CA Part 1 - Apache Lucene Search Engine

Sai Bala Subrahmanya Lakshmi Kanth
19304511
CS7IS3 – Information Retrieval and Web Search

ABSTRACT

Optimisation in search engines is essential for the efficiency of IR systems. This project explores the efficiency of different Lucene Analysers and scoring approaches in enhancing the performance of the Lucene search engine. In this project we explored 3 different Lucene Analysers – English Analyzer, Standard Analyzer and a Custom Analyzer (Custom pre-processing) and 2 different scoring approaches – BM25 and Vector Space Model. The scores for these different models are evaluated against relevancy judgement file using *trec_eval*. These results can be interpreted to choose the most efficient Analyzer and the scoring approach for the Lucene search engine.

PROJECT STRUCTURE

Files

- **cran.all.1400:** Contains a set of 1,400 Cranfield Collection documents for indexing.
- **cran.qry:** Contains 225 example queries that will be used to search and test the Lucene search engine.
- **QRelsCorrectedforTRECeval:** Corrected relevancy judgement file to evaluate the search with trec_eval.

Classes

- **Constants.java:** Contains all the constants for the project.
- **CustomAnalyzer.java:** Creates a custom Lucene Analyzer for content processing.
- **CreateIndex.java**: Creates Cranfield documents and searches using the example queries.

Analyzer Configuration

For this project I have considered 3 different analysers, namely, English Analyzer, Standard Analyzer, and Custom Analyzer.

- **English Analyzer:** The English Analyzer is used as all the Cranfield documents are in English language. Tokenisation and stop word removal performed would be tailored for handling English language specific variations.
- **Standard Analyzer:** The Standard Analyzer is chosen to handle various writing styles and provide a generic solution. English Analyser is a better optimised in this scenario as the Cranfield Collection is in English.
- **Custom Analyzer**: The Custom Analyzer is created combining lowercasing, stop word removal, possessive handling, stemming, and capitalization to suit English language documents like Cranfield Collection. It provides a tailored solution to index the documents. stopWords.txt file, a compilation of the stop words adapted from ranks.nl is used for stop word removal.

Indexing

The code for indexing the Cranfield collection is based on the example code given during the tutorial. The index writer is configured by choosing one of the three analysers chosen by the user. *indexCranfiedCollection()* is a custom method that splits the Cranfield Collection and creates a collection of documents with the fields ID, Title, Author, Bibliography and Words. Once the documents are created, they are indexed using the index writer.

Querying

For this step, the file containing 225 queries is read and only the query strings are stored. This is done because the query IDs are not sequential and are not used later for testing the search. Each of these 225 query strings are parsed as queries which are later used along with the same analyser used for indexing for searching the indexed documents.

Search Methodologies and Scoring

MultiFieldQueryParser() is used to search across multiple fields with varying weights. For this project, the fields "Title" and "Words" from the indexed documents have been used with 40% and 60% as their respective weights.

The user has the option to choose between Vector Space Model and BM25 scoring approaches to evaluate the search results.

- **Vector Space Model (VSM)**: In Apache Lucene, the *ClassicSimilarity()* implements the Vector Space Model for IR. It is the default scoring approach used by Lucene. *ClassicSimilarity()* gives a similarity score between the queries and indexed documents based on Term Frequency (Number of times a specific term appeared in the current document) and Inverse Document Frequency (Higher contribution to the total score from rarer terms).
- **BM25:** Best Match 25 (*BM25*) similarity is based on the probabilistic information retrieval model. Both VSM and BM25 scoring approaches use Term Frequency and Inverse Document Frequency while scoring. However, BM25's scoring is less sensitive to high or low term frequency compared to *ClassicSimilarity* and takes into account the document length normalisation to make sure longer documents won't have a higher score.

RESULTS

Below are the result scores obtained by running the Lucene search engine with different Analysers and similarities.

Map scores

	English Analyzer	Custom Analyzer	Standard Analyzer
VSM	0.3935	0.3935	0.3302
BM25	0.4261	0.4289	0.3947

P_5 scores

	English Analyzer	Custom Analyzer	Standard Analyzer
VSM	0.4160	0.4160	0.3742
BM25	0.4569	0.4551	0.4311

For the above results we can interpret that in the P_5 scores the English Analyzer slightly over performs the Custom Analyzer but when it comes to the map scores, we can see that the Custom Analyser performs better. This is due to the use of custom stop words for stop word removal. We can also see that the Standard Analyzer scores less when compared to the Custom and English because they both are much more customised to English text when compared to the standard Analyzer. Further, BM25 scoring model consistently provides better results compared to VSM.

CONCLUSIONS

This project analyses the significance of selection of Analyzer, scoring approaches to optimise the performance of the Lucene search engine. The English Analyzer is the most suitable Lucene Analyzer to index the Cranfield Collection. However, further work can be done to tailor the Custom Analyzer to more suit the Cranfield Collection. This combined with the versatility of the *MultiFieldQueryParser* in selecting the fields to search from will improve the efficiency of the search engine.

REFRENCES

1) Cambridge University Press. (2009, 04 07). *Okapi BM25: a non-binary model.* Retrieved from https://nlp.stanford.edu/IR-book/html/htmledition/okapi-bm25-a-non-binary-model-1.html

2) Lucene, A. (2013, 06 21). *Apache Lucene - Scoring.* Retrieved from https://lucene.apache.org/core/3_5_0/scoring.html

3) Lucene, A. (n.d.). *Class BM25Similarity.* Retrieved from https://lucene.apache.org/core/8_9_0/core/org/apache/lucene/search/similarities/BM25Similarity.html

4) Lucene, A. (n.d.). *Class ClassicSimilarity.* Retrieved from https://lucene.apache.org/core/8_9_0/core/org/apache/lucene/search/similarities/ClassicSimilarity.html

5) NL, R. (n.d.). *Stopwords*. Retrieved from https://www.ranks.nl/stopwords

6) Turnbull, D. (2015, 10 16). *BM25 The Next Generation of Lucene Relevance*. Retrieved from https://opensourceconnections.com/blog/2015/10/16/bm25-the-next-generation-of-lucene-relevation/

APPENDIX

P_100 scores

	English Analyzer	Custom Analyzer	Standard Analyzer
VSM	0.0576	0.0581	0.0529
BM25	0.0594	0.0600	0.0568

Recall Scores

- **English Analyzer and VSM**

Recall Values	Score
recall_5	0.3176
recall_10	0.4307
recall_15	0.4972
recall_20	0.5363
recall_30	0.6025
recall_100	0.7506
recall_200	0.7506
recall_500	0.7506
recall_1000	0.7506

- **English Analyzer and BM25**

Recall Values	Score
recall_5	0.3525
recall_10	0.4550
recall_15	0.5174
recall_20	0.5754
recall_30	0.6328
recall_100	0.7648
recall_200	0.7648
recall_500	0.7648
recall_1000	0.7648

- **Custom Analyzer and VSM**

Recall Values	Score
recall_5	0.3187
recall_10	0.4386
recall_15	0.5043
recall_20	0.5449
recall_30	0.6078
recall_100	0.7532
recall_200	0.7532
recall_500	0.7532
recall_1000	0.7532

- **Custom Analyzer and BM25**

Recall Values	Score
recall_5	0.3501
recall_10	0.4683
recall_15	0.5343
recall_20	0.5741
recall_30	0.6376
recall_100	0.7718
recall_200	0.7718
recall_500	0.7718
recall_1000	0.7718

- **Standard Analyzer and VSM**

Recall Values	Score
recall_5	0.2831
recall_10	0.3711
recall_15	0.4337
recall_20	0.4800
recall_30	0.5368
recall_100	0.6894
recall_200	0.6894
recall_500	0.6894
recall_1000	0.6894

- **Standard Analyzer and BM25**

Recall Values	Score
recall_5	0.3323
recall_10	0.4287
recall_15	0.4922
recall_20	0.5336
recall_30	0.5915
recall_100	0.7361
recall_200	0.7361
recall_500	0.7361
recall_1000	0.7361