**Introduction to Lucene: Setup, Indexing and Search and Zipf’s Law**

**Setup, Compilation, Execution and Output Guidelines**

This is a JAVA source code based on Lucene version 4.7.2 that aims to index the given CACM corpus, search for given queries and generate the desired output for further analysis. (unique term frequency pairs list, top 100 results for each query as per default document scoring/ranking by Lucene)

For this code to be executed on any terminal, we need to have below pre-requisites met:-

**Setup and Pre-requisites:**

Technical Specifications:

1. Download and setup for lucene 4.7.2
2. Download the available CACM corpus

(<http://www.search-engines-book.com/collections/> ) – Available in the project folder

1. The source code is developed in Java using NetBeans IDE version 8.0.
2. Java Version 1.8.0
3. Lucene, an open source information retrieval library with below essential jar files ([https://lucene.apache.org/](https://lucene.apache.org/core/downloads.html))

* lucene-core-4.7.2.jar
* lucene-queryparser-4.7.2.jar
* lucene-analyzers-common-4.7.2.jar

1. External JAR file used : jsoup-1.8.3.jar

IMP: All required libraries are placed at Project-Lucene Implementation/lib

**Compilation and Execution:**

For this java code to be executed, no command line arguments are required.

(Main Function -> src/lucene/implementation -> LuceneImplementation.java)

Important JAVA src files -> LuceneImplementation.java and FileWrite.java

**LuceneImplementation.java** contains the main function that calls the appropriate functions to execute the task as required in this homework assignment based on lucene in-built dictionary

Step 1 -> Based on the given CACM corpus with 3204 documents, creates a lucene index. (~filepath/ Lucene Implementation/Index)

Step 2 -> Based on given four queries, generates four lists, one per query storing the topmost 100 document ids based on their score. (Based on the default document scoring/ranking by Lucene)

Step 3-> Creates a list of unique term, term\_frequency pairs over the entire corpus collection.

**FileWrite.java** helps as a writer class which writes the respective output to a text file.

**Execution Steps:**

**Steps to run the code->**

**For LINUX**:

* Navigate to the folder where the Project Folder (Lucene Implementation) is present
* Navigate to the src folder inside Lucene Implementation (~/Lucene Implementation/src/lucene/implementation)
* Compile the code (with the External Jar ) from the src folder :::javac -cp “all essential jar files" ~FilePath/ Lucene Implementation/ LuceneImplementation.java

javac -cp "lib/\*.jar" src/lucene/implementation/LuceneImplementation

\*\*\*\* lib contain all four jar files\*\*\*\*\*

**For WINDOWS using NetBeans IDE:**

* Import the Project into the IDE << File --> Open Project >>
* Load the required JAR files as stated in the technical specification
* Right click the program and click on run

**Output:**

Below are the required files generated by this source code:

A lucene index created at below filepath -> ~filepath/ Lucene Implementation/Index

Below files are generated corresponding to given 4 queries which stores the top 100 document id based on their individual scores

* Query\_Qutput1.txt
* Query\_Qutput2.txt
* Query\_Qutput3.txt
* Query\_Qutput4.txt
* Term\_Frequency.txt-> stores sorted (by frequency) list of (term, term\_freq pairs) for the given corpus.

In addition to these text files, we have additional deliverables i.e

**BM25 and Lucene Comparison**.**xlsx** -> gives a statistical comparison of the total number of documents retrieved per given query using this lucene implementation and BM25 based search engine.

**Zipf Curve with Rank\_Term\_freq.xlsx ->** highlights the Zipf’s curve based on the lucene result (Graph of Rank (by decreasing frequency) vs respective probability of occurrence) and (Graph of log(Rank) vs Log(Probability))

**References:**

<http://lingpipe-blog.com/2014/03/08/lucene-4-essentials-for-text-search-and-indexing/>

<https://lucene.apache.org/core/>

<http://stackoverflow.com/questions/2602253/how-does-lucene-index-documents>

<http://stackoverflow.com/hastable>

<http://stackoverflow.com>

---------------------------------------END--------------------------------------------------