University of Granada

Information Retrieval

Implementation of a Information Retrieval System using Lucene

Author:
Mikhail Raudin
mraudin@correo.ugr.es

Lecturer:
Prof. Juan Francisco
Huete Guadix
jhg@ugr.es

December 23, 2020

Contents

1 Introduction **3**

2 Indexing **3**
2.1 Class description 4
2.2 How to use the program? 6

3 Searching **6**
3.1 Searching Class 6
3.2 How to use the program? 8
3.3 Libraries 12

List of Figures

1 Shows the UML diagram of the indexing application. 4
2 Shows how to use the indexing application. 6
3 Shows the UML diagram of the searching application. 7
4 Range Query on the field "size". 8
5 Results of Range Query ordered by relevance, showing size of each document. 9
6 Shows the usage of boolean search 9
7 Using the facet search 10
8 The facets displayed to the user for the query "antibody test". . 11
9 The result of the facet "Country" for the query "antibody test". 11
10 The result of the facet "Institution" for the query "antibody test". 12
11 Shows how to use the drill down facet search. 12
12 Shows the drill down facet search results. 13

1 Introduction

The goal of the project is to create a search engine using Apache Lucene for the COVID-19 Open Research Dataset (CORD-19)[1]. The collection contains 141.000 json files with metadata about the papers in the dataset. Two applications are developed, the first is used to create a Lucene Index, and the second is used to run multiple types of queries on the index, such as

- Free Text Query,
- Boolean Query,
- Phrase Query,
- Search by facets,
- drill down facet query.

2 Indexing

Before constructing the index, we need to change the perspective and look at the possible types of user queries, that will be performed on the index. That way, we decide what type of information should be stored in the index and how it should be processed. The users of the application are epidemiologists, medical scientists and researchers working on COVID-19 and health organization employers (e.g. of the WHO). They speak a scientific language and are familiar with the vocabulary. As might be expected, common queries are searching for a specific term/topic related to COVID research and looking for work from different countries, authors and institutions.

Therefore, the following fields are relevant and stored in each lucene document (representing a json file):

- filename (json)
- title
- exact title
- author
- abstract
- body
- institution
- country
- size

[1] `https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge`

The field title, abstract and body are Lucene Text Fields and tokenized using the English Analyzer. The field exact title (to return in the results) and institution are String Fields, that means they are not tokenized. The field author gets tokenized using a Standard Analyer, because in this case it leads to better quality tokens than the English or Whitespace Analyzer. Finally, the field country uses a Custom Analyzer called Country Analyzer, to improve the retrieval results.

2.1 Class description

Fig. 1 visualizes the structure and design of the Indexing Application. The entry point is the Main class, which creates an instance of the Indexing class. That is the central class of the application, it uses the ParseJSON class to obtain the relevant text from the json files that is being stored in the document fields. To preprocess the country information, the Indexing Class uses the Country Analyzer, which is a subclass of the Lucene Analyzer class. It uses its own TokenFilter, the CountryFilter, which is a subclass of Lucenes Token Filter.

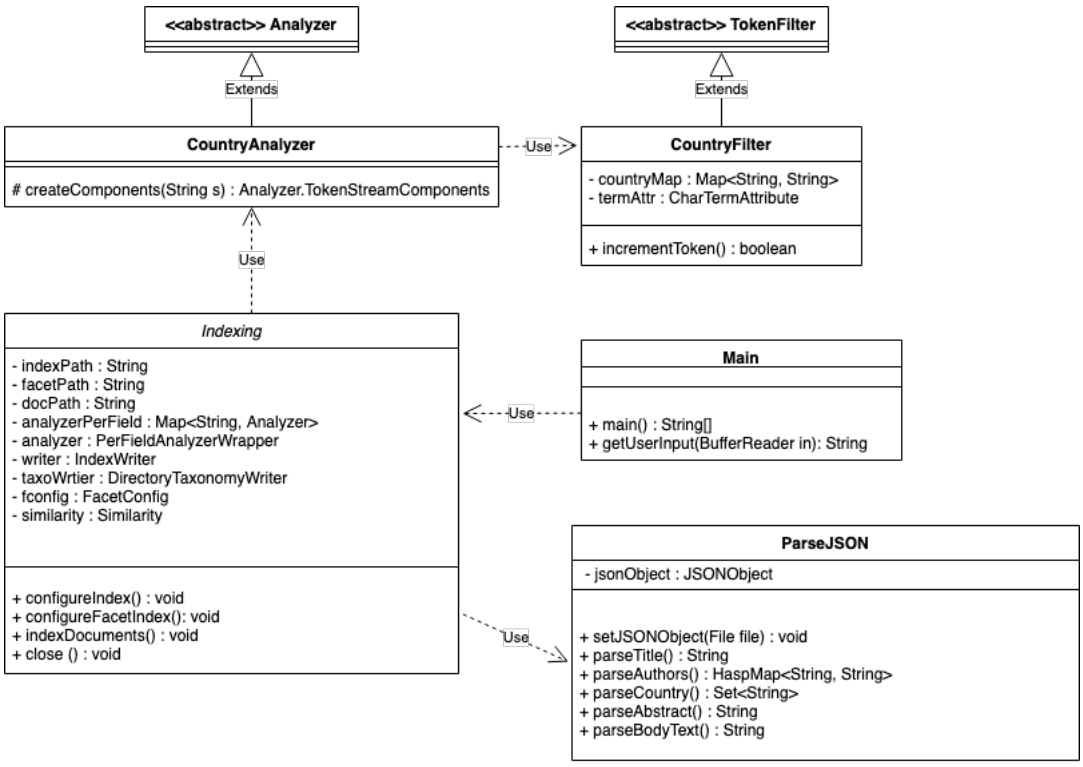

Figure 1: Shows the UML diagram of the indexing application.

ParseJSON Class

This class is responsible to parse json file content of interest using the JSON.simple library. The setJSONObject() method creates a JSONObject for the file passed as an argument. This JSONObject contains all of the json file content and by calling one of the other methods of the class (parseTitle(), parseAuthors(),...) the specific field content is returned.

Indexing Class

This class creates a Lucene Index using the methods of the ParseJSON class to get the document field values. First, it creates and configures the objects being needed to write to an index as well as a sidecar index to store the facets with a Taxonomy Writer. It sets the analyzer for each field. In the indexDocuments() method for each file in the directory, several methods from the ParseJSON class are called to get the field values, which are then added one after another to the documents, providing which of the fields is being stored for the results and which is not. Stored fields can be shown to the user as a result of a query, following fields are stored:

- filename
- author
- exact title
- size

In the end, the changes are commit and the writer objects closed in the close() method.

CountryAnalyzer Class

This is a subclass of the abstract Analyzer class of Lucene and creates tokens for an input string using the Keyword Tokenizer, Country Filter and Lowercase Filter. The class contains only the overriden createComponets() method.

CountryFilter Class

This is a customized Token Filter that takes care of the issue that the most common origin countries of papers in the collection have several different ways of writing the countries name. For instance, USA can be found as

- United States
- United States of America
- USA, USA
- U.S.A

The same is observed for the UK and China. To prevent that each of the names version is seen as a distinct token, and therefore only being relevant for a query using that exact name version ("U.S.A" queries would only retrieve "U.S.A" labeled documents, but not "United States" ones), we override each of the name version with a single one : "USA" (or "UK" or "China").

2.2 How to use the program?

Fig. 2 shows that the user is asked to provide the path to the documents directory, as well as paths to where to store the index and side index. After that the indexing process begins.

```
(base) MacBook-Air-3:Indexing mikhail$ java -jar target/Indexing-1.0.jar
Welcome to the Indexing program. You can create an lucene index.
Please provide the path to the documents directory:
/Users/mikhail/Downloads/archive/document_parses/pdf_json/
Please provide the path where the index should be stored:
/Users/mikhail/index_test
Please provide the path where the side index with the facets should be stored:
/Users/mikhail/facet_test
1 documents indexed!
2 documents indexed!
3 documents indexed!
```

Figure 2: Shows how to use the indexing application.

3 Searching

The Searching application contains besides the Main class only the Searching class, which is responsible for running queries and presenting the results in the CLI. The UML diagram is shown in fig. 3.

3.1 Searching Class

The constructor initializes the similarity measure (BM25) and the Facets Configuration (the same that is used for indexing the documents). For each type of search function exists a method. Each of them sets up the Index Reader and Writer first, by calling that method. Then, the user input (the query) is read and getQuery() method called. The getQuery() methods creates a Lucence Query Object based on the searched field, as shown in table1.

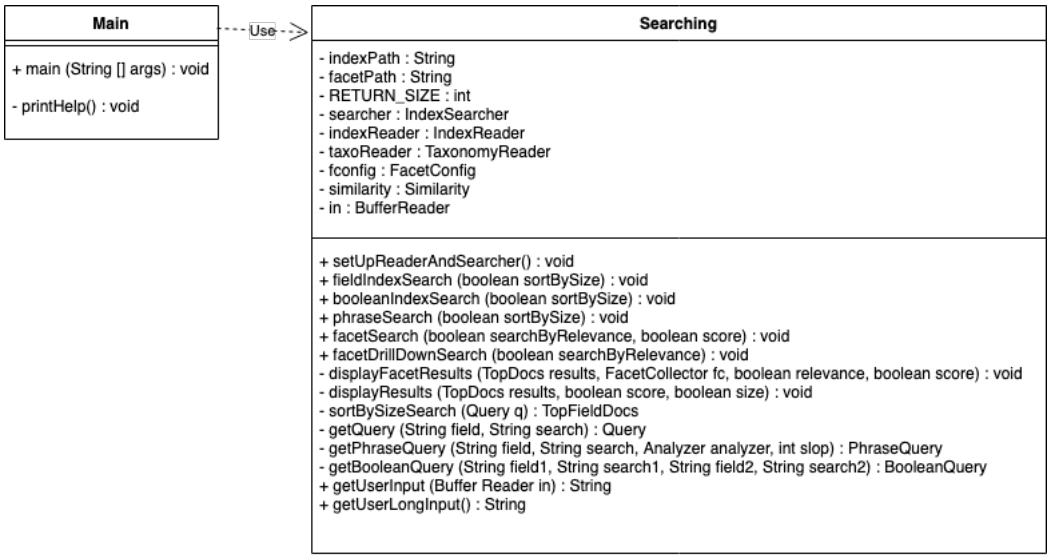

Figure 3: Shows the UML diagram of the searching application.

Table 1: This table shows which Query Object is used for the document fields.

field	Query object
filename	Term Query
institution	Term Query
size	Exact Query or Range Query
author	QueryParser (using Standard Analyzer)
country	QueryParser (using Country Analyzer)
title	Phrase Query
abstract	Phrase Query
body	Phrase Query

For boolean or phrase searches, the getBooleanQuery() or getPhraseQuery() method is used. The results are presented with the displayResults() method and to show the facets, the special displayFacetResults() method is used.

3.2 How to use the program?

After proving the path to the index and side index directory, the user gets asked which search operation he wants to perform. No matter what type of query, the user gets always asked after selecting the search type, if the results should be sorted by size or by the default option, which is the similarity score. If the user selects the sorting by size option, the score of each relevant document is always presented as additional information. As an example of the application usage, fig. 4 shows the free text query on the field "size". When a user searches this field, he can decide between an exact or a range query. If range is selected, then the user is asked for the lower and upper bound. The results (fig. 5) show the size for each of the file, but are sorted by similarity. The amount of documents found as well as the filename and exact title for each document found are printed.

```
(base) MacBook-Air-3:ri-practica3 mikhail$ java -jar target/searching-1.0-SNAPSHOT.jar
What type of search do you want to do?
Type -t for free search by field. Type -b for boolean search. Type -p for phrase proxi
mity search. Type -f for facets search or -df for a drill down facets search:
-t
Do you want the results to be sorted by size (default= similarity score)?[y/n]
n
Field?:
size
Do you want to search for documents having the exact size, or search for documents in
a range of sizes? Type [exact/range]
range
What is the lower bound for doc size?
10000
What is the upper bound for doc size?
15000
```

Figure 4: Range Query on the field "size".

```
What is the upper bound for doc size?
15000
1001 document(s) found
----------------------------------------
Filename: 8871c3e2e979a124705554c0b955fcd49646d315.json
Title: LETTER FROM ASIA-PACIFIC REGION Letter from Hong Kong

Size= 12586
----------------------------------------
Filename: 89125dbd81cb2e98936b681150b5b3d543fef13a.json
Title: SARS-CoV-2 RNA from droplets to faeces: a new focus for proctologists. Correspo
ndence

Size= 10278
----------------------------------------
Filename: ed5f83dc1f7cc94102781ff70051b52197751674.json
Title: Patrolling Monocytes Watch Over Relapse

Size= 13538
----------------------------------------
```

Figure 5: Results of Range Query ordered by relevance, showing size of each document.

Fig. 6 shows an example of a boolean query. If the user does not type an answer for the sorting option, a follow up instruction gets printed. After providing values for the title and country field, the boolean query is executed and the results are shown in the CLI.

```
What type of search do you want to do?
Type -t for free search by field. Type -b for boolean search. Type -p for phrase proxi
mity search. Type -f for facets search or -df for a drill down facets search:
-b
Do you want the results to be sorted by size (default= similarity score)?[y/n]

Type 'y' if sort by size and 'n' if not sort by size
n
Title:
COVID-19 vaccine
AND
Country:
USA
17 document(s) found
----------------------------------------
Filename: 2c70050ff40b538a630333e6708aac990712b718.json
Title: The Optimal Allocation of Covid-19 Vaccines

----------------------------------------
Filename: d0dd87b80696f3a7582f36ee1c8dad8218883c0c.json
Title: COVID-19 vaccine safety

----------------------------------------
Filename: a72a57143cd50bb910eb222c50394cd1314b6f75.json
Title: COVID-19 vaccines: knowing the unknown
```

Figure 6: Shows the usage of boolean search

Fig. 7 shows a facet search. This option asks the user additionally about what order to consider for the facet labels, they can be ordered by the size, so by the number of documents with that label value or by the relevance (the mean similarity score of the documents with that value). The facets are presented after the 20 most relevant result documents (fig. 8, fig. 9, fig. 10).

```
(base) MacBook-Air-3:ri-practica3 mikhail$ java -jar target/searching-1.0-SNAPSHOT.jar
Path to index:
/Users/mikhail/index
Path to sidecar index (containing the facets):
/Users/mikhail/facets
What type of search do you want to do?
Type -t for free search by field. Type -b for boolean search. Type -p for phrase proximity se
arch. Type -f for facets search or -df for a drill down facets search:
-f
Do you want the results to be sorted by size (default= similarity score)?[y/n]
n
Do you want the facet values ordered by size (#n of docs with that value) or relevance (mean
similarity score for all docs with that value)?[s/r]
s
Field?:
title
Query?:
antibody test
72 document(s) found
-------------------------------------
Filename: f2e236f0458f86706985c627ef88d40c8f037492.json
Title: SARS Antibody Test for Serosurveillance

Score= 5.091909
-------------------------------------
Filename: 4a35ac0a934842c639c368058512f68c440bb627.json
Title: Cats, coronaviruses and coronavirus antibody tests

Score= 4.8727903
-------------------------------------
Filename: 6ab2f323dbac8268462f9f755207201c1ecf2539.json
Title: Understanding Antibody Testing for COVID-19

Score= 4.8727903
```

Figure 7: Using the facet search

```
-----------------------------------
Filename: a6e6040c886f3cb42f707ce08f9499b088df1cdd.json
Title: Journal Pre-proof COVID-19: Understanding the science of antibody testing and lessons
from the HIV epidemic COVID-19: Understanding the Science of Antibody Testing and Lessons fro
m the HIV Epidemic

Score= 4.0976925
-----------------------------------
Filename: 5681e3a3fb3b053790cddd2512259d189cfd70cc.json
Title: A Modular Microarray Imaging System for Highly Specific COVID-19 Antibody Testing

Score= 4.009987
Facet Score Summary
Total categories: 3
Category Author
    Label: Steven C Almo, value (#n)->2
    Label: Jonathan R Lai, value (#n)->2
    Label: Algis  Jasinskas, value (#n)->2
    Label: Rie  Nakajima, value (#n)->2
    Label: Saahir  Khan, value (#n)->2
    Label: Phoebe  Vargas, value (#n)->2
    Label: David  Phillips, value (#n)->2
    Label: Martin  Curran, value (#n)->2
    Label: Marlyn  Perales, value (#n)->2
    Label: Richmond  Abeseabe, value (#n)->2
    Label: Daniel  Chapman, value (#n)->2
    Label: Lee  Mynott, value (#n)->2
    Label: Nicholas J Matheson, value (#n)->2
    Label: Bensi  Vergese, value (#n)->2
    Label: Richard  Hardy, value (#n)->2
    Label: Michael P Weekes, value (#n)->2
    Label: Ranalie  De Jesus, value (#n)->2
    Label: Sarah  Clark, value (#n)->2
    Label: Marivic  Fabiculana, value (#n)->2
```

Figure 8: The facets displayed to the user for the query "antibody test".

```
    Label: Jo  Waller, value (#n)->2
    Label: Henry W W Potts, value (#n)->2
Category Country
    Label: USA, value (#n)->7
    Label: Germany, value (#n)->7
    Label: Greece, value (#n)->5
    Label: China, value (#n)->4
    Label: Sweden, value (#n)->4
    Label: UK, value (#n)->3
    Label: Japan, value (#n)->3
    Label: Italy, value (#n)->2
    Label: France, value (#n)->2
    Label: Belgium, value (#n)->2
```

Figure 9: The result of the facet "Country" for the query "antibody test".

```
Category Institution
    Label: university of cambridge, value (#n)->4
    Label: university of thessaly, value (#n)->3
    Label: university of california, value (#n)->2
    Label: harvard medical school, value (#n)->2
    Label: university of california san francisco, value (#n)->2
    Label: university college london, value (#n)->2
    Label: huazhong university of science and technology, value (#n)->2
    Label: massachusetts general hospital, value (#n)->2
    Label: university of science and technology of china, value (#n)->2
    Label: university of kent, value (#n)->2
    Label: albert einstein college of medicine, value (#n)->2
    Label: university of california irvine health, value (#n)->2
    Label: goethe university frankfurt am main, value (#n)->2
```

Figure 10: The result of the facet "Institution" for the query "antibody test".

The last example presents the drill down facet search (fig. 11). The user can provide a specific value for one or more facet categories. In this case we want to know how many documents have the value "Germany" in the category "Country" and "Christian Drosten" in the category "Author". The answer is 79, as seen in fig. 12("drilldown sideways hits: 79 hits"). Furthermore the facets label and values are displayed, where we see again that "Germany" has a value of 79 as well as "Christian Drosten" (fig. 12).

```
Following Facets are available: Author / Institution / Country
Please provide the facets and their specific values you want to search. Type NO if finished
Facet?:
Country
Value?:
Germany
Country Germany
Facet?:
Author
Value?:
Christian  Drosten
Author Christian  Drosten
Facet?:
NO
Filtering query[ +*:* #($facets:CountryGermany) #($facets:AuthorChristian  Drosten)]
```

Figure 11: Shows how to use the drill down facet search.

3.3 Libraries

The maven dependencies are included in the .jar files. To run the application, execute "java -jar Path/to/jarfile"

```
drill sideways hits: 79 hits
[dim=Country path=[] value=-1 childCount=50
  Germany (79)
  Ghana (12)
  The Netherlands (10)
  USA (9)
  Switzerland (8)
  France (8)
  Gabon (8)
  Russia (5)
  the Netherlands (5)
  United Kingdom (4)
  UK (3)
  Saudi Arabia (3)
  Czech Republic (3)
  Democratic Republic of Congo (3)
  People's Republic of China (2)
  PR China (2)
  China (2)
  Brazil (2)
  Sweden (2)
  Central African Republic (2)
, dim=Author path=[] value=-1 childCount=22112
  Christian  Drosten (79)
  †   (38)
  Rolf  Hilgenfeld (37)
  Stefan  Pöhlmann (35)
  John  Ziebuhr (28)
  Marcel A Müller (26)
  Stephan  Becker (26)
  Sandra  Ciesek (26)
  Georg  Herrler (25)
  Ralf  Bartenschlager (23)
  Volker  Thiel (20)
```

Figure 12: Shows the drill down facet search results.