News Search Engine - Using Lucene

Deepak Kumar
S20160010022
deepak.k16@iiits.in

Kaustubh Pandey
S20160010041
kaustubh.p16@iiits.in

Prashant Kumar Mahanta
S20160010066
prashantkumar.m16@iiits.in

Wasim Inshaq Khan
S20160010107
wasim.i16@iiits.in

***Abstract*—Searching relevant news based on the query given by the user is one of the fundamental problems of information retrieval. In this project we aim to build a search engine that can search from news feeds the news that are most relevant to the query given by the user. Our search engine will only require one single query from the user-side as input. The query may contain date, title or content information. Our search engine will only be capable of handling English based news content.**

1. Keywords

Information Retrieval, BM25, BM25F, TF-IDF.

2. Introduction

Making a search engine for retrieving news has many factors that one has to address. In this project we will only be focusing on one such factor i.e. relevance based ranking. News articles are always structured. They contain various zones such as Title, Date of Publication, Body of the News etc. Since the each document is going to be structures so it becomes essential to use a ranking algorithm that takes the structures nature of the documents into account. Lucene by default considers documents to be unstructured and performs their scoring accordingly. So, it won't be nice to use lucene's default ranking. In this project we will be using one such algorithm that takes the structred nature of documents into account. The algorithm is a modified version of BM25 named as BM25F. We will be overriding the default ranking used by lucene i.e. BM25 ranking algorithm with BM25F algorithm.

3. Structured IR

For the last forty years, search engines have been dealing with flat documents, that is, without structure. The main consequence of this approach is the fact that terms within a document are considered to have the same relevance (or value), disregarding their role in the document. This assumption implies a relevance model simplification based on bag of words, and, therefore, useful information is lost.

With the growth of the Web and the increase of the markup languages, the structure of the documents has become explicit. As a result, a more complex information source is available that may, in fact, allow us to improve the performance of the information retrieval. The Semantic Web seeks to add meaning to this structure, trying to replace descriptive mark-up by semantic mark-up.

In fact, around the structured documents a new field has born within IR community named structured IR. This new field, tries to adapt the classical ranking functions and models to the new scenario by means of considering the explicit document structure.

From an IR theoretical standpoint, the main issue to solve is the adaptation of the standard ranking functions, like Vector Space Model (VSM) or BM25, to score structured documents.

One of the main assumptions within IR theory is that exists an statistical independence between terms. From this assumption the standard ranking functions are composed by a term weights summation. The same independence assumption between terms has been extrapolated to independence between the different document parts, that is, fields that represent the document structure.

Dataset We have taken 100 files containing news from Kaggle[5]. We are using 100 documents for making the inverted index. The documents are structured. Each document consists of 3 fields i.e. title, date of publishing, content. The priority of the fields is set as : Date of Publishing, Content and Title.

4. Ranking Algorithms

Given a query q and a collection D of documents that match the query, the task is to rank, i.e. sort the documents in D according to some criterion so that the more relevant results appear ahead in the result list and less relevant documents are displayed later on to the user. We have used 2 different ranking algorithms. First one is BM25 which is the default ranking algorithm in lucene and the other algorithm we used is BM25F which is an extension of the BM25 ranking function adapted to score structured documents.

BM25F performs per-field BM25 calculation, but uses the shared document frequency across multiple fields. For example, sometimes you have scenarios where a common term, say "cat" is actually rare in one particular field (say the "title" field). But when you broaden out to other fields, you then realize that cat is particularly common, and should not be scored so highly.

In other words, BM25F basically does

CombinedIDF * (BM25(title)+BM25(description)+...)

5. Methodology

The vector space model normally doesn't take into account the structure of the document and hence gives no special importance to any field. Although it can be modified for structured documents, it doesn't give good results. Our method BM25F is one of the state-of-the-art methods used for scoring of structured documents and it is an extension of the BM25 Algorithm used for unstructured document.It is well known that the probability of relevance of a document increases as the term frequency of a query term increases, and that this increase is non-linear. For these reasons most modern ranking functions use an increasing saturation function to weight document terms that appear in the query. The term saturation function used in case of BM25 is:

$$\frac{tf(t,d)}{tf(t,d)+k_1}$$

where tf(t, d) is the term frequency function of term t in document d, and k1 is a constant that allows us to control the non-linear growing term frequency function. For computing BM25F similarity, we first calculate a normalized term frequency for each field c:

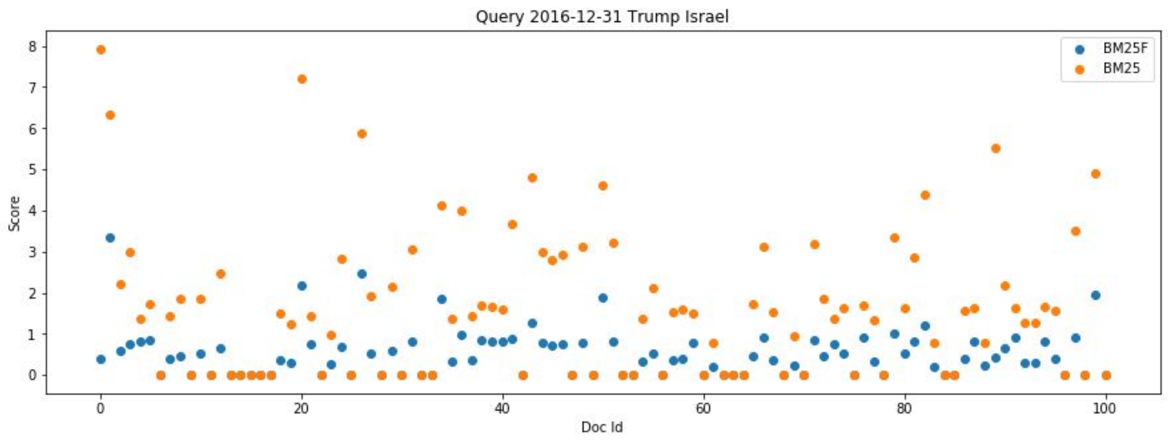
$$tf_c(t,d) = \frac{occurs_c(t,d)}{1+b_c((\frac{l_{d,c}}{l_c}-1))}$$

where $occurs_c(t,d)$ are the occurrences of the term t in the field c of the document d, ld,c is the length of the field, and lc is the average field length for that field. b_c is a field-dependant parameter similar to the b parameter in BM25. b controls to what degree document length normalizes tf values. These term frequencies can be combined linearly using the boost factor w_c:

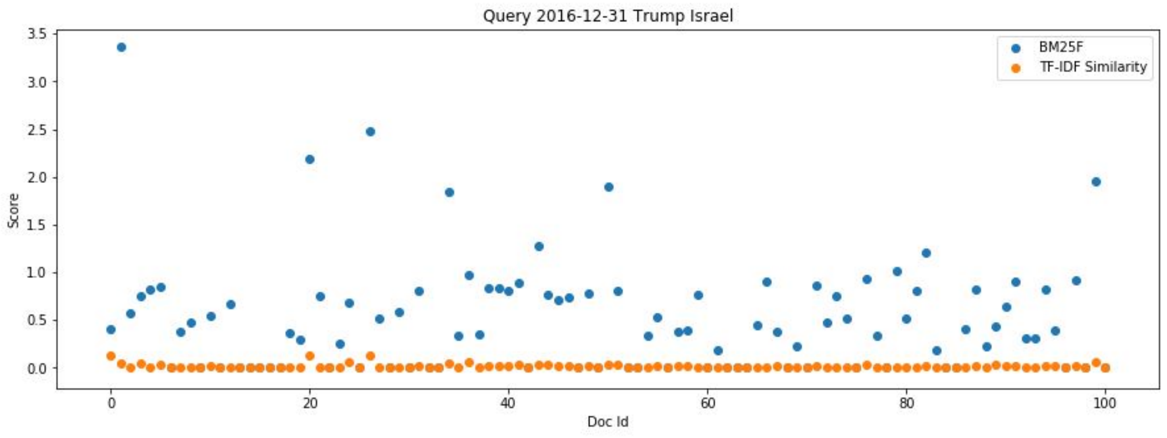
$$tf(t,d) = \sum_{c \in d} w_c * tf_c(t,d)$$

and the BM25F saturation function can be computed as:

$$BM25F_D = \sum_{t \in q \cap d} \frac{tf(t,d)}{tf(t,d)+k_1} * idf(t)$$



(a) BM25 vs BM25F



(b) BM25F vs TFIDF

Figure 1.

where c represents each field contained in the document d, w_c is the weight or boost factor for each field in the document and $tf_c(t,d)$ is the field term frequency function of the term t in the field c. k1 controls non-linear term frequency normalization (saturation). The method used by older versions Lucene was to compute the score of an structured document is based on the linear combination of the scores for each field of the document:

$$score(q,d) = \sum score(q,c)$$

where

$$score(q,c) = \sum tf_c(t,d) * idf(t) * w_c$$

and

$$tf_c(t,d) = \sqrt{freq(t)}$$

As we can see, Lucene's ranking function uses $\sqrt{freq(t)}$ to implement the non-linear method to saturate the computation of the frequencies, but the linear combination of the $tf_c(t,d)$ values that Lucene implements breaks the saturation effect, since field's boost factors w_c are applied after of non-linear methods are used. The consequence is that a document matching a single query term over several fields could score much higher than a document matching several query terms in one field only, which is not a good way to compute relevance and use to hurt dramatically ranking function performance.

6. Results and Conclusions

The performance of different ranking models is evaluated. The figure 1(a) shows the result obtained by Lucene's default BM25 ranking algorithm and BM25F ranking algorithm. The figure 1(b) shows the score result obtained by Lucene's default BM25 and TFIDF (ClassicSimilarity) similarity ranking algorithms. It is evident from figure 1(a) that BM25F scores almost fall within a particular region which lies below a particular threshold. On the contrary, BM25 scores are scattered over the entire region and do not follow any particular trend. A threshold maybe chosen for the scores obtained using BM25F but the same can't be done for BM25. In figure 1(b), it is very clear that TFIDF scores almost follow a uniform distribution but BM25F scores change depending on the document under consideration. So, from the above 2 figures it is clear that BM25F performs better than both the lucene's default BM25 and TFIDF scoring.

References

[1] https://opensourceconnections.com/blog/2016/10/19/bm25f-in-lucene/

[2] N. Craswell, H. Zaragoza, and S. Robertson. Microsoft cambridge at trec-14: Enterprise track. In TREC, 2005.

[3] Dataset : https://www.kaggle.com/snapcrack/all-the-news

[4] http://www.minerazzi.com/tutorials/bm25f-model-tutorial.pdf

[5] https://lucene.apache.org/core/6$_{2}1/core/org/apache/lucene/search/similarities/ClassicSimilarity.html$