CS6370: NLP PROJECT

Submitted by: **TEAM NO. 52:**
Ritwiz Kamal (CS21D700), Nency Bansal (CS21D002), Amogh Patil (EE19B134)
Submitted on: **03/05/2022**

The goal of this project is to address some of the limitations of the basic Vector Space Model (VSM) based Information Retrieval (IR) system we built in Assignments 1 & 2 and attempt to improve upon the same. In this project we attempt to address three specific aspects of the limitations detailed in the section *Limitations of Vector Space Model* below. Thereafter, we give a detailed account of approaches used for addressing these three limitations, our hypotheses behind such approaches and a comparative analysis of the original VSM with our new approaches involving appropriate rigorous hypothesis testing methods. The same dataset as in Assignment 1 and 2 (Cranfield dataset) has been used for this purpose.

1. Limitations of Vector Space Model

We are trying to address the following aspects of the limitations of the VSM:

Aspect 1: Given a two-term query "A B", the vector space model may prefer a document containing A frequently and not B as compared to a document containing both A and B but less frequently. In other words, a simple TF-IDF based Vector Space Model bluntly gives preference to high term frequencies and penalizes document frequency while deciding the relevance of documents with respect to a given query. Moreover, it does not consider the document length at all while deciding on the relevance thereby ignoring an important aspect that may or may not make a certain document relevant irrespective of term frequencies.

Aspect 2: Semantic sensitivity: Documents with similar context but different term vocabulary won't be associated, resulting in a false negative match. In other words, the similarity between synonyms is not captured properly by the vector space model. It is unable to relate the context if they do not share common words.

Aspect 3: Word Sense Disambiguation: Words that have completely different meanings in different contexts. The vector space model, which is incapable of handling context, treats such polysemous words as the same. Example : "Bank of America" vs "Bank of Amazon". The word Bank is Treated the same in the Vector space Model.

2. Addressing Aspect 1 using BM-25:

The simple TF-IDF is a ranking method that is used to rank the set of documents retrieved as per a user's query in an Information Retrieval system by rewarding relevance. It is defined as follows for a term ***i*** in a document ***j***:

$$TF\_IDF(i,j) \;=\; \frac{tf_{i,j}}{T_j} * log(\frac{N}{df_i})$$

Where, $tf_{i,j}$ represents the number of occurrences of the term ***i*** in the document ***j*** ; T_j represents the total number of terms in document ***j*** ; N represents the total number of documents in the corpus and df_i represents the total number of documents containing the term ***i***. Hence the TF_IDF is nothing but a product of the Term Frequency (TF = $\frac{tf_{i,j}}{T_j}$) and the Inverse Document Frequency (IDF = $log(\frac{N}{df_i})$).

The TF-IDF scoring is based on the assumption that a document is relevant to a term if
(i) It contains a high number of occurrences of the term i.e. the more times a document contains a term, the more likely it is to be about that term. That's to say, term frequency (TF) is used as a proxy for relevance.
(ii) If a large number of documents do not contain the term i.e. the term is unique (not a common term) and has a sufficiently high discriminative power to differentiate between relevant/non-relevant documents.

However, the above defined TF-IDF still has some shortcomings that could potentially severely impact the relevance judgements on retrieved documents. These shortcomings are discussed below:

(i) **Term Saturation and Multi-Term Queries**: [1] If a document contains 200 occurrences of a term say "`elephant`" we cannot really say that it is twice as relevant as a document that contains only 100 occurrences of "`elephant`". We could argue that if "`elephant`" occurs a large enough number of times, say 100, the document is almost certainly relevant, and any further mentions don't really increase the likelihood of relevance. To put it a different way, once a document is saturated with occurrences of a term, more occurrences shouldn't have a significant impact on the score. So we would like a way to control the contribution of TF to our score so that it increases fast when TF is small and then increases more slowly, approaching a limit, as TF gets very big.

BM-25 (Best Matching 25) [1-3] computes the Term Frequency slightly differently that addresses the issue of Term Saturation. Instead of using the raw Term Frequency, it uses:

$$TF = TF/(TF + k)$$

Where, $\boldsymbol{k}$ is a hyperparameter whose optimal value has to be found via experimentation. A fortunate side-effect of using TF/(TF + k) to account for term saturation is that we end up rewarding complete matches over partial ones. That's to say, we reward documents that match more of the terms in a multi-term query over documents that have lots of matches for just one of the terms.

Let's say that "`cat`" and "`dog`" have the same IDF values. If we search for "`cat dog`" we'd like a document that contains one instance of each term to do better than a document that has two instances of "`cat`" and none of "`dog`". If we were using raw TF they'd both get the same score. But let's do our improved calculation assuming k=1. In our "`cat dog`" document, "`cat`" and "`dog`" each have TF=1, so each is going to contribute TF/(TF+1) = 1/2 to the score, for a total of 1. In our "`cat cat`" document, "`cat`" has a TF of 2, so it's going to contribute TF/(TF+1) = 2/3 to the score. The "`cat dog`" document wins, because "`cat`" and "`dog`" contribute more when each occurs once than "`cat`" contributes when it occurs twice [1]. Assuming the IDF of two terms is the same, it's always better to have one instance of each term than to have two instances of one of them.

(ii) **Document Length**: If a document happens to be really short and it contains "`elephant`" once, that's a good indicator that "`elephant`" is important to the content. But if the document is really, really long and it mentions "`elephant`" only once, the document is probably not about elephants. So we would like to reward matches in short documents, while penalizing matches in long documents. To address this BM-25 revisits the TF=TF/(TF+k) formula and modifies it to:

$$TF = TF/(TF + k * (1 - b + b * dl/adl))$$

Where, dl is the document length, adl is the average document length in the corpus and is b is again a hyperparameter which needs to be tuned through experimentation. In a sense, $\boldsymbol{k}$ is the knob that controls the term saturation, and $\boldsymbol{b}$ is the knob that controls the importance of document length.

The IDF is defined in BM-25 as $log((N - DF + .5)/(DF + .5))$ where DF is the document frequency and N is the total number of documents. BM-25 redefines vanilla IDF as a probabilistic IDF, the formula for which has been arrived at through rigorous experimentations by researchers working on ranking functions [1-2].

In summary, simple TF-IDF rewards term frequency and penalizes document frequency. BM-25 [1-2] goes beyond this to account for document length and term frequency saturation. The new score is now calculated as follows:

$$BM25(i,j) = \frac{TF_{i,j}}{TF(i,j) + k^*(1 - b + b^*dl_j/adl))} * log(\frac{N - df_j + 0.5}{df_j + 0.5})$$

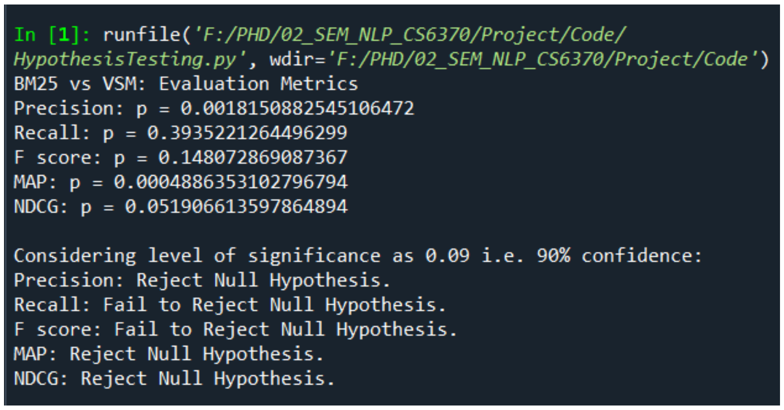
<u>**Comparative Analysis of Performance and Hypothesis Testing:**</u>

Vector Space Model vs BM-25 Model

The Vector Space Model and BM-25 Model were applied to the Cranfield dataset and their respective performances were evaluated with respect to the same five evaluation metrics used in Assignments 1 & 2 namely Precision@k, Recall@k, Fscore@k, Mean Average Precision (MAP@k) and Normalized Discounted Cumulative Gain (NDCG@k). The metric values were averaged over all queries and varied for 10 values of k from 1 to 10. We propose the hypothesis regarding BM-25 Model's performance as follows:- ***Null Hypothesis***: The Mean Evaluation Metric Values for VSM & BM-25 are drawn from the same probability distribution; ***Alternative Hypothesis***: The Mean Evaluation Metric Values for VSM & BM-25 are different and are drawn from the different probability distributions.

$$H_0: Mean\ Evaluation\ Metric_{VSM} = Mean\ Evaluation\ Metric_{BM-25}$$
$$H_a: Mean\ Evaluation\ Metric_{VSM} \neq Mean\ Evaluation\ Metric_{BM-25}$$

In order to test our proposed hypothesis, we employ the Student's Paired sampled t-test for Related Samples where the samples correspond to the evaluation metric values collected over all 225 queries for k = 1 to 10 for both BM-25 and VSM. The evaluation metric values were averaged over all queries per k value. The level of significance was set to **0.09** or **90%**. The following screenshot summarizes the results of Hypothesis testing:

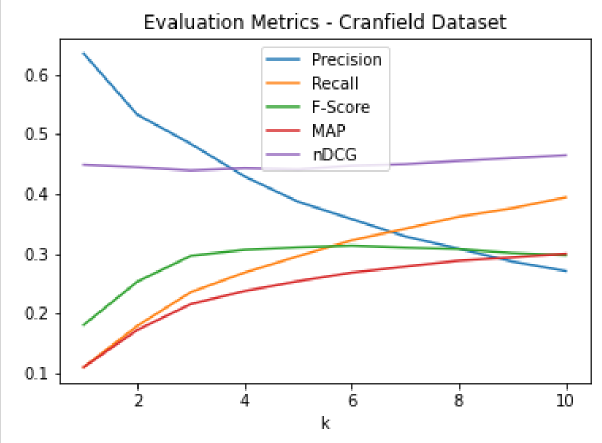


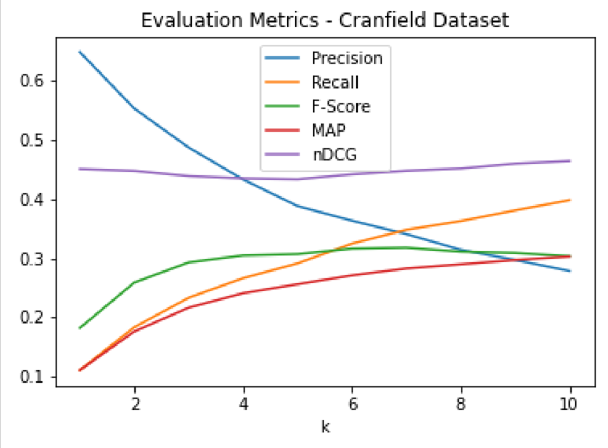
```
In [1]: runfile('F:/PHD/02_SEM_NLP_CS6370/Project/Code/
HypothesisTesting.py', wdir='F:/PHD/02_SEM_NLP_CS6370/Project/Code')
BM25 vs VSM: Evaluation Metrics
Precision: p = 0.0018150882545106472
Recall: p = 0.3935221264496299
F score: p = 0.148072869087367
MAP: p = 0.0004886353102796794
NDCG: p = 0.051906613597864894

Considering level of significance as 0.09 i.e. 90% confidence:
Precision: Reject Null Hypothesis.
Recall: Fail to Reject Null Hypothesis.
F score: Fail to Reject Null Hypothesis.
MAP: Reject Null Hypothesis.
NDCG: Reject Null Hypothesis.
```

Fig: Hypothesis Testing Results: BM25 vs VSM - Evaluation Metrics

It can be seen from the above figure that with 90% confidence we have enough statistical evidence to successfully reject the null hypothesis for Precision, MAP and NDCG evaluation Metrics. However, the statistical evidence is not enough to be able to successfully reject the null hypothesis for Recall and Fscore. Having established statistical support for the hypothesis, we present the comparative analysis of the BM-25 (hyperparameter values k=1.85 & b=0.8 obtained through experimentations, trial and error) and VSM Models below. It can be observed from the comparative plots given below that the BM-25 Model attains a visible improvement over the VSM wrt NDCG metric while it has neck-to-neck on-par performance wrt other metrics like MAP, Fscore, Precision and Recall.

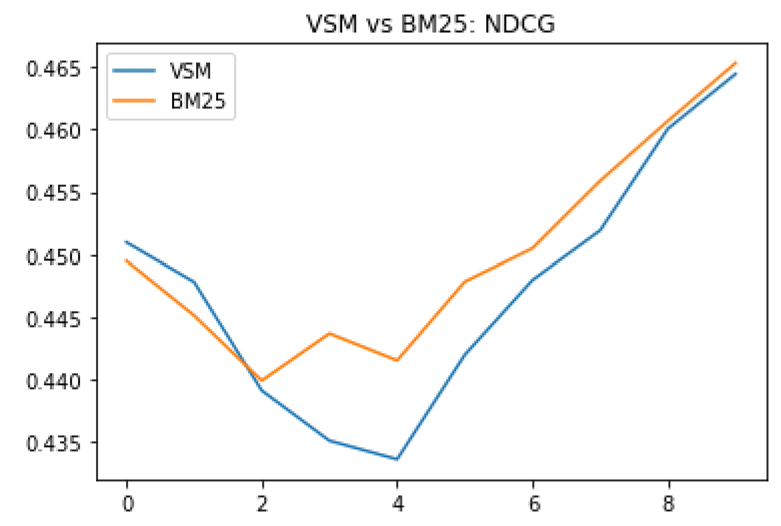


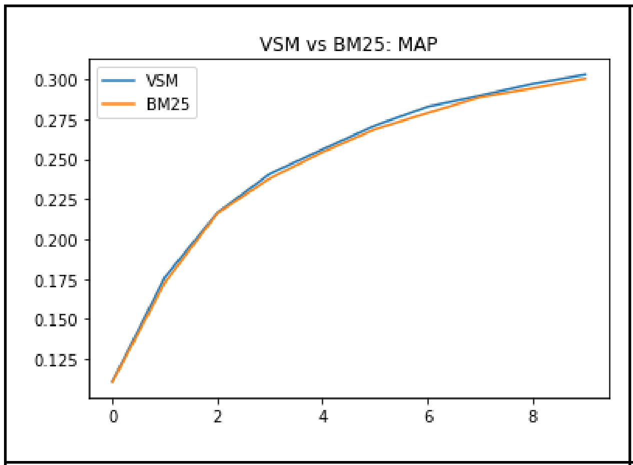


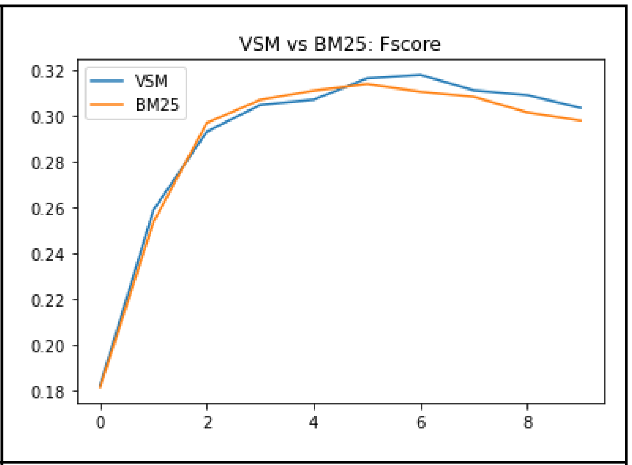
BM-25 **VSM**

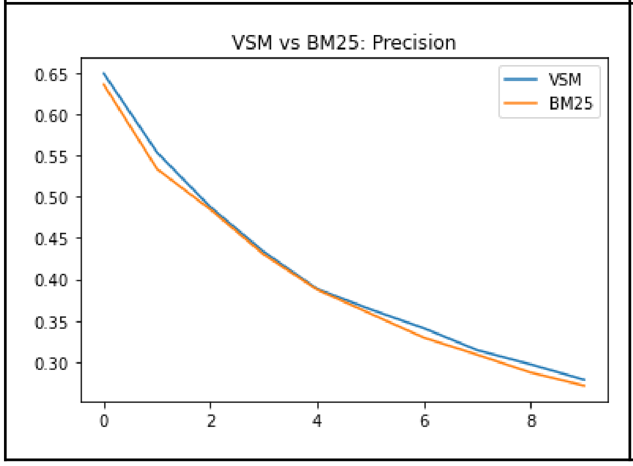
Fig (Top): Evaluation Metric Values vs k | Left: BM-25 | Right: VSM

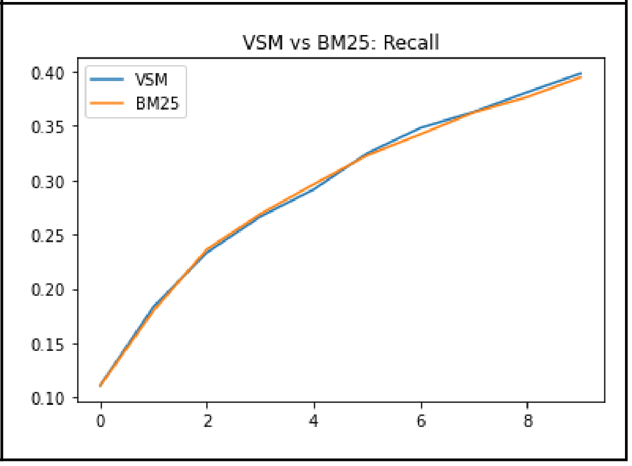
Fig (Below): VSM vs BM25 for Individual Evaluation Metrics







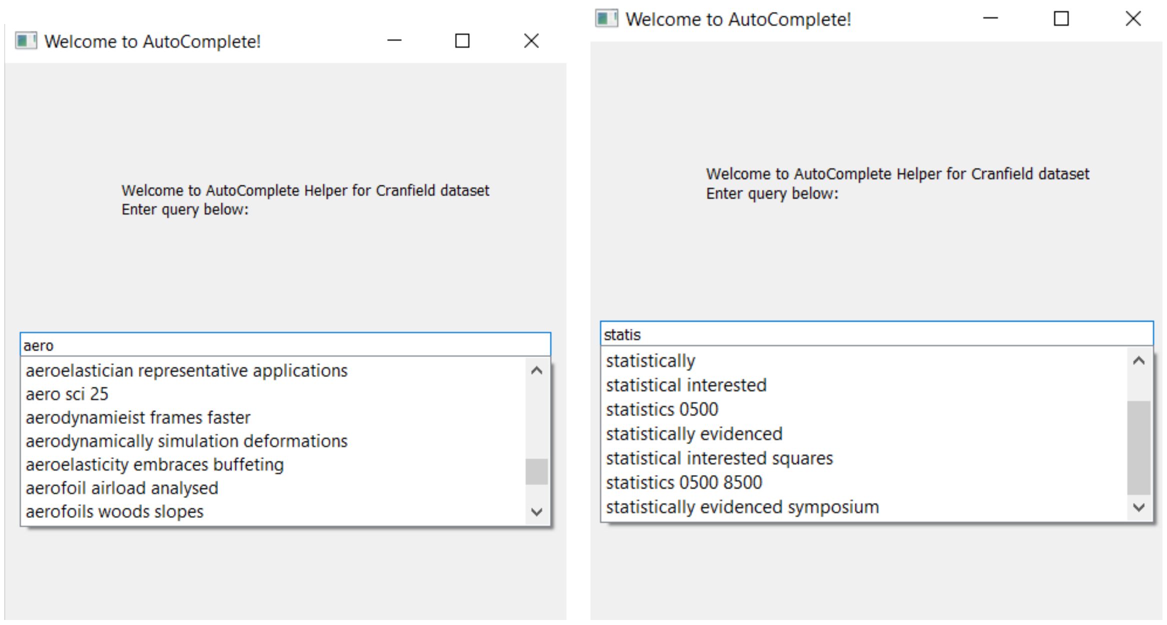




3. Additional Improvement: Query Auto-Completion

As an additional improvement to our Information Retrieval Model, we implemented the functionality of Query Auto Completion [4]. When a user enters a query in a search engine, many times, they end up misspelling the terms or might be genuinely unfamiliar with the exact term(s) they intend to search about. In such cases it is helpful to have an Auto-Complete Suggestion Feature (like in Google) [5] that prompts the user about possible terms they might want to search about depending on the characters they have entered in the input box so far.

We implemented the Auto-Complete Feature using the **unigrams, bigrams** and **trigrams** from the complete set of documents from the Cranfield dataset. This feature does not address any particular limitations of the Vector Space Model and is more of a helpful addition catering to the Handle Custom Query feature supported by our IR system. When a user enters a query through the widget, the same gets stored to a buffer text file and is then fed directly to the handle custom query function. Two sample instances of the AutoComplete Widget are shown below:



Welcome to AutoComplete Helper for Cranfield dataset
Enter query below:

```
aero
aeroelastician representative applications
aero sci 25
aerodynamieist frames faster
aerodynamically simulation deformations
aeroelasticity embraces buffeting
aerofoil airload analysed
aerofoils woods slopes
```

Welcome to AutoComplete Helper for Cranfield dataset
Enter query below:

```
statis
statistically
statistical interested
statistics 0500
statistically evidenced
statistical interested squares
statistics 0500 8500
statistically evidenced symposium
```

Fig: Example of Query Auto-Completer Widget

4. <u>Latent Semantic Analysis (LSA):</u>

We have implemented LSA (Latent Semantic Analysis) [6] to improve the performance of the previous vector space model.

For our Information Retrieval System, LSA performs much better than the Vector Space Model in terms of computational time.

Vector space model is taking around 450 sec whereas the LSA is taking around 80 sec to run.

LSA is an efficient way of analyzing the text and finding the hidden topics by understanding the context of the text. In LSA, the term frequency matrix is first subjected to singular value decomposition (SVD). It reproduces the matrix using the space of latent semantic dimensions and then dimensions are reduced.

LSA has following assumptions:

1. Words that are used in the same context are analogous to each other.
2. The hidden semantic structure of data is unclear because of ambiguity in chosen words.

In the vector space model, the term doc matrix, where rows represent unique words and columns represent documents, is constructed using the complete dataset. In LSA, a statistical technique known as Singular Value Decomposition (SVD) is used to reduce the size of the matrix while preserving the similarity structure between the columns. We can set the value of k as the required number of dimensions.

$X = U \sum V^T$ where,

X is the original term document matrix of shape (m,n)

U contains distribution of words across different context of shape (m, k)

$\sum$ is the diagonal matrix of the association among the contexts of shape (k, k)

V contains the distribution of contexts across the different documents of shape (k, n)

A very significant feature of SVD is that it allows us to truncate a few contexts which are not necessarily required by us. The $\sum$ matrix provides us with the diagonal values which represent the significance of the context from highest to the lowest. By using these values we can reduce the dimensions and hence this can be used as a dimensionality reduction technique too.

Initially the size of tf-idf matrix was (13313, 1400) where,

Vocabulary size = 13313

Number of docs = 1400

Then after applying SVD (k=100) in LSA we reduced it to:

Terms Representation Matrix: 13313, 100

Docs Representation Matrix: 1400, 100

Comparative Analysis of Performance and Hypothesis Testing:

Vector Space Model vs LSA

The Vector Space Model and LSA were applied to the Cranfield dataset and their respective performances were evaluated with respect to the Execution Time.

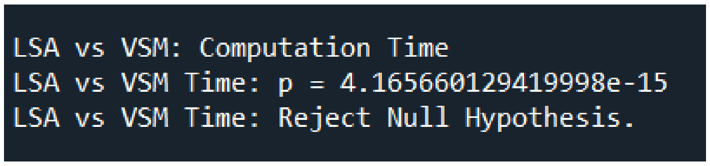
In our information retrieval system we observed that, when the value of k is quite small like k = 5, 10, 20, then it is not performing well in terms of precision, recall, or F1 score . Performance improves as we increase the value of k and saturate at around k = 100. If we increase the value of k above 100, then we can observe only a marginal increase in the performance.

As there is significant reduction in the size of the matrix, hence total time reduction is also significant.

We propose the hypothesis regarding LSA's performance as follows:- ***Null Hypothesis***: The Average Computation Time for Vector Space Model is equal to that of LSA; ***Alternative Hypothesis***: The Average Computation Time for Vector Space Model is not equal to that of LSA

$$H_0: Average\ Computation\ Time_{VSM} = Average\ Computation\ Time_{LSA}$$
$$H_a: Average\ Computation\ Time_{VSM} \neq Average\ Computation\ Time_{LSA}$$

In order to test our proposed hypothesis, we employ the Student's Paired sampled t-test for Related Samples where the samples correspond to the average computation time collected over all 225 queries for both LSA and VSM.



```
LSA vs VSM: Computation Time
LSA vs VSM Time: p = 4.165660129419998e-15
LSA vs VSM Time: Reject Null Hypothesis.
```

Fig: Hypothesis Testing: LSA vs VSM Computation Time

It can be seen from the above figure that we have enough statistical evidence to successfully reject the null hypothesis for computation time.
As we can see in the plot below, there is a significant difference in time taken by both the different models. We have measured the running time by using the time function of the time module. We have calculated the start time, end time and then subtracted them to get the execution time. We have plotted the below graph using that execution time of different runs.

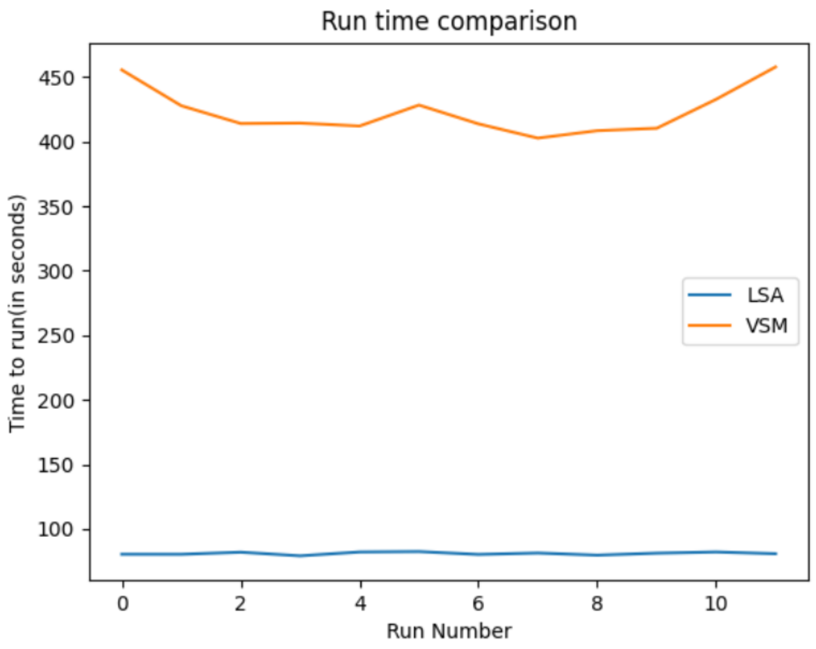


Fig: LSA vs VSM Computation Time Plot

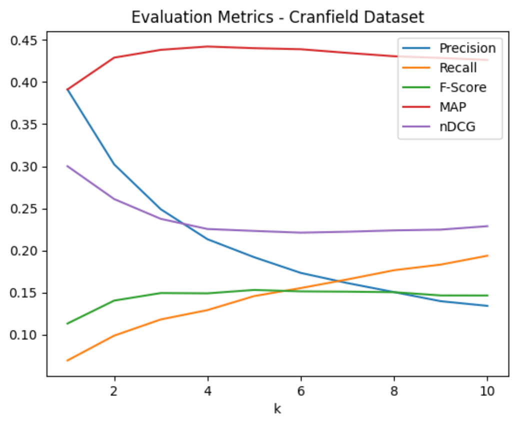
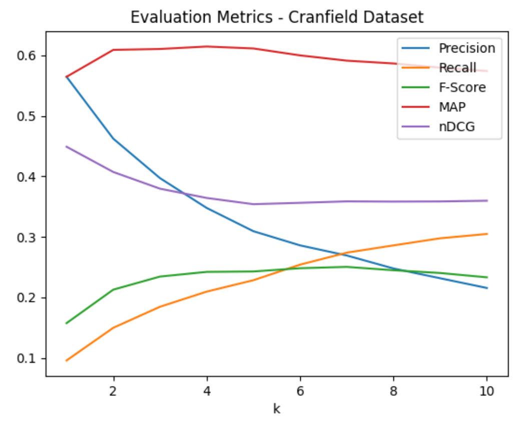
So we can conclude that Approach 2 i.e LSA performs better than Approach1 (Vector Space Model) on task of information retrieval from documents on Evaluation measure as Execution time on Cranfield dataset under the assumptions that other things are kept constant.

5. Explicit Semantic Analysis

Explicit semantic analysis (ESA) is a vectoral representation of text that uses a document corpus as a knowledge base. In ESA, a word is represented as a column vector in the tf–idf matrix of the text corpus and a document is represented as the centroid of the vectors representing its words.

Here we have used a bunch of Wikipedia articles as the text corpus.
Given the practical limitations of not being able to process the entire Wikipedia database, a subset of relevant articles related to the titles of documents in the cranfield dataset were extracted(200 articles). Since there a limited number of articles, some words in the cranfield data set are not found in any of the 200 articles. Hence such words are just assumed to be articles themselves and hence act as the article representative of the word(call it the partial ESA model).

We compare the partial ESA model and the Base model in the extrinsic evaluation of Information Retrieval on the cranfield dataset with the help of the evaluative measures of precision, recall,... .
Null Hypothesis : The partial ESA model does better as ESA would help in bringing non-orthogonality to words and help in word sense disambiguation.
Alternative Hypothesis : Base model does better.

Partial ESA Model	Base Model
	

From above we can say that the Base Model does better.
Due to practical limitations, the full scope of ESA is not being realized.

6. Spell Check

We have implemented a spell check mechanism while pre-processing the queries as well as documents after the tokenization. We have tried spell check using 3 different mechanisms as edit distance, pyspellchecker, textblob.

1. Edit distance: In edit distance measure distance between two words are measured by calculating the number of edit operations required for converting a particular word to another word. It takes each misspelled word, and tries to correct it with the word, which has minimum distance to it.
2. Pyspellchecker [7]: It uses a Levenshtein Distance algorithm to find permutations within an edit distance of 2 from the original word. It then compares all permutations (insertions, deletions, replacements, and transpositions) to known words in a word frequency list. Those words that are found more often in the frequency list are more likely the correct results.
3. Textblob [8]: This is a python library used in natural language processing for correcting spelling mistakes. It has a correct() method which takes input of the incorrect words and returns words with correct spelling.

All these spell check techniques are taking a lot of time for execution and are giving only marginal improvement in the performance of the information retrieval system.

==

REFERENCES:

[1] UNDERSTANDING TF-IDF AND BM-25:
https://kmwllc.com/index.php/2020/03/20/understanding-tf-idf-and-bm-25/

[2] Okapi BM25:
https://en.wikipedia.org/wiki/Okapi_BM25

[3] 3 Vector-based Methods for Similarity Search (TF-IDF, BM25, SBERT):
https://youtu.be/ziiF1eFM3_4

[4] pyqt auto complete: https://pythonbasics.org/pyqt-auto-complete/

[5] Fei Cai and Maarten de Rijke. 2016. A Survey of Query Auto Completion in Information Retrieval. Now Publishers Inc., Hanover, MA, USA. https://dl.acm.org/doi/10.5555/3099948

[6] UNDERSTANDING LSA:
https://towardsdatascience.com/latent-semantic-analysis-intuition-math-implementation-a194aff870f8

[7] Pyspellchecker:
https://pyspellchecker.readthedocs.io/en/latest/

[8] Text Blob:
https://stackabuse.com/spelling-correction-in-python-with-textblob/