CS6370 NLP Project: Enhancing Vector Space based IR System

Shubham Patel (ME19B170) & Bagul Amit Sunil (MM19B023)

Abstract

We provide a comprehensive overview of the use of the Vector Space Model (VSM) for search engine applications in this report. The Vector Space Model is a mathematical representation that enables efficient retrieval and ranking of documents based on their relevance to user queries. This report provides an overview of the different terminologies of the VSM, its implementation in a search engine and its advantages as well as limitations. After that, we discuss various techniques and considerations for improving the performance of the search engine based on the VSM.

1 Introduction

Search engines have become an integral part of our daily lives and it is hard to imagine life without it. It enables us to quickly find relevant information on the web. The Vector Space Model is one of many techniques and is widely used in search engine technology to retrieve relevant documents. By representing documents and queries as vectors in a high-dimensional space, the VSM provides a foundation for effective information retrieval.

1.1 Overview of the Vector Space Model

1. The Vector Space Model represents documents and queries as vectors in a multi-dimensional space. This vector should capture information of document such as terms. The VSM enables the calculation of document-query similarity based on various measures, such as the cosine similarity, dot product. By comparing how close/similar the document and query vectors are, the VSM ranks documents in order of their relevance to the query.

1.2 Steps involved in the VSM

1. Implementing the Vector Space Model in a search engine involves several key steps:

2. *Text Preprocessing:* Before building the vector representation, text preprocessing techniques such as sentence segmentation, tokenization, stop-word removal, stemming, and normalization are applied to both documents and queries. These steps help to standardize the representation of text and improve retrieval accuracy.

3. *Term Frequency-Inverse Document Frequency (TF-IDF):* The TF-IDF weighting scheme is commonly used to assign weights to terms in the vector representation. It considers both the frequency of a term in a document (Term Frequency, TF) and the rarity of the term across the entire document collection (Inverse Document Frequency, IDF). The TF-IDF score boosts

the importance of terms that are frequent in a document but rare in the entire collection, thus capturing the unique characteristics of documents.

4. *Document-Query Vector Construction:* Using the preprocessed text and TF-IDF weights, the search engine constructs the document and query vectors. Each document is represented by a vector where each dimension corresponds to a term and its weight. Similarly, the user query needs to be transformed into a vector representation based on the same process.

5. *Document-Query Similarity Calculation:* Using these the document and query vectors, the search engine calculates their similarity using measures such as cosine similarity, dot product . The cosine similarity measures the angle between two vectors, where a smaller angle indicates a higher similarity. Dot product measures the direction alignment. Then search engine ranks the documents based on their similarity scores to the query.

1.3 Implemetation of the VSM

1. Storing documents and queries in order using dictionaries:

   - `docQueryDict` with key as index and value - corresponding document or query. `docQueryDict` = {0: d_1, 1: d_2,...,1399: d_{1400}, 1400: q_1,..., 1624: q_{225}} and similarly, `docDict` with key - index and value - corresponding document. `docDict` = {0: d_1, 1: d_2,..., 1399: d_{1400}}
   - `inv_idx` is calculated with key - token and value - list of indices of documents having that token. Example: `inv_idx` = {t_1: $[0, 1]$, t_2: $[3, 7, 9]$,...,t_n: $[1, 5, 6]$}

2. **TF-IDF** matrix calculation uses the following:

   - Term frequency dictionary: `term_freq` with key - token and value - number of times a token occurs in a particular document. `term_freq` = {t_1: 2, t_2: 5,...,t_n: 1} for say document d_2.
   - `tf_list` is a list of all `term_freq` dictionaries, one for each document or query in `docQueryDict`.
   - Inverse Document Frequency: `idf_dict` with key - token and value - IDF.

3. **Weight matrix**

   - `weight_matrix` is calculated using the TF-IDF model with each element:

     $$w_{ij} = \text{frequency}_{\text{ij}} \cdot idf_{ij}$$

     where w_{ij} is the weight of term t_i with respect to document d_j, $\text{frequency}_{\text{ij}}$ is the frequency of t_i in d_j and idf_{ij} is the inverse document frequency.
   - Document or Query vector is represented by a column of the `weight_matrix`.

4. **Cosine similarity** for ranking:

   - Similarity calculation: Document vector d_i is used with a query vector q_j. Note that the first 1400 columns of the `weight_matrix` are document vectors and the remaining 225 are query vectors.

5. **Evaluation Metrics**: using Recall, Precision, ..

1.4 Results

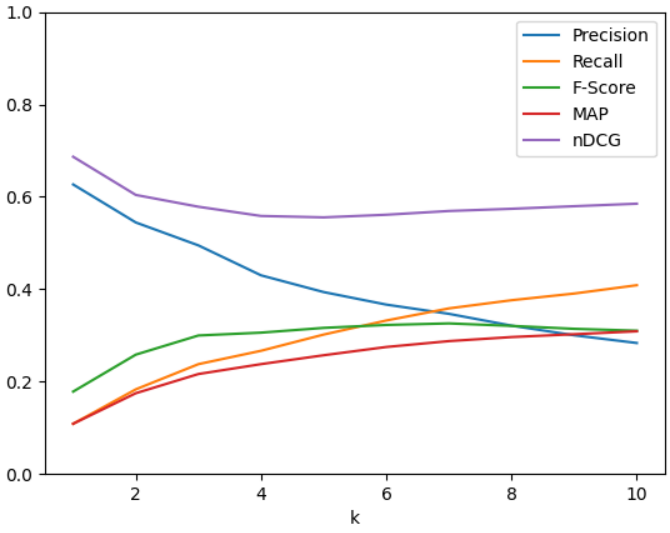

Figure 1: This is plot of Evaluation metrics for simple VSM model.

1.5 Limitations of the Vector Space Model

1. The current search engines suffers from the Dimensionality curse. As the number of documents and queries increase, it takes more storage and time to compute. It becomes computationally expensive to process the data with this current model. Also the longer documents will get more relevance score due to more words and likewise, the smaller ones will get less score.

2. Vector space model is such that all important quantities pertaining to documents need to be recalculated when a new document or term is added.

3. As the current model only uses the presence of words without considering adjacent words or sentences (context indifferent) for similarity calculation, Order in which term and sentences occurs is lost.

4. *Polysemy:* The current model does not take into consideration the meaning of a particular word in a given context. A query with polysemous words may get a high similarity score with

irrelevant documents (having same words but different meaning) and thus may suffer from precision.

5. *Synonymy:* The current model does not identify different words that have same meaning as related words in similarity calculation. Thus it may assign a less similarity score even for relevant documents if words are synonyms.

2 Improvements

2.1 Proposed Hypotheses

1. **Latent Semantic Analysis (LSA):** It can be used to solve above mentioned limitations namely, synonymy and to some extent polysemy. Instead of thinking documents as "bag of words" with each word occurring separately, topic model (word or text clustering) can be used to improve document representation The reason that it can overcome above mentioned limitations is because it is a process that finds unstructured data to get hidden relationships between terms and concepts using singular value decomposition. The term matrix is used for dimensionality reduction (Latent semantic space). It can help reduce noise and data is placed according to correlation which aids retrieval.

2. **BM25:** It uses different rankings than TF-IDF cosine similarity. BM25 improves TF-IDF by using two parameters to do the following:

   - *Term saturation Parameter* k is used to detect saturation of TF after which, no additional relevance is provided. For example if two document contains many occurrences of 'dog', one with frequency 200, other with 100. Both will have different relevance to query 'dog' with TF-IDF but relevance likelihood can be controlled in BM25 using k.
   - *Document Length normalization Parameter* b is used to penalised the documents with lengths larger than average document length and smaller ones are rewarded. This is to normalize the size of documents and have fair judgement unlike TF-IDF.

3. **Modifying TF-IDF weights:** For the original model, a simple $frequency_{ij}$ (term frequency) times idf_{ij} formula was used which would induce bias if a word was repeated enough times by assigning it a relatively higher weight. So, we use the *Glasgow model* for calculating weights. The `weight_matrix` is calculated using the Glasgow model with each element:

$$w_{ij} = \frac{\log_2 (frequency_{ij} + 1)}{\log_2 (length_j)} \cdot idf_{ij}$$

   where w_{ij} is the weight of term t_i with respect to document d_j, $frequency_{ij}$ is the frequency of t_i in d_j, $length_j$ is the number of unique tokens in d_j and idf_{ij} is the inverse document frequency.

4. **Query expansion:** Using only keywords is not very efficient way. Generally, if a query is small and results in poor document retrieval due to lack of sufficient information. Query expansion is a process used to improve the quality of user query. If Query expansion is used along with the above mentioned models, improvement in retrieval performance is possible. By Adding additional relevant terms to the terms of query vocabulary mismatch can be addressed increasing efficiency of retrieval system.

5. **A combination** of above mentioned models. This can potentially help in improving the overall performance of our search engine.

2.2 Flow of modelling

1. Preprocessing the documents from carnfield dataset and query
2. *Expand query using WordNet and DBpedia* ***partially done**
3. Perform Latent Semantic Analysis (LSA) using Singular Value Decomposition (SVD)
4. LSA model vector representation for query.
5. Build BM25[1]
6. Use of combination of models
7. Rank the documents
8. Retrieve top $k \in (1, 10)$ documents
9. Evaluate and compare the results

2.3 Evaluation Measures

- Hypothesis testing methods such as, Precision@k, Recall@k, F-score, MAP@k, nDCG@k for different values of $k \in (1, 10)$.

3 Experiments

In order to evaluate the effectiveness of our proposed improvements to the Vector Space Model, we conducted several experiments.

1. *Top m singular values*: We compared the performance of LSA with the original weight matrix calculation formula (simple TF-IDF) over different values of m. We experimented with values of m equal to 100, 200, 500, and 1000, and measured the corresponding precision and recall scores. We also experimented with the Glasgow model of weights using the same values of m, and measured the corresponding precision and recall scores. Based on the results of these experiments, we then selected the best value of m, which gave the highest precision and recall scores.
2. We compared the use of LSA with $m = 200$ with the original model wherein LSA was not employed, to determine the influence of using LSA.
3. We compared the performance of the Glasgow model of weights with the original weight matrix calculation formula using $m = 200$, to determine the superior method.
4. We experimented with the BM25 model and its influence on the performance. We use a model wherein we multiply the similarity matrices of LSA with the one obtained using BM25 element-wise. We find the best m value again.

5. We compare the above model (BM25·LSA) with simple LSA over recall and precision metrics for $m = 200$.

Finally, we analyzed the results of all the experiments to draw conclusions about the effectiveness of the proposed improvements to the Vector Space Model. The results and conclusions of our experiments are discussed in the following sections.

4 Results

1. Plots for Precision and Recall comparing original weight matrix method (TF-IDF) with Glasgow method for different m values. Clearly, we obtain the best results for $m = \mathbf{200}$.

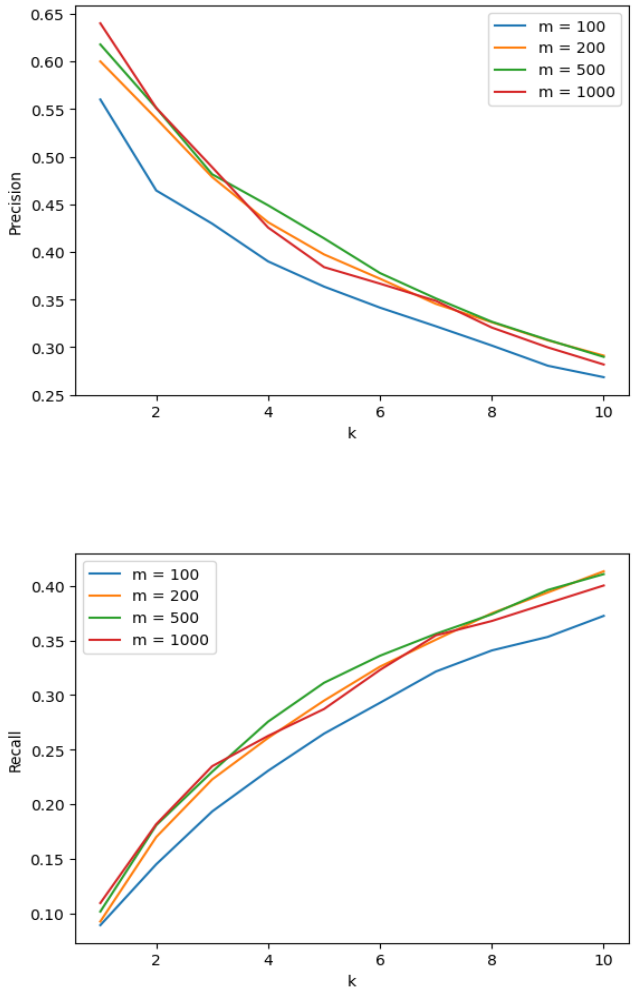

Figure 2: TF-IDF Method

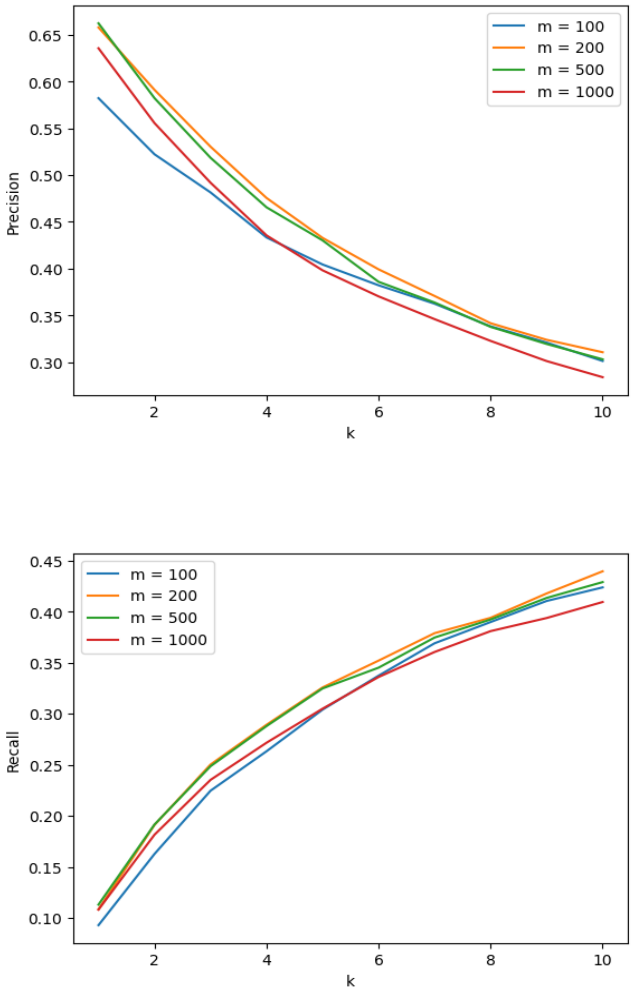

Figure 3: Glasgow Method

2. Plots comparing the effect of using Glasgow for $m = 200$.

3. Plots comparing the effect of using LSA versus without for $m = 200$.

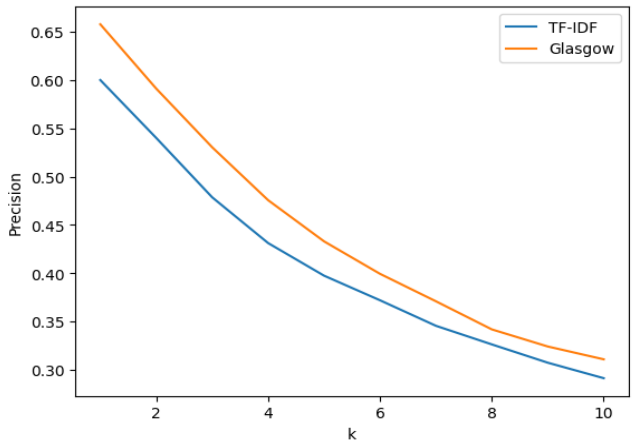

Figure 4: Comparing precision

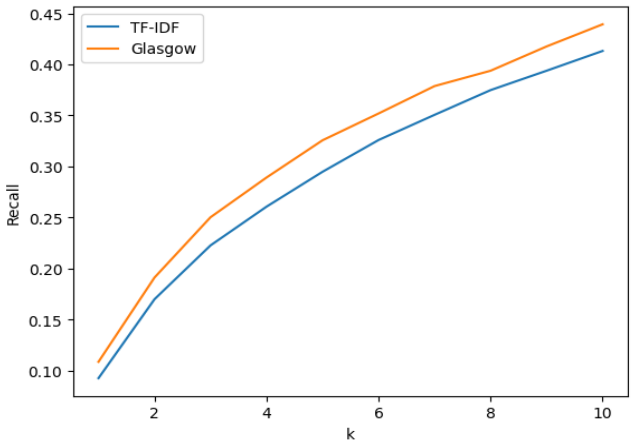

Figure 5: Comparing recall

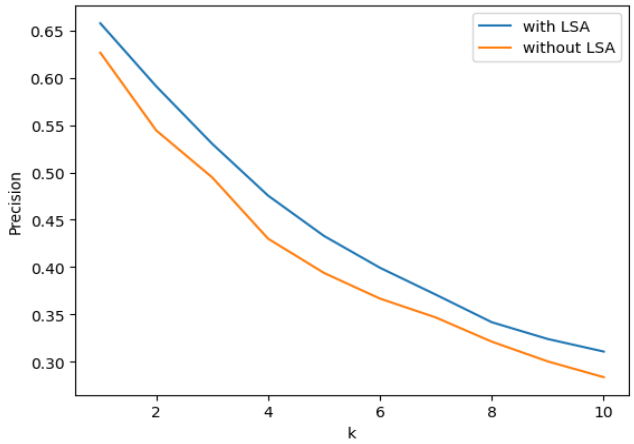

Figure 6: Comparing precision

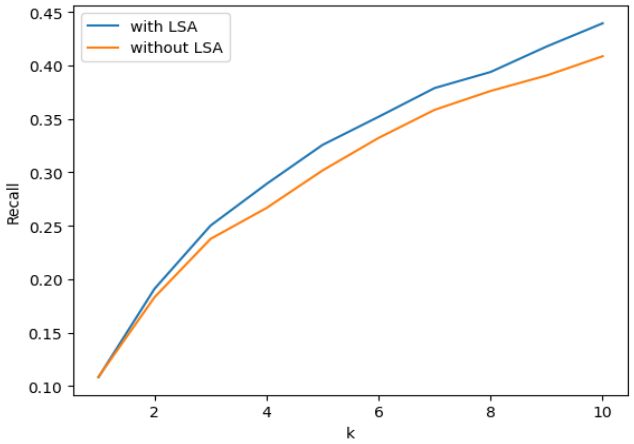

Figure 7: Comparing recall

5 Discussion

The use of different approaches gave different results as above. The best results were observed for m=200 for LSA combined with BM25. Multiplying the BM25 scores and LSA similarity scores was used to emphasize documents that are not only semantically similar but also have higher BM25 relevance are retrieved. This gave better results compared to different weighted average of LSA and BM25. Query expansion and other relevant techniques can be used to better the model using domain specific knowledge in future.

6 Acknowledgements

Our experiment was more complicated than initially intended, and we could not have done it by ourselves. We would like to thank Adwait Sir (TA) for providing the guidance on the project. Additionally, we would like to thank Profesor Sutanu for wonderful teaching and for providing

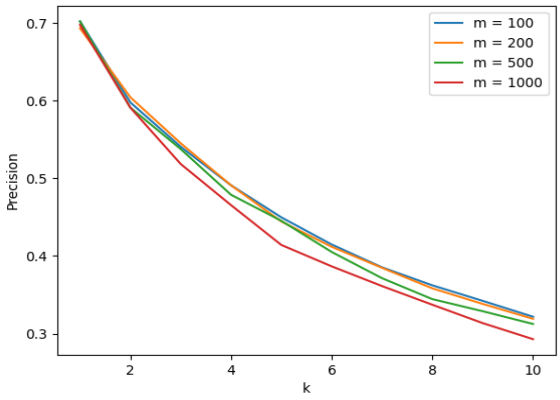

Figure 8: Plot of precision of sim=BM25*LSA

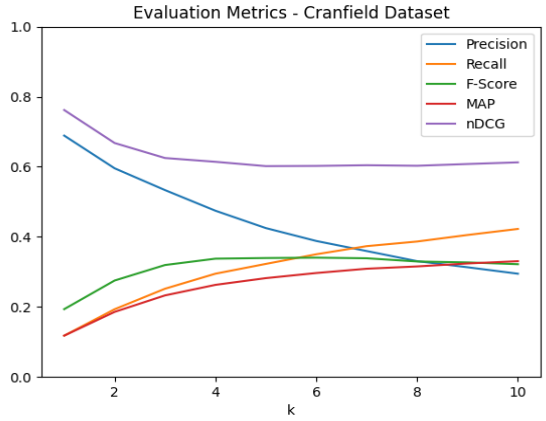

Figure 9: Plot of evaluation of sim=0.25*BM25+0.75*LSA

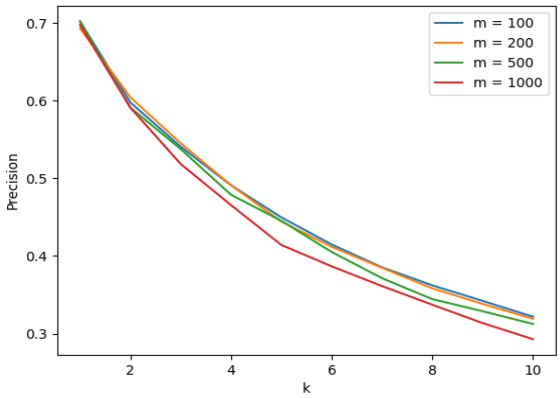

Figure 10: Plot of precision of sim=BM25*LSA

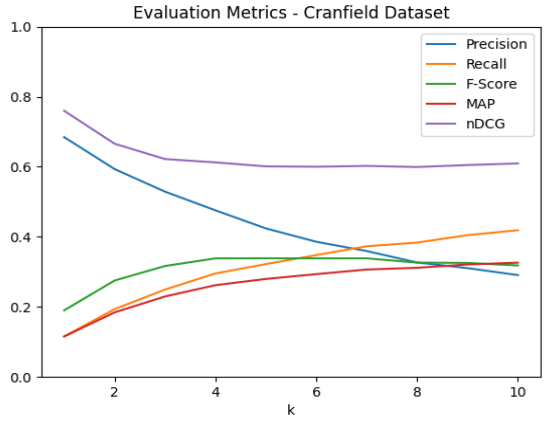

Figure 11: Plot of evaluation of sim=0.5*BM25+0.5*LSA

hands on practice in form of assignments and project. Finally, We would like to thank everyone that helped us in one way or other to complete this project.

References

[1] https://pypi.org/project/rank-bm25/

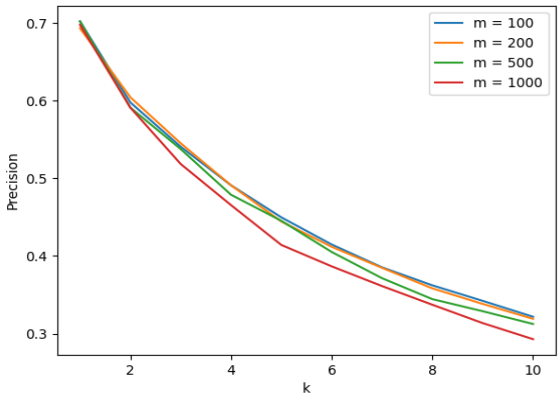

Figure 12: Plot of precision of sim=BM25*LSA

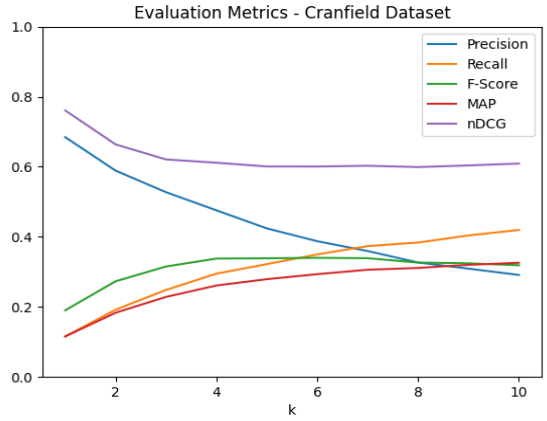

Figure 13: Plot of evaluation of sim=0.75*BM25+0.25*LSA