Information Retrieval System

Philip, Joel John
Mudgal, Kartik
Sajith, Madhuri
Jayanand, Vishnu
Hegde, Nitin

November 29, 2019

1 Introduction

Information Retrieval is the process of searching for information (relevant to the user query) within documents or a collection of documents or the metadata describing these documents. The following documentation describes an Information Retrieval project to search and retrieve text from indexed documents.

2 Softwares Used

Programming Language : Java
Documents Used : .txt (Text Files), .html (HTML Files)
Libraries : Lucene, Jsoup
Lucene : Lucene-core-7.5.0.jar, lucene-queryparser-7.5.0.jar, lucene-analyzer-common- 7.5.0.jar, lucene-demo-7.5.0.jar.
Jsoup : Java HTML parser (Version 1.11.3): Jsoup is a Java library for working with real-world HTML. It manipulates HTML elements, attributes and text and parses HTML file. Jars used: jsoup-1.11.3.jar

3 Implementation

The *Runner.java* uses args4j library to parse command line arguments. After this the control is passed over to the *CommandLineAPI.java* which manages the complete program flow. The *CommandLineAPI.java* uses the *BasicIndexer.java* to get an *IndexReader.java*. This *IndexReader.java* instance may represent a newly created index or an old index according to the command line arguments. After this, the *Searcher.java* is used to query the IndexReader after we get the results they are printed to the console.

3.1 Indexing

The *BasicIndexer.java* takes in three parameters - target path, Boolean value indicating whether to create new index and an index directory. If we are creating a new index then the directory tree is walked and all files and subfolders

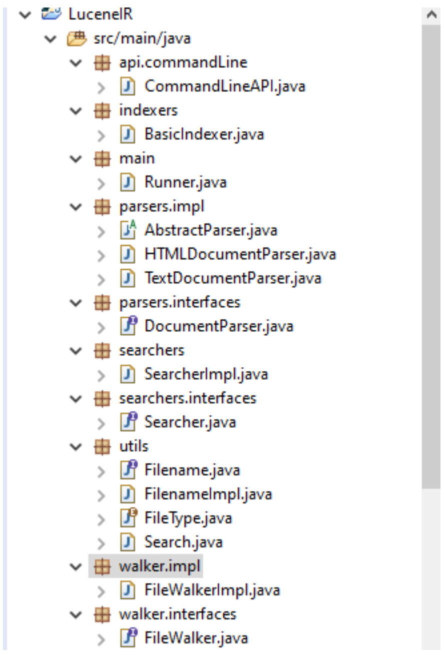
```
LuceneIR
  src/main/java
    api.commandLine
      CommandLineAPI.java
    indexers
      BasicIndexer.java
    main
      Runner.java
    parsers.impl
      AbstractParser.java
      HTMLDocumentParser.java
      TextDocumentParser.java
    parsers.interfaces
      DocumentParser.java
    searchers
      SearcherImpl.java
    searchers.interfaces
      Searcher.java
    utils
      Filename.java
      FilenameImpl.java
      FileType.java
      Search.java
    walker.impl
      FileWalkerImpl.java
    walker.interfaces
      FileWalker.java
```

Figure 1: List of all entities

are indexed. .html and .txt files are parsed separately using *HTMLDocumentParser.java* and *TextDocumentParser.java*. Both are implementations of the *DocumentParser.java* interface. The *DocumentParse.java* interface provides the ability to extract different fields from a document such as the body, date, title, filename, last modified date, summary, etc.,. After extracting all the fields, they are added to an index. Then an index reader for this directory is returned. If in case the user does not ask to create a new index, then the old index is simply read from the disk and an index reader is returned.

3.2 Searching

The searching takes place through *Searcher.java* interface. It provides two methods - one to set the query string and another to trigger the search. The search method takes in the number of pages to search for, hits per page and an index reader to search against. It returns a *Search.java* instance through which the results of the query can be accessed.

3.3 Stemming

"Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word for."[1] Furthermore, "all stop words, for example, common words, such as a and the, are removed from multiple word queries to increase search performance".[2]While either processing the queries or indexing documents, they are pre-processed by the *EnglishAnalyzer.java*. This class uses a *StandardTokenizer.java* and *English-PossessiveFilter.java*, *LowercaseFilter.java*, *StopFilter.java* and *PorterStemFil-*

ter.java. This step performs the stemming and stop-word removal.

3.4 Code Sections and tasks fulfilled

3.4.1 Runner.java

takes in command line arguments and sets them up for command api processing. It then calls 'run' method of *CommandLineAPI.java*.

3.4.2 CommandLineAPI.java

It checks if the user has asked for new index to be created or to use an existing old index itself. User has to provide the location of the files to be indexed, index directory and whether to index the files or not and/or a search query. If search query has been provided after getting a suitable index the query is run against the index by a searcher interface.

3.4.3 BasicIndexer.java

- The *BasicIndexer.java* takes in three parameters - target path, Boolean value indicating whether to create new index and an index directory.

3.4.4 Search.java

The *Searcher.java* creates a query and collates the results of the query.

4 Example query

```
java -jar '.\IR P01.jar' "C:\Users\abc\Desktop\tests" -ri -idx "C:\Users\Kartik
\Desktop\tests\index" -q "copy"
```

Argument 0 - Path to document folder

-ri/–reindex - flag must be present if you want to create a new Index

-idx/–index_dir - Path to the directory in which the index is stored/ Where to write the index

-q/–query - Query to run against the index

References

[1] https://en.wikipedia.org/wiki/Stemming

[2] https://www.ibm.com/support/knowledgecenter/en/SS8NLW_12.0.0/com.ibm.discovery.es.ta.doc/iiysalgstopwd.html