CS6370 Project
Information Retrieval System

Keerthana S
Computer Science Department
Indian Institute of Technology, Madras
Chennai, India
cs20b039@smail.iitm.ac.in

Vaibhav Mahapatra
Mechanical Engineering Department
Indian Institute of Technology, Madras
Chennai, India
me19b197@smail.iitm.ac.in

Abstract

In this project, we aim to implement a search engine for the Cranfield dataset. In this document, we will analyse the results of the search engine implemented using a simple Vector Space Model. We look for the model's shortcomings and avenues to overcome them and develop hypotheses on addressing the retrieval failures. We then attempt to test these hypotheses using appropriate testing methods and present the results.

Contents

I. Introduction

Throughout the project, our team explored various ways to tackle the shortcomings of the Vector Space Model (VSM) based search engine for the Cranfield dataset. Some of these shortcomings of the VSM model were:

- The VSM model using TF-IDF measures is a bag of words model, meaning that it does not take word order or word meanings into consideration.
- The representations of words in the documents are considered to be orthogonal to each other when the term-document matrix is constructed and used to find similarities between documents. But in reality, words are highly related through synonymy, polysemy and by being related to similar concepts.
- The problem of circularity, that similar words are those that occur in similar documents, and similar documents are those that contain similar words, goes unaddressed.
- The TF-IDF measure that denotes the relatedness of a term and a document does not consider the document length, which can play a major role in deciding the relevance of a term to a document.
- The model, when used on larger documents and corpora might become very computationally expensive to use, since the representations of the documents become very large.
- The model does not account for human errors such as spelling mistakes that might be present in the queries and/or documents.
- The central concept of the document evinced by the title was ignored and the title was considered as one sentence in the body of the document.

We have tackled some of these issues and implemented additional features including:

1) Okapi BM25 ranking method, over TF-IDF to account for additional important aspects like document length and term frequency saturation, making it a more robust approach.
2) Latent Semantic Analysis (LSA), which uses Singular Value Decomposition (SVD) to derive concepts which overcome the orthogonal representations of words in the model. Word relatedness, especially synonymy is not overlooked this way. The problem of circularity can be overcome as well.
3) The current VSM does not account for each document's title while building document vectors. In our project, we have considered the title with additional emphasis.
4) We implemented a spellcheck mechanism to the model to deal with errors in the query inputs.

The details of each of the models implemented are described in the coming sections.

II. The Dataset

The Cranfield dataset [1] is a collection of documents that was compiled for use in the evaluation of information retrieval systems. The dataset consists of 1,404 technical documents, primarily in the field of aeronautics and related disciplines, along with a set of 225 manually generated queries and relevance judgments for the documents corresponding to the queries. Each document in the dataset includes a title, author name, source information, and the full text of the document. After having implemented an information retrieval system using the Vector Space Model for this dataset, we aim to improve the same using novel methods.

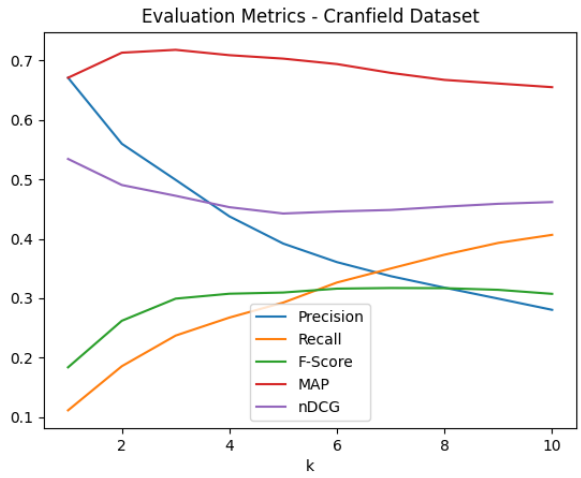


Figure 1. Plot of Evaluation Metrics of VSM Model against k

III. Evaluation metrics

1) Precision: A measure of how many of the top k documents retrieved are actually relevant

$$P@k = \frac{N_k}{k}$$

where N_k is the number of relevant documents in k retrieved documents.
2) Recall: A measure of how many of the relevant documents have been retrieved by the system.

$$P@k = \frac{N_k}{N_{rel}}$$

3) F-Score: The harmonic mean of precision and recall.

$$F_1 = \frac{2 \times P \times R}{P + R}$$

where P is the precision and R is the recall.
4) Mean Average Precision: A measure to take into account if relevant documents are retrieved in the earlier ranks,

which is desirable over retrieving them in later ranks.

$$AP@k = \frac{\sum_{i=1}^{k} P@i}{N_k}$$

where $AP@k$ is the average precision at rank k and $P@i$ is the precision at rank i.

$$MAP@k = \frac{\sum_{i=1}^{N} AP@k}{N}$$

$MAP@k$ is the Mean Average Precision at rank k

5) nDCG: Takes into account the relevance judgments provided in the dataset to see if more relevant documents are retrieved in the earlier ranks. nDCG is calculated as $\frac{DCG}{iDCG}$ where

$$DCG = \sum_{i=1}^{N} \frac{rel_i}{\log_2(i+1)}$$

where rel_i is the graded relevance of the retrieved document at position i. $iDCG$ is also calculated similarly but with the ideal retrieved documents and the corresponding ideal rel_i values.

IV. Hypothesis Testing Methods

In order to test if the models we have implemented perform better than the previous ones in retrieving appropriate results, we have used the **One-sided Paired T-Test** [2]. This test is used to compare two population means where we have two samples in which observations in one sample can be paired with observations in the other sample.

The formula used by the Paired t-test is given as

$$t = \frac{\bar{X}_D - \mu_0}{s_D/\sqrt{n}}$$

where $\bar{X}_D$ and s_D are the average and standard deviation of the differences between all pairs. The constant μ_0 is set to zero since we want to test whether the average of the difference is significantly different. The degree of freedom used is $n-1$, where n represents the number of pairs.

The **one-sided** paired t-test that we have used helps in accepting or rejecting the hypotheses:

- H0: The two related or repeated samples have identical average (expected) values.
- H1: The mean of the distribution underlying the first sample is less than the mean of the distribution underlying the second sample.

This test makes the assumptions that

1) Data values are independent and continuous
2) Data is normally distributed
3) Data is randomly sampled

Such a test is used for instance, when the population means of marks scored by students of a class in two different examinations need to be compared. In our case, the queries in the Cranfield dataset on which our model performance is evaluated can be considered as the students, and the values of the different evaluation metrics (averaged over all the ranks) can be thought of as their marks.

Hence, for each evaluation metric, we consider the two samples to be:

1) The value of the evaluation metric calculated for every query using the model M1 at a given rank
2) The value of the evaluation metric calculated for every query using the model M2 at a given rank

where M1 and M2 are the models to be compared. Since it is more meaningful to seek the top few documents retrieved by an IR system, we run the hypothesis tests at rank 5. To give an idea of the overall performance, we also perform the tests for the evaluation metric values averaged over all ranks.

This has been implemented using the ttest_rel function of the SciPy module in Python [3], with an attribute 'less' to make it one-sided.

The models implemented and their evaluations using the aforementioned metrics, as well as their results when hypothesis tests are carried out, are presented in the following sections.

V. BM25 ranking method

A. The Model

So far, we have used the Term Frequency - Inverse Document Frequency (TF-IDF) value to measure the term-document relevance. It gives more weight to terms that occur in larger numbers in a document over ones that are not as frequent.

$$TFIDF(t,d) = TF(t,d) \times \log_2 \frac{N}{df(t)}$$

where $TF(t,d)$ is the number of times term t occurs in document d, N is the total number of documents in the collection and $DF(t)$ is the number of documents in the collection that contain term t. Okapi BM25 is an improvement over this measure. It is calculated as [4]

$$BM25(t,d) = \frac{TF(t,d)}{TF(t,d) + k1 \times (1 - b + b \times \frac{dl(d)}{adl})} \times IDF(d)$$

where $IDF(j)$ is calculated as

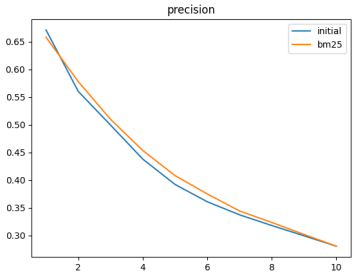
$$\log\left(\frac{N - df_t + 0.5}{df_t + 0.5}\right)$$

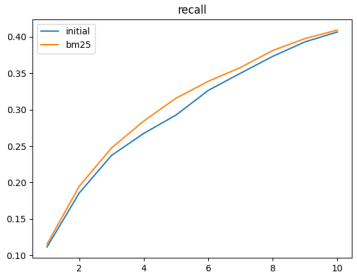
where $TF(t,d)$ is the number of times term t occurs in document d, $k1$ and b are parameters, $dl(d)$ is the length of document d, adl is the average document length, df_t is the number of documents in which term t occurs. This is an improvement in two ways.

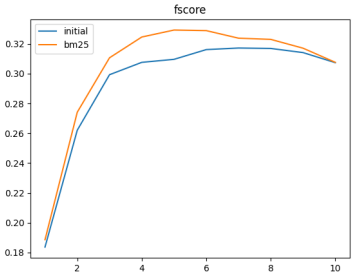
- **Term Saturation**:
  If a document contains 200 occurrences of a word say "elephant", it does not mean that it is *twice* as relevant as a document containing 'elephant' 100 times. We could argue that if "elephant" occurs a large enough number of times, say 100, the document is almost certainly relevant, and any further mentions don't really increase the likelihood of relevance. In other words, once a document is saturated with occurrences of a term, more occurrences shouldn't have a significant impact on the score. BM25 controls the TF term this way by saturating it as $\frac{TF}{TF+K}$ where K is a constant that depends on the parameters, as shown in the formula previously.
- **Document Length**:
  If a document happens to be really short and it contains "elephant" once, that's a good indicator that it is important to the content. But if the document is very long and it mentions "elephant" only once, the document is probably not about elephants. So we'd like to reward matches in short documents while penalizing matches in long documents. This is done when BM25 takes into account the document length and average document length.

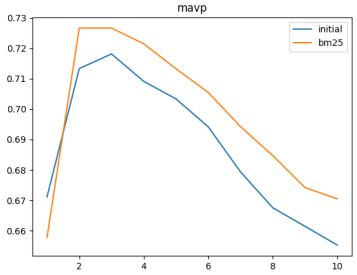
  The IDF is calculated differently from usual, and is called a 'probabilistic IDF'. Its practical use is that it drops the IDF for stop words like 'the' that may be present in 98% of the documents, without ignoring important non-stop words that may occur in say 70% of the document. [4]
  We have replaced the TF-IDF measures in the Vector Space Model with the BM-25 measure with parameters $k1 = 2$ and $b = 0.9$. Notes on hyperparameter tuning can be found in section XII.

B. Results









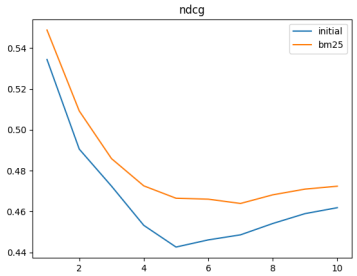


Figure 2. Comparison of VSM with TF-IDF vs BM25, Evaluation Metrics plotted across ranks

The results of the hypothesis tests at rank 5, and averaged over all ranks, are also given in Figure 3.

```
Testing at rank = 5
precision:
P: 0.026. Reject the null hypothesis at the 95.0% confidence level.
recall:
P: 0.001. Reject the null hypothesis at the 95.0% confidence level.
fscore:
P: 0.002. Reject the null hypothesis at the 95.0% confidence level.
mavp:
P: 0.217. Fail to reject the null hypothesis at the 95.0% confidence level.
ndcg:
P: 0.001. Reject the null hypothesis at the 95.0% confidence level.


Testing over values averaged over all ranks
precision:
P: 0.116. Fail to reject the null hypothesis at the 95.0% confidence level.
recall:
P: 0.021. Reject the null hypothesis at the 95.0% confidence level.
fscore:
P: 0.02. Reject the null hypothesis at the 95.0% confidence level.
mavp:
P: 0.207. Fail to reject the null hypothesis at the 95.0% confidence level.
ndcg:
P: 0.014. Reject the null hypothesis at the 95.0% confidence level.
keerthanasenthil@Keerthanas-MacBook-Pro project %
```

Figure 3. Results of hypotheses tests of Initial VSM vs. BM25 Models

It is clear that the BM25 measure is an improvement over TF-IDF.

VI. Latent Semantic Analysis

A. The Model

Latent Semantic Analysis is an approach to factorise the word-document matrix into three matrices multiplied by each other, using Singular Value Decomposition (SVD) [5]. The main idea behind this approach is to capture and represent words and documents as a combination of abstract *underlying* concepts. Mathematically, SVD decomposes the tf-idf matrix $A_{w\times d}$ (w here is words in the vocabulary, and d here is the documents in the corpus) as below:

$$A_{w\times d} = U_{w\times c}\Sigma_{c\times c}\left(V_{d\times c}\right)^T$$

Each row of the U matrix captures words as a linear combination of concepts and each row of the V matrix captures documents in the concept space. The Σ matrix is a diagonal matrix that captures the variance in the data. This decomposition can be treated as similar to eigenvalue decomposition but done for a non-square rectangular matrix.

Once we've obtained this decomposition, by selecting the highest k eigenvalues in Σ, we can obtain a transformation matrix expressing a set of k features that are mathematically most relevant ones as follows:

$$T_{w\times k} = U_{w\times k}\Sigma_{k\times k}$$

With this matrix, any document in the corpus or new input query once expressed as a vector v_i from tf-idf formulation, can be represented as a compressed vector v_o by applying

$$v_o = v_i^T T$$

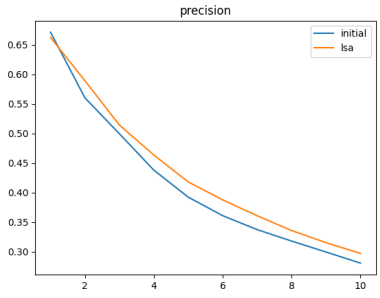
The above approach was implemented in our project using the scikit-learn library with the in-built function *Truncated-SVD* [6]. This is mainly because the library computes the decomposition in an optimised manner for large tf-idf matrices which are fairly sparse in nature. Mathematically, the variance in the data caught by choosing k features is the ratio sum of the first k eigenvalues in Σ with the sum of all eigenvalues. We chose to capture 80% of the variance by reducing the feature space from 6329 to 475.

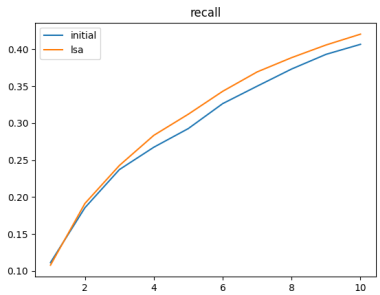
The key advantages of using this approach are:

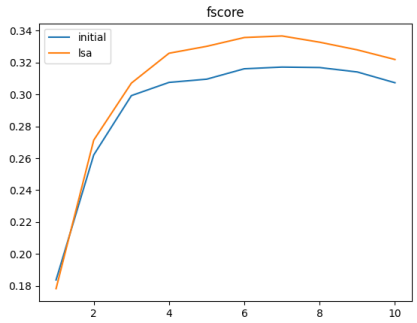
- When words are expressed as derived concepts, they are no longer orthogonal and seem to capture notions of similarity and dissimilarity.
- Synonymous words can be identified by estimating *closeness* of two words in the concept space.
- Similar documents are those associated with similar concepts in the concept space - the problem of circularity has been addressed.
- Data compression- the earlier stored documents in thousands of features are expressed in 475. This reduces space occupied tremendously and speeds up information retrieval as well.

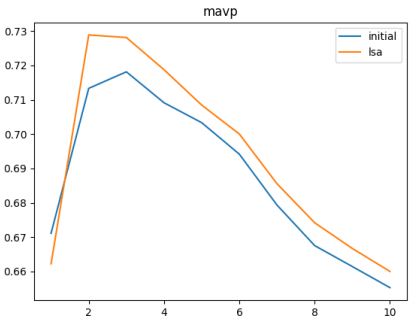
When the initial VSM model's feature space was replaced by Latent Semantic Indexing performed on the TF-IDF document term matrices, we got the following results.

B. Results









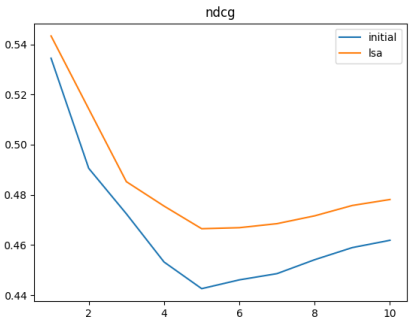


Figure 4. Comparison of VSM with TF-IDF vs LSA, Evaluation Metrics plotted across ranks

The graphs show that our LSA-based model outperforms the initial VSM on each metric. Subsequently, The results of the hypothesis tests at rank 5, averaged over all ranks, are also given in Figure 9.

```
Testing at rank = 5
precision:
P: 0.001. Reject the null hypothesis at the 95.0% confidence level.
recall:
P: 0.003. Reject the null hypothesis at the 95.0% confidence level.
fscore:
P: 0.001. Reject the null hypothesis at the 95.0% confidence level.
mavp:
P: 0.334. Fail to reject the null hypothesis at the 95.0% confidence level.
ndcg:
P: 0.001. Reject the null hypothesis at the 95.0% confidence level.


Testing over values averaged over all ranks
precision:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
recall:
P: 0.002. Reject the null hypothesis at the 95.0% confidence level.
fscore:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
mavp:
P: 0.301. Fail to reject the null hypothesis at the 95.0% confidence level.
ndcg:
P: 0.007. Reject the null hypothesis at the 95.0% confidence level.
```

Figure 5. Results of hypotheses tests of Initial VSM vs. LSA Models

VII. Document Title Incorporation

A. The Model

The initial VSM model looks purely at the content in the document to generate a tf-idf matrix without laying any emphasis on the title of the documents. The title is included once as a sentence in the body, by default in the Cranfield dataset. But this is not sufficient to capture its importance to the document. This is evident in the below-highlighted case

```
Enter query below
dynamics of a dissociating gas

Top five document IDs :
317
110
656
625
167
keerthanasenthil@Keerthanas-MacBook-Pro initial %
```

Figure 6. Example of title being disregarded

Here, document 110's title is "dynamics of a dissociating gas", and document 317's title is "inviscid hypersonic flow past blunt bodies". We would naturally prefer that 110 was returned at a higher rank than 317, but that is not the case in the system. This proves that the IR system ignores the document's central concept, evinced by the title.

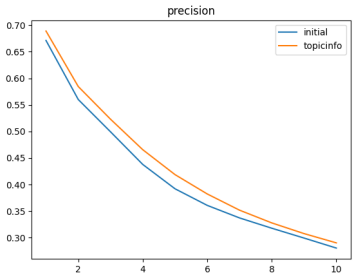
We incorporate the titles in the tf-idf index we created by giving extra weight to words in the title. We denoted this weight given to the title with the variable *title_k*. The document vector representation in the TF-IDF term-document matrix is updated as follows:

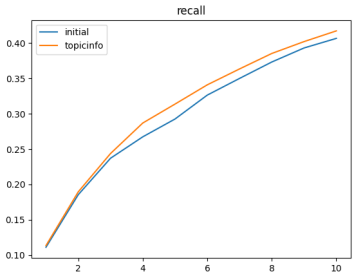
$$vec_{doc} = vec_{doc} + title_k \times vec_{title}$$

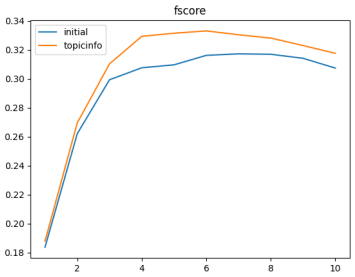
Where k can take floating point values as well, this modification has been done to the initial VSM model, to its term-

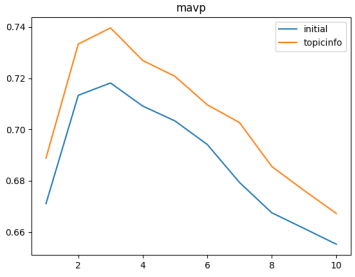
document TF-IDF vectors. After tuning parameters, we arrived at $k = 1.75$ which worked reasonably well with this corpus and queries.

B. Results









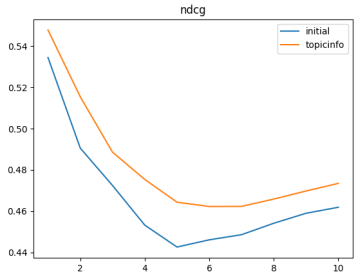


Figure 7. Comparison of the initial VSM with topic inclusion, Evaluation Metrics plotted across ranks

The graphs show that including titles improves the model performance across metrics. Subsequently, The results of the hypothesis tests at rank 5, averaged over all ranks, are also given in Figure 8.

```
Testing at rank = 5
precision:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
recall:
P: 0.001. Reject the null hypothesis at the 95.0% confidence level.
fscore:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
mavp:
P: 0.096. Fail to reject the null hypothesis at the 95.0% confidence level.
ndcg:
P: 0.002. Reject the null hypothesis at the 95.0% confidence level.


Testing over values averaged over all ranks
precision:
P: 0.001. Reject the null hypothesis at the 95.0% confidence level.
recall:
P: 0.002. Reject the null hypothesis at the 95.0% confidence level.
fscore:
P: 0.001. Reject the null hypothesis at the 95.0% confidence level.
mavp:
P: 0.076. Fail to reject the null hypothesis at the 95.0% confidence level.
ndcg:
P: 0.012. Reject the null hypothesis at the 95.0% confidence level.
```

Figure 8. Results of hypotheses tests of Initial VSM vs. Title Inclusive Models

Additionally, the error that occurred in the example shown previously has been rectified.

```
Enter query below
dynamics of a dissociating gas

Top five document IDs :
110
167
317
656
625
keerthanasenthil@Keerthanas-MacBook-Pro topicinfo %
```

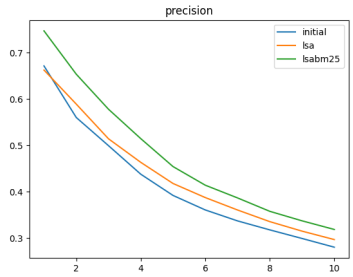
Figure 9. More relevant document retrieved first

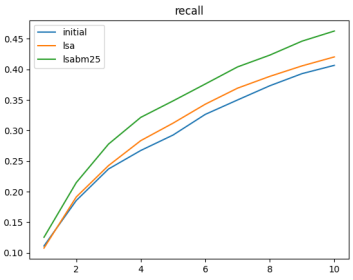
Since the previous three models have independently performed much better than the primitive TF-IDF VSM model, we attempt to combine them for better results.

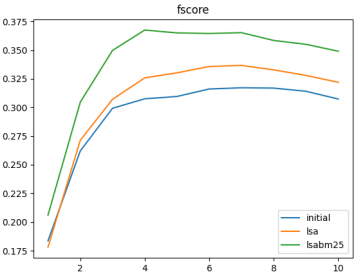
VIII. Combined Model: BM25+LSA

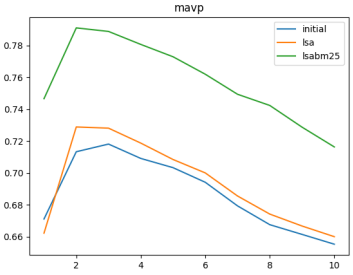
In this model, we have used the BM25 measures to find the term-document relevance measures and performed latent semantic indexing upon the thus obtained term-document matrix. Considering the methods individually seemed to improve document performance, we expected it to boost performance while combining them in one model.

A. Results









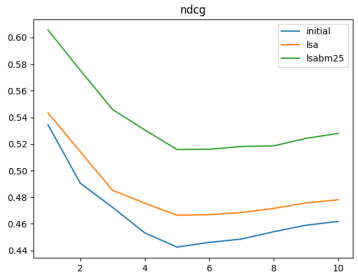


Figure 10. Comparison of the initial VSM, LSA and LSA+BM25, Evaluation Metrics plotted across ranks

The graphs show that the combined model outperforms the initial VSM model. Subsequently, to statistically compare its performance with the initial model, the results of the hypothesis tests at rank 5, averaged over all ranks, are also given in Figure 11.

```
Testing at rank = 5
precision:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
recall:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
fscore:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
mavp:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
ndcg:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.

Testing over values averaged over all ranks
precision:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
recall:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
fscore:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
mavp:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
ndcg:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
```

```
Testing at rank = 5
precision:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
recall:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
fscore:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
mavp:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
ndcg:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.

Testing over values averaged over all ranks
precision:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
recall:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
fscore:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
mavp:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
ndcg:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
```

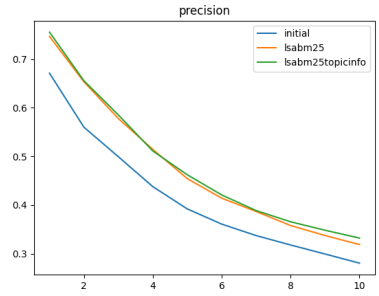
Figure 11. i) Results of hypotheses tests while comparing Initial model with the combined model LSA+BM25 ii) Results of hypotheses tests while comparing lsa and LSA+BM25

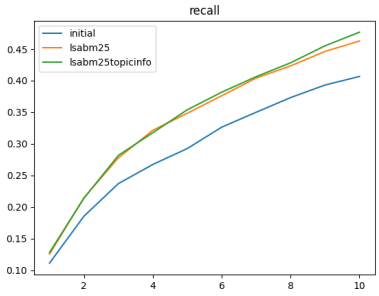
Since the Combined LSA+BM25 model is a clear winner over the initial VSM, we also check to see if it is significantly better than the plain LSA model. Figure 11 also shows the hypothesis tests where the first model is LSA and the second is LSA+BM25. The hyperparameters used for this model are $k1 = 1.95$, $b = 0.8$, and the number of components in LSA are 350.

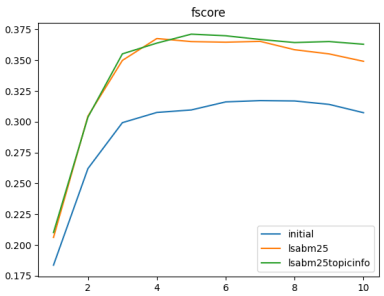
IX. Combined Model: BM25+LSA+Titles

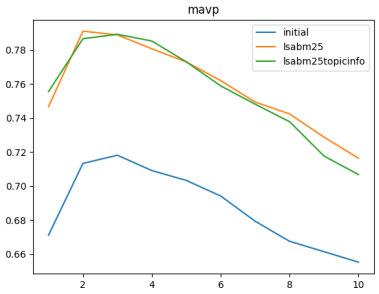
In this model, we have combined the previous model with the incorporations of title information more than once, as shown in section VII. This is done by adding the title vector k times to the document vectors in the BM-25 term-document matrix.

A. Results









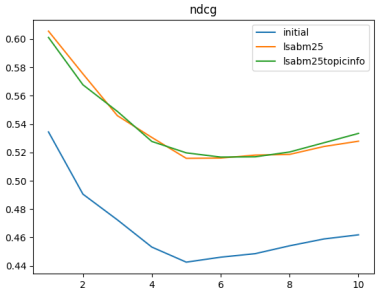


Figure 12. Comparison of the initial VSM with lsa_bm25_topic, Evaluation Metrics plotted across ranks

The results of the hypothesis tests at rank 5, averaged over all ranks, to determine if the additional topic inclusion improved the combined model's performance are given in Figure 13.

```
Testing at rank = 5
precision:
P: 0.085. Fail to reject the null hypothesis at the 95.0% confidence level.
recall:
P: 0.098. Fail to reject the null hypothesis at the 95.0% confidence level.
fscore:
P: 0.086. Fail to reject the null hypothesis at the 95.0% confidence level.
mavp:
P: 0.496. Fail to reject the null hypothesis at the 95.0% confidence level.
ndcg:
P: 0.204. Fail to reject the null hypothesis at the 95.0% confidence level.


Testing over values averaged over all ranks
precision:
P: 0.062. Fail to reject the null hypothesis at the 95.0% confidence level.
recall:
P: 0.08. Fail to reject the null hypothesis at the 95.0% confidence level.
fscore:
P: 0.049. Reject the null hypothesis at the 95.0% confidence level.
mavp:
P: 0.596. Fail to reject the null hypothesis at the 95.0% confidence level.
ndcg:
P: 0.491. Fail to reject the null hypothesis at the 95.0% confidence level.
```

Figure 13. Results of hypotheses tests while comparing LSA+BM25 and LSA+BM25+TopicInfo

The addition of title information to the LSA+BM25 seems to falter. If we consider the 90% confidence interval of hypothesis testing, however:

```
Testing at rank = 5
precision:
P: 0.085. Reject the null hypothesis at the 90.0% confidence level.
recall:
P: 0.098. Reject the null hypothesis at the 90.0% confidence level.
fscore:
P: 0.086. Reject the null hypothesis at the 90.0% confidence level.
mavp:
P: 0.496. Fail to reject the null hypothesis at the 90.0% confidence level.
ndcg:
P: 0.204. Fail to reject the null hypothesis at the 90.0% confidence level.


Testing over values averaged over all ranks
precision:
P: 0.062. Reject the null hypothesis at the 90.0% confidence level.
recall:
P: 0.08. Reject the null hypothesis at the 90.0% confidence level.
fscore:
P: 0.049. Reject the null hypothesis at the 90.0% confidence level.
mavp:
P: 0.596. Fail to reject the null hypothesis at the 90.0% confidence level.
ndcg:
P: 0.491. Fail to reject the null hypothesis at the 90.0% confidence level.
```

Figure 14. Results of hypotheses tests while comparing LSA+BM25 and LSA+BM25+TopicInfo

So we can say that it does improve the performance for the better, but not significantly enough for a 95% confidence interval. This can be avoided by using finer parameter tuning methods as we will describe in section XII. Since it makes meaningful sense to emphasise title information, we retain this feature in our model.

The parameters used were $k1 = 1.75$, $b = 0.9$, $title_k = 1.75$ and the number of components in LSA are 350.

X. Spellcheck on queries

To perform spelling checks on the corpus, we utilised the *pyspellchecker* [7] library to identify candidate words and choose the most effective one. Pure Python Spell Checking is based on Peter Norvig's article [8] that implements a simple spell checker using Bayesian inferencing and the noisy channel model. Key features of this library and its methods are:

- 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.
- Additionally, it supports multiple languages including English, Spanish, German, French, Portuguese, Arabic and Basque. This feature can be handy for Information Retrieval systems that need machine translation to search across an array of documents in different languages.

Spell-checking the entire corpus including documents and queries resulted in a huge time overhead, without compensatory improvements in performance, since the Cranfield dataset does not seem to have spelling errors. We have hence implemented the spell check feature for handling custom queries. An example is as shown in Figure 15:

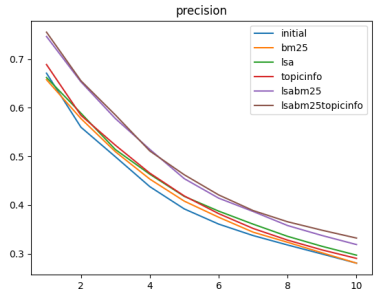
```
Enter query below
Papers on Airodynamics

Query after correcting any spelling errors:
Papers on aerodynamics

Top five document IDs :
925
137
1066
1379
860
```

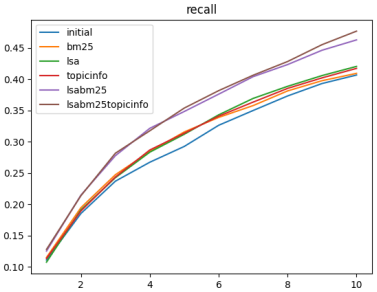
```
Enter query below
fluyd flow in airplains

Query after correcting any spelling errors:
fluid flow in airplane

Top five document IDs :
783
807
389
810
806
```

Figure 15. Spellcheck on queries

XI. Final Results

All in all, the plots in Figure 18 showcase how all the models discussed so far perform against each other.





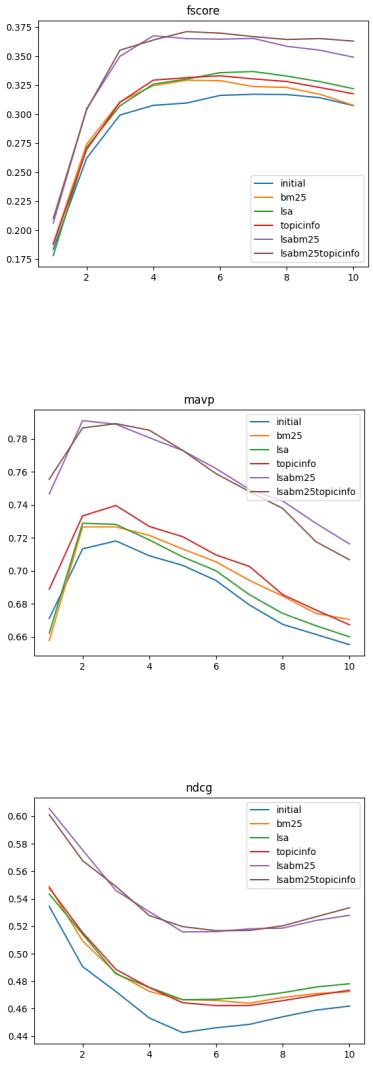
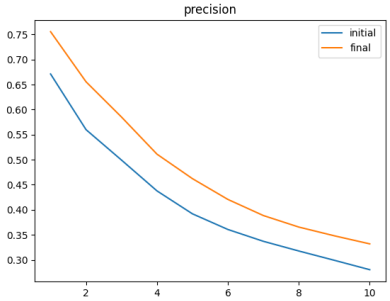


Figure 16. Comparison of all models assessed so far

So our **final, chosen model** takes the BM-25 representation of the term-document matrix, adds the title vectors to the document vectors the desired number of times, performs Latent Semantic Indexing on it, and also checks the spellings of custom queries. The improvement over the initial model is as shown.



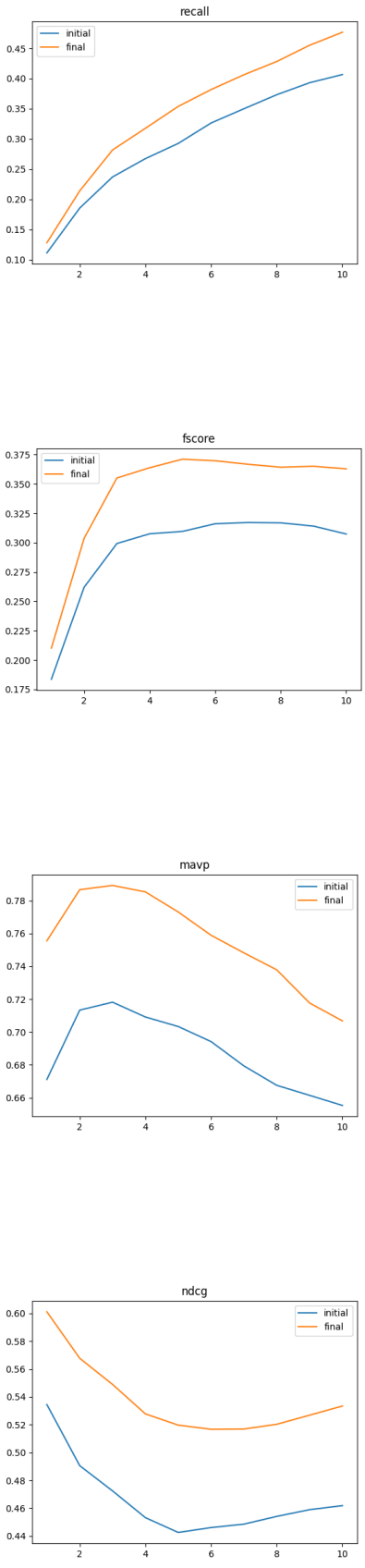


Figure 17. Performance of final model

```
Testing at rank = 5
precision:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
recall:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
fscore:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
mavp:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
ndcg:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.


Testing over values averaged over all ranks
precision:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
recall:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
fscore:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
mavp:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
ndcg:
P: 0.0. Reject the null hypothesis at the 95.0% confidence level.
```

Figure 18. Hypothesis tests on final model

XII. Hyperparameter tuning

Hyperparameters of the models were:

- $k1$ of BM25 document representations
- b of BM25 document representations
- number of components of Latent Semantic Indexing
- weight given to title vector in the document

We used the Optuna [9] framework to search for the optimal set of hyperparameters for each model, over a user-defined search space. It uses a technique called Bayesian optimization to efficiently search the hyperparameter space. Bayesian optimization builds a probabilistic model of the objective function based on the results of previous trials and uses this model to decide which hyperparameters to try next. Since the relevance scores of documents for each query are available to us, we defined the objective function to be mean nDCG score averaged over all ranks.

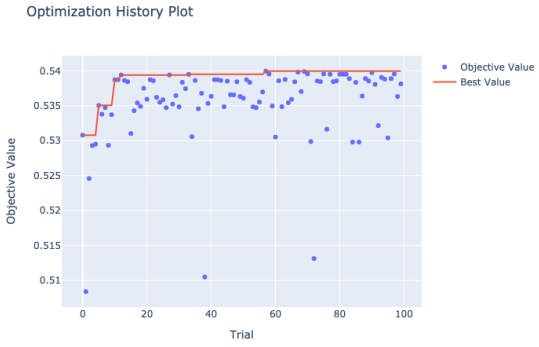


Figure 19. Optimisation History of Hyperparameter Tuning

It is possible to find better hyperparameters by defining a finer and larger search space (we stuck to the recommended values of 100-500 for LSA components, and incremented $title_k$ in steps of 0.25, for instance). This could've contributed to improving the performance of the model. The search for hyperparameters could also have been sped up by harnessing parallelisation of code, which Optuna offers.

XIII. Conclusion

We have thus identified the shortcomings of the Vector Space Model of Information Retrieval and taken suitable steps to address it using methods such as Best Matching 25 ranking of documents and Latent Semantic Indexing. Appropriate hypotheses were devised and used to test the results of every model. The final model which includes emphasis on titles as well as a spell checking feature has been presented.

References

[1] Donna Harman. Overview of the third text retrieval conference (trec-3). *NIST special publication*, 500-226:1–20, 1992.
[2] Paired t-tests. https://www.statstutor.ac.uk/resources/uploaded/paired-t-test.pdf.
[3] T-test on related samples of scores. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_rel.html.
[4] KMW LLC. Understanding tf-idf and bm25. https://kmwllc.com/index.php/2020/03/20/understanding-tf-idf-and-bm-25/, 2020.
[5] Navdeep Uppal. Truncated singular value decomposition (svd) using amazon food reviews, January 2021.
[6] Truncated svd. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html.
[7] Lucas Ou-Yang. pyspellchecker. https://pypi.org/project/pyspellchecker/, 2021.
[8] Peter Norvig. How to write a spelling corrector. *norvig.com*, 2011.
[9] Optuna. https://optuna.org/.