Information Retrival and Web Search Individual Assignment-1: Lucene and Cranfield

Vishal Kumar
MSc. Computer Science
Trinity College Dublin
Dublin, Ireland
Kumarv1@tcd.ie
Github: https://github.com/vishalkumarmishra7/LuceneCRANSearch.git

I. Introduction

This document explains concept and method used to make a search engine using Apache Lucene 8.4.1 library in java. Created application uses maven 3.8 for build and deployment. Amazon Web Services instance server is used to deploy application and for evaluation of results Trec_eval 9.0.7 is used with graded judgement available. Gnuplot is used to plot a graph demonstrating performance of different similarity methods. Total 12 combinations of Analysers and Similarity methods are applied to find differences in result, as Follows:

Analyzer	**Similarity**
English	TFIDF
English	BM25
English	LMDirichlet
Standard	TFIDF
Standard	BM25
Standard	LMDirichlet
Simple	TFIDF
Simple	BM25
Simple	LMDirichlet
WhiteSpaceAnalyzer	TFIDF
WhiteSpaceAnalyzer	BM25
WhiteSpaceAnalyzer	LMDirichlet

II. Concept

Cranfield collection contains 1400 text files which acts as document database where search engine searches for CRAN queries. Document database and query file are parsed, document indexes are done on indexer method and query search is perfomed using scoring methods. Retrived documents are then ranked as per alorithm used.

III. Implementation

Project is implemented with maven 3.8 which can be deployed on any local or cloud machine. After compilation, running jar file executes all combination of Analyzer and Similarity available.

A. Project Structure

Java project structure is followed. Package ie.icd.irws.assignment1 has all source java files. startLucene is main class file which is the starting point of application. 12 combinations of Analyzer and Similarity are passed to Indexer and searcher as String array inside loop which is used to perform operations for indexing and searching.

B. Parsing CRAN data and query file

Since Cranfield collection has a number of parts, it contains cran.all.1400 that represents 1400 documents that application indexes. Each document has specific format of fields as:

- .I: It is the unique ID of the CRAN document with range between 1 and 1400.
- .T: It is the title of the document and which of string type.
- .A: Author of a document and of string type.
- .B: It represents Bibliographic information for document of type string.
- .W: It is the actual content of document where search engine performs search.

These tags come in the begining of each line. Any document can have nultiple tags for .A and .B.

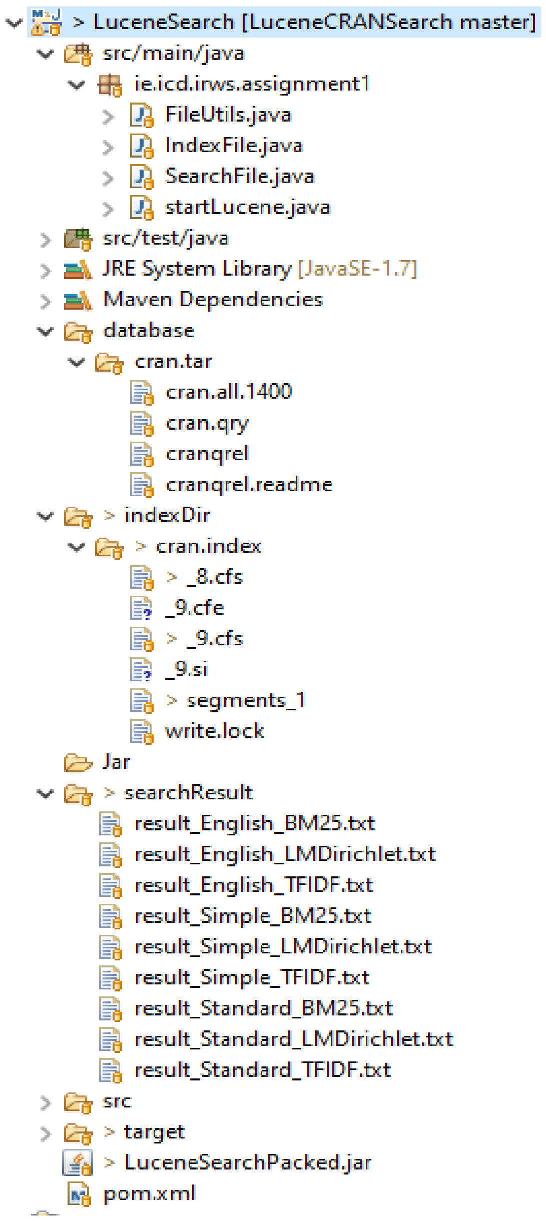
CRAN query file contains total queries as 225 and each query has two tags as:

- .I: It is the unique ID of the CRAN query.
- .W: It is the actual content of query which is searhed by search engine.

FileUtils.java is responsible for Parsing CRAN data file and Cran query file. Inside databaseParser method dictionary map is created using fileList as Cran.all.1400 based on tag defined. And, parseQuery method isi creating qDictionary as query dictionary based on query ID, QueryNo, and query content.

C. Indexing Data Files

When Parsing is done map type List parsedCranFileList is created , IndexFile is responsible for Indexing files using this index parsedCranFileList. Method createCranIndex in IndexFile gets index list and check passed Analyzer and Similarity mode and creates a Index writer based on conguration. addCranFiles method uses index writer to add files as proper



```
> LuceneSearch [LuceneCRANSearch master]
  src/main/java
    ie.icd.irws.assignment1
      FileUtils.java
      IndexFile.java
      SearchFile.java
      startLucene.java
  src/test/java
  JRE System Library [JavaSE-1.7]
  Maven Dependencies
  database
    cran.tar
      cran.all.1400
      cran.qry
      cranqrel
      cranqrel.readme
  > indexDir
    > cran.index
      > _8.cfs
      _9.cfe
      > _9.cfs
      _9.si
      > segments_1
      write.lock
  Jar
  > searchResult
     result_English_BM25.txt
     result_English_LMDirichlet.txt
     result_English_TFIDF.txt
     result_Simple_BM25.txt
     result_Simple_LMDirichlet.txt
     result_Simple_TFIDF.txt
     result_Standard_BM25.txt
     result_Standard_LMDirichlet.txt
     result_Standard_TFIDF.txt
  src
  > target
  > LuceneSearchPacked.jar
  pom.xml
```

Fig. 1. Project Structure.

indexed document based on ID, Title, Location, Authors, and Abstract. Index writer writes index file into path indexDir.

D. Seacrhing Data Files

searchQueries method in SearchFile is resposible for searching parsed cran query queryList into indexes of document. It make map type resultDictionary to store search results. Initially queryList is parsed to get each individual query and based on argument Similarity type and hit per page as 1000 query is searched and result is stored in TopDocs object. Using this TopDocs object curResultList is created and added to List type searchResultData in proper format as per trec_eval requirement. Finally all result data is written with specific Analyzer and Similarity type as result file name and stored in directory searchResult.

.

E. Result Evaulation and Plotting Graph

For evaluating result, generated result files are compared to judgement Qrel file given. And using Precision and Recall data of output graph is generated for Analyzer type English and compared to all 3 types of Similarity used in project.

IV. Discussion

In Fig 2. Comparing results for Analyzer English and Similarity as BM25, LMDirichlet and TFIDF. IN graph it is clear that BM25 is performing better than rest two evenif recall goes to maximum range. In Similar way P/R graph can be generated for other comparisions too. Maximum Mean Average Precision found as **0.42** in case of English Analyzer and BM25 Similarity.

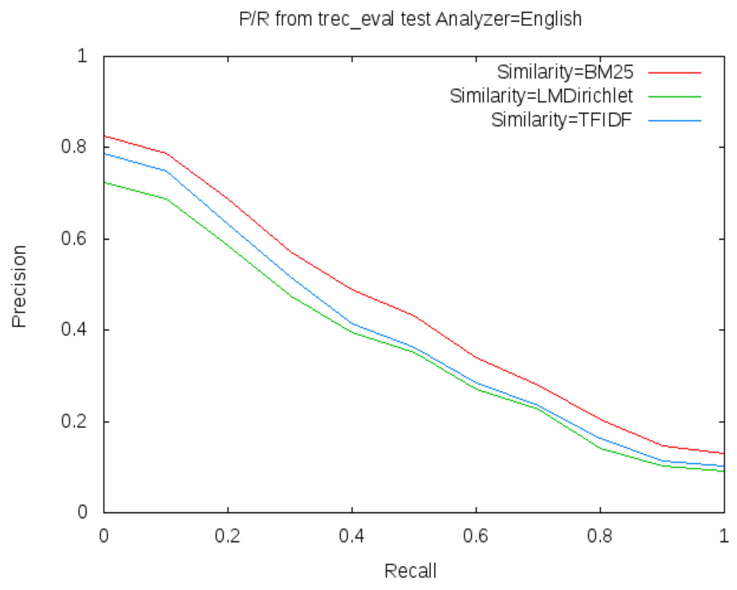


Fig. 2. P/r Graph for English Analyzer and 3 different Similarities..