**Project Members:** Fibin Francis Assissi

Jatin Taneja

Sugandha Kher

Work division:

**Literature and Resources:**

Phase1: Indexing and Retrieval:

Task1:

For our retrieval models, we reused our Cosine Similarity retrieval model and Lucene.

We implemented BM25 and tf-idf concept this time.

BM25 is based on binary independence model.

For BM25 the formula we used:



Where k1, k2 and K are parameters whose values are set empirically.

Typically value for k1 is 1.2 and k2 ranges from 0 to 1000, in our retrieval model we took k1 as 1.2 and k2 as 500.

Reason for it being, for k1 the document score list was……..

Second model we implemented is tf-idf, scoring function we used,

1. Find the term frequency(tf) of each term in the document.
2. Inverse document frequency:

Idf(term) = 1.0 + log(float (len (allDocuments)) / numDocuments with this Term)

1. Weight of a term = tf\*idf
2. In this step we are deriving vectors for documents. The vector for a document is obtained by taking the product of tf\*idf for each unique term in the document. The weights are obtained for all the documents.
3. ???? The same process is repeated for query. For query, the normalised term frequency is the frequency of that term in the query divided by the total number of terms in the query. And the inverse document frequency is 1.

For Cosine Similarity

Cosine Similarity (Query, Document1) = Dot product(Query, Document1) / ||Query|| \* ||Document1||

Here we are calculating the score for each document based on the query.

Dot product (Query, Document1) is the sum of product of weight of a term in the document and its weight in the query for all the terms.

||Document1|| is the magnitude of document vector

||Query|| is the magnitude of query vector

Typically, the tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

TF: Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:   
  
TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

IDF: Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:   
  
IDF(t) = log(Total number of documents / Number of documents with term t in it).

**SCORING FUNCTION USED BY LUCENE**

In case of Lucene, we are inputting the corpus obtained from the HW3. The parsing and scoring is done using the algorithms already defined in Lucene. For scoring, Lucene combines Boolean model (BM) of Information Retrieval with Vector Space Model (VSM) of Information Retrieval.

Cosine Similarity (q, d) = V(q). V(d) / |V(q)| |V(d)|

Where V(q) · V(d) is the [dot product](http://en.wikipedia.org/wiki/Dot_product) of the weighted vectors, and |V(q)| and |V(d)| are their [Euclidean norms](http://en.wikipedia.org/wiki/Euclidean_norm#Euclidean_norm).

The weights of term in the vector is calculated as follows:

tf (t in d) correlates to the term's frequency, defined as the number of times term t appears in the currently scored document d. Documents that have more occurrences of a given term receive a higher score. Note that tf (t in q) is assumed to be 1 and therefore it does not appear in this equation, however if a query contains twice the same term, there will be two term-queries with that same term and hence the computation would still be correct (although not very efficient). The default computation for tf (t in d) in [DefaultSimilarity](https://lucene.apache.org/core/3_6_0/api/core/org/apache/lucene/search/DefaultSimilarity.html#tf(float)) is:   
  tf (t in d) = frequency 1/2

idf(t) stands for Inverse Document Frequency. This value correlates to the inverse of docFreq (the number of documents in which the term t appears). This means rarer terms give higher contribution to the total score. idf(t) appears for t in both the query and the document, hence it is squared in the equation. The default computation for idf(t) in [DefaultSimilarity](https://lucene.apache.org/core/3_6_0/api/core/org/apache/lucene/search/DefaultSimilarity.html#idf(int, int)) is:

  idf(t) = 1 + log (numDocs / docFreq+1)

3 queryNorm(q) is a normalizing factor used to make scores between queries comparable. This factor does not affect document ranking (since all ranked documents are multiplied by the same factor), but rather just attempts to make scores from different queries (or even different indexes) comparable. This is a search time factor computed by the Similarity in effect at search time. The default computation in [DefaultSimilarity](https://lucene.apache.org/core/3_6_0/api/core/org/apache/lucene/search/DefaultSimilarity.html#queryNorm(float)) produces a [Euclidean norm](http://en.wikipedia.org/wiki/Euclidean_norm#Euclidean_norm): 

queryNorm(g) = queryNorm (sumOfSquaredWeights) = 1 / sumOfSquaredWeights

Task2:

Expanded query: what articles exist which deal with tss, time sharing system   an operating system for ibm computers performance developed design tutorial simulation display search retrieval operations interarrival data distribution communication network time-sharing.

For query expansion, we used the technique, pseudo relevance.

Taking the run from our search engine with BM25 as a retrieval model, we performed query expansion using pseudo-relevance feedback.

The approach used is, simply taking the top 10 scored documents from the BM25 score run. Take out the most frequent words from the 10 documents. From the given common\_words.txt document, we took the words and eliminate these words from our most frequent words. The remaining words are used in expanding our query.

Task3:

In task3, we took the BM25 as base search engine. Using the stop words given, common\_words.txt, we removed the common words while indexing the BM25 retrieval model.

Also again indexing is performed on the given indexed version of the corpus, cacm\_stem.txt

Phase2: Evaluation

Using the Cosine Similarity run along with the stopping done, we performed retrieval effectiveness. We performed precision and recall on each query using the formula:





**IMPLEMENTATION:**

In Cosine Similarity measure we tried 2 different methods for implementing the formula, one using site,

<https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/>

Using cosine similarity formula as:

Cosine Similarity (d1, d2) = Dot product (d1, d2) / ||d1|| \* ||d2||

Dot product (d1, d2) = d1[0] \* d2[0] + d1[1] \* … \* d1[n] \* d2[n]

||d1|| = square root (d1[0]2 + d1[1]2 + … + d1[n]2)

||d2|| = square root (d1[0]2 + d1[1]2 + … + d1[n]2)

We used two approaches in Cosine similarity. Firstly, we normalised the denominator of the cosine similarity formula by taking total of all the terms in the documents. In our second approach we took only the terms in the query to normalise the denominator.

Since the second approach was giving better results, we carried forward the approach where we took the query terms to normalize.

For BM25, we tried different values of b ranging from 0 to 1 and k2 which ranges from 0 to 1000. Upon taking k2 as 500 and b as 1, we got the results in which our relevant document 1410 was coming on 7th rank but on taking b as 0.75 and k2 as 100, the relevant document 1410 shifted to 4th rank, which according to Reciprocal Rank which would be incorrect, since the lower the ranked position of a relevant document, the less useful it is for the user, since it is less likely to be examined.

In tf.idf a simple approach was followed wherein, we calculated the term frequency for all the terms per document and then then took out the normalised term frequency by dividing term frequency by total number of terms in particular document.

Then calculate the idf

Idf = log N/ nk where N is the total number of documents in corpus and nk is the number of documents where term occurs.

Calculate the weight using tf\*idf for the documents as well as for the queries.

In Lucene, we used the concepts cited at:

<https://lucene.apache.org/core/2_9_4/api/all/org/apache/lucene/search/Similarity.html>

Lucene combines Boolean model (BM) and Vector Space Model (VSM). In VSM, documents and queries are represented as weighted vectors in a multi-dimensional space, where each distinct index term is a dimension, and weights are tf.idf values.

VSM does not require weights to be tf.idf values but the tf.idf values are believed to produce search results of high quality, and so Lucene uses tf.idf

For Task2, using the run from BM25 we performed query expansion using pseudo-relevance,

We are reformulating the query by adding terms. In our approach, we ran an original query using BM25 retrieval method. Then, related terms are extracted from top 5 documents. We are taking 3 words from each document.

Observation results:

Taking 1 word from each document:

Doc #1572, at 97th position

Doc #1410, at position 4th

Taking 5words from each document:

Doc #1605, at position 37th

Taking 3 words from each document:

Doc #1572, at 19th position

Doc #1410, at position 1,

therefore, Reciprocal rank increased, recall increased

Doc #1605, at position 20th

Similarity score improved efficiently to 26.84 which was 8.17 before taking 5 words from each document.

For expansion, we performed 7 runs, namely, Cosine Similarity, BM25, tf-idf, Lucene, pseudo-relevance, BM25 after stopping, indexing stemmed corpus.

**CONCLUSION AND OUTLOOK:**

**Top 5 Results of Cosine Similarity for Q1:**

Results after taking total words in the document to normalize the denominator in Cosine Similarity formula.

CACM-1591 1 0.200544279326

CACM-1033 2 0.178576068885

CACM-2542 3 0.171556342261

CACM-1519 4 0.166745872741

CACM-1680 5 0.16317130098

Whereas on taking query words to normalize the denominator we get results as:

Final result

CACM-1410 1 0.636555216239

CACM-1046 2 0.572388333005

CACM-1698 3 0.568103542632

CACM-2947 4 0.549628104412

CACM-1506 5 0.547936673554

On the basis of reciprocal rank, CACM-1410 is a relevant document which is coming as the top result in our second approach. Hence, we take this approach to measure cosine similarity.

**Top 5 Results of BM25 for Q1:**

On taking b as 1 and k as 500

1680 1 9.3760522355

2319 2 8.94587362091

2371 3 8.94579413578

1236 4 8.78018991948

1168 5 8.72134862702

Taking b as 0.75 and k2 as 500

2319 1 9.86481989693

1680 2 9.72607004789

1236 3 9.08566397854

1410 4 8.96081225322

2371 5 8.81367981145

Taking b as 0.75 and k2 as 100

2319 1 9.83499435029

1680 2 9.69123838624

1236 3 9.05707365678

1410 4 8.93717960334

2371 5 8.77743879692

Since by taking b as 0.75 and k2 as 100, the document 1410 which is a relevant one is coming at the 4th position, hence the reciprocal rank would increase, so we take the approach with b=0.75 and k2=100.

**Top 5 Results of tf-idf for Q1:**

CACM-2371 1 0.305168830315

CACM-1591 2 0.299880050314

CACM-1033 3 0.291805418555

CACM-1519 4 0.285964768691

CACM-1304 5 0.266288027745

Results of the tf-idf retrieval system takes simple the term frequency and the document frequency and calculated tf\*idf score for documents.

**Results for Pseudo-relevance:**

Taking 10 top documents and 3 most frequent words from each document:

1410 1 28.1957921664

1680 2 24.1344962859

3025 3 23.003652679

1750 4 21.4841937177

2371 5 20.2388846723

Taking 5 top documents and 1 most frequent word from each document:

1410 1 21.5352739427

1680 2 13.7295950175

2371 3 11.6226071628

2319 4 10.9805107712

1236 5 10.5747088717

Taking 5 top documents and 5 most frequent word from each document:

1410 1 32.8569885643

2371 2 27.9724258879

1680 3 20.3444028742

2951 4 16.4704709489

1236 5 16.2623335671

Taking 5 top documents and 3 most frequent words from each document:

1410 1 26.8420581312

1680 2 19.1894345712

2371 3 18.1518815441

1236 4 12.6071621183

2951 5 11.9733665547

As described above the taking 5 top documents and among that taking 3 most frequent words from each document, gives us better results such as relevant document is obtained at position 1 which increases the reciprocal rank. Also recall is increased, so we take 3 words per document for query expansion.

**BIBLIOGRAPHY:**

https://www.tfidf.com

https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/

https://lucene.apache.org/core/2\_9\_4/api/all/org/apache/lucene/search/Similarity.html

Search Engines: Information Retrieval in Practice by Croft, Metzler and Strohman:   <http://ciir.cs.umass.edu/irbook/>

Introduction to Information Retrieval by Manning, Raghavan, and Schütze (2008) <http://www-nlp.stanford.edu/IR-book/>

1. Lucene in Action (2nd Edition). McCandless, Hatcher, and Gospodnetić. (Manning)

Fast Generation of Result Snippets in Web Search by Felix Geller