# CS 6200 INFORMATION RETRIEVAL

## Spring 2017


**Project members**

Lokesh Kodali

Nagasai Vinaykumar Kapalavai

Rahul Kondakrindi

**Instructor**: Nada A. Naji

# 2. Introduction

The project involves designing and building search engines based on various retrieval models, implementing several optimization techniques on these models and evaluating the retrieval models in terms of their retrieval effectiveness.

The various retrieval models implemented were:

- TF-IDF
- BM25 ranking algorithm
- LUCENE
-

The optimization techniques implemented on the retrieval models were:

- Query expansion using pseudo relevance feedback
- Stopping
- Stemming

The retrieval models were evaluated using the following measures:

- MAP (Mean Average Precision)
- MRR (Mean Reciprocal Rank)
- P@K (Precision at rank K = 5, 20)
- Precision and Recall (at each rank)

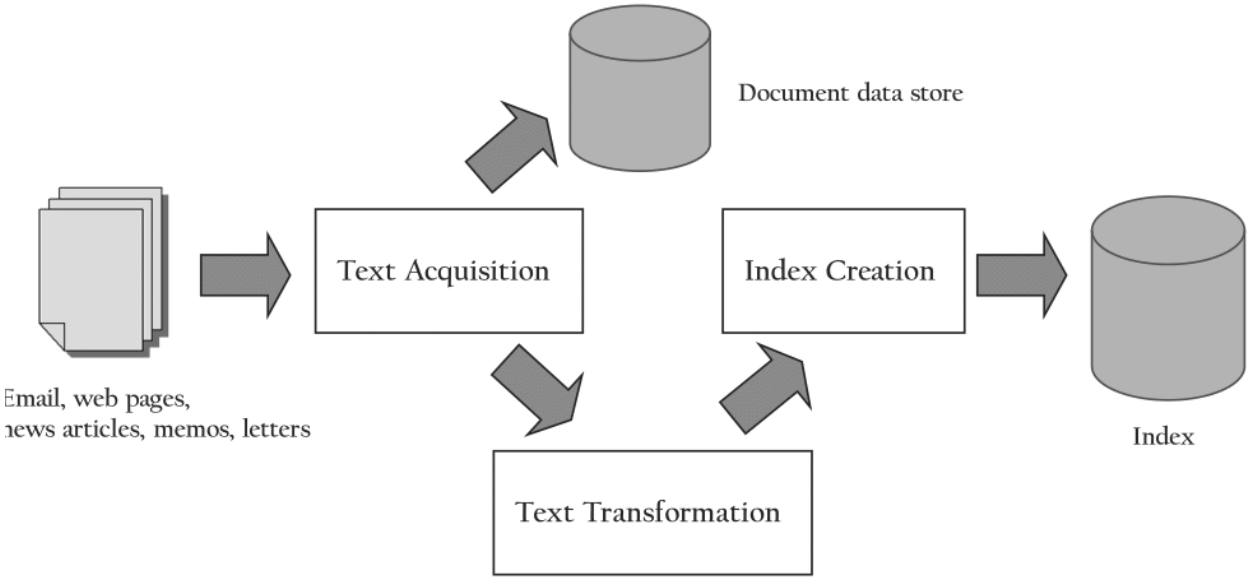
## 2.1. Contribution of members

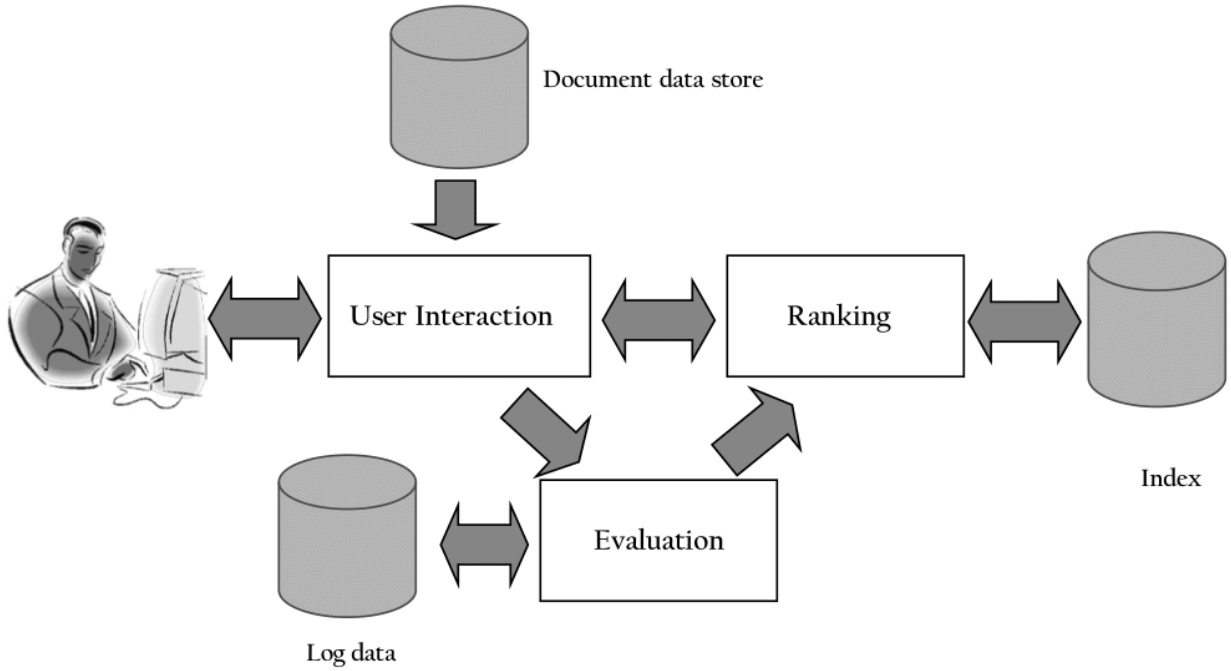
Lokesh Kodali : Task2, Task3(b), t-tests and Documentation

Nagasai Vinaykumar Kapalavai: Task1:BM25, Task1: Lucene, Task3(a), Documentation

Rahul Kondakrindi: Task1:tf-idf, Task2, Snippet Generation and Documentation

# 3. Literature and resources



A need for information is the primary motivation behind a search request submitted by the user. Information retrieval is defined as the process of finding material of an unstructured nature that satisfies an information need from within large collections.

Search Engines are the systems that perform the information retrieval. A search engine consists of two major functions.

- Indexing process
- Query process

### 3.1 Tf-Idf Retrieval Model

The tf-idf weighting determines the importance of a word to a document in the corpus. This weighting consists of two components. The tf weight reflects the importance of the term within the document. The frequency of a term is directly proportional to the tf weight of a term. The idf weight reflects the importance of the term in the corpus. The idf weight of a term is inversely proportional to the number of documents in which it occurs.

tf is calculated as follows:

tf = term frequency in the document / total no.of terms in the document

idf is calculated as follows:

idf = log( total no.of documents in corpus / no.of documents containing the term)

The tf and idf components are calculated and multiplied to score the documents in the corpus.

### 3.2 BM25 Retrieval Model

The BM25 model is a probabilistic model. It is an improvement over the binary independence model in that it includes document and query term weights in the scoring process. The formula for BM25 scoring is given as follows:

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

The score is computed for each document over only the terms that occur in the query. The terms that do not occur in the query do not contribute to the BM25 score.

Here N = total number of documents in the corpus

n = number of documents that contain the particular term

R = total number of relevant documents for a query

r = number of relevant documents for the particular term

qf = term frequency in the query

f = term frequency in the document

k1, k2 = constants that are set empirically
Avdl = average length of the document

K is calculated using the following formula

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

### 3.3 Lucene

Lucene is an open source full featured text search engine library. It is developed in Java. The components and concepts of Lucene scoring are defined by Similarity. Lucene combines Boolean model with Vector Space model. Tf-idf values are believed to produce results of the highest quality and hence the Similarity in Lucene uses tf-idf weighting.

### 3.4 Query Expansion

Pseudo relevance feedback using Rocchio algorithm is chosen as the query expansion technique. In pseudo relevance feedback, an initial ranking is obtained based on an initial query. The top k results are assumed to be relevant to the user and the system reformulates the query by reweighting the terms in the relevant documents with a higher weight and in the process decreasing the weight for the that appear in the non-relevant documents. This modified query is used to produce a new ranking.

Using derivational or inflectional variants might result in addition of variants to the query that do not reflect the information need of the user anymore. When using thesauri and ontologies, synonyms of a query term may be added to the query which might change the entire meaning of the query and produce non-relevant documents to the user. Hence, keeping these issues and the information need of the user in mind, we have chosen pseudo relevance feedback as the query expansion technique.

### 3.5 Third Party Libraries

- BeautifulSoup
- Lucene libraries
  →lucene-queryparser-VERSION.jar
  →lucene-analyzers-common-VERSION.jar
  →lucene-core-VERSION.jar
- Python Scipy

## 4. Implementation and Discussion

### 4.1 Task-1

A new parser was built from scratch to parse the cacm files accordingly. In the implementation of tf-idf retrieval model, a score for each document is computed. This is done by calculating and multiplying the tf weight and idf weight for each term that occurs in both the query and the document and then summing the tf-Idf weights of all such terms to produce a single score for the document. The cut-off value is determined empirically to be set at 0.5.

tf = term frequency in the document / total no.of terms in the document
idf = log (total no.of documents in corpus / no.of documents containing the term)

In the implementation of BM25 retrieval model, the score for each document is computed in a similar fashion to the Tf-Idf model. For each document, if a term is present in both the query and document, then the score is computed and is summated over all such terms to obtain a single BM25 score for the document.

The constants k1 and k2 in the formula are set at k1 = 1.2 and k2 = 200 based on the TREC experiment results. The constant k1 determines how the term frequency component in the BM25 formula changes as fi ( frequency of term in the document) increases. A low value of 1.2 suggests that the effect of fi is nonlinear (i.e) the frequency does not matter after the 4th or 4th occurrence in a document. The function of k2 is similar to k1 and a high value of 200 is set for k2 as query term frequencies are much low and less variable than the document term frequencies.

### 4.2 Task-2

Query expansion is performed by using the concept of pseudo relevance feedback which is implemented by the Rocchio algorithm. While taking the pseudo relevance feedback, the information from the top 12 documents retrieved is considered and only the top 5 most frequent terms were considered for expanding the query to keep the length of expanded query to a minimum.

The Rocchio algorithm is based on the concept of an optimal query which maximizes the difference between the average vector representing the relevant documents and the average vector representing the non-relevant documents. The Rocchio algorithm modifies the initial query to produce a new query according to the following formula

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

Here qj = initial weight of query term j.

Rel = set of identified relevant documents
NonRel = set of identified non relevant documents
$D_{ij}$ = weight of $j^{th}$ term in document i

The initial query term weight is modified by adding a component based on average weight in the relevant documents and subtracting a component based on the average weight in the non-relevant documents.
In this project, we considered $\alpha = 1$, $\beta = 1$ and $\gamma = 0$.

### 4.3 Task3

Stopping in task 3 was performed by removing the stop words from the query as well as the documents by using the common_words.txt file which contains a list of the stop words. For Stemming, the stemmed corpus ( cacm_stem.txt ) and the stemmed queries ( cacm_stem.query ) were used as inputs to the retrieval models instead of the regular corpus and queries. Both the queries and the documents were stopped and stemmed because the preprocessing of the queries should match that of the corpus.

### 4.4 Query-by-Query Analysis

The query analysis is done by taking the runs from the stopped version (task 3(a)) and the stemmed version (task3(b)) of the BM25 model and comparing them. The top 10 documents from the result lists of both the versions are taken. Comparison is done by finding the number of relevant documents retrieved by both the versions in the top 10 documents as well as the positions at which the relevant documents were retrieved. At the end, the number of common documents in the top 10 documents of both the versions are found out.

**(i) Query ID = 13**
**Original query** = code optimization for space efficiency
**Stemmed query** = code optim for space effici

- In stemmed version, 4 relevant documents were retrieved.
- In stopped version also, 4 relevant documents were retrieved.
- There are 7 common documents between the results for both versions.

**(ii) Query ID** = **24**
**Original query** = Applied Stochastic processes
**Stemmed query** = appli stochast process

- In the stemmed version, 3 relevant documents were retrieved.
- In the stopped version, 2 relevant documents were retrieved.
- There are 8 common documents between the results for both versions.

**(iii) Query ID = 23**
**Original query =** Distributed computing structures and algorithms
**Stemmed query =** distribut comput structur and algorithm

- In the stemmed version, 3 relevant documents were retrieved.
- In the stopped version, 4 relevant documents were retrieved.
- There are 4 common documents between the results for both versions.

### 4.4 t-Test

For calculating t-values we considered average precision as retrieval effectiveness.

Initially we considered Lucene as base line run and compared it with other six runs.

t-value is calculated using the below formula,

$$t = \frac{\overline{B - A}}{\sigma_{B-A}} . \sqrt{N}$$

p-value is calculated using scipy library. Obtained values are as shown below

| Run | t-value | p-value |
|---|---|---|
| bm25 | 6.45244 | 1.9930 e-08 |
| tfidf | -2.6458 | 0.9945 |
| bm25_with_query_expansion | 5.3428 | 1.0744 e-06 |
| tfidf_with_query_expansion | -3.2589 | 0.9989 |
| bm25_stopping | 6.8813 | 4.1910 e-09 |
| tfidf_stopping | -1.2874 | 0.8981 |

We considered 0.05 as a significance value. From the values above for all the runs with bm25 p-value are less than significant value 0.05.

$$P < 0.05$$

So, we can reject the null hypothesis (no difference). That is all the runs with bm25 is more effective than Lucene.

Similarly, as the p-values for all the runs with tfidf is greater than 0.05, our assumption of null hypothesis is true.

### 4.5 Snippet Generation

The snippet generation is an effective way for user to judge if the document satisfies his/her information need.

We implemented following steps to obtain snippet for the results generated by the effective model.

- Initially the user is prompted for the query no.
- Then the query is given to the retrieval model to obtain documents, from that we consider top 10 documents for snippet generation.
- Then query and document is tokenized to consider the lines based on the query terms.
- Then we compute a scoring function which will give a list of the matching terms with the given query. Finally, the matched terms in the document lines are highlighted to represent its importance.

## 5. Results

| | MAP | MRR |
|---|---|---|
| **BM25** | 0.578098827972 | 0.841079059829 |
| **TF-IDF** | 0.326359083749 | 0.568511745679 |
| **Lucene** | 0.413729205069 | 0.683367885593 |
| **BM25 with Query Expansion** | 0.573946659671 | 0.846153846154 |
| **TF-IDF with Query Expansion** | 0.284633555826 | 0.491913329814 |
| **BM25 with Stopping** | 0.573973407647 | 0.854700854701 |
| **TF-IDF with Stopping** | 0.379649482698 | 0.626883741259 |

Summary of Results

The results for the seven retrieval models are mentioned below:

**TASK1_BM25:**

task1_bm25_map.txt

task1_bm25_mrr.txt

task1_bm25_p_20.txt

task1_bm25_p_5.txt

task1_bm25_precision_recall.csv

**TASK1_TF-IDF:**

task1_tfidf_map.txt

task1_tfidf_mrr.txt

task1_tfidf_p_20.txt

task1_tfidf_p_5.txt

task1_tfidf_precision_recall.csv

**TASK1_Lucene:**

task1_lucene_map.txt

task1_lucene_mrr.txt

task1_lucene_p_20.txt

task1_lucene_p_5.txt

task1_lucene_precision_recall.csv

**TASK2_BM25 with Query Expansion:**

task2_bm25_map.txt

task2_bm25_mrr.txt

task2_bm25_p_20.txt

task2_bm25_p_5.txt

task2_bm25_precision_recall.csv

**TASK2_TF-IDF with Query Expansion:**

task2_tfidf_map.txt

task2_tfidf_mrr.txt

task2_tfidf_p_20.txt

task2_tfidf_p_5.txt

task2_tfidf_precision_recall.csv

**TASK3_BM25 with Stopping:**

task3a_bm25_map.txt

task3a_bm25_mrr.txt

task3a_bm25_p_20.txt

task3a_bm25_p_5.txt

task3a_bm25_precision_recall.csv

**TASK3_TF-IDF with Stopping:**

task3a_tfidf_map.txt

task3a_tfidf_mrr.txt

task3a_tfidf_p_20.txt

task3a_tfidf_p_5.txt

task3a_tfidf_precision_recall.csv

## 6. Conclusion and Outlook

The BM25 retrieval model has highest MAP and MRR values followed by Lucene and TF-IDF models. Query expansion and Stopping seem to have a negligible effect on the BM25 model where as they have considerable effect on the TF-IDF model. Query expansion decreases the MAP value of TF-IDF model while Stopping increases the MAP value of the model. TF-IDF with query expansion is the least effective retrieval model. BM25 has the highest retrieval effectiveness as it considers relevance during the scoring of documents.

The effectiveness of BM25 can be increased further by fine tuning the values of the constants k1 and k2 , considering other document quality features like page rank and also by considering other features that incorporate the user relevance.

## 7. Bibliography


1. Yang, Christopher C. "Search engines information retrieval in practice." (2010): 430-430.
2. https://en.wikipedia.org/wiki/Rocchio_algorithm
3. https://nlp.stanford.edu/IR-book/
4. https://lucene.apache.org/
5. https://nlp.stanford.edu/IR-book/html/htmledition/results-snippets-1.html