INFORMATION RETRIEVAL

Fall-2016-Final Project

Guided by: Professor Nada Naji

Team Members:

- **Anurag Obulampalli**
- **Meghna Tulasi**
- **Vyshaal Narayanam**

Table of Contents

Introduction

1.1 Overview:

- The goal of the project is to design and build our own Information Retrieval systems, evaluate and compare their performance levels in terms of retrieval effectiveness by using our own search engines.

- We have designed seven distinct runs which include 4 baseline runs, 1 query expansion run, 1 stopping , 1 query expansion combined with stopping and 1 stemming run.

- We have used the following retrieval models:

  - BM25
  - tf-idf
  - Cosine Similarity
  - Lucene Systems

- We have three additional runs using the following techniques:

  - Pseudo relevance feedback query expansion technique
  - Cosine Similarity retrieval model along with Stopping technique
  - Cosine Similarity retrieval along with Stemming technique

- We assess the performance of these seven distinct runs in terms of the following retrieval effectiveness measures:

  - Mean Average Precision (MAP)
  - Mean Reciprocal Rank (MRR)
  - P@K measure where K=5 and K=20
  - Precision & Recall

- We have Snippet Generation and Highlighting for our search engine.

1.2 Contribution of Team Members:

- The project helps us understand which retrieval model works better for different types of queries and documents such as long queries, short queries, lengthy documents, documents with lots of stop words, short documents etc.
- The detailed contribution of each member is given below:

  - Vyshaal Narayanam: Analysis and implementation of retrieval models in Python programming language. Implemented BM25, Cosine Similarity, tf-idf, query expansion run, stopping run and query expansion combined with stopping run.

  - Meghna Tulasi: Built a search engine using Lucene using Java programming language. Implemented a snippet generation technique for query term highlighting. Contributed to the documentation of the project. Analysis of results.

  - Anurag Obulampalli: Implementation of evaluation of seven distinct runs by calculating MAP, MRR, P@K for K = 5 and 20, Recall and Precision values to compare the performance of all the runs using Python. Contributed to the documentation of the project.

Literature and Resources

2.1 Techniques:

- BM25 Model: 'relevance_dict' file contains set of relevant documents has been used. The values of '*K*', '*k1'* and *'k2'* are chosen as per TREC results.
- Cosine Similarity: For calculating cosine similarity, we have used tf*idf as the term weights for the query and document terms.
- tf-idf measure: For calculating tf-idf for each document, we used the raw value of term frequency for the document * inverse document frequency.
- Lucene: We have used the standard Lucene open source library to perform indexing and search operations.
- Query Expansion: Vector space model using Rocchio algorithm for pseudo relevance feedback has been implemented for query expansion using k=5 and N=12.
- Stopping: The stop list – 'common_words.txt' has been used to perform stopping
- Stemming: We have performed stemming using Cosine Similarity model with the help of the stemmed version of the corpus 'cacm_stem.query' and query 'cacm_stem.query'.
- Precision, Recall MAP, MRR, P@K values have been calculated for each run.
- Snippet Generation and Highlighting

2.2 Third Party Tools:

The following third party tools have been used when required:

- **Beautiful SOUP**: Beautiful SOUP has been used for parsing the documents in the corpus and the given queries.
- **PyStemmer** :provides access to efficient algorithms for calculating a "stemmed" form of a word.
- **Pickle**: Implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure.
- **Collection:Counter** : Used to calculate term frequency
- **csv**: Import and export format for spreadsheets and databases.
- **matplotlib.pyplot : matplotlib** is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
- **Lucene Libraries**: Following referenced libraries are used to run the Lucene code:
  - lucene-queryparser-VERSION.jar
  - lucene-analyzers-common-VERSION.jar
  - lucene-core-VERSION.jar

2.3 Research Article References:

- http://nlp.stanford.edu/IR-book/pdf/06vect.pdf
- http://bendemott.blogspot.com/2013/11/installing-pylucene-4-451.html
- http://lucene.apache.org/pylucene/
- https://www.academia.edu/2787890/Relevance_feedback_and_inference_networks
- web.stanford.edu/class/cs276/handouts/lecture9-queryexpansion.ppt
- http://www.seobythesea.com/2013/02/google-snippets-search-results/

Implementation

3.1. BM25

- The bm_25_value function takes document, query and query id and finds query terms that matches in the document. When a token in the query matches with the tokens in the document, the rank is calculated.

- BM25 extends the scoring function for the binary independence model to include document and query term weights.

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

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

* Where N -> Total number of documents
* R -> Number of relevant documents for query
* r_i -> Number of relevant documents containing term i
* n_i -> Number of documents containing term i
* f_i -> frequency of term i in the document
* qf_i-> frequency of term i in the query
* k_1, k_2, b and K are parameters whose values are set empirically.
* dl -> length of the document
* avdl -> average length of a document in the collection.

3.2. Cosine Similarity

- The cosine_sim_value function takes document, query , document and query magnitude as input. When a token in the query matches with the tokens in the document, the query term frequency is incremented and is multiplied with query idf. And the document tf-idf is normalized.
- get_ranked_list function creates list where in document scores are sorted in descending order stored.

$$Cosine(D_i, Q) = \frac{\sum_{j=1}^{t} d_{ij} \cdot q_j}{\sqrt{\sum_{j=1}^{t} {d_{ij}}^2 \cdot \sum_{j=1}^{t} {q_j}^2}}$$

  - D_i ->Document Vector for i term
  - Q -> Query Vector for query

- The calculate_document_magnitude function and calculate_ query_magnitude calculates document and query tf-idf.

$$d_{ik} = \frac{(\log(f_{ik}) + 1) \cdot \log(N/n_k)}{\sqrt{\sum_{k=1}^{t} [(\log(f_{ik}) + 1.0) \cdot \log(N/n_k)]^2}}$$

  - d_{ik} -> Document magnitude for k term in i document
  - f_{ik} -> Frequency of k term in i document
  - N -> Total number of documents
  - n_k->number of documents having term k

3.3. TF-IDF

- tf_idf_value function takes document and query as input and matches query token with document token and calculates term frequency and inverted document frequency.

- In tf_idf_value function we are implementing weighting scheme-2 from the following table. This is to reduce the dependency of term on the frequency as more documents that a term occurs in, the less discriminating the term is between documents and, consequently, the less useful it will be in retrieval.

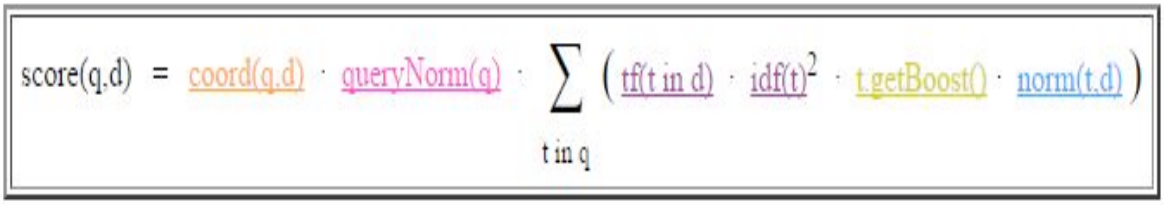
Recommended TF-IDF weighting schemes

weighting scheme	document term weight	query term weight
1	$f_{t,d} \cdot \log \frac{N}{n_t}$	$\left(0.5 + 0.5 \frac{f_{t,q}}{\max_t f_{t,q}}\right) \cdot \log \frac{N}{n_t}$
2	$1 + \log f_{t,d}$	$\log(1 + \frac{N}{n_t})$
3	$(1 + \log f_{t,d}) \cdot \log \frac{N}{n_t}$	$(1 + \log f_{t,q}) \cdot \log \frac{N}{n_t}$

* N -> Total number of documents
* n_t -> number of documents containing term t
* $f_{t,d}$ -> Frequency of term t in document d

3.4. Lucene

- Lucene scoring uses a combination of the Vector Space Model (VSM) of Information Retrieval and the Boolean model to determine how relevant a given Document is to a User's query. In general, the idea behind the VSM is the more times a query term appears in a document relative to the number of times the term appears in all the documents in the collection, the more relevant that document is to the query.
- The Lucene takes in corpus and index location and provides document scores for given query file in Lucene.csv.



  - ***tf(t in d)*** -> correlates to the term's *frequency*
  - ***idf(t)*** -> Inverse Document Frequency
  - ***coord(q,d)*** -> score factor based on how many of the query terms are found in the specified document.
  - ***queryNorm(q)*** -> normalizing factor used to make scores between queries comparable.
  - ***t.getBoost()*** -> search time boost of term *t* in the query *q* as specified in the query text
  - ***norm(t,d)*** -> encapsulates a few (indexing time) boost and length factors:
  - **Document boost** - set by calling doc.setBoost() before adding the document to the index.
  - **Field boost** -> set by calling field.setBoost() before adding the field to a document.
  - **lengthNorm**(field)- computed when the document is added to the index in accordance with the number of tokens of this field in the document, so that shorter fields contribute more to the score.

3.5. Pseudo Relevance Feedback

- We have implemented query expansion technique using Rocchio algorithm where top 12 documents are considered relevant(γ =0) and the highly relevant terms found in those documents are added to the query terms.
- The basic assumption is high frequency terms in the feedback documents are considered relevant to the query with respect to the user.
- The number of expansion terms can be reduced with this technique and even with reasonably small number the retrieval effectiveness can be greatly increased. This allows us to control query traffic within acceptable range and make the method more feasible in the search engines.

$$q_j' = \alpha.q_j + \beta.\frac{1}{|Rel|}\sum_{D_i \in Rel} d_{ij} - \gamma.\frac{1}{|Nonrel|}\sum_{D_i \in Nonrel} d_{ij}$$

- The technique adds additional terms extracted from feedback documents to improve query expansion and retrieval effectiveness.

3.6. Query Expansion Combined with Stopping

- Pseudo Relevance Feedback using Rocchio algorithm has been implemented after removing stop words to compare its retrieval efficiency with the previous runs.

3.7. Retrieval Effectiveness Metrics

- P@K measure
  - Precision at where K=5 and K=20

- Mean Average Precision (MAP)
  - Average of precision values of the retrieved relevant documents.

- Mean Reciprocal Rank (MRR)
  - Reciprocal rank is the reciprocal of the rank at which the first relevant document is retrieved.
  - Average of reciprocal rank over a set of queries is MRR.

- Precision & Recall
  - Recall is the proportion of the relevant documents that are retrieved
  - Precision is the proportion of the retrieved documents that are relevant

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

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

A -> Relevant Documents
B -> Retrieved Documents

3.8. Snippet Generation and Highlighting

- In Snippet Generation, the query is read from 'cacm.query' file and is converted to tokens and is matched in the corpus.
- Query tokens with minimum match are effective and score is number of query tokens/total no of words in document.
- The term matched is highlighted and printed along with particular sentence.

3.9. Query Analysis

- Query ID: 7

- BM25:

  First 7 documents out of the top 10 documents retrieved are relevant. The document at rank 1 is CACM-3043 as the tokens "processes communicate and synchronize" from the query were found in that document. The implementation of BM25 takes into account the total number of relevant documents while calculating the score this is why 7 out of top 10 documents retrieved were relevant.

- Lucene:

  First 3 documents out of the top 10 documents retrieved are relevant. The document at rank 1 is CACM-3043. This is because "processes communicate and synchronize" tokens from the query were found in the document CACM-3043.

  Due to VSM extended to Boolean, instead of giving no relevant documents, it provides 3 relevant documents with more number of query terms matching in the retrieved documents.

- Query Expansion:

  Only the First document CACM-3148 of the top 10 documents retrieved was relevant. The tokens that were similar in the query and document were "distributed", "particular", and "message". In Pseudo Relevance feedback technique which is used to implement the query expansion, the top 10 documents are considered relevant and the high frequency terms from these documents are extracted and added to the query. This results in retrieving documents that are topically relevant to the query irrespective of user's relevance.

- Query Expansion with stopping:

  Only the First document CACM-3043 of the top 10 documents retrieved was relevant. The tokens that were similar in the query and document were "processes communicate and synchronize". This technique is used to implement the query expansion with stopping, the top 10 documents are considered relevant even if stop words are removed. This results in retrieving documents that are topically relevant to the query irrespective of user's relevance.

- Stemming:

  The first 60 retrieved documents are non-relevant with respect to the given 'cacm.rel' relevance judgement file. The document CACM-1811 at rank1 has tokens similar to "program" and "distributed" in query. CACM-2685 at rank 61 has no matching tokens. This implementation results in retrieving documents that are topically relevant based on high frequency terms.

- Stopping:

  First 6 documents out of the top 10 retrieved documents are non-relevant. The document is CACM-2226 at rank1 has tokens similar to "program", "algorithm" and "correctness" which are present in the query. Stop words like "the", "for" etc., resulted in retrieving unnecessary documents that did not contain query terms earlier before which is not the case here. CACM-3043 at rank 2 has matching tokens. This implementation results in retrieving documents that are relevant to the query irrespective of user's relevance.

- TF-IDF:

  First 2 documents out of top 10 retrieved documents are relevant. The document CACM-2342 at rank1 has tokens similar to "progress", "communicate" and "correctness" which are present in the query. CACM-3043 at rank 3 has matching tokens. This implementation results in retrieving documents that are relevant to the query irrespective of user's relevance.

- Cosine-Similarity:

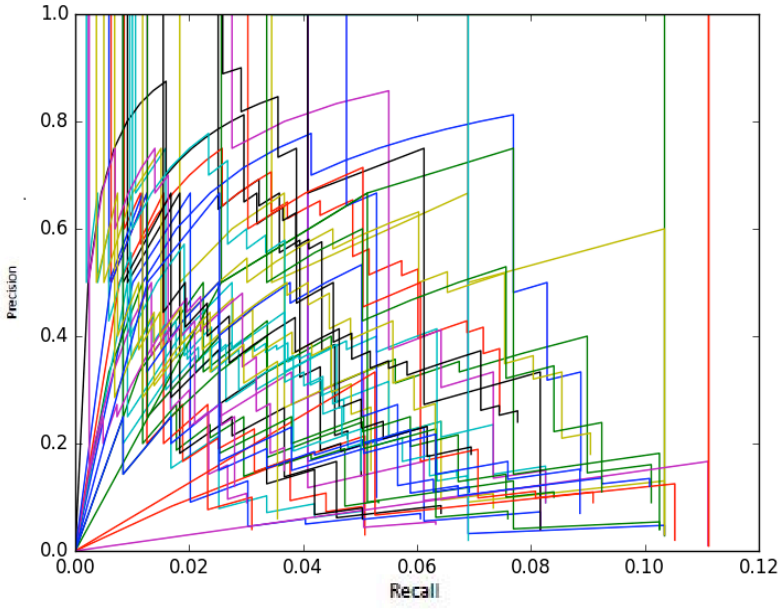
  First 10 documents out of top 10 retrieved documents are non-relevant. The document is CACM-2950 at rank1 has tokens similar to "process" and "correctness" which are present in the query. The document CACM-2342 at rank1 has tokens similar to "progress", "communicate" and "correctness" which are present in the query. This implementation results in retrieving documents that are topically relevant and also considers proximity.

Results

- Query ID :7
- Precision V/s Recall graphs are provided under respective retrieval model.
- BM25:

7	CACM-3043	1	0.00358423	R
7	CACM-3128	1	0.00716846	R
7	CACM-2865	1	0.01075269	R
7	CACM-2912	1	0.01433692	R
7	CACM-2376	1	0.01792115	R
7	CACM-2342	1	0.02150538	R
7	CACM-3141	1	0.02508961	R
7	CACM-2949	0.875	0.02508961	NR
7	CACM-1854	0.77777778	0.02508961	NR
7	CACM-0046	0.7	0.02508961	NR
7	CACM-2890	0.63636364	0.02508961	NR

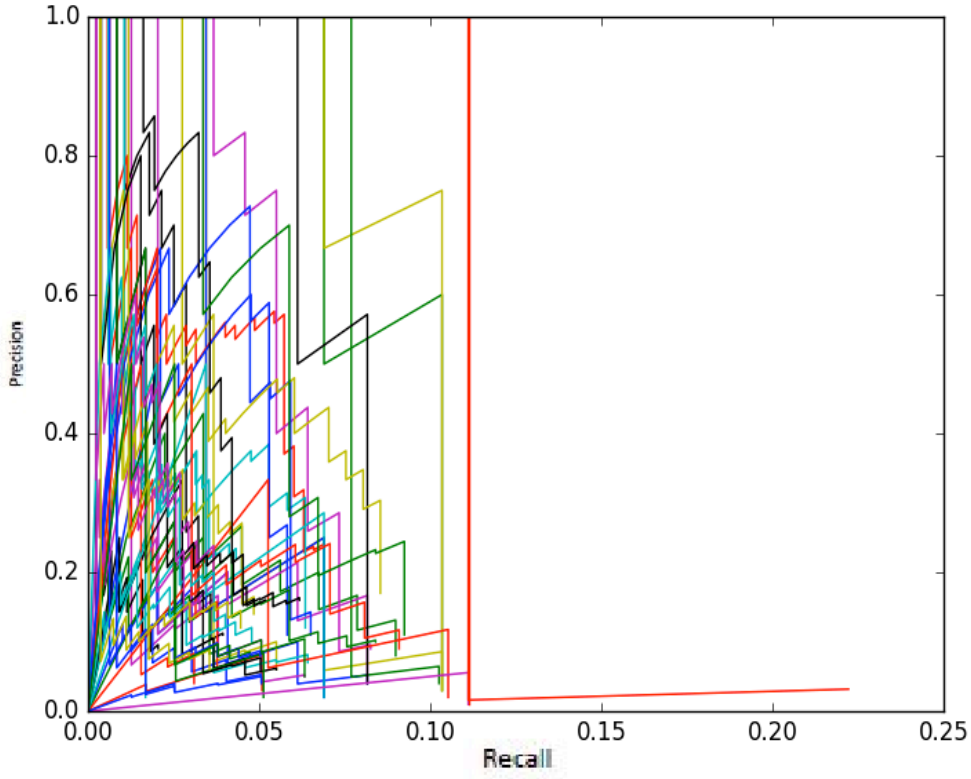
MAP: 0.526222869543779
MRR: 0.83076923076923
P@5,20: 1,0.55



- Lucene:

7	CACM-3043	1	0.00358423	R
7	CACM-2912	1	0.00716846	R
7	CACM-2342	1	0.01075269	R
7	CACM-2111	0.75	0.01075269	NR
7	CACM-2865	0.8	0.01433692	R
7	CACM-3128	0.83333333	0.01792115	R
7	CACM-1623	0.71428571	0.01792115	NR
7	CACM-3141	0.75	0.02150538	R
7	CACM-2903	0.66666667	0.02150538	NR
7	CACM-3148	0.7	0.02508961	R
7	CACM-2317	0.63636364	0.02508961	NR

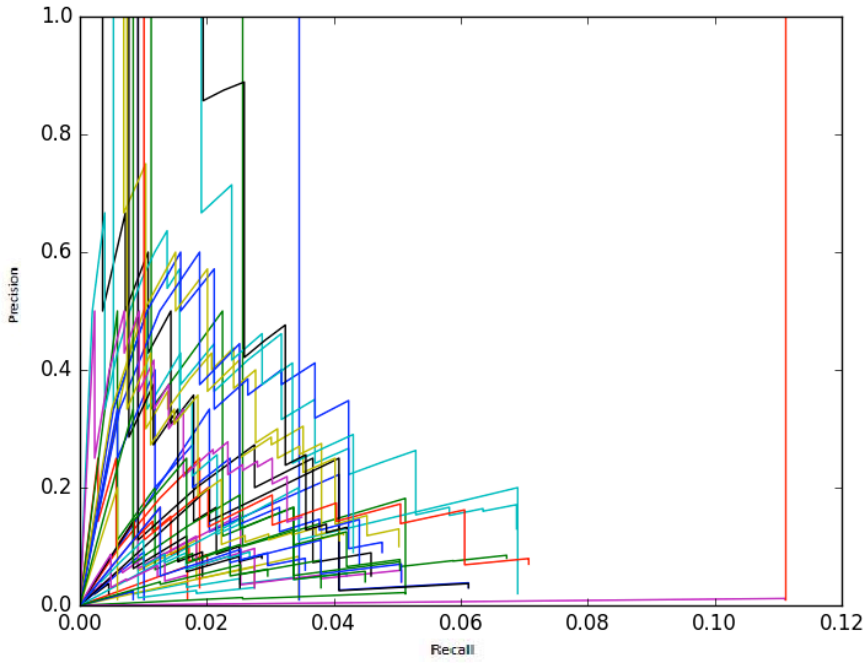
MAP: 0.405588746737929
MRR: 0.689137116362071
P@5,20: 0.8,0.4



- Pseudo Relevance Feedback:

7	CACM-3043	1	0.00358423	R
7	CACM-1366	0.5	0.00358423	NR
7	CACM-3148	0.66666667	0.00716846	R
7	CACM-1854	0.5	0.00716846	NR
7	CACM-2342	0.6	0.01075269	R
7	CACM-1135	0.5	0.01075269	NR
7	CACM-2645	0.42857143	0.01075269	NR
7	CACM-2865	0.5	0.01433692	R
7	CACM-3080	0.44444444	0.01433692	NR
7	CACM-2378	0.4	0.01433692	NR
7	CACM-1939	0.36363636	0.01433692	NR

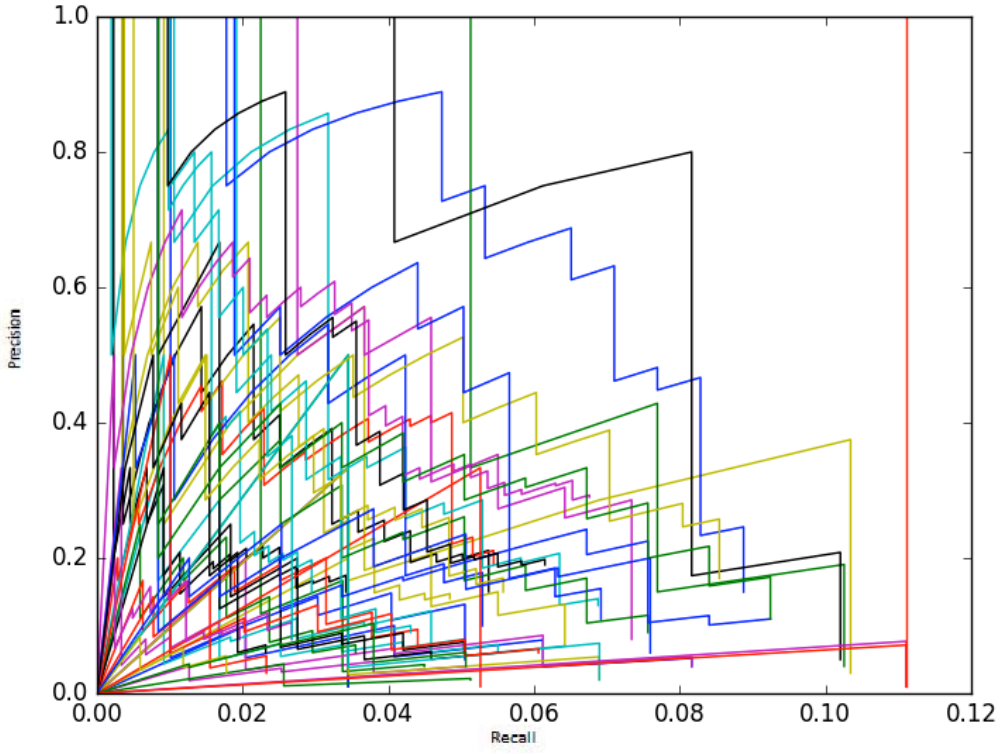
MAP: 0.249722660524584
MRR: 0.476771538543472
P@5,20: 0.6,0.1



- Query Expansion with stopping:

7	CACM-3043	1	0.00358423	R
7	CACM-2902	0.5	0.00358423	NR
7	CACM-2668	0.33333333	0.00358423	NR
7	CACM-3006	0.25	0.00358423	NR
7	CACM-3148	0.4	0.00716846	R
7	CACM-2912	0.5	0.01075269	R
7	CACM-3141	0.57142857	0.01433692	R
7	CACM-1626	0.5	0.01433692	NR
7	CACM-2669	0.44444444	0.01433692	NR
7	CACM-2320	0.5	0.01792115	R
7	CACM-2342	0.54545455	0.02150538	R

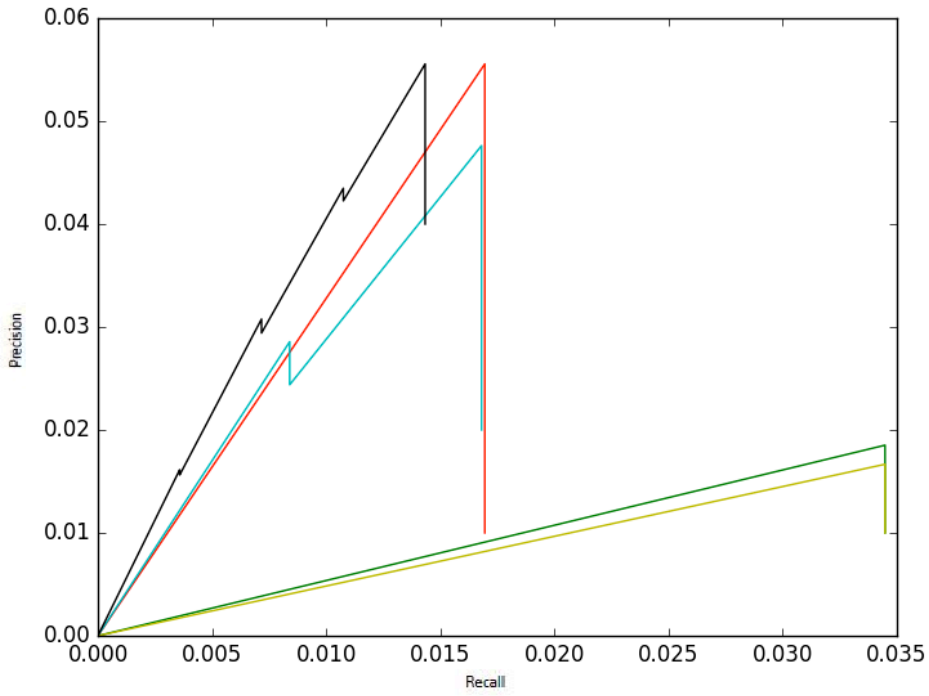
MAP: 0.339907797426867
MRR: 0.595127424436887
P@5,20: 0.4,0.35



- Stemming:

7	CACM-1811	0	0	NR
7	CACM-1457	0	0	NR
7	CACM-3134	0	0	NR
7	CACM-0891	0	0	NR
7	CACM-1699	0	0	NR
7	CACM-2664	0	0	NR
7	CACM-1927	0	0	NR
7	CACM-2278	0	0	NR
7	CACM-2288	0	0	NR
7	CACM-1613	0	0	NR
7	CACM-0634	0	0	NR

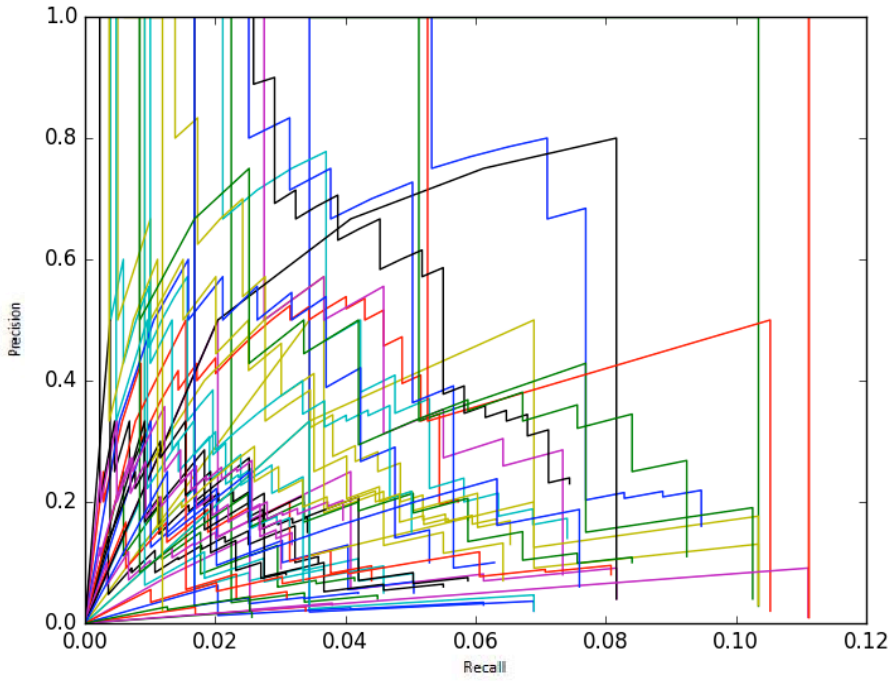
MAP: 0.0236169998141546
MRR: 0.0270882403140467
P@5,20: 0,0



- Stopping

7	CACM-2226	0	0	NR
7	CACM-3166	0	0	NR
7	CACM-0371	0	0	NR
7	CACM-2884	0	0	NR
7	CACM-2950	0	0	NR
7	CACM-1658	0	0	NR
7	CACM-3043	0.14285714	0.00358423	R
7	CACM-2325	0.125	0.00358423	NR
7	CACM-2903	0.11111111	0.00358423	NR
7	CACM-3006	0.1	0.00358423	NR
7	CACM-1551	0.09090909	0.00358423	NR

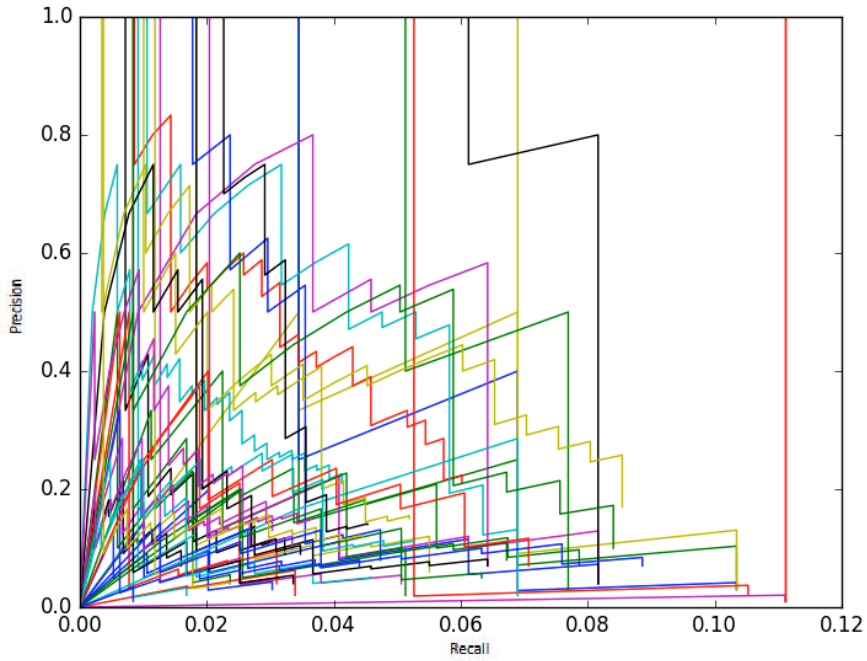
MAP: 0.390508594015654
MRR: 0.617180042164413
P@5,20: 0,0.5



- TF-IDF:

7	CACM-2342	1	0.00358423	R
7	CACM-3043	1	0.00716846	R
7	CACM-1665	0.66666667	0.00716846	NR
7	CACM-2289	0.5	0.00716846	NR
7	CACM-2033	0.4	0.00716846	NR
7	CACM-3077	0.33333333	0.00716846	NR
7	CACM-3128	0.42857143	0.01075269	R
7	CACM-2344	0.375	0.01075269	NR
7	CACM-2903	0.33333333	0.01075269	NR
7	CACM-2931	0.3	0.01075269	NR
7	CACM-1135	0.27272727	0.01075269	NR

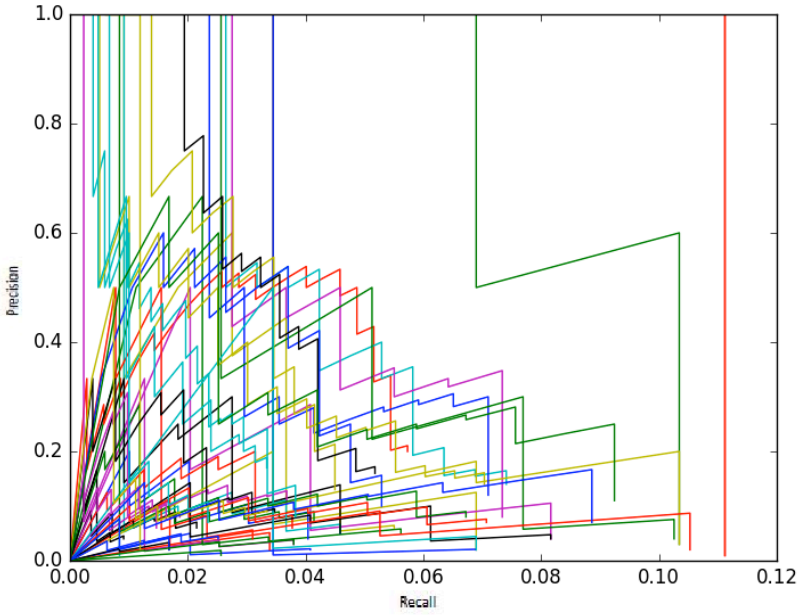
MAP: 0.360316273939787
MRR: 0.608595277191336
P@5,20: 0.4,0.2



- Cosine-Similarity:

7	CACM-2950	0	0	NR
7	CACM-2289	0	0	NR
7	CACM-0409	0	0	NR
7	CACM-3080	0	0	NR
7	CACM-2903	0	0	NR
7	CACM-1834	0	0	NR
7	CACM-3006	0	0	NR
7	CACM-1387	0	0	NR
7	CACM-1366	0	0	NR
7	CACM-1854	0	0	NR
7	CACM-2342	0.09090909	0.00358423	R

MAP: 0.315845356638497
MRR: 0.52074785185218
P@5,20: 0,0.5



Conclusion

- BM25 model takes into account the total number of relevant documents while calculating the score this makes it more efficient based on relevant documents list. Models other than BM25 don't consider the number of relevant documents while computing the score. Hence, they are less efficient compared to BM25 for the given CACM files.

- In Lucene. VSM is extended to Boolean. The document scores are not considered on basis of relevance.

- Query expansion uses Pseudo Relevance feedback, where the top 10 documents are considered relevant and the high frequency terms from these documents are extracted and added to the query which results in retrieving documents that are topically relevant to the query irrespective of user's relevance.

- All other retrieval models use the query and document weights while computing the scores, resulting in retrieving the documents which contain more number of query terms. This makes them less efficient compared to BM25 for CACM files.

- From the above analysis, it is clear that BM25 works better in this case but in general **Cosine Similarity** is the best. There is no explicit definition of relevance in the vector space model, there is an implicit assumption that relevance is related to the similarity of query and document vectors. In other words, documents **"closer"** to the query are more likely to be relevant.

- **Outlook**:
  - To incorporate "suggestions" to the query as the user starts typing a query in the search engine which enables user to decide the query as user is usually unsure of what query to provide.

Bibliography

- https://github.com/rohanag/ADB-relevance-feedback-rocchio
- http://people.cs.georgetown.edu/~nazli/classes/ir-Slides/QueryExpansion-RF-13.pdf
- https://www.youtube.com/watch?v=WTlomM_iDVA&list=PLBv09BD7ez_6kGMKFo9JwQf8Ld13WyHRd&index=23