CS6370 - Project - IR system

Aswin Balamurugan, Tushar Bhutada, Ashwin Krishna

Indian Institute of Technology Madras, Guindy, Chennai, 600036, India.

Contributing authors: ch20b018@smail.iitm.ac.in; ch20b025@smail.iitm.ac.in; ch20b017@smail.iitm.ac.in;

Abstract

This paper covers the problems faced in a basic information retrieval system and ways to tackle them. We start with a baseline Vector Space Model (VSM) method to create the search engine. Multiple preprocessing techniques were tried and implemented to find the most optimum for our case. The search engine was modified with different models like LSA and spell-check. The Cranfield dataset was used to evaluate the search engine. Additionally, some manual queries were used to gauge and tune the performance of the search engine. We have used hypothesis testing to validate the improvement in the search engine.

Keywords: IR system, Vector Space Model, Pre-processing, LSA, Evaluation

1 Introduction

In this report, we will be going through the details of the IR system that we build. All the relevant methods used, all the pre-processing done, evaluation used and performed will be discussed in detail. The evaluation is primarily done over the Cranfield data provided, but we also used certain custom queries to realize the improvement as well as the drawbacks. We, in general, used the paired t-test to realize improvements from the previous model by estimating the p-value over n-DCG values calculated over the queries. We also use plots then and there to show the improvement visually. The IR system works on the basis of a vector space model so we will look at that first.

2 Vector space Model (VSM)

The Vector Space Model (VSM) is a framework used in Natural Language Processing to represent text data as vectors in a multi-dimensional space. In this model, each document and query is represented as a vector, with each dimension in the vector corresponding to a term or word in the document. The value of the vector in each dimension is typically a weight that reflects the importance of that term in the document, and we use tf-IDF values in this case. During retrieval, the cosine similarity between the query and document is used for retrieving and ranking the documents.

3 Features of the previous model

3.1 Pre-Processing

3.1.1 Sentence segmentation and Tokenization

In order to use the vector space model, we have to pre-process the documents and queries, which usually are a bunch of sentences. First, the discourse is split into sentences then the sentences are themselves split into words. The first one is done when we encounter a punctuation mark such as a full stop or question mark. The second one is done based on white spaces as well as other punctuation and symbols. We also convert all the words to lower cases for ease of processing.

3.1.2 Stop Word Removal

In a query or a document, not all words are important and carry meaning. These words include words like articles, prepositions, and conjunctions. These words are known as stop words, and they include words such as "the", "and", "a", "an", "in", "of", etc. The reason for removing stop words is that they can add noise to the data and make it more difficult to extract meaningful patterns and insights from the text. There are multiple ways to do it, but we here use a pre-defined stop words list to identify and remove stop words.

3.1.3 Stemming and Lemmatization

A drawback of the VSM is that the word forms should be exact so that it retrieves better data. So we use stemming and lemmatization to counter this. The difference between stemming and lemmatization is that stemming just removes prefixes and suffixes of the root form, while lemmatization changes it to the correct word. So in general, lemmatization is preferred, and we use the same here as well.

3.2 Working of IR system based on VSM

After all the pre-processing, we get all the words in the form of tokens, using which we calculate the tf-Idf values of each word in a document and query. TF-IDF frequency essentially consists of two terms, TF and IDF. TF known as the term frequency, is essentially the no of times a particular type/word appears in a document. IDF known as the inverse document frequency, is a measure of how discriminating a particular term is. It is given by

$$\text{idf} = \log_2 \left(\frac{N}{n} \right)$$

Here N = The total no of docs and n = No of docs with the particular type
Now for the TF-IDF values we have to multiply frequency (TF) and the IDF of each term

$$\text{tf-idf} = \text{frequency} \times \text{idf}$$

We do it for all the terms in documents and find the representation in terms of all the available words. If the word is not in a document, its tf value is zero, thus the idf-value turns out to be zero. Thus we get a vector representation of all the documents and queries in terms of words.

To retrieve the relevant documents to a query, we find the cosine similarity of the query with all the available documents and return the top documents with the highest values. The time it takes to perform the IR process is around 16 seconds, including the evaluation process.

3.3 Evaluation

The performance is evaluated by checking the following measures for the quality of the retrieved documents:

– **Precision@K**: Precision refers to the fraction of retrieved documents that are relevant to the query. Precision@K is evaluated at a given cut-off rank, considering only the topmost results returned by the system.

– **Recall**: Recall is the fraction of the relevant documents that are successfully retrieved. Similar to precision, recall@k is evaluated at a given cut-off rank.

– **F-Score**: It is the harmonic mean of precision and recall, hence tbalances both the measures.

- **MAP**: Mean Average Precision provides a single-figure measure of quality across recall levels. For a single information need, Average Precision is the average of the precision value obtained for the set of top k documents existing after each relevant document is retrieved, and this value is then averaged over information needs. That is, if the set of relevant documents for an information need $q_j \in Q$ is $d_1, ...d_{m_j}$ and R_{j_k} is the set of ranked retrieval results from the top result until you get to document d_k, then

$$\text{MAP}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(R_{j_k}) \tag{1}$$

- **nDCG**: NDCG (Normalized Discounted Cumulative Gain) is a metric used to evaluate the effectiveness of ranking algorithms. The basic idea behind NDCG is to measure the quality of a ranking of items by comparing it to an ideal ranking. The ideal ranking(IDCG) is obtained by assuming that the relevance of each item to the user is known and sorting the items in decreasing order of relevance. Like precision at k, it is evaluated over some number k of top search results. For a set of queries Q, let $R(j, d)$ be the relevance score assessors gave to document d for query j. Then

$$\text{DCG}(Q, k) = \sum_{m=1}^{k} \frac{R(j, m)}{\log_2(1 + m)} \tag{2}$$

$$\text{NDCG} = \frac{DCG_k}{IDCG_k} \tag{3}$$

4 Limtations

4.1 Limitations of VSM

1. It assumes independence between terms: The Vector Space Model assumes that each term in a document is independent of the others, which is not always true in natural language. For example, the meaning of a sentence can depend on the order and relationship between its words, which the Vector Space Model does not capture.
2. It requires the same terms as in the documents to be used. Although we counter word forms using lemmatization, completely different synonyms of words will give poor results if used in a query.
3. It is computationally more expensive. Since we deal with a lot of words, around 145000 words to be precise here. Each document is being represented in terms of 145000 entities and calculating cosine similarity with each document takes a lot of time.
4. Polysemy is not handled at all. All polysemous words are treated equally here with no consideration of their context.

4.2 Limitations to be addressed

1. Independence of terms and using related word forms in the documents and query
2. Use of incorrect words and spellings in the query
3. Computation time of Quering over documents
4. Use of polysemous words in the documents and query

5 Performance of the model

To evaluate the performance, we primarily use the MAP and nDCG scores calculated over the queries in the Cranfield dataset for the top ten documents retrieved by each query. We also calculate precision, recall and F-score, although we don't use that much. The results are given below.

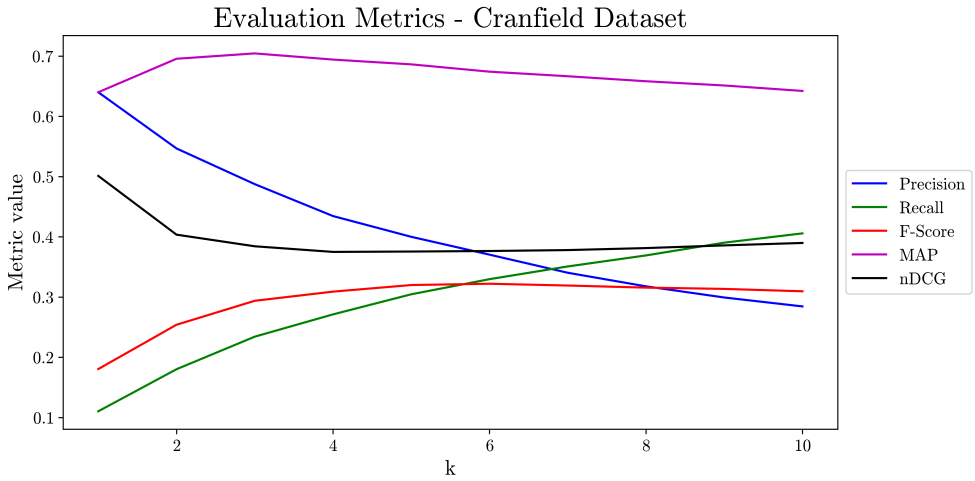

Fig. 1 Performance: VSM Model

k	Precision	Recall	F-score	MAP	nDCG
1	0.6400	0.1104	0.1805	0.6400	0.5011
2	0.5467	0.1802	0.2540	0.6956	0.4036
3	0.4874	0.2342	0.2938	0.7044	0.3843
4	0.4344	0.2712	0.3090	0.6942	0.3749
5	0.4000	0.3046	0.3199	0.6863	0.3755
6	0.3704	0.3297	0.3221	0.6741	0.3764
7	0.3403	0.3508	0.3191	0.6666	0.3780
8	0.3178	0.3691	0.3156	0.6582	0.3813
9	0.2993	0.3903	0.3135	0.6512	0.3857
10	0.2844	0.4057	0.3096	0.6421	0.3898

Table 1 Evaluation Metrics

5.1 Observations

1. As expected the precision and recall decrease and increase respectively as the k value increase
2. F1 score increases and seems to stabilize around k = 3 and almost remains almost constant
3. MAP value is high overall and tends to increase and then decrease to stabilise
4. nDCG values are on the lower side, and they tend to decrease and then increase to a constant value

6 Improving the model

6.1 Latent Semantic Analysis

A Naive, term-frequency-based information retrieval system works poorly because it tries to find word-for-word matches between queries and documents. Hence, it depends totally on the morphology of the words. It can not handle synonymy(different words with similar meanings) and polysemy(same words with different meanings).

LSA deals with these issues by understanding the underlying semantic structure of the text. It does so by constructing a "semantic" space where terms and documents that are closely associated are placed near one another. It is based on Singular-value decomposition which captures the important patterns

in the term-doc space and rejects the unimportant ones. As a result, terms that did not appear in a document may still end up close to the document.

$$X_o = T_o S_o D_o' \tag{4}$$
$$X \approx TSD' \tag{5}$$
$$Q \approx QT \tag{6}$$

The term-doc matrix can be expressed according to the equation (4) using SVD. When only a specific number of singular values are selected from the S_o matrix, we get the compressed version of X_o, X. This compressed version of the term-doc matrix has weights rearranged compared to the original matrix, and this rearrangement is according to the semantic relations between terms and documents and eliminates the unimportant concepts.

The similarity between the two terms can be analyzed by the similarity(cosine) between corresponding rows of the matrix X. This is equivalent to taking the dot product between the i and j rows of the matrix TS (Term- Term space). Similarly for two documents, one can take the dot product between the i and j rows of the matrix DS (Doc-Doc space). But for IR system application we need Term-Doc similarity.

To get an estimate of the term-doc similarity a query is first represented in term-vector. It is then converted to a compressed vector according to the equation (6). Its cosine similarity is then checked with each of the doc vectors.

6.1.1 Performance

The evaluation metrics are listed in the table. One can observe a significant increase in the metrics of the model using LSA compared to the initial one. The improvement is validated by the paired t-test performed on the evaluation metrics obtained before and after the LSA.

k	Prev - Map	Prev - nDCG	MAP	nDCG
1	0.6400	0.5011	0.6667	0.5344
2	0.6956	0.4036	0.7111	0.4160
3	0.7044	0.3843	0.7167	0.3937
4	0.6942	0.3749	0.7111	0.3929
5	0.6863	0.3755	0.7001	0.3901
6	0.6741	0.3764	0.6906	0.3916
7	0.6666	0.3780	0.6818	0.3942
8	0.6582	0.3813	0.6764	0.3984
9	0.6512	0.3857	0.6667	0.4033
10	0.6421	0.3898	0.6601	0.4042

Table 2 Performance Comparison of LSA

The Null hypothesis (Ho): The model is not improved, and the values of the performance metric nDCG are not much different.

The Alternate hypothesis (Ha): The model is different with improved values.

The mean of previous nDCG = 0.3951
The mean of new nDCG = 0.4119
d: Difference per paired value
n: Number of samples = 10

$$t = \frac{\Sigma d}{\sqrt{\frac{n(\Sigma d^2) - (\Sigma d)^2}{n-1}}}.$$

$$t = 8.3861$$

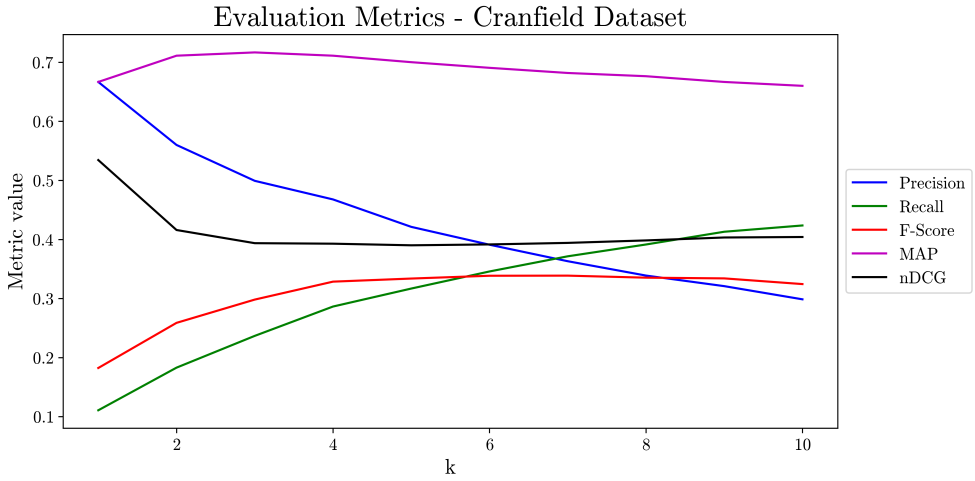

Fig. 2 Performance: LSA Model

$$Degree\ of\ freedom = 10 - 1 = 9$$

$$two\ tails\ p - value = 1.5 \times 10^{-5} < 0.05$$

The p-value is way less than 0.05, so we reject the null hypothesis that they are equal and conclude that they are significantly different. This means the new nDCG is better than the old one and LSA has improved the system drastically.

6.1.2 Limitations of LSA

As SVD tries to find 'similar' words by the occurrence pattern in documents, it handles synonymy impressively. At the same time, as it assumes that words have a single meaning, it performs poorly when dealing with words that have multiple meanings (polysemy). Take this query for an example.

What is the potential of aerodynamic study

- 682: the lift of twisted and cambered wings in supersonic flow.
- 814: stability derivatives of cones at supersonic speeds.
- 1098: an experimental investigation of ablating material at low and high enthalpy potentials.
- 363: n alternative formulation of the problem of flutter in real fluids.
- 1383: on the theory of laminar boundary layer involving separation.

What is the impact of aerodynamic study

- 183: properties of impact pressure probes in free molecule flow.
- 010:the theory of the impact tube at low pressure.
- 1066: wind tunnel measurements of aerodynamic damping derivatives of a launch vehicle vibrating in free-free bending modes at mach numbers from 0. 70 to 2. 87 and comparisons with theory.
- 719: the theory of the impact tube at low pressure.
- 36: supersonic flow around blunt bodies.

Both queries are similar and mean the same thing, but the term potential in query 1 is polysemous and is mapped wrongly; hence though both queries should retrieve the same thing, they retrieve completely different documents.

6.2 Assigning Different weights

Information retrieval systems are designed to match the user's search queries with the documents that are most relevant to them. A good information retrieval system should be able to return relevant documents when queries are about the author/title of the document or somehow related to it. To do so we have added the corresponding strings in the documents.

The dataset already had the title of the documents incorporated at the start/end of the document. So we adjusted the weight given to the title strings in the term frequency matrix to reflect their relative importance compared to the other text in the document. Through careful tuning via grid search, we were able to identify the optimum weight that would yield the most effective results. However, we found that the author information was not included in the dataset. To address this, we added the author's name to each document so that it would be included as part of the text. Although the evaluation metrics showed very minor changes, these changes significantly improved the performance when queries were based on the author's name or the title of the document.

The parameters: weights for the title and author strings as well as the number of components retained in the SVD were tuned using gridsearch. The best performance was obtained for 400 retained components and (2.0,2.0) weights for the title and author respectively. The following plot shows the results obtained after gridsearch. One can clearly see that nDCG is maximum for $K = 400$. The loops in the plot are due to iterations over different weights for title, for a particular K.

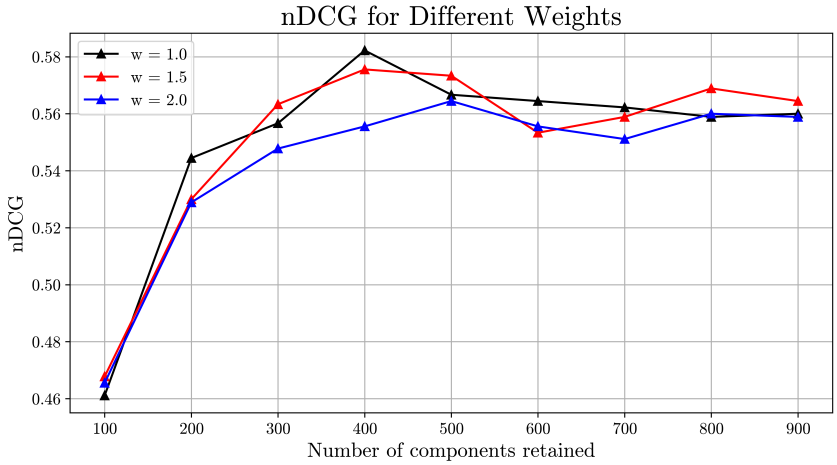

Fig. 3 Grid search results for LSA with weights

6.2.1 Performance

We checked the scores of the system(having LSA) after implementing these weights for the title and the author. The following table shows the improvement in the scores compared to the system having LSA.

k	Prev - Map	Prev - nDCG	MAP	nDCG
1	0.6667	0.5344	0.6667	0.5467
2	0.7111	0.4160	0.7156	0.4306
3	0.7167	0.3937	0.7170	0.4062
4	0.7111	0.3929	0.7132	0.3937
5	0.7001	0.3901	0.7098	0.3944
6	0.6906	0.3916	0.6951	0.3967
7	0.6818	0.3942	0.6851	0.4023
8	0.6764	0.3984	0.6751	0.4078
9	0.6667	0.4033	0.6572	0.4146
10	0.6601	0.4042	0.6519	0.4163

Table 3 Performance Comparison after incorporating title and author

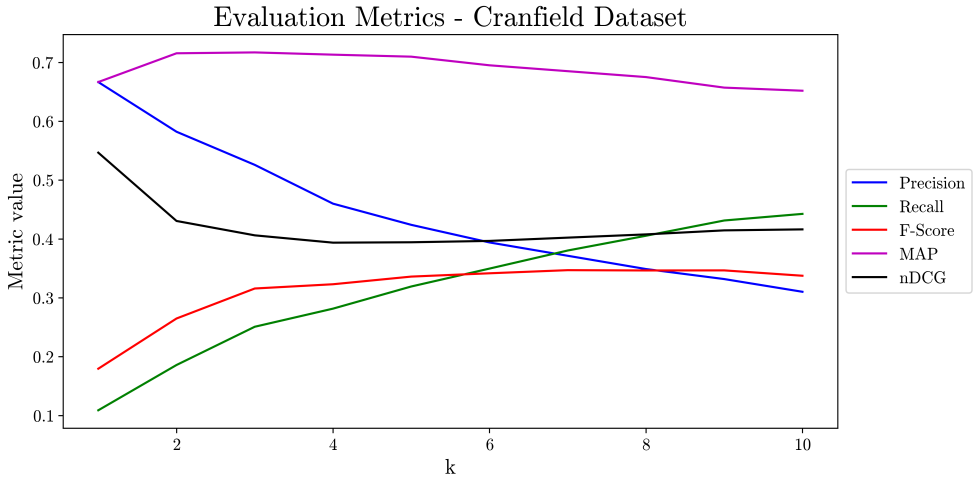

Fig. 4 Performance: LSA and Title+Author weights

The Null hypothesis (Ho): the values of the performance metric nDCG have not changed.
The Alternate hypothesis (Ha): The model is different with improved values.

The mean of previous nDCG = 0.4119
The mean of new nDCG = 0.4209
d: Difference per paired value
n: Number of samples = 10

$$t = \frac{\Sigma d}{\sqrt{\frac{n(\Sigma d^2)-(\Sigma d)^2}{n-1}}}.$$

$$t = 6.4922$$

$$Degree\ of\ freedom = 10 - 1 = 9$$

$$two\ tails\ p{-}value = 1.125{\times}10^{-4} < 0.05$$

The p-value is way less than 0.05, so we reject the null hypothesis that they are equal and conclude that they are significantly different. More importantly, the incorporation of title and author is even more fruitful in case of manual queries consisting title or name of the author of the document. For instance, when we pass an author name as a custom query: *libby*, we get the following top-5 relevant documents:

For the IR-system without weights for the author:

- 2: simple shear flow past a flat plate in an incompressible fluid of small viscosity.
- 3: the boundary layer in simple shear flow past a flat plate.
- 389: simple shear flow past a flat plate in a compressible viscous fluid.
- 324: vorticity effect on the stagnation point flow of a viscous incompressible fluid.
- 664: the boundary layer on a flat plate in a stream with uniform shear.

Which does not consist of any of the papers contributed by the author. Conversely, the IR-system with weights for the author retrieves 5 out of 5 papers contributed by the Libby. (ID's: 17, 84, 123, 1180, 1374).

- 84: experimental investigation of the downstream influence of stagnation point mass transfer.
- 1374: theoretical analysis of turbulent mixing of reactive gases with application to supersonic combustion of hydrogen.
- 123: the downstream influence of mass transfer at the nose of a slender cone.
- 1180: approximate analysis of the slot injection of a gas in laminar flow.

- 17: remarks on the eddy viscosity in compressible mixing flows.

6.3 Spell checker

A spellchecker is a tool that checks the spelling of words in a text document and suggests corrections for misspelt words. The basic operation of a spellchecker is to compare each word in a text document against a dictionary of correctly spelt words. If a word in the text document is not found in the dictionary, it is flagged as potentially misspelt, and the spellchecker suggests one or more possible correct spellings for the word. It gives suggestions based on techniques like Edit Distance, where the words with minimum number/distance of edits get suggested first. We have not built a spell-check model from scratch but have incorporated some pre-built ones in the project.

6.3.1 Spell Checker Model

The next form of improvement comes in the form of a spell checker. We are implementing these only on the queries and not document words. The words in the documents are assumed to be error-free. We have three levels of checking the errors. We do this because the spell checker that we use, Textblob, is not capable of handling scientific words.

The first level is to check if the words are actually in the document since the document is assumed to be free of error. If it isn't, we pass the word to the Textblob spell checker to correct it. But if it is a scientific word, the spell checker doesn't correct it and returns the incorrect word back, so comes the final layer, which is the enchant corrector. This enchant corrector has actually been fed with all the words in the Cranfield corpus. So if there is an incorrect word, it returns the closest match in the corpus based on edit distance.

6.3.2 Performance

The overall performance of the IR system, when tested using the Cranfield data set, doesn't improve by a lot as the number of erroneous words in the Queries listed out were very less. However, it shows significant retrieval accuracy when we use a mistyped custom query. The system without a spell checker lists out completely irrelevant documents when compared to the IR system with a spell checker. So we made a custom dataset of queries by slightly modifying the Query file in the Cranfield data set. A word was made erroneous in each of the top hundred queries to evaluate the performance. The modified query file is also attached for reference. The results are as below.

k	Prev - Map	Prev - nDCG	MAP	nDCG
1	0.6667	0.5467	0.7067	0.5833
2	0.7156	0.4306	0.7444	0.4488
3	0.7170	0.4062	0.7433	0.4227
4	0.7132	0.3937	0.7379	0.4083
5	0.7098	0.3944	0.7337	0.4079
6	0.6951	0.3967	0.7173	0.4111
7	0.6851	0.4023	0.7047	0.4143
8	0.6751	0.4078	0.6928	0.4215
9	0.6572	0.4146	0.6787	0.4282
10	0.6519	0.4163	0.6701	0.4300

Table 4 Performance Comparison of the spell checker

The overall performance improvement on the Cranfield data set is plotted below. The overall time of running the Cranfield data set is about 57 seconds which is slightly slower since we are using the spell check to check every word in the query.

To compare the performance with and without spell check, we use t-test to calculate the probability. This is done by using the mean of nDCG values with and without the spell checker.

The Null hypothesis (Ho): The model is not improved, and the values are not much different.

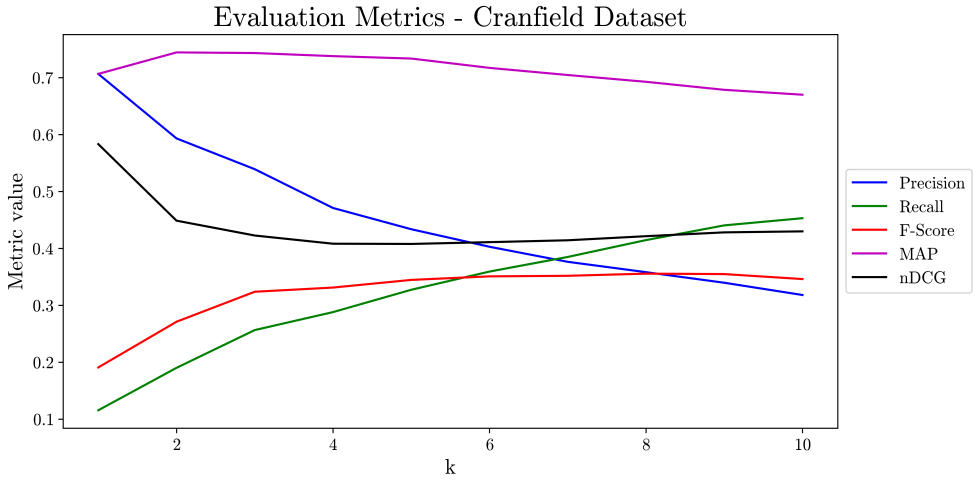

Fig. 5 Performance: Spell-Checker Model

The Alternate hypothesis (Ha): The model is different with improved values.

The mean of previous nDCG = 0.4209
The mean of new nDCG = 0.4376
d: Difference per paired value
n: Number of samples = 10

$$t = \frac{\Sigma d}{\sqrt{\frac{n(\Sigma d^2)-(\Sigma d)^2}{n-1}}}.$$

$$t = 7.3152$$

$$Degree\ of\ freedom = 10 - 1 = 9$$

$$two\ tails\ p{-}value = 4.49{\times}10^{-5} < 0.05$$

The p-value is less than 0.05, so we reject the null hypothesis that they are equal and conclude that they are significantly different. This means the new nDCG is better than the old one.

6.3.3 Limitations

1. It only works as long as it is a common word and word in the corpus. If it is an entirely new scientific word, it corrects it to the closest word in the corpus, which may not be right.
2. It increases the overall execution time as well since we are doing a spell check on every word in the query doc.

7 Conclusion

The overall performance of the search engine has improved by a considerable margin. It is now faster, less prone to spelling errors, and retrieves files pertaining to the author, owing to which the accuracy of the search engine has gone up. The improvement in both the accuracy and time of execution can be seen in the plots given. LSA is the most significant contributor in this performance enhancement. However, the search engine isn't still devoid of errors. It struggles with new words outside of the corpus; It also struggles with polysemous words; Spell check is not error-free as well, but it does show a considerable

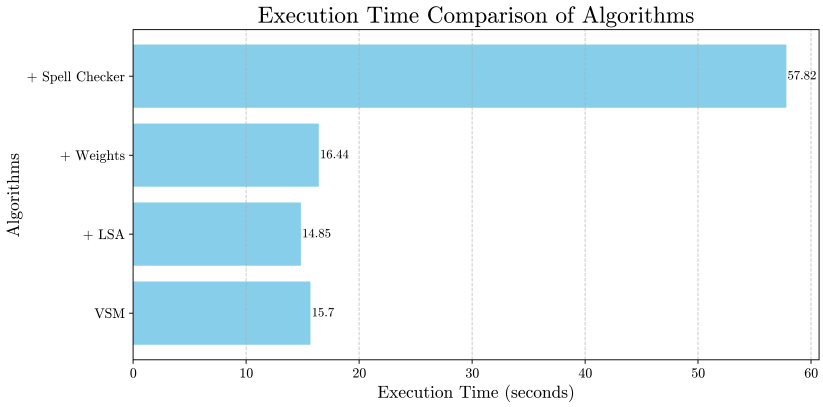

Fig. 6 Time taken for different models

jump from the previous model, which can be visually shown in the form of scores as below.

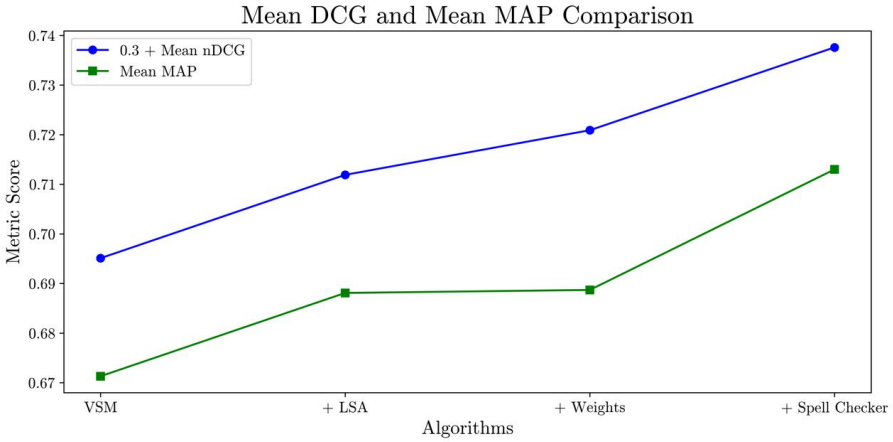

Fig. 7 Comparisons of Models

References

[1] Prakhar, Python – Lemmatization Approaches with Examples, https://www.geeksforgeeks.org/python-lemmatization-approaches-with-examples/

[2] Nishant N, TextBlob Spelling Correction, https://towardsdatascience.com/textblob-spelling-correction-46321fc7f8b8

[3] Yash R, Python – Spelling checker using Enchant, https://www.geeksforgeeks.org/python-spelling-checker-using-enchant/

[4] The Two-Sample t-Test, https://www.jmp.com/enin/statistics-knowledge-portal/t-test/two-sample-t-test.html

[5] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Richard Harshman, Indexing by latent semantic analysis, https://doi.org/10.1002/(SICI)1097-4571(199009)41:6

[6] Moussa Taifi, MRR vs MAP vs NDCG: Rank-Aware Evaluation Metrics And When To Use Them, https://medium.com/swlh/rank-aware-recsys-evaluation-metrics-5191bba16832

[7] Ioana, Latent Semantic Analysis: intuition, math, implementation, https://towardsdatascience.com/latent-semantic-analysis-intuition-math-implementation-a194aff870f8