Ritika Sharma
21334470
sharmari@tcd.ie
MSc Computer Science – Intelligent Systems

Assignment 1 : Lucene and Cranfield

This project requires to work on Lucene Search engine to design Lucene Analysers for content processing and test the engine on Cranfield dataset.

About Lucene

Lucene is a powerful text-search library built on Java to introduce text search to an application. It is a simple yet a sophisticated library under the hood. It searches text or documents by adding index to the content. It allows us perform queries on the index and then rank those results on the basis of relevance.

Indexing

The reason why Lucene can perform searches faster than a regular database is because it searches an index rather than going through the entire content directly.

Lucene has analysers used for analysing text while indexing and searching the content. It splits the content into tokens. The different analysers have different combination of splitting the content into tokens.

Lucene has many types of Analysers. We will dive into three types for this assignment.

1. English Analyser – It is suitable for plain English text that is found in the content. It provides the feasibility to have stop words feature as well.

2. Standard Analyser – It is one of the most sophisticated analyser. It deals with emails, names and addresses. It removes common words and punctuations found in the text and lowercases tokens.
3. Classic Analyser – It is the basic form of analyser provided by Lucene.

Project

For this assignment, we were given Corpus dataset on which we had to perform the indexing after parsing the dataset.

First, we create a Parser class to parse all the fields given in the corpus dataset. We parse individual rows given in the document and split them into individual tokens like Id, Author, Content, Bibliography and Title using regular expression. These same fields are added into the document using document.add method.

After the rows of data is parsed and split into tokens, the data is passed into different types of analysers in the Indexer class. EnglishAnalyzer, ClassicAnalyzer, StandardAnalzer. The data is again checked for Similarity. We use BM25, LMDirichletSimilarity, ClassicSimilarity, BooleanSimilarity.

The parsed and tokenized indexs are passed into the Query Searcher, ListQuery Class. This step is done to query the data and retrieve the results.

Table 1: Standard Analyzer for BM25Similarity		
runid	all	STANDARD
map	all	0.4021
gm_map	all	0.2578
P_5	all	0.4471

sharmari@tcd.ie
MSc Computer Science – Intelligent Systems

Table 2: English Analyzer for BM25 Similarity		
runid	all	STANDARD
map	all	0.4257
gm_map	all	0.3039
P_5	all	0.4524

Table 3: Standard Analyzer for BM25 Similarity		
runid	all	STANDARD
map	all	0.4257
gm_map	all	0.3039
P_5	all	0.4524

Table 4: Classic Analyzer for LMDirichletSimilarity		
runid	all	STANDARD
map	all	0.3274
gm_map	all	0.1862
P_5	all	0.3627

Table 5: Standard Analyzer for LMDirichletSimilarity		
runid	all	STANDARD
map	all	0.3267
gm_map	all	0.1861
P_5	all	0.3627

Table 6: Standard Analyzer for Classic Similarity		
runid	all	STANDARD
map	all	0.3419
gm_map	all	0.2025
P_5	all	0.3858

Here I have added just a few of the combinations of Analyser and Similarities that were tried in the project. The table states map and gm_map to predict the accuracy of the search engine created.

The best combination is English Anlayser for BM25 Similarity. BM25 is the default similarity and has built in normalisation. This works best with English Analyser as the combination provides document length normalisation and BM25 works best for short fields.

This provides best result with map = 0.4257 which is a great score for the dataset among all the others.

Steps to run the program-

1. Login into VMware using the username - azureuser
2. Run cd ..
3. Run cd /opt/info-retrieval
4. Run bash script.sh