DESIGN AND IMPLEMENTATION OF RETRIEVAL MODELS

1. Introduction:

1.1 Overview:

The main purpose of this to put the core Information Retrieval concepts into practice by building and using our very own retrieval systems.

The Goal of the project is to design and build retrieval systems, evaluate and compare their performance levels in terms of retrieval effectiveness.

As part of *phase1* of this project, we have carried out three tasks:

Task1: As part of this task, we have implemented four runs each representing a retrieval model.

- BM25 retrieval model
- Tf-idf retrieval model
- Smoothed Query Likelihood retrieval model
- Lucene’s default retrieval model

So, this task includes **four baseline runs**.

Task2: As part of this task, we have implemented pseudo relevance feedback on BM25 retrieval model.

Task3: As part of task 3, we have implemented following models:

- BM25 with stopping, and using stemmed corpus and stemmed query subset
- Tf-idf with stopping, and using stemmed corpus and stemmed query subset
- Smoothed Query Likelihood with stopping, using stemmed corpus and stemmed query subset

So, this task includes **six runs**.

As part of *Phase2* of this project, we have implemented a snippet generation technique and query term highlighting within the results in the BM25 run.

As part of *Phase3* of this project, we have implemented an evaluation system to assess the performance of our retrieval systems in terms of their effectiveness. The evaluation gives the values for the following evaluation parameters:

- MAP: Mean Average Precision
- MRR: Mean Relevance Rank
- P@5,P@20: Precision at rank 5 and 20
- Full tables showing Precision and Recall for all the ranks

As part of Extra Credit task, we have modified our unigram indexer to hold the term positions for proximity enabled search on BM25 model. This is performed with stopping and also without it.

2. Literature and Resources:

2.1 Overview of the techniques used in the project:

- <u>BM25</u>: We have implemented this model by using the formula cited above to score each and every documents related to the terms in the query. Values for K1 and K2 are chosen as directed in the problem statement.
- <u>Tf-idf:</u> This assigns to a term t, a weight in document d that is,
  - Highest when t occurs many times within a small number of documents
  -lower when the term occurs fewer times in a document, or occurs in many documents.
  -lowest when the term occurs in virtually all documents.
- <u>Smoothed Query Likelihood</u>: Foundation for this model is taken using Baye's rule which states as below

  $$p(D|Q) \overset{rank}{=} P(Q|D)P(D)$$
- <u>Lucene</u>: We have used Lucene 4.7.2 to get the ranking for the documents. This program is slightly modified so that each query is read from the document and result is printed in the desired format.
- <u>Stopping</u>: It's the process of removing common words from the stream of tokens that become index terms. Stop words are taken into a list and while parsing the document for indexing, if the word is present in the stop word list, that word is not added to the inverted index.
- <u>Stemming</u>: The task of stemming component is to group words that are derived from a common stem. Here we are using cacm-stemmed corpus to create the new inverted index containing stemmed words and documents related to it. We ignored the digits appearing at the end of each document while creating the stemmed corpus.
- <u>Query refinement approach</u>: Ideally the query and corpus should be refined in the same way. While refining the query, we are initially removing numbers that exist between the braces. Other than '.' ',' ''' and ':' all other punctuations are ignored. These three are retained only in between the numbers. Case - folding is done and extra spaces are removed.
- <u>Query Expansion</u>: We have carried out Query Expansion using pseudo relevance feedback technique.
- <u>Snippet Generation</u>: We are carrying out this in the following way:
  -Initially tri-grams of query is compared against the top ranked document. If it is found, then 50 characters before and after the tri-gram along with the tri-gram is considered for the snippet and displayed. The matched tri-gram is highlighted in the snippet.
  -If we don't find the query's tri-grams in the document, the same process is carried out for bi-grams and unigrams of the query.
- <u>Evaluation technique</u>: We use precision and recall to find the effectiveness of the retrieval model.

-Recall: It gives the proportion of the relevant documents that are retrieved.

$$Recall = \frac{|A \cap B|}{|A|}$$

-Precision: It gives the proportion of the retrieved documents that are relevant.

$$Precision = \frac{|A \cap B|}{|B|}$$

-MAP: Mean Average Precision

Sum of average precisions/Total number of queries

-MRR: Mean Relevance Rank

Sum of RR of each query/Total number of queries

2.2 Research Articles Referred:

- We have referred the following articles for implementing term proximity in BM25 model.
  https://nlp.stanford.edu/IR-book/html/htmledition/query-term-proximity-1.html
  https://link.springer.com/chapter/10.1007/978-3-642-36981-0_44
  https://link.springer.com/chapter/10.1007/978-3-540-78646-7_32?no-access=true

  We went through various research articles regarding different perspectives of calculating term proximity and have come up with our own formula for calculating score for documents using term proximity.

3. Implementation and Discussion:

This section contains detailed description of the chosen models and design choices made during the implementation of these models. It also includes query-by-query analysis of the run performed in the task2 of phase1.

3.1 Task1- Phase1:

- BM25 Model: It's an effective ranking algorithm based on binary independence model which also includes document and query term weights.

$$\sum_{i \in Q} \log \frac{(r_i+0.5)/(R-r_i+0.5)}{(n_i-r_i+0.5)/(N-n_i-R+r_i+0.5)} \cdot \frac{(k_1+1)f_i}{K+f_i} \cdot \frac{(k_2+1)qf_i}{k_2+qf_i}$$

-K1, K2 – parameters whose values are set empirically. We have takes K1 and K2 values as 1.2 and 100 respectively.

$$K = k_1((1-b) + b \cdot \frac{dl}{avdl})$$

-dl= document length.

-b= 0.75, avdl= average document length

-ri= number of relevant documents containing the term i. We have taken ri=0

-R= number of relevant documents for the query. We have taken R=0

-ni= number of documents containing the term i.

-N= total number of documents in the collection.

-fi= frequency of the term i in the document

-qfi= frequency of the term i in the query

- Procedure:
- The list of documents is taken from the inverted index for each term in the query.
- The score for each document is calculated using the above formula.
- The scores are stored in the dictionary and sorted in the descending order of their scores.
- The scores along with the document name is printed in the text document in the below format

QueryID 'Q0' DocID Rank BM25_Score ' BM25_Model'

- Tf-idf Model: This is the simplest model which only considers term frequency and document frequency.

Score = (fi/dl) * (1+ log (N/(ni+1)))
-dl= document length
-fi= frequency of the term i in the document
-N= total number of documents in the corpus
-ni= number of documents containing the term i

- Procedure:
- The list of documents is taken from the inverted index for each term in the query.
- A table containing unigram tokens with the name of the document is used to get document length for each of the document
- Score for each document for each term is calculated using the above mentioned formula.
- The document and its respective score is stored as a dictionary and sorted based on the score in descending order.
- The result is printed into a text file in the below format
QueryID 'Q0' DocID Rank BM25_Score 'tf-idf_Model'

- Smoothed Query Likelihood model: In this model documents are ranked by the probability that the query text could be generated by the language model. Query generation is the measure of how likely it is that a document is about the same topic as the query. Smoothing is included in this to avoid various estimation problem and to overcome data sparsity.

$$\log P(Q|D) = \sum_{i=1}^{n} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})$$

-fqi,D= number of times the term i occurs in the document D
-|D|= length of the document
-lambda= 0.35 as given in the problem statement
-Cqi= number of times the term i occurs in the whole collection of documents
-|C|= total length of all the documents present in the corpus.

- Procedure:

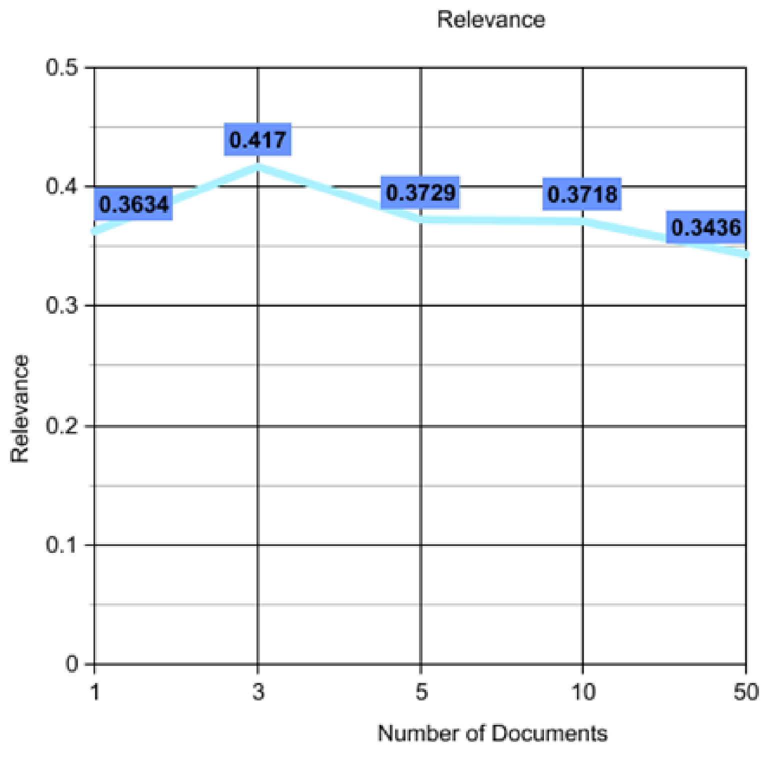
- The list of documents for each term of the query is taken from the inverted index.
- fqi is calculated for each term with the help of inverted index generated.
- Document length is found out using the unigram tokens generated
- Cqi for each term is calculated using the unigram_tf table. This table contains each term and its frequency in the whole collection.
- The score for each document is calculated using the above formula and stored as a dictionary.
- The dictionary is sorted in the descending order based on the score generated
- The result is printed into a text file in the below format
  QueryID 'Q0' DocID Rank BM25_Score 'QLikelihood'

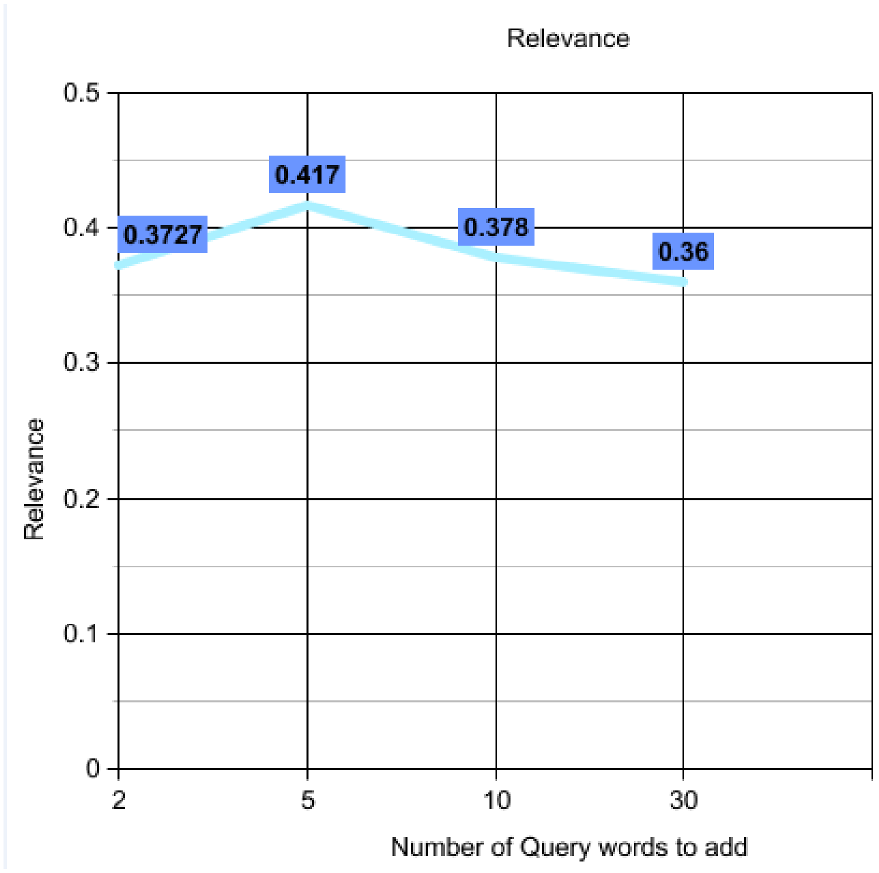
3.2 Task2- Phase1:

➢ Pseudo Relevance Model: We have implemented this for the run BM25.
Procedure:

- Rank documents using BM25 score for a query Q
- Select some number of top ranked documents to the set of C
- Calculate the relevance model probabilities P(W|R) using the estimate for P(W,Q1….Qn)
- Rank documents again using BM25 score function.

➢ Optimum number of documents and number of queries is chosen with the help of the following graphs:





3.3 Query-by-Query Analysis:

Analysis for Extra credit task:
Effect or Query By Query Analysis:
Effectives went up from .38 to .40
Scenario 1:

```
56 Q0 CACM-2921 5 17.2623752838 BM25_Unigram_Casefolding
56 Q0 CACM-2931 6 16.3745477394 BM25_Unigram_Casefolding
56 Q0 CACM-2931 5 14.6226345734 BM25+TermProximity_Unigram_Casefolding
56 Q0 CACM-2921 6 14.3319992293 BM25+TermProximity_Unigram_Casefolding
```

Consider the following, BM25 Consider CACM-2921 to have ranking than CACM-2931 But in Bm25+Term Proximity Considers CACM-2931 to have ranking.
Reason: Although CACM-2921 might have slightly more query terms appearing in the document. But CACM-2931 has more query terms appearing in close proximity (of automata and) compared to CACM-2921 hence ranked higher than 2931.
Scenario 2:

```
56 Q0 CACM-2327 1 24.6129189391 BM25_Unigram_Casefolding
56 Q0 CACM-2650 2 21.5131034173 BM25_Unigram_Casefolding
56 Q0 CACM-2327 1 20.5592385028 BM25+TermProximity_Unigram_Casefolding
```

56 Q0 CACM-2650 2 17.9489875612 BM25+TermProximity_Unigram_Casefolding
Here CACM-2650 has lower ranking than CACM-2337,
Although CACM 2337 has more query terms co appearing "the study of" than CACM 2650 which as only two query terms co appearing "regular languages" CACM2337 is ranked higher because, it has more query terms appearing in the document which are not in close proximity to one another, since this is considered by BM25, The score of CACM-2337 still remains higher than the score of CACM-2650.

3.4 Term proximity score:

Main Idea: Higher scores to be allocated to document that have query terms in the same order in the document.

Scenarios Considered: In the following examples

Document 1 is assumed to have (some or all) query terms in the same order in the document. Document 2 is assumed to have query terms but query terms do not appear in close proximity in the document.

- If both Document 1 and Document 2 have the approximately same number of query terms in the document, then Document 1 has higher ranking than Document 2 since the query terms appear together.
- If Document 2 has significantly more query terms than Document 1, (Or Doc 1 has some query terms appearing in close proximity) document 2 should have higher ranking as it has more query terms and consequentially is more relevant than doc 1 (although doc 1 has query terms appearing together, it may have just query terms in close proximity).

First, we score the document by BM25 model, then allocate some score for each term co-occurrence. For a document score (81% of score is based on BM25 and 19% based on term proximity)
Document Sore = (1 – Lambda) BM25 Score + (Lambda) Term Proximity Score
Our Experiments show that for Lambda = .19 the relevance is effective. Further drop in lambda value will shift to binary relevance model. Further increase in lambda will drop the relevance score of the model.
Proximity Term Measure:
Proximity Score = (Number of times the terms co-appear in the document)
Consider the query "The quick brown fox":
For the document we count the number of times words co-occur in the document
"The quick" "quick brown" "brown fox" all terms co appear right next to each other hence their score is 3.
Allocating Term Proximity Score:
If two words appear together then they are given highest score. If they appear further away they are given less weight.
(Started out with 1, 0.75, 0.50,0.33)

Example:
The quick: Score = 1
The very quick: Score = 0.95
The very very quick: Score = .90
The very very very quick: Score = 0.80

4. Conclusions and Outlook:

4.1 Conclusion:

Examining the results of all the run we can conclude that

- BM25 with pseudo relevance has the highest retrieval efficiency, given that right parameters are chosen pseudo relevance increases the relevancy from base BM25 model (MAP 0.417)
- BM25 with stopping also increases the relevance from base BM25 model run, as we remove the unwanted noise from the query (words like the which are present in numerous documents)
- Base BM25 model has the highest relevance as compared to query likelihood. BM25 has higher MAP and MRR than query likelihood
- Lucene has higher relevance (0.417) given that it incorporates query/term boosting and a mixture of Boolean and vector space model makes it efficient
- Tf.idf has the lowest MRR(0.46) and MAP (0.26) just goes to show it is not a very promising when chosen as a retrieval model
- BM25 with Term Proximity increases the relevance of the model significantly and BM25 (from 0.38 to 0.40) with Term Proximity and Stopping goes further to greatly enhance the MAP. Given all the models so far BM25 with Term Proximity and Stopping has the higher Relevance

4.2 Outlooks:

In this project we are mainly considering the frequency of terms, i.e. number of times the word occurs in the document and in the collection. We have considered smoothing factors and logarithms to normalize the score even if the document contains very few query terms.
There are few areas which can be improved to get the score and ranking of the documents in a better way:

- ➢ We can add fields to the inverted list which can contain information whether the query terms appear in the heading of the document or in the body of the document. In this way, relevancy score can be improved.
- ➢ Users usually prefer a document in which most of the query terms appear close to each other, because this is the evidence that the document has text focused on their query intent. So this calculating this term proximity can also be included so that users find their relevant documents at top.
- ➢ In case of snippet generation we are just taking few characters and displaying to the user. Going forward, we can check if the snippet forms a meaningful sentence.

5. Bibliography:

- Croft, W.Bruce; Metzler, Donald; Strohman, Trevor Search Engines: Information Retrieval in Practice. Pearson Education 2015
- Manning, Christopher D; Raghavan, Prabhakar; Schutze Hinrich An Introduction to Information Retrieval. Cambridge England: Cambridge University Press 2009
- http://people.cs.georgetown.edu/~nazli/classes/ir-Slides/LanguageModel-13.pdf
- https://nlp.stanford.edu/IR-book/html/htmledition/okapi-bm25-a-non-binary-model-1.html
- http://ir.dcs.gla.ac.uk/~ronanc/papers/cumminsSIGIR09.pdf
- https://lucene.apache.org/core/3_6_0/scoring.html