Romanian Information Retrieval System

Information Retrieval Course, Project I

Raluca Tudor, Group 512 – NLP

Nov 10, 2023

Project Introduction

The project requires developing an Information Retrieval (IR) system for Romanian language.

Technologies Used

- **Java** – version "16" 2021-03-16
- Apache **Lucene** – version 8.11.2 (the latest in the 8.x series)
- Apache **Tika** – version 2.6.0

User Guide

There are 2 main components that shall be executed in order to use the IR system developed: the indexer and the searcher. While these can be run from the IDE (I preferred to use JetBrains IntelliJ IDEA), one can also run them from CLI as such:

```
java -cp %DEPS% RoIndexer <documents_directory_path>
```

and then:

```
java -cp %DEPS% RoSearcher <query>
```

Where:

- `documents_directory_path` = the path for the directory holding the documents to be indexed. If no value is given, it defaults to `"resources"`, directory residing in the project folder (see image below).

- `query` = the query to be searched in the index. For inputting a query formed of more than 1 word, wrap the query around `""`. If the query is not provided, it defaults to the word “incredere”.

- **DEPS** is a temporary environment value encompassing the JAR paths for the libraries used: Lucene core JAR, common analysis JAR and queryparser JAR (i.e. files taken from the directory created when extracting the Lucene archive). This is done because I did not put these files in the Java CLASSPATH, thus having to use the -cp parameter when starting Java.

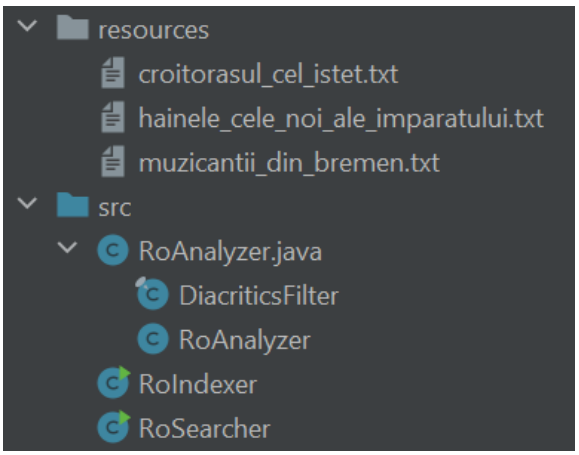
```
set
DEPS=C:\Users\Raluca\IdeaProjects\info-retrieval-ro-system\out\production\info-retrieval-ro-system;C:\Users\Raluca\Desktop\lucene-8.11.2\core\lucene-core-8.11.2.jar;C:\Users\Raluca\Desktop\lucene-8.11.2\analysis\common\lucene-analyzers-common-8.11.2.jar;C:\Users\Raluca\Desktop\lucene-8.11.2\queryparser\lucene-queryparser-8.11.2.jar;C:\Users\Raluca\.m2\repository\org\apache\tika\tika-core\2.6.0\tika-core-2.6.0.jar;C:\Users\Raluca\.m2\repository\org\slf4j\slf4j-api\2.0.3\slf4j-api-2.0.3.jar;C:\Users\Raluca\.m2\repository\commons-io\commons-io\2.11.0\commons-io-2.11.0.jar;C:\Users\Raluca\.m2\repository\org\apache\tika\tika-parser-text-module\2.6.0\tika-parser-text-module-2.6.0.jar
```

Note: Make sure that the PATH includes the Java bin directory, in order to run the `java` command.

Data Used For Building the Index

The documents used represent 3 stories taken from Povești, basme, povestiri și nuvele pentru copii de orice vârstă (povesti-pentru-copii.com).

Each story was pasted into its own `.txt` file, and all files were placed under a `resources` directory.



In terms of the encoding, Tika handles this by default.

Implementation Details

The implemented information retrieval algorithm is based on the following components:

- The **Indexer** – handles building an inverted index from a set of documents in Romanian, using Lucene & Tika, as such:
  - Open a specified directory and instantiate a custom Analyzer (detailed below). Also, opens an index writer in the *CREATE* open mode, which fully reindexes the index (another alternative is to use *CREATE_OR_APPEND*, which I mentioned in a comment).
  - For each document found within the directory, create a Lucene Document, representing *the unit of indexing and search*. For this document, set the fields to be the file-name and the text content, parsed using Tika.
    - Then, each document is added to the index.
- The **Analyzer**
  - I implemented a custom analyzer by extending the `Analyzer` abstract class and overriding the `createComponents()` method.
  - This represents the pre-processing pipeline, which breaks up the text into *indexed tokens* and performs the following normalization operations on these:
    1. **filtering out** unwanted tokens – **Romanian stop words**.
       I used RomanianAnalyzer (Lucene 8.11.2 API), which holds the method `getDefaultStopSet()`, returning the set of Romanian default stop words.
    2. **Stemming** - I used SnowballFilter (Lucene Analysis), along with RomanianStemmer, since `org.tartarus.snowball.ext` had this available stemmer for Romanian.
    3. The last preprocessing step is the **removal of Romanian diacritics**. For this, I have implemented the filter myself, by extending the `TokenFilter` (Lucene 8.11.2 API) class and overriding the `incrementToken` method.

- The **Searcher** – prompts the user for a query and uses the **same** custom Romanian **Analyser**, to execute the same pre-processing on the query as on the documents. Then, using the index searcher, search for the query (within the index, of course) and display the results, by using a Top Docs collector from Lucene Search.

Bibliography

1. Overview (Lucene 9.1.0 demo API) (apache.org)
2. Apache Tika – Tika API Usage Examples
3. Custom Token Filter Lucene - NathanCHEN