Northeastern University

College of Computer and Information Science

Information Retrieval CS6200 – Fall 2017

<u>Project-Report</u>

<u>Members:</u>

Divyavijay Sahay
Vaibhav Shekhar Dave
Nicholas Jared Carugati

<u>Batch:</u>
Tuesday-Thursday

<u>Instructor:</u>

Prof. Nada Naji

II. Introduction:

We aimed to make our own information retrieval system to generate relevant output for the queries.

We have implemented three retrieval models:

- BM25
- TF – IDF
- Smoothed Query Likelihood Model.

And utilized Lucene - an open source information retrieval library.

We have also evaluated the performance of the above-mentioned systems under conditions like:

- Given corpus, given query
- Given corpus and refined query.
- Stopped corpus and stopped queries.

Extra Credit – Proximity enabled search

We have implemented a new retrieval model as well as a new index to support proximity enabled search. This retrieval model scores a document using term proximity weight in combination with BM25 score. The occurrence of term pairs maintains the order as per the query. After testing this model with the stopped and non-stopped versions of the given queries and corpus documents. The results suggest that this retrieval model favors the documents which contains high number of query term pairs.

Contributions:

Vaibhav Shekhar Dave

- Implemented Smoothed Query Likelihood Model.
- Implemented TF-IDF
- Integrated snippet generator.

Divyavijay Sahay

- Implemented BM25 Model.
- Managed Extra Credit Section.
- Integrated performance evaluator.

Nicholas Jared Carugati

- Implemented Snippet Generator and managed Phase 2.
- Implemented performance evaluator and managed Phase 3.

Combined Efforts:

Documentation of the Project:

III. Literature and Resources cited:

1) https://lucene.apache.org/

2) https://en.wikipedia.org/wiki/Tf-idf

3) W.B. Croft, D. Metzler, T. Strohman, 2015

4) Rasolofo, Yves, and Jacques Savoy. “Term Proximity Scoring for Keyword-Based Retrieval Systems.” Lecture Notes in Computer Science Advances in Information Retrieval, 2003, pp. 207–218., doi:10.1007/3-540-36618-0_15.

IV. Implementation and discussion:

In [2] there are three recommeded Tf-idf weighing schemes and

Out of which we choose the following:

document term weight(dw): $(1+ \log(f_{t,d}))$

Query term weight(qw): $(1+ \log(N/n))$

```
For(each document)
{
For(each query term)
    {
        Score = $\sum$ (dw * qw)
    }
}
```

$f_{t,d}$, - term frequency of tf(t,d)
N - corpus size
n – Document frequency
as this gave the best results for the given corpus out of the other two.
The parameter used to evaluate all three approaches - Mean Average Precision.

Implementation of Phase1 - Task 2:

Referred – [3]

- Baseline run used for implementing task 2 – TF-IDF
- Number of relevant documents taken into consideration – 30.
- Number of top frequency terms appended in the query – 10
  Ignoring the numbers, stopwords and terms already present in the query.
  Reasons:
  It's a small corpus so its not too much overhead to read top 30 files.
  Aimed at improving the precision for queries where the P@5 was 0.
- Observation:
  Although the Mean Average precision dropped by 0.08 % comparing the TfIdf evaluation on corpus for Task 1 and Task 2 if we take top 100 results into consideration.
  However, showed improvement in Mean Average Precision if we only consider top 5 results. For example, query no. 16, 29 42 had no relevant documents in the top 5 results but after pseudo relevance their P@5 improved.

Discussion of Phase1 - Task 3 - B:

Models used – BM25, TFIDF, LUCENE

1)**Query - portabl oper system**

Top BM25 Result:
DocID – 1591
Terms present and term frequency:
portable - 0 , oper - 6, system – 12
document length – large
Inference – The document clearly talks about an operating system but not a portable one.
Relevance – partially.

Top Lucene and TFIDF Result:
DocID – 3127
Terms present and term frequency:
portable - 4 , oper - 3, system – 5
document length – medium
Inference- The document clearly talks about all aspects of the query.
Relevance – Relevant.

2) Query: Code optim for space effici

Top BM25 and Lucene Result:
DocID – 2748
Terms present and term frequency:
Code – 5, optim – 0, for – 1, space- 1, effici -1.
document length – small

DocID – 2491
Terms present and term frequency:
Code – 7, optim – 1, for – 0, space- 1, effici -0.
document length – small

Inference - One document talks about space efficiency but not about optimization and the other vice versa.
Relevance – Partially relevant

Top TF-IDF Result:
DocID – 2033
Terms present and term frequency:
Code – 5, optim – 0, for – 3, space- 4, effici -2.
document length – large

DocID – 2897
Terms present and term frequency:
Code – 11, optim – 10, for – 4, space- 0, effici -0.
document length – large

Inference : Relevance seems comparable to the previous two models, however TFIDF seems to be biased towards longer documents

3) Query: distribut comput structur and algorithm

Top TFIDF and Lucene Result:
DocID – 2276
Terms present and term frequency:
Distribut-9, comput- 3, structur – 0, and – 8, algorithm - 2
document length – large.

Inference – Document clearly seems to touch upon the aspect asked in the query.
Relevance – Relevant

Top BM25 Result:

DocID – 2430
Terms present and term frequency:
Code – 0, optim – 0, for – 0, space- 0, effici -4.
document length – very small

Inference: Irrelevant

Observation: TFIDF clearly favors documents containing terms with high term frequency and large sized documents. BM25 seems to remove the bias, but cases where the size of the document is too small, and one out of 5 query terms are present in the document it seems to favor that document over a large sized relevant document. Lucene results are seemed to be a mixture of TFIDF and BM25's properties.

<u>Implementation of Phase 2:</u>

Referred – [3]

The snippet handles two types of phrases. It handles sentences and pseudo-sentences. Sentences are phrases found in the corpus which start with a capital letter and end with a punctuation mark given no case folding or punctuation stripping in corpus. Pseudo-sentences are phrases captured by the snippet which either have A window starting at the capital letter or a relevant term and ends at the max threshold phrase window (20), which is the standard for an average complete sentence. If a sentence is too small (7 words or less). Then it will expand the phrase by 5 windows and then recalculate the phrase with respect to the new window sizes.

The main algorithm retains the count of relevant for a given window and then highlights them accordingly when encountered. When a window is complete, the phrase is then formulated, and uses Luhn's significance factor for ranking using the equation as shown:

Rank = t^2/window – t = number of relevant windows

When there are no more words to search within the contents of the corpus, the ranking structure is sorted by the best rank. Then a snippet payload is used to compile the snippet by best significance factor.

Discussion of Phase 2:

Syntax of each snippet: “Quick <hl>brown</hl> fox jumps over the lazy dogs.” Q = “ brown”

Snippet sentence/pseudo-sentence limit: 5
Queries generated: 64
Max/Min Window Size Threshold: 20/7
Rank lists used: BM25, Proximity Score, Lucene, TF-IDF

Snippet example:

Query: “setl very high level languages”

Top 2 Query results using proximity score:

CACM-1923	175.28015
CACM-2699	126.08873

Snippet Example:

```
CACM-1923
-----------------------------------------------------------------
<hl>high</hl> <hl>level</hl> <hl>languages</hl> basic
Input output support facilities shown technique provide
potentially inexpensive methods programs communicate deeply
embedded facilities command language
```

Inference: The algorithm captures words such as “high”, “level”, and “languages” but more terms are highlighted than other this is due to the ranking order.

Extra Credit

Implementation and discussion:

The retrieval model presented in [4] was developed for a retrieval system supporting distributed indexes. The scoring method suggested by them was designed in consideration of large index and corpus size. After testing their scoring function against the index and corpus used in this project. We noticed the scores to be very small values and was also not giving the desired results for few queries and evaluation of that system was comparatively low. In response to these observations we modified our system to support BM25 model as represented in [3] and we modified the values of b (document length) to 0.75 and k2 (constant for query term weight) 100.

The scoring function:

Score(d,q) = Score$_{BM25}$ (d,q) + TPRSV(d,q)

Where,

Score$_{BM25}$ (d,q) = BM25 score for the document and the given query.

TPRSV(d,q) = TermProximity weight for the document and the given query.

Term Proximity weight:

$$TPRSV(d,q) = \sum_{(t_i,t_j)\in S} w_d(t_i,t_j)\cdot \min(qw_i, qw_j)$$

Where, qw$_i$ represents weight of the qwery term t$_i$ as calculated in BM25 of [3].

$$w_d(t_i,t_j) = (k_1+1)\cdot \frac{\sum_{occ(t_i,t_j)} tpi(t_i,t_j)}{K + \sum_{occ(t_i,t_j)} tpi(t_i,t_j)}$$

Where k1 and K are the values present in BM25 of [3].

$$tpi(t_i,t_j) = \frac{1.0}{d(t_i,t_j)^2}$$

Where, d(t$_i$, t$_j$) is the distance expressed in number of words between query term t$_i$ and t$_j$.

Score calculation steps:

1. Rank the documents for the given query using BM25 retrieval model.
2. For the top 100 ranked documents calculate and multiply a term proximity weight to the document's current score.
3. Sort all the documents again with the new scores.
4. The sorted documents represent the result.

Implementation description:

Changes in Inverted Index:

Inverted index was modified to include the position of the terms in a document.

Sample representation
Term { [DocID, (term_frequency, no_of_positions, (position1, position2,…))] }

Assumptions:

1. if a document contains sentences having at least two query terms within them in the order as per the query, the probability that this document will be relevant must be greater.
2. Shorter the distance between term pairs more relevant is the document to the topic of the query.
3. Queries are short and only contains relevant terms. Stopwords are removed.

Query-by-Query analysis:

With stopping:

Query1: setl high level languages

Top 2 result and score:

BM25:
1. CACM-2699 - 21.268663
2. CACM-1923 - 15.97896

Proximity model:
1. CACM-1923 - 175.70007
2. CACM-2699 - 126.006805

Analysis:
1. Both the documents have 2 pairs of query terms "high level"
2. Size of 1923 is comparatively small so it is favored because in proximity score we are again including the document length normalization in the formula of $w_d(t_i,t_j)$.

Query2: id find articles describing singular decomposition digital image processing applications include finding approximations original image restoring images subject noise article subject andrews patterson outer product expansions digital image processing american mathematical andrews Patterson

Top 1 result and score:

BM25: CACM-1686 - 49.025223
Proximity model: CACM-2065 - 693.1738

Analysis:
1. Though the length of document CACM-2065 is comparatively more it ranks higher in proximity model because it has 4 query term pairs "image processing" whereas CACM-1686, short in length but have only 1 pair.

Without stopping:

Query: find all discussions of optimal implementations of sort algorithms for database management applications

Top 1 result and score:

BM25: CACM-3164 - 14.080444
Proximity model: CACM-2484 - 45.248596

Analysis:

1. The document CACM-2484 was not present in the top 5 results of BM25 model but since it has 2 query terms pair “algorithms for” it ranks first in proximity model.
2. Whereas the top result of BM25, CACM-3164 ranks 4th in proximity model as it contains only 1 pair of query terms “for database”.

Conclusion:

Evaluation results:
without stopping
MAP: 0.39787150468838217
MRR: 0.6257450882450883

With stopping
MAP: 0.38703630933351535
MRR: 0.6048755994408167

We can conclude that this model ranks documents higher having:

1. High number of query term pairs.
2. Short length in case 2 documents of different length having same number of query term pairs.

Results: refer the results.txt present in the Results folder.

Conclusion:

From the evaluation results of the baseline queries it is evident that Smoothened Query model seemed the most inefficient followed by Lucene, BM25 and TFIDF.

Statistically TFIDF seems to work best according to the Mean average precision but by little margin to BM25.

As BM25 is also a Tf * Idf equivalent model both seem to be almost equal in their precision.

TF-IDF seemed to work better as the corpus documents were relatively of same size but if the document sizes vary start varying by a lot its relevance judgement might be affected as its biased towards larger documents

Future Scope: We have already started to implement a basic spell check using Soundex to implement in this model on real world data.

Bibliography:

- W.B. Croft, D. Metzler, T. Strohman, 2015
- https://en.wikipedia.org/wiki/Tf-idf
- https://lucene.apache.org/
- Rasolofo, Yves, and Jacques Savoy. “Term Proximity Scoring for Keyword-Based Retrieval Systems.” Lecture Notes in Computer Science Advances in Information Retrieval, 2003, pp. 207–218., doi:10.1007/3-540-36618-0_15.