NLP Course Project
Improving the Simple
VSM Search Engine

Naveen Vakada[1] and Arup Das[2]

[1]Roll Number: CS20S012
[2]Roll Number: CS20S016
{cs20s012,cs20s016}@smail.iitm.ac.in

Abstract. We analyze the results of our search engine implemented using a simple Vector Space Model (VSM). list the shortcoming(s) and explore improvisations to improve the search engine by addressing its current limitations.

Keywords: TF IDF, LSA, Query Expansion, BM25.

1 Introduction

An information retrieval system(IRS) is a system developed to help a user seeking to learn some information. When supplied with a query, the system attempts to produce results relevant to the Query. As time passed by, the system evolved. However, the biggest drawback was that it enlists thousands of documents for a certain Query out of which only few may be relevant to the user. Hence, with no doubt it can be said that the prerequisite of an information retrieval model is to have a good sense of relevance judgment. A good ranking model which can precisely capture the intent of the user query and produce results by predicting the grouping of document-query pairs accurately is a non-negotiable requirement.

This project report begins by describing the problem definition, the motivation behind attempting to improve the current search engine. It then lists the background and related work done using the approaches that we have explored for improvisation. We then discuss the proposed methodology, the experiments performed, results obtained and finally we conclude our experiments with convincing evidence obtained from the statistical hypothesis tests that we have performed to reach the conclusion.

2 Problem Definition

The objective of this project is to design and build an information retrieval system which overcomes the limitations of the current search engine implemented using a simple Vector Space Model (VSM) where the document vector is represented using the TF IDF scores and the retrieved documents are ranked in the non-increasing order of cosine similarity to the query terms. We evaluate and compare the performance of our improvised search engine with the current search engine in terms of nDCG, Mean Precision, Mean Recall and F-score evaluation metrics to determine the retrieval effectiveness.

3 Motivation

3.1 Limitations of the current search engine.

The limitations of the current search engine primarily lie in the algorithm used to find the relevant documents as per the user query.

1. The current search engine fails to classify documents containing synonymous words or documents with the same context as relevant documents.
2. It suffers from the problem of polysemy.
3. In the current search engine the terms are mutually independent whereas in reality there are many words which tend to co-occur and the Vector Space Model does not make use of this co-occurrence relation.
4. The Vector space model does not take into consideration the order in which the terms appear in the document.
5. We ignored the titles of the documents. However, in most cases the user decides to view a document based on the title of the document. So titles are extremely informative in information retrieval.

4 Background and Related Work

Below we discuss the work done in the past based on the various approaches that we have used to improve the current search engine.

4.1 Latent Semantic Analysis (LSA)

LSA finds the low rank approximation of the term-document matrix [1]. The terms (unique words) in the corpus are represented as concepts. The column vectors of V^T represent the document in terms of the concept space. Each concept is weighted by the entries in the diagonal matrix Σ where every diagonal entry(singular values) are arranged in non-increasing order from top to bottom left to right. Hence the weights denote the concepts which are more important. Since we only take the top k singular values we are essentially producing a k-rank approximation of the original term document matrix. This alleviates the problem of synonymous terms as the low rank approximation combines the dimensions of terms having similar meanings. In LSA, it is assumed that terms used in similar contexts have similar meanings and hence are synonymous. LSA also partially alleviates the problem of polysemy because the dimension of the polysemous words which are in the same direction as that of the relevant context get added to dimensions of the words that have the similar meanings.

In our project, we found the size of the latent dimension by cross-validation. It was found that for unigrams, ‘325’ gave good results, whereas in case of bigrams ‘500’ proved to be better. We did not go beyond bi-grams because it is a computationally expensive approach and it takes time for cross-validation.

Interestingly, it has been observed that several neural word embeddings implicitly perform matrix factorisation [2]. The Word2Vec model word2vec factorizes a matrix M which is related to the PMI based co-occurrence matrix (very similar to what SVD does). This is the same case with the NCE embedding model and skip gram with negative sampling. Hence low dimensional vectors are preferred, it has been shown that factorisation with SVD can be at least as good as these neural word embedding models. Since in the Cranfield Dataset there are roughly 1000 documents, the vocabulary size is not that huge so the vector dimensions are low. Hence using SVD for learning the distributed representations [3] is more than sufficient.

There are two drawbacks commonly shared by both LSI and the bag of words vector space approach: there is no proper way of expressing negations (find documents that contain German but not shepherd), and no way of enforcing Boolean conditions.

4.2 Best Match 25 (BM25)

BM25 is a bag of words approach that ranks documents based on the query terms appearing in each document, regardless of the inter-relationship between the query terms within a document. It has been derived from a probabilistic framework [4]. In fact, the formula for idf also comes from the probabilistic model. A probabilistic model of retrieval has a mathematical formalism of relevant and non-relevant sets. This restricts us from ad-hoc engineering (like: tweak, run, observe and tweak the model) due to the constraints provided by the probabilistic framework.

One of the main reasons why significant developments were fostered in the probabilistic model is attributed to the Probability Ranking Principle (abbreviated as PRP). The Probability Ranking Principle (PRP) states that when documents are ranked in increasing order of their posterior probability of relevance P(rel = 1 | Doc) then the ranking obtained will be the best optimal ranking in any of the evaluation metrics used [5].

Assumptions made in the Probabilistic models [6]:

1. One random variable {0,1} to indicate the presence or absence for every word in the document collection.
2. The words are mutually independent of each other.
3. An empty document, that is, a document where all the words are absent is equally likely to belong to the relevant and the non-relevant class.
4. If the word is not in the query then it is equally likely to be present in the relevant and the non relevant category.
5. On an average a query word will occur in half the relevant documents.
6. Almost the entire collection of documents is non relevant with respect to a query.

Retrieval functions like BM25 were designed keeping in mind the objective to improve the probabilistic model by relaxing the assumptions. The BM25 formula derived from the 25th iteration of tweaking the relevance computation is as given below:

$$\Sigma_{t \in q} IDF(t) * \frac{(k_1 + 1) * tf(t, d)}{k_1[(1 - b) + \frac{b * dl}{avdl}] + tf(t, d)} * \frac{(k_2 + 1) * tf(t, q)}{k_2 + tf(t, q)} \quad (1)$$

where, $IDF(t) = log\left(\frac{N}{df(t)}\right)$

In the presence of full relevance judgement,

$$IDF(t) = log\left(\frac{\frac{r_t + 0.5}{R - r_t + 0.5}}{\frac{df(t) - r_t + 0.5}{N - df(t) - R + r_t + 0.5}}\right)$$

$$\Rightarrow IDF(t) = log\left(\frac{[r_t + 0.5] * [N - df(t) - R + r_t + 0.5]}{[df(t) - r_t + 0.5] * [R - r_t + 0.5]}\right) \quad (2)$$

Here,

- N is total number of documents in the corpus,
- r_t is the number of relevant documents containing the term t,
- R is the number of relevant documents for the query q,
- df(t) is the number of documents containing the term t,
- dl is document length measured in terms of number of words in the document,
- avdl is the average length of the document in the collection.
- tf(t,d) is the frequency of term t in document d,
- tf(t,q) is the frequency of term t in query q,

- K_1 is a non-negative tuning parameter which scales the document term frequency,
- k_2 is a non-negative tuning parameter which scales the query term frequency and
- b is a non-negative tuning parameter which scales the document length

$df(t) - r_t$ denotes the number of documents in the judged sample which contain the term t but are not considered to be relevant to the query q.

$P(tf(t) \mid rel) \approx \frac{r_t}{R - r_t}$ represents the standard maximum likelihood estimate of the probability of a term in a document contributing to its relevance.

$\frac{df(t) - r_t}{N - df(t) - R + r_t}$ denotes the probability of choosing an irrelevant document containing the term t.

Thus, $\frac{\frac{r_t + 0.5}{R - r_t + 0.5}}{\frac{df(t) - r_t + 0.5}{N - df(t) - R + r_t + 0.5}} = \frac{[r_t + 0.5] * [N - df(t) - R + r_t + 0.5]}{[df(t) - r_t + 0.5] * [R - r_t + 0.5]}$ represents the ratio between the term's relevance odds and the term's non relevance odds where 0.5 is a smoothing correction applied to deal with the limiting cases when any of the components in the formula become 0 [7].

On applying log to the formula obtained above we get what is popularly known as Robertson Sparck Jones Weight (w^{RSJ}_t)

In the absence of the relevance information which is the most commonly experienced scenario, P(tf(t) | rel) is fixed and is set to 0.5 (assumption 5 in PRP). As roughly, the entire collection of documents is irrelevant to the query (assumption 6 in PRP) we get $R = r_t = 0$.

Thus, $w_t^{RSJ} = log\ [\frac{[0 + 0.5] * [N - df(t) - 0 + 0 + 0.5]}{[df(t) - 0 + 0.5] * [0 - 0 + 0.5]}]$

$$\Rightarrow w_t^{RSJ} = log\ [\frac{0.5 * [N - df(t) + 0.5]}{[df(t) + 0.5] * 0.5}]$$

$$\Rightarrow w_t^{RSJ} = log\ [\frac{N - df(t) + 0.5}{df(t) + 0.5}] \quad (3)$$

If we closely observe, due to assumption 5 in PRP, $log\ [\frac{N - df(t) + 0.5}{df(t) + 0.5}]$ will be negative for terms which are there in more than half of the documents in the collection because log values between 0 and 1 are negative. Our desire is to avoid negative values arriving out of our ranking function since the presence of a query term in a document should not contribute to a lower score than if the term was simply absent from the document. Hence to avoid this scenario we add a 1 to the component [8].

$$log\ [1 + \frac{N - df(t) + 0.5}{df(t) + 0.5}] = log\ [\frac{df(t)}{df(t)} + \frac{N - df(t) + 0.5}{df(t) + 0.5}]$$
$$\Rightarrow log\ [1 + \frac{N - df(t) + 0.5}{df(t) + 0.5}] \approx log\ [\frac{df(t)}{df(t)} + \frac{N - df(t)}{df(t)}]$$
$$\Rightarrow log\ [1 + \frac{N - df(t) + 0.5}{df(t) + 0.5}] = log\ [\frac{N}{df(t)}] \quad (4)$$

Hence in the absence of relevance judgements, IDF(t) in BM25 reduces to the IDF(t) in the traditional tf idf computation. Therefore, the global weights in BM25 and traditional tfidf are obtained from Robertson Sparck Jones (RSJ) weight. We now focus on how the formula for local weights are obtained.

The ultimate goal of traditional tfidf is that if a term is repeated in several documents then it should be given a low score whereas if a term is rarely observed in multiple documents then it should be given more weightage considering that the occurrence of the term is significant in judging the relevance of the document to the query.

From a logical point of view, the term frequency of a term can potentially climb up to infinity due to the absence of any boundary to the term frequency score. So, by this approach the lengthier documents which have rare terms being repeated several times would have an unfair advantage over shorter documents which are actually more relevant and the term frequency can rise linearly to a very high value (close to infinity). This has happened because **traditional tf idf rewards term frequency and penalises document frequency**.

BM25 will overcome this situation using **term frequency saturation** and **document length normalisation**. It exploits the latent property between a document-term pair called "***eliteness***". Eliteness describes the *aboutness* of a

document with respect to its terms. It makes two key assumptions.

1. The occurrence of a term in a document depends on its eliteness.
2. There is a relationship between the eliteness of a term and the relevance to a query.

On the basis of the above two assumptions it is concluded that term frequency tf is independent of relevance.

$$P(TF_t = tf \mid rel) = p(E_t = elite \mid rel)p(TF_t = tf \mid E_t = elite) + (1 - p(E_t = elite \mid rel))p(TF_t = tf \mid E_t = \neg elite)$$

$$P(TF_t = tf \mid rel) = p_{t1}E_{t1}(tf) + (1 - p_{t1})E_{t0}(tf) \quad (5)$$

The third assumption made by BM25 is that eliteness is a binary value. The distribution of term frequencies obeys a binomial distribution which can be approximated by a Poisson distribution. This distribution is unique based on if the term is elite or not . This is called the "2-Poisson model" introduced by Harter [9].

Recalling that the objective of PRP is to rank the documents as per the posterior relevance of the documents we get

$$P(rel = 1 \mid Doc) = \frac{P(rel = 1 \mid Doc)}{P(rel = 0 \mid Doc)}$$

From the Bayes rule:

$$P(rel = 1 \mid Doc) = \frac{P(Doc \mid rel = 1)}{P(Doc \mid rel = 0)} * \frac{P(rel = 1)}{P(rel = 0)}$$

This model has a generative form. We choose a generative model because we can never have enough data to accurately estimate the probability of relevance of a document to a query and making estimations are easier for a generative model particularly in cases where we lack observations.

We can drop the priors $\frac{P(rel = 1)}{P(rel = 0)}$ because they do not serve any purpose in the ranking process since they are the same for every document in the collection. Hence $\frac{P(rel = 1)}{P(rel = 0)}$ is a constant and does not affect the ranking process.

Now we have,

$$P(rel = 1 \mid Doc) = \frac{P(Doc \mid rel = 1)}{P(Doc \mid rel = 0)}$$

From the six assumptions made in PRP we can reduce the above formula to:

$$P(rel = 1 \mid Doc) = \Sigma_{t \in q, tf_t > 0} w_t(tf) \quad (6)$$

where,

$$w_t(tf) = w_t^{elite}(tf) * w_t^{RSJ} \quad (7)$$

and

$$w_t^{elite}(tf) = log\left[\frac{[p_{t1}E_{t1}(tf) + (1 - p_{t1})E_{t0}(tf)] * [p_{t0}E_{t0}(0) + (1 - p_{t0})E_{t0}(0)]}{[p_{t1}E_{t1}(0) + (1 - p_{t1})E_{t0}(0)] * [p_{t0}E_{t0}(tf) + (1 - p_{t0})E_{t0}(tf)]}\right] \quad (8)$$

By design,

1. $w_t^{elite}(0) = 0$
2. $w_t^{elite}(tf)$ increases monotonically with increase in tf.
3. However, $w_t^{elite}(tf)$ asymptotically reaches a maximum value as $tf(t) \to \infty$

   where, $lim_{tf(t) \to \infty} w_t^{elite}(tf) = log\left[\frac{p_{t1}(1 - p_{t0})}{(1 - p_{t1})p_0}\right]$ (9)

Hence as per the BM25 assumptions the only relationship between tf and relevance is through eliteness, that is, the most useful information we can get from a term is whether the document is truly elite for that term.

It has been shown that how many ever times a term occurs in a document its contribution to the document score cannot exceed an asymptotic limit called the *saturation point.* This saturation behaviour does not apply when we assume that the eliteness property for every term in the query coincides with the relevance for the query, that is, when $p_{i1} = 1$ and $p_{i0} = 0$. This makes the limit infinite and weight grows linearly with the increase in tf. Thereby, exhibiting the characteristics of tf in the traditional tf idf computation. However, the non linear saturating function of tf has proven itself to deliver better results than the traditional tf idf.

$p(E_t = elite \mid rel)$, $p(TF_t = tf \mid E_t = elite)$ and $p(TF_t = tf \mid E_t = \neg elite)$ are difficult to calculate due to the absence of an appropriate generative corpus model. So it has been approximated using a parametric curve $\frac{tf(t)}{k + tf(t)}$, $k > 0$.

Higher the value of k, higher is the contribution made by increments in tf to the document score. Lower the value of k, implies that the contribution made by the

increments in tf to the document score will reach a maximum value very rapidly. This is clearly observed from the graph below.

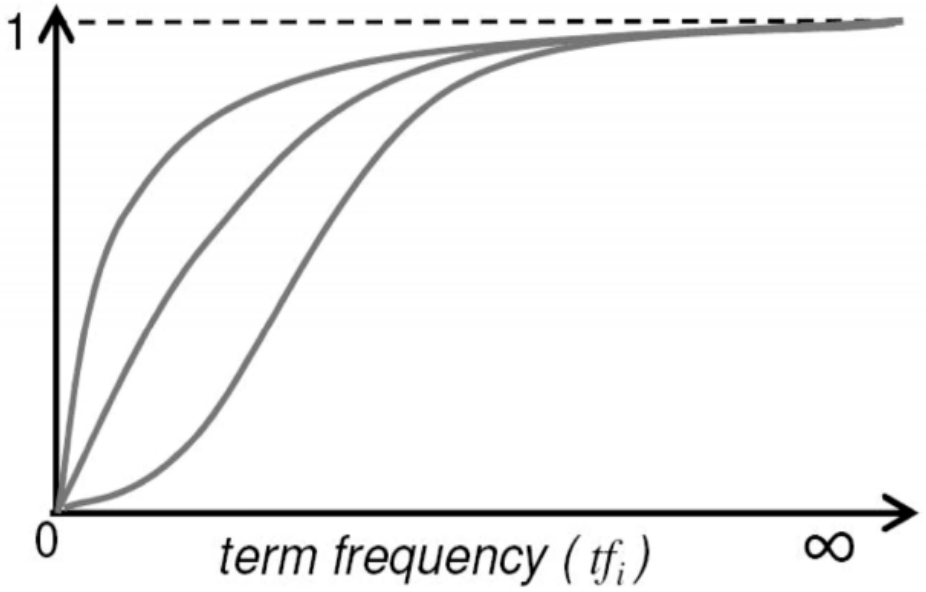

Fig. 1. Saturation functions generated by the 2-Poisson model (Source: Robertson et. al., The Probabilistic Relevance Framework: BM25 and Beyond, Foundations and Trends in Information Retrieval, 2009)

In the real world, documents mostly appear in varying lengths and the reason for such a behaviour is partly explained explained by each of the below two hypotheses:

1. **Verbosity Hypothesis**: Some authors tend to use more words to convey a particular concept.
2. **Scope Hypothesis**: Some authors may create a document which conveys more than one concept or they may also concatenate two or more documents and create a single document.

The term $k_1[\,(\,1 - b\,) + \frac{b*dl}{avdl}]$ is a soft or adjustable normalisation technique used to normalise the term frequencies so as to obey the two contrasting hypotheses: Verbosity and Scope Hypothesis ideally observed in document collections [10].

b = 1 will perform full length document normalisation while b = 0 will disable the document length normalisation. Hence b gives BM25 the flexibility of choosing the importance of document length as per the collection of the documents.

The python library rank_bm25 which we have used sets the default value of k_1 and b to 1.5 and 0.75 respectively. To conduct our experiments we have used these default values.

On applying the soft normalisation to the saturation function (approximated using the parametric function) we get:

$$\frac{\frac{tf(t,d)}{(1-b)+\frac{b*dl}{avdl}}}{k_1+\frac{tf(t,d)}{(1-b)+\frac{b*dl}{avdl}}} = \frac{\frac{tf(t,d)}{(1-b)+\frac{b*dl}{avdl}}}{\frac{k_1(1-b)+\frac{b*dl}{avdl}+tf(t,d)}{(1-b)+\frac{b*dl}{avdl}}} = \frac{tf(t,d)}{k_1[\,(1-b)+\frac{b*dl}{avdl}]+tf(t,d)}$$

The (k_1+1) factor in the numerator doesn't alter the ranking because it is present in the calculation of all the terms. However its usage makes the term score 1 when tf = 1 [11]. Also, since it does not affect the scores, in many implementations (for e.g. in Lucene) the (k_1+1) factor in the numerator is simply ignored. The below graph shows the behaviour of the saturation function when the document length (defined as $dl = \Sigma_{t \in V} tf(t)$) is the less than, same as or greater than the average document length.

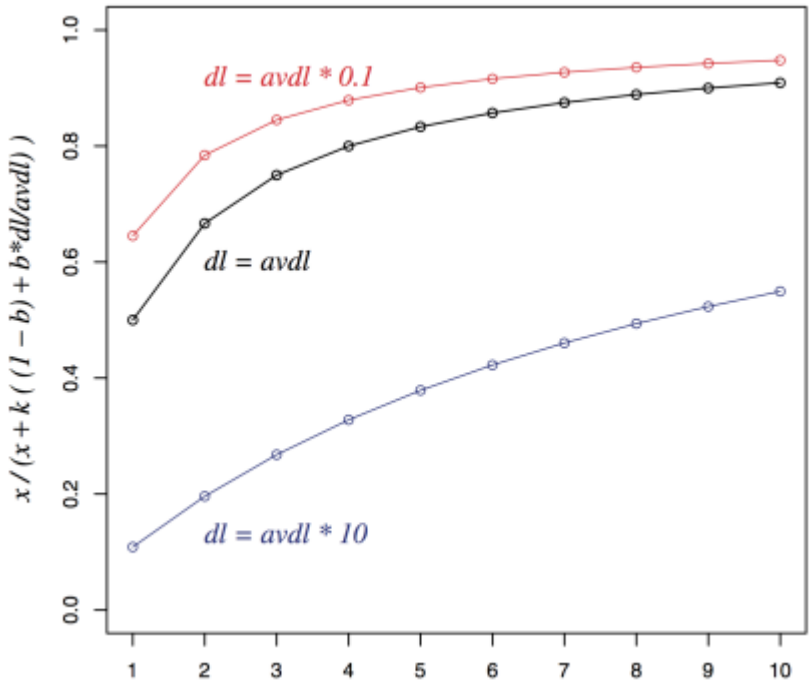

Fig. 2. Strong saturation is obtained with smaller values of k and with shorter documents. Here k = 1 and b = 0.5 (Source: Robertson et. al., The Probabilistic Relevance Framework: BM25 and Beyond, Foundations and Trends in Information Retrieval, 2009)

The TF-IDF implementation produces documents relevant to the query irrespective of user's relevance whereas BM25 takes into account the total number of relevant documents while calculating the score. This makes BM25 more efficient based on the relevant documents list. Also, even if the global weights in BM25 behaves like the idf in traditional tf idf computation due to the way BM25 computes the local weights it tends to perform better than the traditional tf idf.

The second component, $(k_2 + 1) * tf(t, q)$ is required only for long queries, say for example if a query has information requirements that are a paragraph long. However, it is unnecessary for short queries. In our project as all the queries are not more than one line in general hence we have not considered this term in our computation of the BM25 scores.

There is no saturation function of term frequencies of query terms because each time the documents are retrieved with respect to a single fixed length query. So there is no length normalisation required for the frequencies of the query terms. This phenomenon has also been observed experimentally.

Thus, the local weights are obtained from a parameterised version of the 2-Poisson model as described above and the global weights are obtained from Robertson Sparck Jones (RSJ) weight.

One of the limitations of the BM25 retrieval function is that it does not give any guidelines on how to set the values of the hyperparameters. Hence to find the right configuration of the hyperparameter values one of the commonly followed approaches is performing optimization on an evaluated set of queries along with relevance judgements. From several major experiments performed in the past it has been suggested that b = 0.75 and $k_1 \in [1.2, 2]$ are some of the appropriate values.

Another drawback of Okapi BM25 is that it tends to extensively penalise very long documents. This is because on applying document length normalisation in the saturation function the term frequency could become very small and may approach to 0 indicating the absence of the query term in the document. Hence the presence of a query term in a very long document fails to differentiate it clearly from other documents where the query term is absent suggesting that BM25 overly penalises very long documents [12]. However, in our project the Cranfield Dataset does not have very long documents and hence BM25 performs well in our case.

4.3 Query Expansion

Since Query Expansion will introduce additional tokens or phrases to a query ultimately enriching the user's initial query with synonymous words and words which represent a similar context. Hence Query expansion will significantly help in improving the recall.

In order to do query expansion, we need to have the similarity scores between all the pairs of words in the vocabulary. HEnce we need to calculate the word similarity matrix where an element in the i^{th} row and j^{th} column represent the word similarity between the i^{th} word and the j^{th} word in the vocabulary.

There are many ways to calculate this word similarity, one way is to use the wordnet to get the similarity score [13]. Another approach is to use the vector representation of the words in the concept space obtained from LSA.

We can use the vector representation of the words in the vocabulary obtained from LSA within unigrams and then compute the similarity matrix by taking cosine similarity between all the word pairs. We chose the latter approach as the expanded query would have terms already present in the document collection [14].

Query expansion suffers from the problem of intent drifting. As with our current approach we have no way of determining which of the terms in the query (obtained after data pre-processing) are more important. We adopt a random approach of choosing any 3 additional query tokens (this selection is uniformly random) during the query expansion phase. Such an approach helped us to curb the extent of intent drifting.

4.4 Convex Combination

Retrieval models like latent semantic analysis associate a query against the documents from the collection in the latent semantic space. Contrary to this, the bag of words approach such as tfidf and BM25, find exact matches of query terms in the textual content of the documents to determine the relevance of a document to the query. We hypothesize that latent semantic representations complement matching (Inexact matching) with bag of words model representations (exact matching), and that a combination of the two is favourable.

Hence by the idea of ensemble methods where a combination of multiple models produce more accurate results than a single model, our objective here is to combine the LSA model and the BM25 model where the weightage of combination is dictated by the degree of usefulness of the models in the given evaluation metric.

$$\alpha = \frac{a}{a+b} \text{ and } \beta = 1 - \alpha \tag{10}$$

where a = nDCG(LSA) @K = 3 and b = nDCG(BM25)@K = 3

$$Final\ Score = \alpha * CosSim(LSA) + \beta * BM25Score \tag{11}$$

Note that the Cosine similarity score and the BM25 scores are scaled between 0 and 1 prior to applying the convex combination. This is done so as to ensure that both the components are in the same scale giving us a result between 0 and 1.

5 Proposed Methodology

Below algorithm enlists the steps that would be adopted to test our improvised approach.

1. Pre-process the corpus (Cranfield Dataset) and the query.
    a. Sentence Segmentation.
    b. Word Tokenization.
    c. Stopword Removal.
    d. Stemming
2. Create TF IDF vector representations of the documents in the collection.
3. Build the LSA model with n-grams where n = 2, by computing the SVD decomposed matrices U, Σ and V^T.
4. Represent the documents in the corpus in the latent semantic space.
5. Create a word similarity matrix using LSA with a unigram model.

6. Expand the query using the word similarity matrix obtained from the previous step.
7. Represent the expanded query in the latent semantic space.
8. Compute the cosine similarity scores between the documents in the collection and the expanded query represented in the latent semantic space.
9. Compute the BM25 scores of the tokenized documents in the corpus with respect to the tokenized expanded queries.
10. Scale the cosine similarity and BM25 scores between 0 and 1.
11. Perform the convex combination of the two scores obtained in steps 9 and 10.
12. Rank the documents in the decreasing order of the scores obtained in step 10.
13. Output the top k retrieved results.

The below diagram shows the final recommended architecture of the proposed approach:

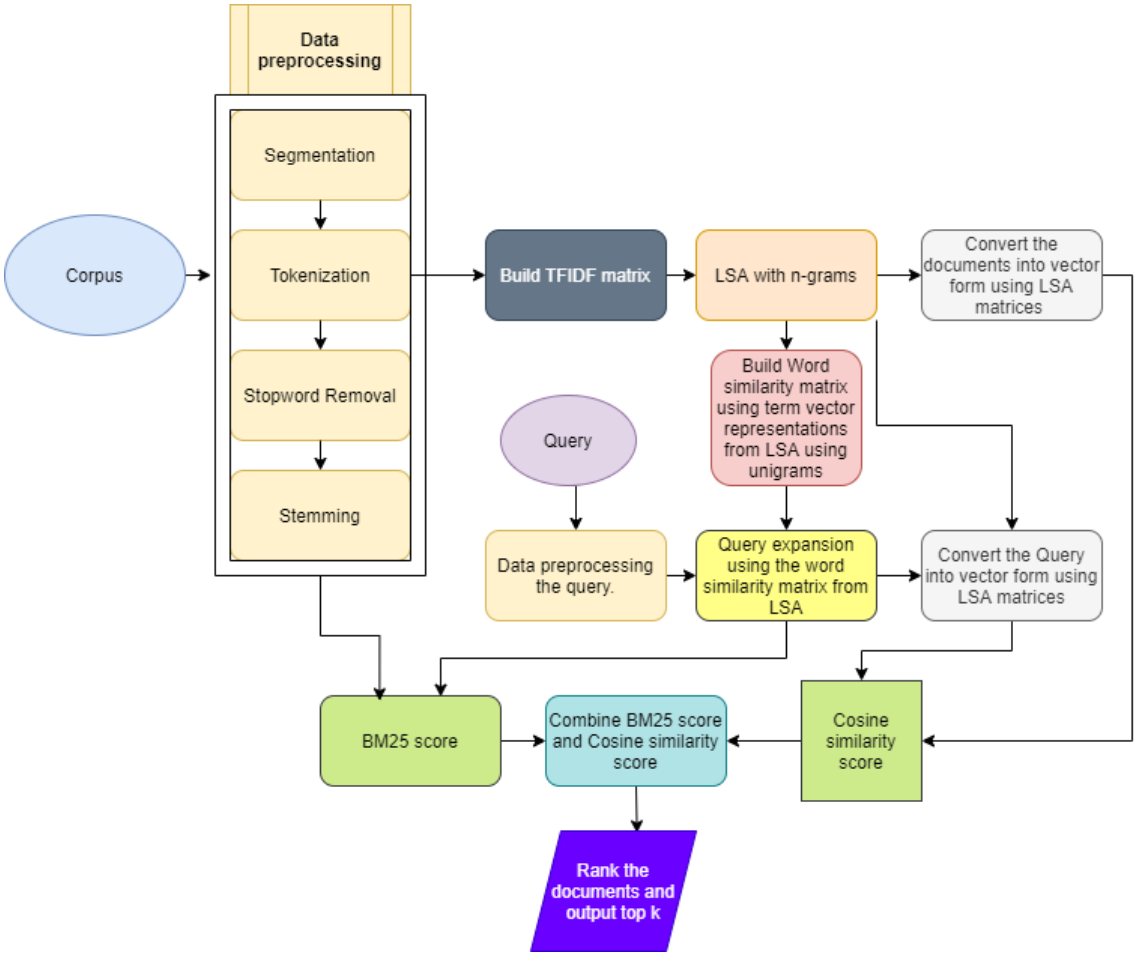

Fig. 3. Information Retrieval System Architecture

The implementation for the proposed methodology along with the experiments and test results can be found in the ipython notebook file: NLP_final_project.ipynb provided in the code folder.

5.1 Custom Query Search Application

Apart from testing the improvised approach we have also created a custom query search application where the user can get a seamless experience out of the various approaches suggested upon entering a custom query. The user is provided with the flexibility to choose the retrieval function and the number of top K results that they want to view.

Below is the screenshot of the custom search application that we have created. As it is quite natural for a user to make errors while typing the queries, we have used a spell check algorithm based on Peter Norvig's spell correction algorithm [15].

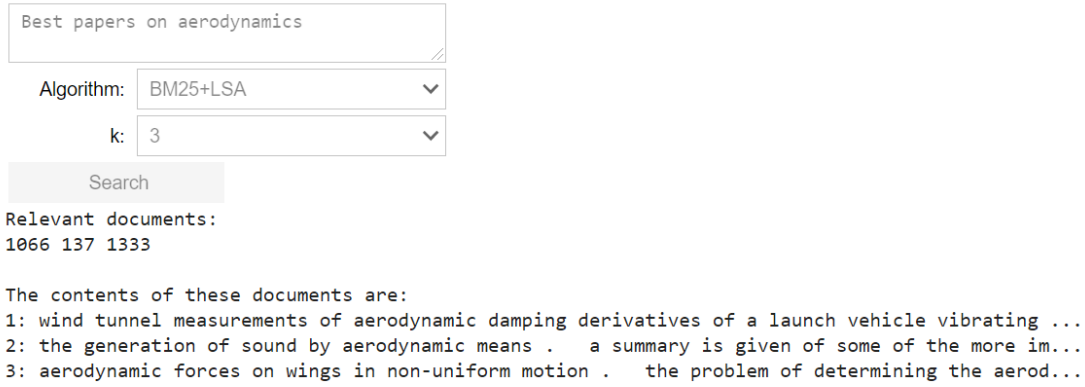

Fig. 4. Information Retrieval System with Custom Search

The implementation for the same can be found in the ipython notebook file: NLP_final_project_custom_search_application.ipynb provided in the code folder.

6 Experiments

We attempt to improve the current state of the art of the search engine using the below hypotheses:

Candidate Hypothesis (H_1): Latent Semantic Indexing instead of just TF-IDF vectorial representation of documents can better handle the problem of synonymy and co-occurrence relation. We even attempt to solve the problem of sequencing to some extent using the n-gram approach in LSA.

Candidate Hypothesis (H_2): Using Query Expansion along with Candidate Hypothesis 1 will improve the search results.

Candidate Hypothesis (H_3): Using BM25 (Best Match 25) as a ranking function instead of TF-IDF will produce better relevant results.

Candidate Hypothesis (H_4): A convex combination of Query Expansion with Candidate Hypothesis 1 and Candidate Hypothesis 3 will perform better than the current search engine.

Please note that we have taken 20 samples where each sample contains 10 queries. Further, we have tested the above hypothesis at k = 3(top 3 results from the retrieval) and the evaluation metrics used are nDCG, Mean Precision, Mean Recall and Mean F-Score. The evaluation metric graphs were plotted for values of k ranging from 1 to 7 because the usefulness of an Information Retrieval System depends primarily on the first few top results. Hence to understand the variations in the test results between the proposed approaches and the existing approach better we took a mid value of k which is 3.

The hypothesis testing method used here is: *comparing the means of two populations with unknown variances* [16].

Null Hypothesis (H_0): Mean of the evaluation metric of a candidate hypothesis does not give a better score when compared to that of the current search engine.

Alternate Hypothesis (H_4): Mean of the evaluation metric of a candidate hypothesis gives a better score when compared to that of the current search engine.

7 Results

The results of the above candidate hypothesis is given below:

```
Hypothesis testing for LSA vs Tfidf using nDCG metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.6729155133501977 y_:0.6347976098868618 sx :0.03156679790347412 sy: 0.026145862823338594 sp: 0.02885633036340636
Test statistic:  0.7095915826432786
t-alpha:  1.3042302030325095
p-value:  0.24114454627814164
alpha:  0.1
Conclusion: No evidence for proving new method is better
```

```
Hypothesis testing for LSA + Query exapsnion vs Tfidf using nDCG metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.674511758875355 y_:0.6347976098868618 sx :0.03289232183558089 sy: 0.026145862823338594 sp: 0.02951909232945974
Test statistic:  0.7309602510007985
t-alpha:  1.3042302030325095
p-value:  0.23464341386159515
alpha:  0.1
Conclusion: No evidence for proving new method is better
```

```
Hypothesis testing for BM25 vs Tfidf using nDCG metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.685363678150915 y_:0.6347976098868618 sx :0.029312145269132384 sy: 0.026145862823338594 sp: 0.02772900404623549
Test statistic:  0.9602671031730882
t-alpha:  1.3042302030325095
p-value:  0.17149570572168504
alpha:  0.1
Conclusion: No evidence for proving new method is better
```

```
Hypothesis testing for BM25+LSA vs Tfidf using nDCG metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.7309630375198581 y_:0.6347976098868618 sx :0.02250387497621015 sy: 0.026145862823338594 sp: 0.024324868899774373
Test statistic:  1.9498163734987124
t-alpha:  1.3042302030325095
p-value:  0.029302512533899816
alpha:  0.1
Conclusion: mean of x is greater than mean of y (new method is better than old method)
```

```
Hypothesis testing for LSA vs Tfidf using Mean precision metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.475 y_:0.45625 sx :0.010657894736842106 sy: 0.01183388157894737 sp: 0.011245888157894738
Test statistic:  0.5591191819261845
t-alpha:  1.3042302030325095
p-value:  0.2896804593655382
alpha:  0.1
Conclusion: No evidence for proving new method is better
```

```
Hypothesis testing for LSA + Query Expansion vs Tfidf using Mean Precision metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.47375 y_:0.45625 sx :0.010360197368421052 sy: 0.01183388157894737 sp: 0.011097039473684211
Test statistic:  0.5253327585334656
t-alpha:  1.3042302030325095
p-value:  0.3012019812854241
alpha:  0.1
Conclusion: No evidence for proving new method is better
```

```
Hypothesis testing for BM25 vs Tfidf using Mean Precision metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.4699999999999999 y_:0.45625 sx :0.010894736842105261 sy: 0.01183388157894737 sp: 0.011364309210526315
Test statistic:  0.40787884160855886
t-alpha:  1.3042302030325095
p-value:  0.3428258368413517
alpha:  0.1
Conclusion: No evidence for proving new method is better
```

```
Hypothesis testing for BM25+LSA vs Tfidf using Mean Precision metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.51125 y_:0.45625 sx :0.008123355263157895 sy: 0.01183388157894737 sp: 0.009978618421052632
Test statistic:  1.7411150986408148
t-alpha:  1.3042302030325095
p-value:  0.04487767248270624
alpha:  0.1
Conclusion: mean of x is greater than mean of y (new method is better than old method)
```

```
Hypothesis testing for LSA vs Tfidf using Mean Recall metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.29712964278509413 y_:0.29228076656552826 sx :0.008132068940914457 sy: 0.008283757156909804 sp: 0.00820791304891213
Test statistic:  0.16924845941442407
t-alpha:  1.3042302030325095
p-value:  0.43324948760521975
alpha:  0.1
Conclusion: No evidence for proving new method is better
```

```
Hypothesis testing for LSA + Query Expansion vs Tfidf using Mean Recall metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.3008810462019387 y_:0.29228076656552826 sx :0.008556739989657197 sy: 0.008283757156909804 sp: 0.0084202485732835
Test statistic:  0.2963808343184353
t-alpha:  1.3042302030325095
p-value:  0.3842765550494184
alpha:  0.1
Conclusion: No evidence for proving new method is better
```

```
Hypothesis testing for BM25 vs Tfidf using Mean Recall metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.30188562985358114 y_:0.29228076656552826 sx :0.00808946969840244 sy: 0.008283757156909804 sp: 0.008186613427656121
Test statistic:  0.3356904961251618
t-alpha:  1.3042302030325095
p-value:  0.36947535044218993
alpha:  0.1
Conclusion: No evidence for proving new method is better
```

```
Hypothesis testing for BM25+LSA vs Tfidf using Mean Recall metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.3263359364913878 y_:0.29228076656552826 sx :0.006950026930596502 sy: 0.008283757156909804 sp: 0.007616892043753154
Test statistic:  1.2339403089636
t-alpha:  1.3042302030325095
p-value:  0.11240129507800733
alpha:  0.1
Conclusion: No evidence for proving new method is better
```

```
Hypothesis testing for LSA vs Tfidf using Mean F-score metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.3404391442668425 y_:0.33100601541598923 sx :0.007656477333043275 sy: 0.008343585799317679 sp: 0.008000031566180477
Test statistic:  0.33351081093398727
t-alpha:  1.3042302030325095
p-value:  0.3702910543089566
alpha:  0.1
Conclusion: No evidence for proving new method is better
```

```
Hypothesis testing for LSA + Query Expansion vs Tfidf using Mean F-score metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.34303658440840035 y_:0.33100601541598923 sx :0.008042838658610336 sy: 0.008343585799317679 sp: 0.008193212228964008
Test statistic:  0.4202996861939884
t-alpha:  1.3042302030325095
p-value:  0.3383169027471973
alpha:  0.1
Conclusion: No evidence for proving new method is better

Hypothesis testing for BM25 vs Tfidf using Mean F-score  metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.3431563886400467 y_:0.33100601541598923 sx :0.008035718980488098 sy: 0.008343585799317679 sp: 0.008189652389902888
Test statistic:  0.42457741069146276
t-alpha:  1.3042302030325095
p-value:  0.3367695540229121
alpha:  0.1
Conclusion: No evidence for proving new method is better

Hypothesis testing for BM25+LSA vs Tfidf using Mean F-score metric
x-new method in this test, y- old method(Tfidf)
null hypothesis: mean of x is less than or equal to mean of y
alternate hypothesis: mean of x is greater than mean of y
x_:0.3713594987313148 y_:0.33100601541598923 sx :0.0063201375660292165 sy: 0.008343585799317679 sp: 0.007331861682673448
Test statistic:  1.4903006722487013
t-alpha:  1.3042302030325095
p-value:  0.07219984852595207
alpha:  0.1
Conclusion: mean of x is greater than mean of y (new method is better than old method)

From the above experiments we get:
alpha (Weight for LSA) :  0.46549453953992714
beta (Weight for BM25):  0.5345054604600729
```

After performing the above tests we can conveniently conclude that the BM25+LSA retrieval model is significantly better than the Vector Space model using just the tf idf vector representation. Here we have considered nDCG, mean precision and mean F-score as the evaluation metrics at $\alpha = 0.1$ (Significance level).

All the results and observations discussed above are accessible over here .

We know that for convex combination, we need both α and β, the values of which are dependent on $'a'$ and $'b'$, To find the values of α and β we have performed the above experiment 20 times and determined the final values of α and β by taking the average of $'a'$ in the 20 runs. As we have used a randomized algorithm from the scikit-learn library available in python to find the SVD decomposed matrices so the values of $'a'$ vary in every run while the values of $'b'$ remains a constant . The values of $'a'$ in every run can be found in the file a-values.csv provided in the Hypothesis testing data folder.

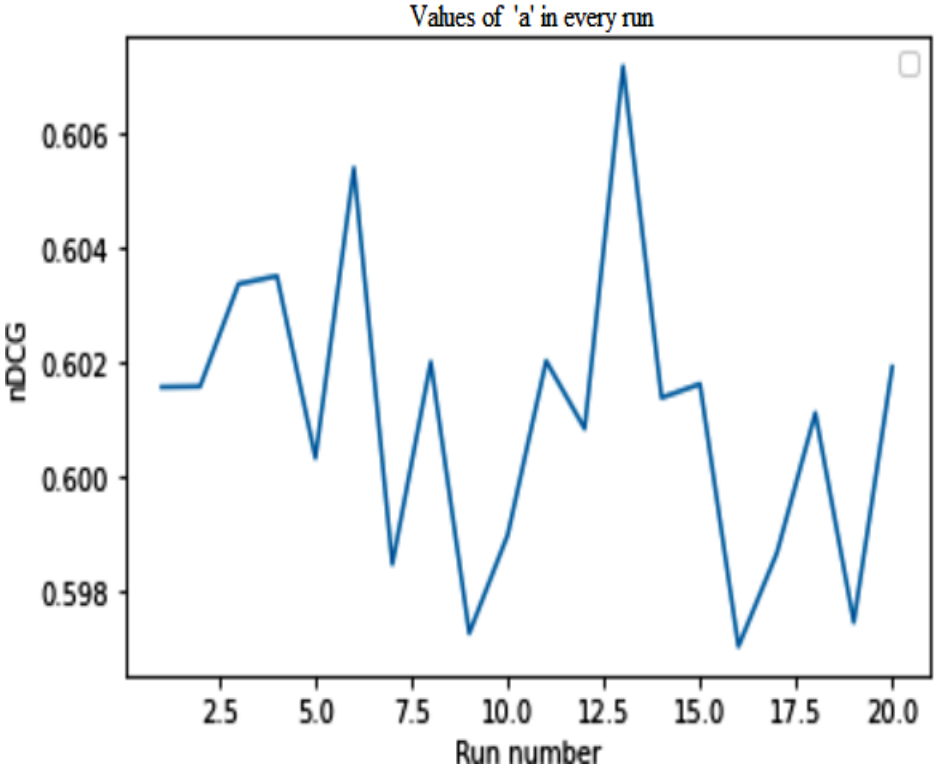

Fig. 4. Evaluation Metric: nDCG

Below are the graphs of the evaluation metrics as a function of k:

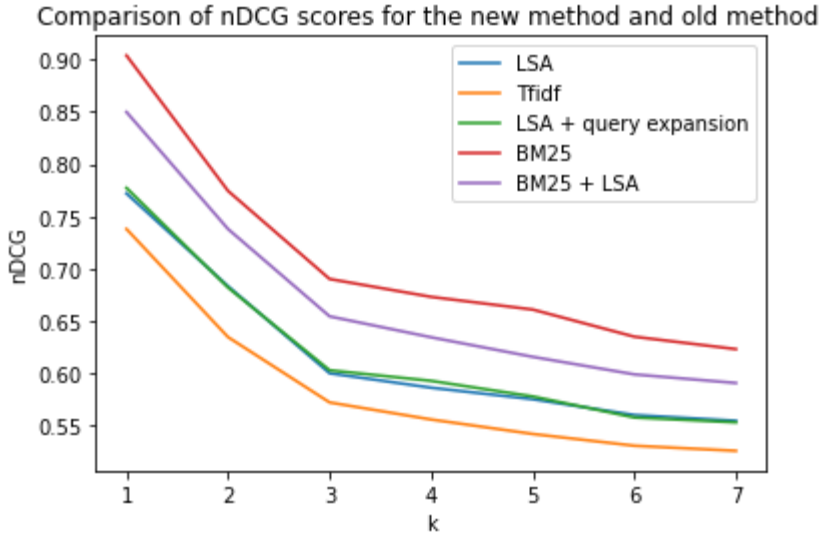

Fig. 4. Evaluation Metric: nDCG

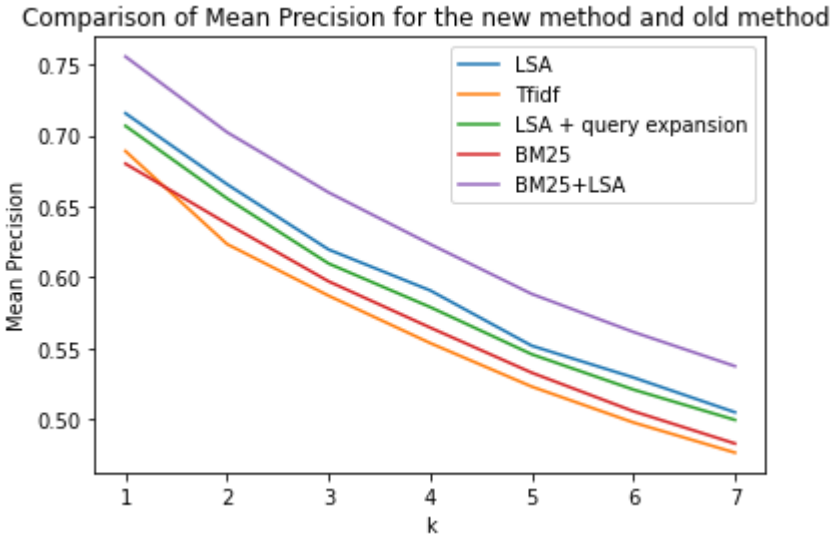

Fig. 5. Evaluation Metric: Mean Precision

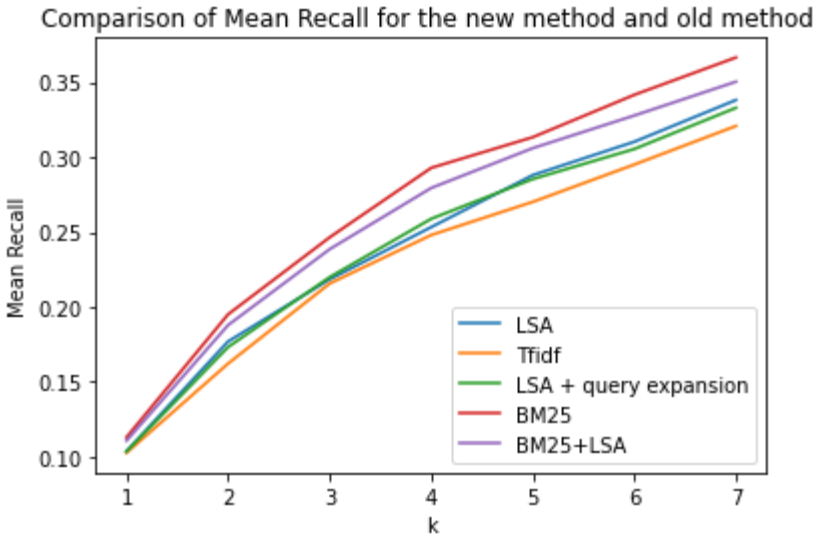

Fig. 6. Evaluation Metric: Mean Recall

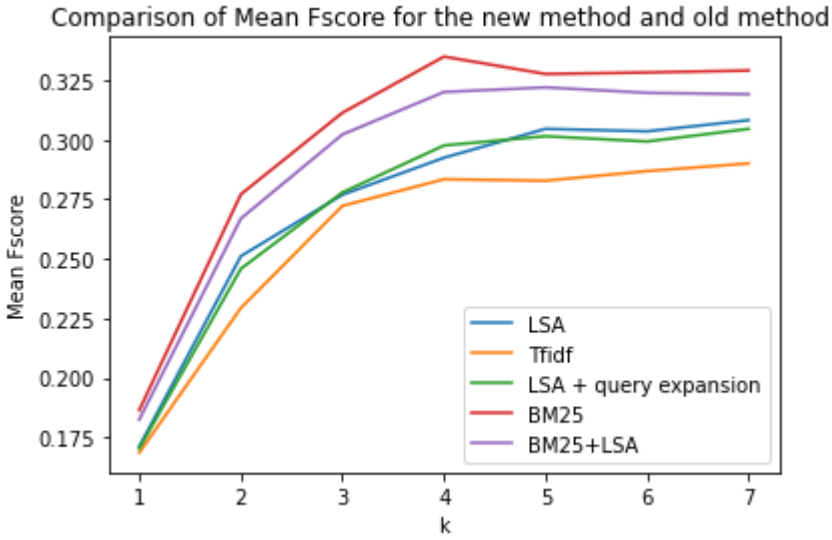

Fig. 7. Evaluation Metric: Mean F-score

From the above graphs, it is evident that BM25 clearly outperforms other retrieval techniques except when the evaluation metric used is mean precision. In case of mean precision, it is the convex combination of BM25 and LSA which outperforms all the other retrieval methods. However, from a statistical viewpoint, BM25 + LSA is the overall best model that we found from our experimental results.

8 Conclusion and Future Enhancements

From our overall understanding of BM25 and LSA, we know that BM25 certainly is better than the traditional TF IDF and LSA with the n-gram approach is clearly better in representing co - occurrence relations, solving the problem of synonymy and in handling the sequence problem. This has also been demonstrated from our statistical experiments and the graphs obtained using the evaluation metrics. Hence from the convincing results obtained above we can conclude that the combination of BM25 and LSA along with n-gram approach and query expansion is a better modelling choice than the current search engine.

While there are machine learning algorithms using Learning to Rank algorithms with hand crafted features to learn ranking models. These are domain specific and obtaining and validating such hand crafted features restricted to only a specific task is time consuming. Hence more focus is being laid on deep learning models which just takes a text as input, learns some abstract way of representing the text information and learns a ranking function. The biggest challenge in deep learning models is thus how to represent the data that accurately represents its relationship to the query. Hence having a good embedding model is the most important requirement to enable quality search in the Neural Network Era. In order to further improve the retrieval results in our project we can explore the possibility of using various neural models for information retrieval.

References

1. Introduction to Information Retrieval, Cambridge University Press. 2008: https://nlp.stanford.edu/IR-book/html/htmledition/latent-semantic-indexing-1.html
2. Levy, Omer and Y. Goldberg. “Neural Word Embedding as Implicit Matrix Factorization.” NIPS (2014). Available here: https://papers.nips.cc/paper/2014/file/feab05aa91085b7a8012516bc3533958-Paper.pdf
3. Mitra, Diaz and Craswell, "Learning to Match using Local and Distributed Representations of Text for Web Search", Proceedings of the 26th International Conference on World Wide Web, April 2017 Pages 1291–1299, https://doi.org/10.1145/3038912.3052579(2016).
4. BIR.11 Estimation without examples: https://www.youtube.com/watch?v=5IirVFBO0lg&list=PLBv09BD7ez_56YwzSyQOOsrAE4Js7XuAB&index=11
5. ROBERTSON, S.E. (1977), "THE PROBABILITY RANKING PRINCIPLE IN IR", Journal of Documentation, Vol. 33 No. 4, pp. 294-304. https://doi.org/10.1108/eb026647, https://www.researchgate.net/publication/235253512_The_Probability_Ranking_Principle_in_IR
6. BIR.13 Summary of assumptions: https://www.youtube.com/watch?v=W3NvM1m3qKQ&list=PLBv09BD7ez_56YwzSyQOOsrAE4Js7XuAB&index=13
7. S.E. Robertson and K. Spärck Jones, Relevance weighting of search terms. Journal of the American Society for Information Science 27, 129-46 (1976). Reprinted in: P. Willett (ed.), Document Retrieval Systems. Taylor Graham, 1988. (pp 143-160), Available here: https://www.staff.city.ac.uk/~sbrp622/papers/RSJ76.pdf
8. https://kmwllc.com/index.php/2020/03/20/understanding-tf-idf-and-bm-25
9. HARTER, S. P. A probabilistic approach to automatic keyword indexing. part I: On the distribution of specialty words in technical literature. Journal of the ASIS 26 (1975), 197–216.
10. http://www.minerazzi.com/tutorials/okapi-bm25-model.pdf
11. https://web.stanford.edu/class/cs276/handouts/lecture12-bm25etc.pdf

12. Yuanhua Lv and ChengXiang Zhai. 2011. When documents are very long, BM25 fails! In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval (SIGIR '11). Association for Computing Machinery, New York, NY, USA, 1103–1104. DOI:https://doi.org/10.1145/2009916.201007
13. Introduction to Information Retrieval, Cambridge University Press. 2008: https://nlp.stanford.edu/IR-book/pdf/09expand.pdf
14. Pilato, Giovanni, et al. "A simple solution for improving the effectiveness of traditional information retrieval systems." WSEAS Transactions on Information Science & Applications 2 (2005): 189-194.
15. Peter Norvig's spell correction algorithm: http://norvig.com/spell-correct.html
16. http://www.r-5.org/files/books/computers/algo-list/statistics/probability/Sheldon_M_Ross-Introduction_to_Probability_and_Statistics-EN.pdf