**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:

Sravanthi:

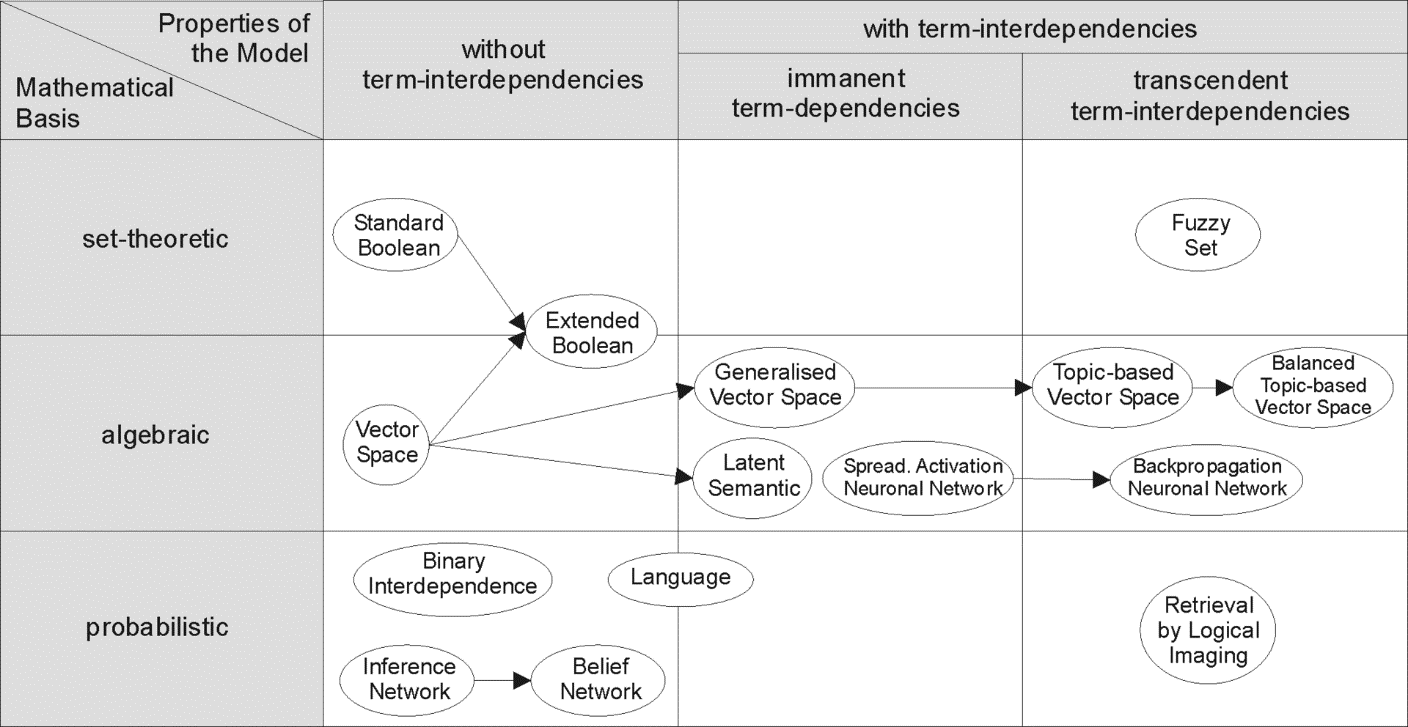
Frenia:

**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 = loge(Total number of documents / number of documents with the term t in it)

We multiplied these terms 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.

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.

Σ 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

k1, k3 are constants

q = term frequency in the query

f = term frequency in the document

K = k1((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.

**Implementation and Discussion**

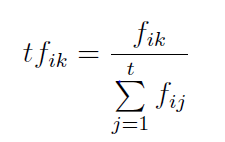
**Task 1:**

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)

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.



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 15 documents and the top 25 high frequency words for expanding the query.

**Task 3:**

Since, the query has to be processed same 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:

**Results**

|  |  |  |
| --- | --- | --- |
| **Retrieval Model** | **MAP** | **MAR** |
| BM25 | 0.558 | 0.832 |
| Cosine Similarity | 0.299 | 0.436 |
| Tf-idf | 0.305 | 0.532 |
| Cosine Similarity using Query Expansion | 0.299 | 0.498 |
| Cosine Similarity using Stopping | 0.378 | 0.580 |
| Cosine Similarity using Stemming | 0.003 | 0.002 |
| Lucene |  |  |

**Conclusions and Outlook**

**Bibliography**

1. <http://wwwhome.cs.utwente.nl/~hiemstra/papers/IRModelsTutorial-draft.pdf>
2. <https://en.wikipedia.org/wiki/Information_retrieval#Model_types>
3. https://lucene.apache.org/core/4\_6\_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html