CS6200-Information Retrieval

(Fall 2016)

Members

Ashok Koduru

Sravanthi Kethireddy

Frenia Pinto

Instructor

Professor Nada Naji

Introduction

The project demonstrates designing and building information retrieval systems, evaluating and comparing their performance levels in terms of retrieval effectiveness. The algorithms that are implemented for retrieval are:

1) Tf-idf
2) Cosine similarity
3) Lucene
4) BM-25 algorithm

The project also includes implementation of the query expansion technique using Pseudo Relevance feedback. Using stopping and stemming on corpus, two other runs are produced.

The runs produced by the retrieval models are evaluated using:

1) MAP
2) MRR
3) P@K, K = 5 and 20
4) Precision and Recall

Contribution of the team-members:

Ashok:

- Performed BM25, Cosine Similarity and Tf-idf
- Snippet Generation

Sravanthi:

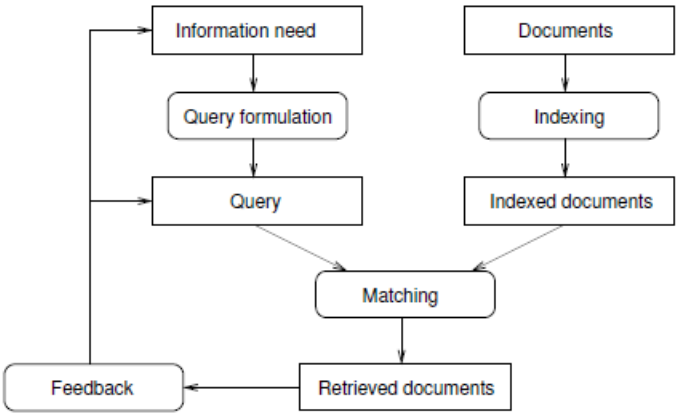
- Documentation
- Stopping and stemming

Frenia:

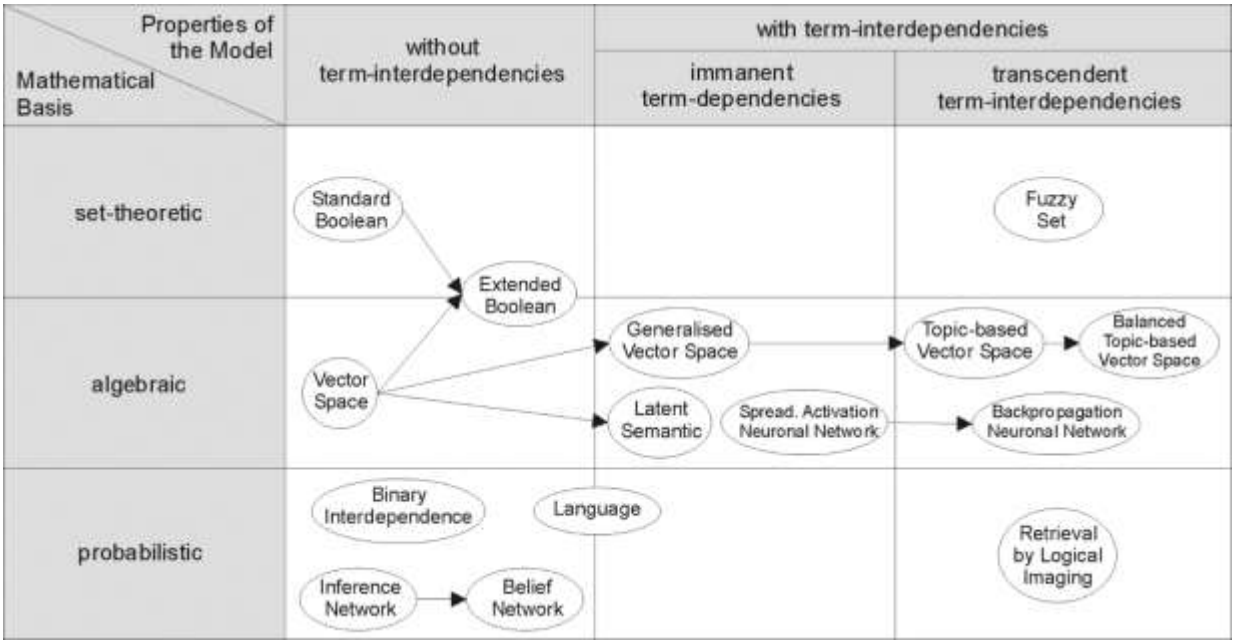
- Query Expansion
- Documentation
- Lucene

Literature and Resources

The purpose of Information Retrieval(IR) is to provide the documents which satisfy the users information need. The process of IR can be depicted in the following diagram.



There are a wide range of retrieval models which are classified as follows:



Tf-idf retrieval model:

The tf-idf weight is the statistical weight measure used to determine importance of the term in a corpus. The term importance is proportional to the term frequency in a document but inversely proportional to frequency of term in corpus.

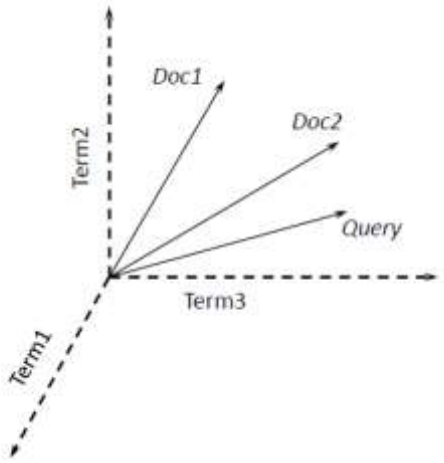
Tf = (occurrence of a term in a document) / (number of terms in the document)

Idf = $\log_e$(Total number of documents / number of documents with the term t in it)

We multiplied these factors to rank the documents according to the given query.

Cosine similarity:

A cosine similarity is a measure of the direction-length resemblance between vectors. In order to compute cosine similarity two document vectors are needed where each unique term in the index is a vector and the value at that index is the measure of term importance in that document. This representation of set of documents as vectors is called vector space model.



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

Vector representation of documents and query

To retrieve the ranked documents, we utilized the vector space similarity model(VSM). The VSM score of a document d for query q is the cosine similarity of weighted query vectors V(q) and V(d).

Cosine-similarity (q, d) = V(q). V(d) / |V(q)| |V(d)|

V(q). V(d) is the dot product of the weighted vectors

|V(q)| and |V(d)| are the Euclidean norms

Lucene:

Lucene is an open source information retrieval software library which is developed in java. Lucene scoring is done using the TfIdf similarity model, we implemented the same and ranked the documents accordingly.

BM25 retrieval model:

We utilized the following formula available for BM25 to compute the ranks of the documents.

$\Sigma \log ((r + 0.5)\ (N - n - R + r + 0.5))/\ ((n - r + 0.5)\ (R - r + 0.5)) * ((k_3 + 1)\ q)/\ ((k_3 + q)) * ((k_1 + 1)\ f)/\ ((K + f))$

Where:
r = number of relevant documents indexed by the term
R = total number of relevant documents
N = number of documents in the collection
n = number of documents in the collection indexed by the term
k_1, k_3 are constants
q = term frequency in the query
f = term frequency in the document
K = k_1((1-b) +b*(document length/ average document length))

For implementation of the query expansion, 'Pseudo Relevance Feedback' approach is being used. The expansion terms generated by pseudo-relevance feedback will depend on the whole query, since they are extracted from documents ranked highly for that query, but the quality of the expansion will be determined by how many of the top-ranked documents in the initial ranking are in fact relevant. The derivational/inflectional variants, thesauri, ontologies are used to generate language-specific terms.

We have used this approach because derivational/inflectional variants are used to add the variants (parts of speech) terms to the query which may change the entire meaning of the query terms entered by the user. The thesauri and ontology adds synonyms to the query terms. This approach may not allow the user to find the exact document he is looking for. Also, pseudo relevance feedback is most effective and widely used.

Query expansion with stopping is done by eliminating the stop words from the query and the corpus which is followed by implementation of Rocchio's pseudo relevance feedback algorithm

Implementation and Discussion

Task 1:

To compute the Tf-idf for each query term present in the document, we have calculated the values as follows:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document)

IDF(t) = log (Total number of documents / Number of documents with term t in it)

The cutoff value which applied is 0.5. Generally a high weight in tf-idf is obtained by a high term frequency in a particular document and low document frequency in the whole corpus since it is an inverse relationship.

The ratio inside the logarithm will be always greater than 1. Because no of documents a term appears will be less than or equal to total number of documents. As a term appears in more documents the ratio will be closer to one. which means the log of that value will move closer to zero. So the lower the weightage more frequent is the term

If the value is much closer to zero the retrieval model does not give proper documents since most of the documents(which makes the value inside the log closer to 1) contains that term and the querying will not be efficient.

To compute the Cosine Similarity to rank the documents, we have normalized the term frequency values in each document by dividing it by the length of the document.

$$tf_{ik} = \frac{f_{ik}}{\sum_{j=1}^{t} f_{ij}}$$

The weighting factor in Cosine Similarity is the tf.idf for each term in each document.

For the BM25 algorithm, we have assumed k1 = 1.2 since in TREC experiments, a typical value for k1 is 1.2, which causes the effect of term frequency to be very non-linear and k2=200 since performance is less sensitive to k2 than it is to k1.

Task 2:

For the query expansion task, we have implemented Rocchio algorithm along with query expansion. This algorithm produces a new query by a modifying the query term weights by adding a component based on the average weight in the relevant documents and excluding the previous query terms. For the implementation of pseudo relevance feedback, we have considered the top 12 documents and the top 5 high frequency words for expanding the query. The value of k and the number of documents are assumed based on the experimental approach, we ran the retrieval models with random values of k and number of documents and the values 12 and 5 proved to produce efficient results

Task 3:

Since, the query has to be processed in a similar way as the corpus, for the stopping task, we have performed stopping on the corpus as well as the query. The same process is applicable for stemming as well.

Query-by-query analysis:

Query 1: portabl oper system

The top 5 results for this query for stemming are: CACM-3127, CACM-2319, CACM-1680, CACM-2379, CACM-3068 while stopping (Query = portable operating system) produces the following top 5 results: CACM-2319, CACM-2379, CACM-1591, CACM-1749, CACM-2629

Two of the documents retrieved are same for both runs. While going through the other documents for both runs, all the documents retrieved are related to the query.

1591- occurrence of more "operating" and "system"

1749- more “systems”

2629,2379- “operating system” together occurred more

Even in case of stemming, the top 5 results do not deviate much away from the query terms.

Query 5: appli stochast process

Stemming results are: CACM-1696, CACM-2080, CACM-2727, CACM-3043, CACM-2999. Stopping produces the following top 5 results (Query = applied stochastic processes): CACM-2342, CACM-3043, CACM-1696, CACM-0268 and CACM-2376

2342,3043 - occurrence of "process"

1696 - occurrence of “stochastic”

0268 - only one “stochastic”

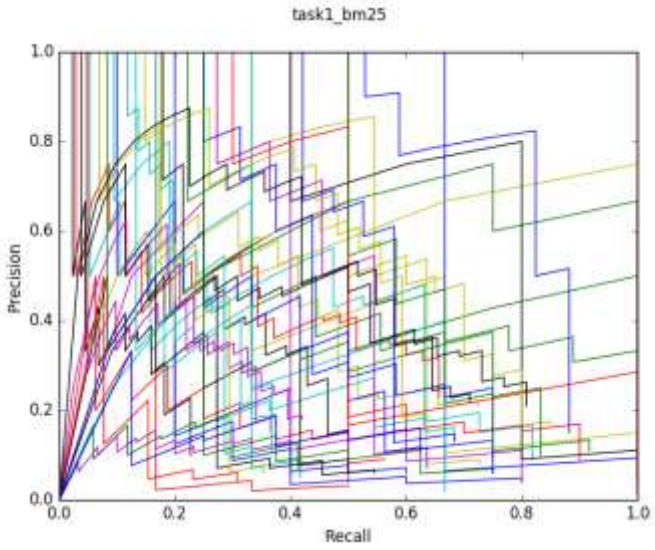
2376 - occurrence of “process”

Except for CACM-1696 which is common for both the runs, rest all documents that are retrieved for stemming are completely irrelevant to the query. Also, the original query term is ‘applied’ but the stemmed version of this term is ‘appli’, hence it also considers documents with ‘application’ which is stemmed to ‘appli’ in the corpus.

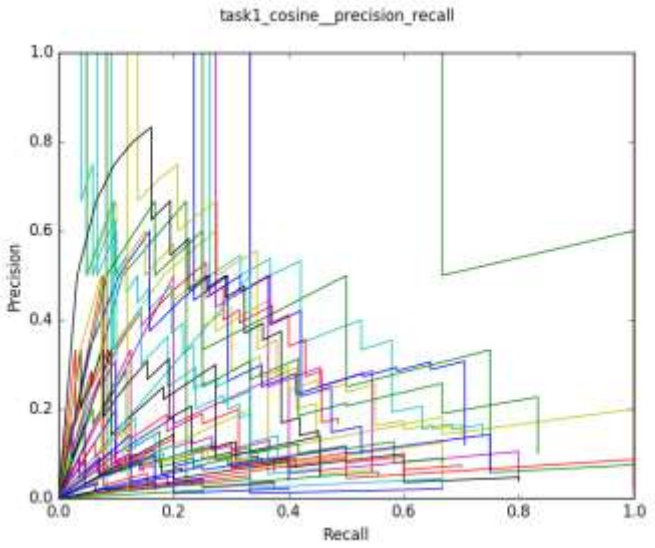
Results

Retrieval Model	MAP	MRR
BM25	0.573541161786	0.850961538462
Cosine Similarity	0.309177359049	0.483167592773
Tf-idf	0.31525653275	0.544080756808
Cosine Similarity using Query Expansion	0.256135996535	0.471745959884
Cosine Similarity using Stopping	0.381993742348	0.599559023129
Cosine Similarity using Stemming	0.0236169998142	0.027088240314
Lucene	0.39072537126	0.65372171798

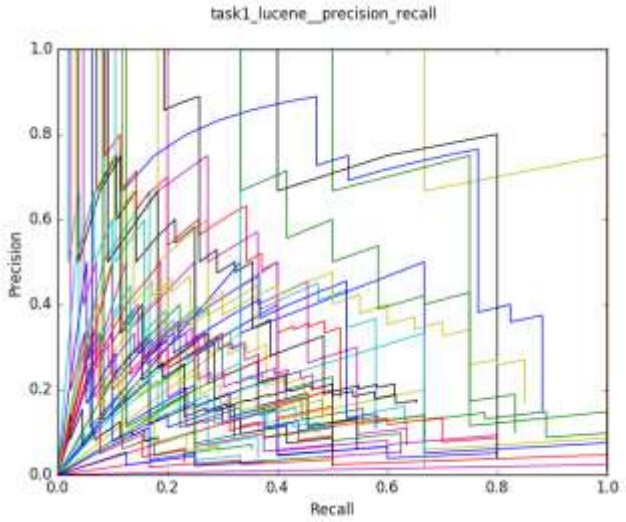
The graph plotted between the precision and recall for the 7 runs are as following:



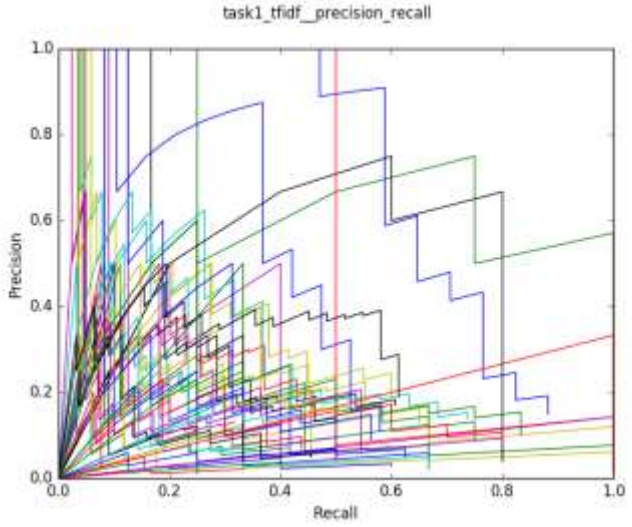
task1_bm25__precision_recall.xlsx



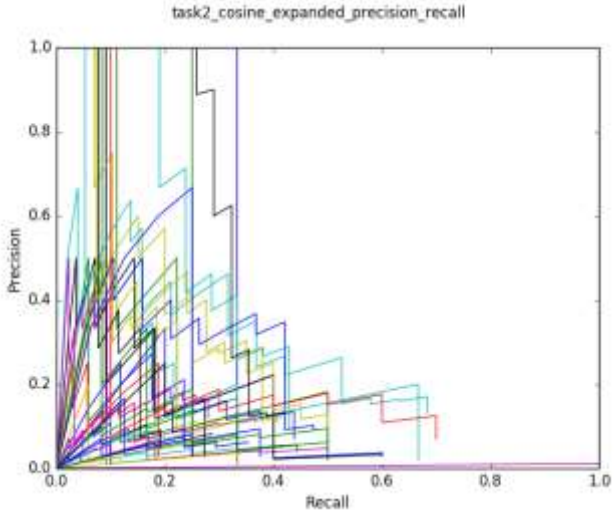
task1_cosine__precision_recall.xlsx



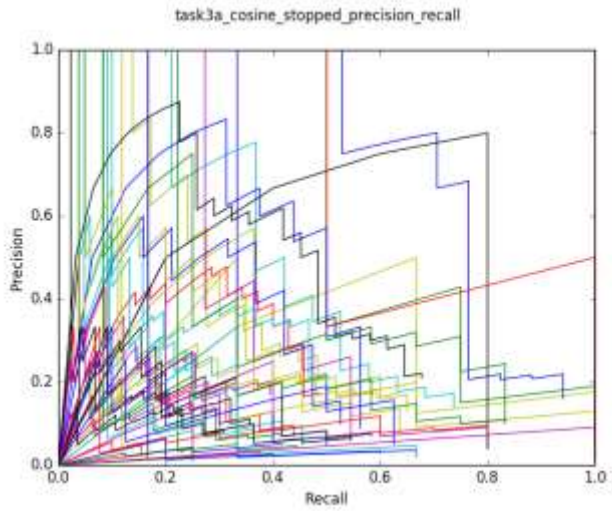
task1_lucene__precision_recall.xlsx



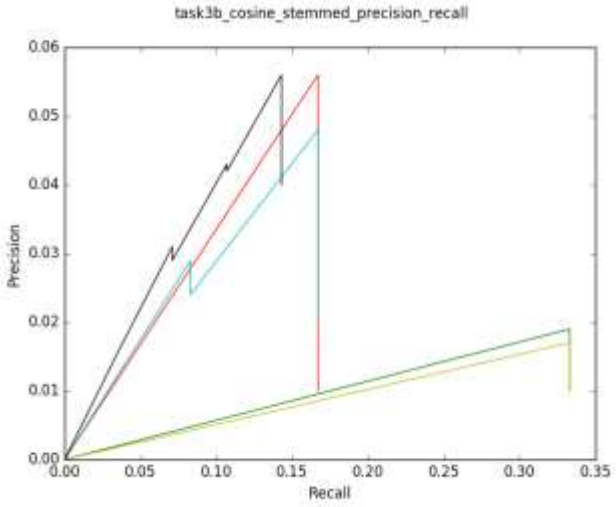
task1_tfidf__precision_recall.xlsx



task2_cosine_expanded_precision_recall.xls



task3a_cosine_stopped_precision_recall.xlsx



task3b_cosine_stemmed_precision_recall.xls

The tables for all the runs are attached below:

tashphase2_cosine_stopped_expanded.xlsx

task1_bm25_.xlsx

task1_cosine_.xlsx

task1_lucene_.xlsx

task1_tfidf_.xlsx

task2_cosine_expanded.xlsx

task3a_cosine_stopped.xlsx

task3b_cosine_stemmed.xlsx

Snippet generation:

In order to generate a snippet according to the query, we used an algorithm whose flow is as follows:

- From the given query, retrieve the top 10 documents using a retrieval model.
- Now, we have a query and the top 10 relevant documents
- The query and documents are tokenized

- Now the document is split into lines based on the query term.
- Next step is to score. Score is the ratio of number of matching keywords in the fragment to the number of occurrences of the keyword in the document. The keyword being the token in the query.
- Using this score the fragmented lines are formatted and displayed.

Conclusions and Outlook

After performing sufficient tests on the given four retrieval models, we deduced, it is the BM25 model which is effective in terms of ranking the documents according to the query. This is because BM25 incorporates relevant data information during the retrieval process. However, if no relevant data is available, the working of this model would be similar to any weighting model. All this is evident from the graphs above that demonstrate high precision values for majority of the queries compared to other retrieval models.

If the relevant document data set information is not available, cosine similarity model with stopping produces good results. This is because it produces results based on the topical relevance of the query terms.

Bibliography

http://wwwhome.cs.utwente.nl/~hiemstra/papers/IRModelsTutorial-draft.pdf

https://en.wikipedia.org/wiki/Information_retrieval#Model_types

https://lucene.apache.org/core/4_6_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html

Search Engines Information Retrieval in Practice by W. Bruce Croft Donald Metzler Trevor Strohman

http://www-nlp.stanford.edu/IR-book/

http://www.ccs.neu.edu/course/cs6200f13/cs6200-f13-7.pdf