Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CS7IS3 INFORMATION RETRIEVAL AND WEB SEARCH Assignment 1

I have read, and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at: http://www.tcd.ie/calendar

I have also completed the Online Tutorial on avoiding plagiarism 'Ready, Steady, Write', located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write

Student Name & Number	Aditya Kumar Shrivastava TCD ID: 19323354
Module	CS7IS3 INFORMATION RETRIEVAL AND WEB SEARCH
Name of Demonstrator	Dr YVETTE GRAHAM
Title of Assignment	Assignment 1: Continuous Assessment Part 1
Date	24-Oct-2021

Describing the retrieval system:

Text Acquisition: The documents to be indexed were obtained from the Cranfield dataset available on the web.

Document Processing and Indexing:

To index the content in the Cranfield documents for this project, we need to process the documents to convert the words in each document into index terms.
This project implements four inbuilt analysers provided by the Lucene API (analyser-common) and a custom analyser to process the text in the documents. The analysers used are:

1. Standard Analyser. 2. Simple Analyser. 3. Whitespace Analyser. 4. English Analyser, and,

5. A custom analyser that uses Standard tokenisation, filters tokens based on lowercase, uppercase, stop words and implements the porter stem token filter that provides stemming for the English language

Following the analysis (tokenising, streaming tokens, stopping and stemming), we configure the index writer used in the index program to implement each of the analysers mentioned above and the BM25 similarity scoring.

While indexing the Cranfield documents, we store all the fields present (ID, title, author, bibliography and words)in each document. This ensures that the field's full text is stored and returned with the search results when the document is searched.

Searching strategy.

In this project, we implement three main components to search the created indexes. These are:

1. Index Reader to access the created indexes.
2. A multi-field query parser to parse the 225 queries available in the cran.qry file from the Cranfield dataset. For the multi-field query parser, we use the same Analyser that we used to index documents. And finally,
3. Index Searcher to search over a single Index Reader.

To optimise search performance, we also take an additional step in merging all the segments created while indexing into one segment using *indexWriter.forceMerge(1);*

Scoring function.

While performing searching on the indexes, we implement the following scoring strategies:

1. BM25
2. TF-IDF
3. LMDirichletSimilarity: based on Latent Dirichlet Allocation(LDA) (see api documentation)
4. Jelinek-Mercer Similarity
5. MultiSimilarity model combining BM25Similarity and LMDirichletSimilarity.

To perform a search based on a query, we use IndexSearcger.search().scoreDocs to find the top n hits for a query. Following the search and obtaining the score for each document for a query, we write the results in a results.txt file. The results.txt file is in a particular format such that it can be evaluated using trec_eval.

Present and discuss results:

The programmed indexer and searcher were run on the Cranfield documents and cran.qry files. To evaluate the models, we first generate a series of result files for each Analyser coupled with each similarity scoring technique. We then use trec eval to evaluate the retrieval and search system based on the Cranfield query relevance and ranking of each document stored in the results_similarity_analyzer.txt files.

The main metrics used for evaluating the retrieval system were the Mean average precision (MAP) scores and the interpolated recall – precision averages at 0.50 scores (iprec_at_recall_0.50).

I chose iprec_at_recall_0.50 since it gives us a good idea of the average recall level precision from the 11 standard recall levels used to compare the performance of the retrieval system.
I also chose the MAP score since it quantifies how good our model is at performing or searching well for a given query

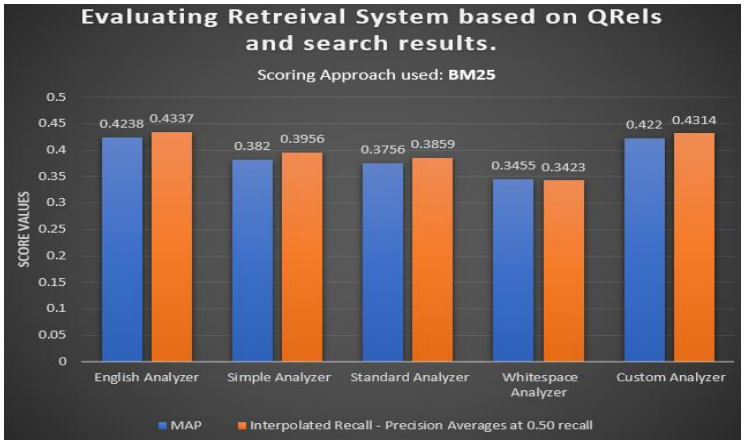
Discussion:

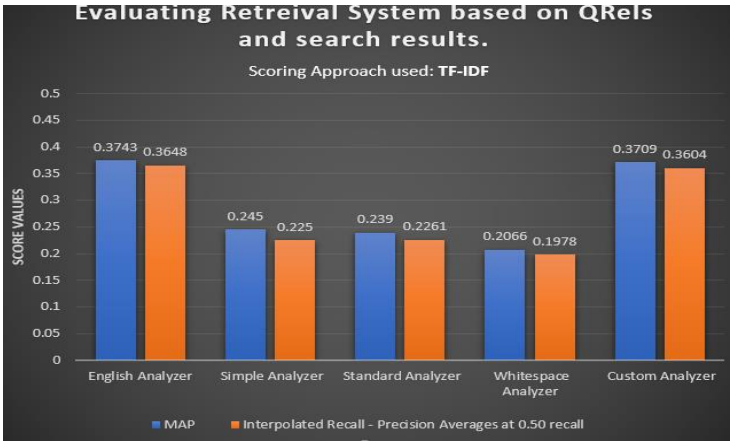
The plots below show that we obtained the best MAP and iprec_at_recall_0.50 scores when implementing the **English Analyser** and the **BM25** scoring technique. This indicates that this combination allows the retrieval system to return the best results for a given query.

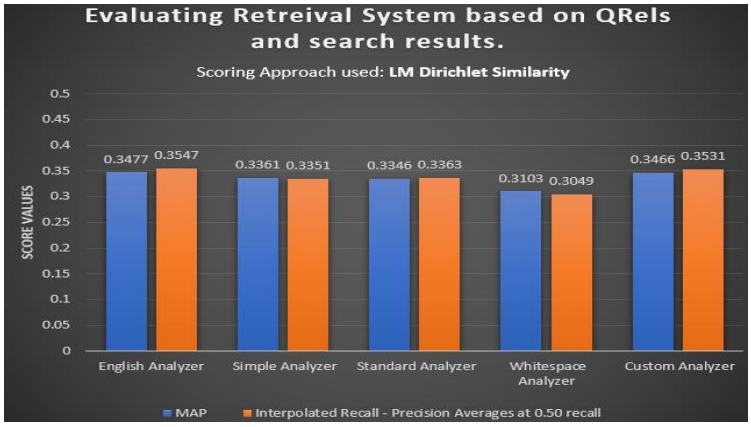
We can also observe from each of the plots below that the English and custom analysers are very close in their scores. This is because the custom analyser implementation in code is very similar to the English Analyser. The only difference is that the custom analyser capitalises tokens, and the English Analyser doesn't.

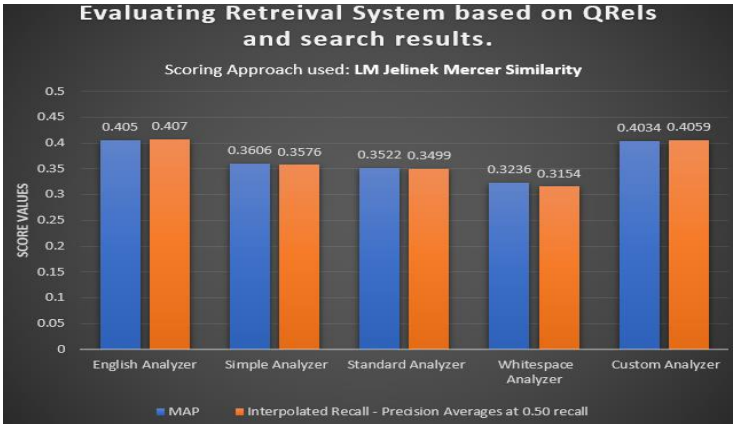
The plots below also show that the multi-scoring technique combining the BM25 and Drichlet similarity scoring techniques performs very well. The multi scoring technique can yield scores and performance very close to the English Analyser's and would be a good implementation for the Cranfield dataset.

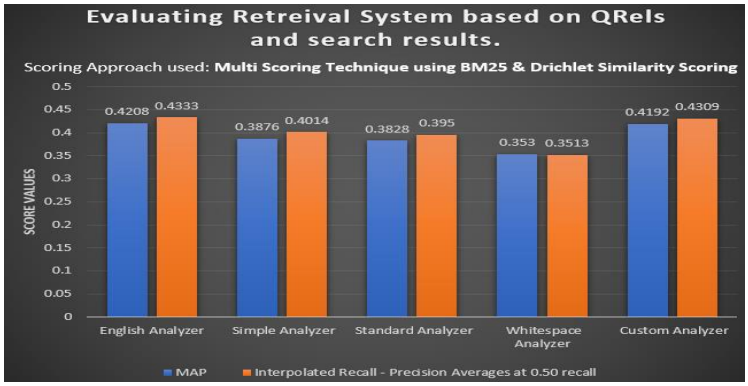
From the evaluation, we can conclude that the BM25 similarity scoring performs the best out of all the implemented scoring techniques. We can also observe that the English Analyser performs the best out of all the implemented analysers for indexing the Cranfield dataset.











How to access my linux box and the repository:

1. Linux VM Public IP Address: 20.105.177.112
2. Connect using a client and use the attached .ppk file: ashrivas.ppk
3. Login as user: azureuser
4. Once logged in, switch user to root using the command: 'sudo su –'
5. Type in the command: 'whoami' and you should see the output as root.
6. To run the indexer and searcher, switch to the directory containing the assignment code. The name of the directory would be "ahsrivas_CS7IS3_Assignment1"
7. Follow the instructions on the readme file

Bibliography

badges, BabakBabak 16111. "How to Evaluate a Search/Retrieval Engine Using Trec_eval? - Stack Overflow." *Stack Overflow*, https://stackoverflow.com/questions/4275825/how-to-evaluate-a-search-retrieval-engine-using-trec-eval. Accessed 24 Oct. 2021.

badges, Katedral PillonKatedral Pillon 14. 6k2020. "Lucene Field.Store.YES versus Field.Store.NO - Stack Overflow." *Stack Overflow*, https://stackoverflow.com/questions/32940650/lucene-field-store-yes-versus-field-store-no. Accessed 24 Oct. 2021.

Contributors to Wikimedia projects. "Latent Dirichlet Allocation - Wikipedia." *Wikipedia, the Free Encyclopedia*, Wikimedia Foundation, Inc., 2 Apr. 2006, https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation.

"How to Read a UTF-8 File in Java - Mkyong.Com." *Mkyong.Com*, 28 Apr. 2009, https://mkyong.com/java/how-to-read-utf-8-encoded-data-from-a-file-java/.

"LMDirichletSimilarity (Lucene 8.0.0 API)." *Apache Lucene - Welcome to Apache Lucene*, 13 Mar. 2019, https://lucene.apache.org/core/8_0_0/core/org/apache/lucene/search/similarities/LMDirichletSimilarity.html.

"LMJelinekMercerSimilarity (Lucene 8.0.0 API)." *Apache Lucene - Welcome to Apache Lucene*, 13 Mar. 2019, https://lucene.apache.org/core/8_0_0/core/org/apache/lucene/search/similarities/LMJelinekMercerSimilarity.html.

"MultiSimilarity (Lucene 8.0.0 API)." *Apache Lucene - Welcome to Apache Lucene*, 13 Mar. 2019, https://lucene.apache.org/core/8_0_0/core/org/apache/lucene/search/similarities/MultiSimilarity.html.

"Org.Apache.Lucene.Search.Similarities (Lucene 8.0.0 API)." *Apache Lucene - Welcome to Apache Lucene*, 13 Mar. 2019, https://lucene.apache.org/core/8_0_0/core/org/apache/lucene/search/similarities/package-summary.html.

ProgrammingKnowledge2. *Maven Tutorial - How To Create a Maven Project Using Eclipse IDE*. YouTube, 13 Mar. 2016, https://www.youtube.com/watch?v=sNEcpw8LPpo.

"Simon Says: Optimize Is Bad for You.... - Trifork Blog." *Trifork Blog*, https://www.facebook.com/Trifork/, 21 Nov. 2011, https://blog.trifork.com/2011/11/21/simon-says-optimize-is-bad-for-you/.

"TFIDFSimilarity (Lucene 8.0.0 API)." *Apache Lucene - Welcome to Apache Lucene*, 13 Mar. 2019, https://lucene.apache.org/core/8_0_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html.