language recognition?
  - How do you extract some useful information from the text, e.g.:
    - verbs, nouns,
    - named entities,
    - dates,
    - phone numbers,
    - e-mail addresses,
    - locations,
    - document’s title,
    - numerical data (weight, height, etc.)?

    - Do you normalize the extracted data?
    - Do you use some interesting data structures to keep the data?
    - Present some examples for each step. In particular, you may:
        - Present some interesting text fragment and show & discuss the outcomes of subsequent steps.
        - Obviously, your program may not work perfectly and some data may not be extracted correctly. It will be interesting if the report provides these examples, followed by the discussion that refers to:
            - why does the program perform incorrectly?
            - how may it be improved?
- <u>You should address topic-related questions</u>:
    - For instance, if you build a search engine:
        - What data is indexed?
        - What data is stored?
        - Do you use some query expansion techniques to support searching? Describe them briefly.
        - What are the search options? Can the user, e.g., provide an input string (user's query), select a category, select some terms, or provide a Boolean query?
        - If is suggested to present some scenarios (use cases): the user's input followed by the program's output and the discussion.