Lab 6 – Apache Lucene

Deadline: +1 week

1 Exercises

Your task is to create a simple Apache Lucene application. This exercise is divided into two parts:

- Firstly, you will use Apache Lucene to index a collection of HTML documents.
- Then, you will make several queries and search the collection (index) for the most relevant documents.

Before going further:

1. Download Indexer.java, Searcher.java, Constants.java, and pages.zip from http://www.cs.put.poznan.pl/mtomczyk/ir/lab6/. Create a java project using any IDE and add these files to the project (unzip pages.zip!).
2. Download lucene-7.2.1.zip (www.cs.put.poznan.pl/mtomczyk/ir/lab6/) and unzip it to 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.
3. Download tika-app-1.17.jar (http://www.cs.put.poznan.pl/mtomczyk/ir/lab2/) and add it to your project.

1.1 Exercise 1

Firstly, use Apache Lucene to index the collection of HTML files:

1. Go to **indexer.java**. It generates a Lucene index.
2. Seek for a **indexDocuments** method. There are several **TODOs** that must be completed.
3. Firstly, take a look at a **getHTMLDocuments** method. It loads the files from the collection, iterates over the files stored in "pages" folder, and constructs Document (Apache Lucene class) objects. This method is complete. Go to a **getHTMLDocument** method, which processes a single file.

4. Read the comments carefully. Focus on the Document $\leftrightarrow$ Field relation. What is the Field? What do STORED and INDEXED flags mean?

5. Finish this method (you may notice that Apache Tika is used for content extraction).

6. Go back to the **indexDocuments** method. Now, having a list of Documents, you should create an index using **IndexWriter**. Follow the comments.

7. If you had completed all **TODOs**, run Indexer.java. You should see that the index was generated and stored in "index" folder.

1.2 Exercise 2

Let's search for some documents.

1. Go to **Searcher.java**. It is suggested to start with a **printResultsForQuery** method.

2. This method is invoked after a Query object is constructed. As you may notice, an **IndexSearcher** object is passed as an argument. This object provides a method for searching for the first N documents which are the most relevant to the query. The method should notify (log) about the found documents. The log, for each document, should be in the following form:
SCORE: FILENAME (Id=...ID) (Content=CONTENT) (Size=SIZE)

3. Now, go to the **main** method. Follow the comments and construct several queries:

   (a) **TermQuery**: for seeking for documents that contain some term.

   (b) **BooleanQuery**: boolean query, e.g., "medicine AND drug".

   (c) **RangeQuery**: for seeking for documents whose attribute values are between a range of some numbers, e.g., size of the document $\in [50B, 2000B]$.

   (d) **PrefixQuery**: like TermQuery, but the prefix is provided instead of a whole string (i.e., "antelope" and "ant" match the "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**: for seeking for terms that are similar to the provided term, e.g., "mammal" matches "mamml".

   (g) Use **QueryParser** to parse human-entered query strings.

2 Programming Assignment

Your task is to use information retrieval techniques which 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 (default option),
- gather some data and extract some useful information from it (it may 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 tasks involving information retrieval. One of the most important things 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 on the topic, you may explore and analyse the data differently. For instance:

- Sports: You may build a search engine which 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 describing 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. 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 Maps. After you get the data 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 containing 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. The reports (PDFs) shall be sent via e-mail. The deadline is $8th$ lab -3 days. If your project is application-oriented, then you are expected to present it during the last lab. Some hints/requirement about the aspects the reports should cover:

- Motivation (short): what is the main idea? What data did you 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 plot these data (as a function of iteration).
- Data processing & analysis: It is recommended to discuss the data processing step-by-step:
  - How do you extract textual (or other) data?
  - Do you extract some metadata?
  - Do you perform language recognition?
  - How do you extract some useful information, 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?
  - It is suggested to show some examples. 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: (a) why does the program perform incorrectly? (b) how may it be improved?
- You should address topic-related questions. 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.