Document Retrieval Project Report

CS6370: Natural Language Processing

Chathur (CS20B018) and Santosh G (EE19B055)†

†These authors contributed equally to this work.

Abstract

The project aims at improving the performance of the Search engine by addressing the limitations previously faced. The search engine has been built to perform Information Retrieval from the Cranfield Dataset. The dataset contains about 1400 documents related to Aerodynamics. We have experimented with and implemented query processing and various methods of documents-and-concept representations to improve the system's performance.

Keywords: Information Retrieval, Cranfield Dataset, LSA, Bi-Grams

1 Introduction

Information Retrieval (IR) is retrieving relevant documents and raningk them in the order of relevance to the query. It is very crucial to achieve this, especially when there is large data; starting with accurately disambiguating the query, we face diversity and complexity of linguistics as a huge challenge.

There have been various improvements and new methods to improve IR; the previously submitted system (for Assignment-2) implements the **"tf-idf"** model to retrieve relevant documents, whereas in this project, we have identified issues with the previous model and tried addressing a few issues by implementing Latent Semantic Analysis (LSA) and also processing the query to obtain and match the bi-grams (of the query) with the corpus to extract the relevant documents.

2 Baseline and its limitations

2.1 Baseline

- The model that has been submitted for assignment 2, which is based on **Vector space Model** is being used as the baseline model for this project.
- Normalised "tf-idf" has been implemented in the baseline model, which works by computing the individual term frequencies and inverse document frequencies, which are then multiplied to compute the weights in the vector.
- If a term is commonly present across large proportion of documents, the IDF value would be small and hence wouldn't contribute a lot while evaluating the relevancy of documents.

2.2 Limitations

As mentioned in the proposal, the following are the shortcomings of the Vector space model (VSM):

- **Assumption of Term Independency (Orthogonality)**: The model assumes the unique terms are unrelated to other terms, disregarding the semantic understanding and resulting in inaccurate search results.
- **Polysemy and Synonymy :** VSM tries to match the exact form of the word and will not be able to handle synonymy or polysemy, hence resulting in improper results

3 Evaluation Metrics

Various metrics such as Precision, Recall, F-Score, Mean Average Precision (MAP) and nDCG are being computed and compared to evaluate the performance of the model

3.1 Recall

The fraction of relevant documents that have been retrieved.

$$recall = \frac{|relevant \cap retrieved|}{|relevant|}$$

3.2 Precision

The fraction of retrieved documents which are relevant.

$$precision = \frac{|relevant \cap retrieved|}{|retrieved|}$$

3.3 F-score

It is the harmonic mean of precision and recall.

$$F - score = \frac{2 \times precision \times recall}{precision + recall}$$

3.4 Mean Average Precision

It is the mean of the Average Precision @ K over all the queries.

$$MAP@K = \frac{\sum_{i=1}^{N} AvgPrecision@K(i)}{N}$$

3.5 nDCG

It is the ratio of DCG@K(Discounted Cumulative Gain) and IDCG@K(Ideal DCG).

$$nDCG_k = \frac{DCG_k}{IDCG_k}$$

$$DCP_k = \sum_{i=1}^{k} \frac{rel_i}{log_2(i+1)}$$

4 Proposed Methodology

4.1 Latent Semantic Analysis (LSA)

Latent Semantic Analysis (LSA) generates the term-document matrix with the tf-idf scores, and then we obtain a compressed representation of it by applying Singular Value Decomposition (SVD).
SVD generates a low-rank approximation of the above matrix.

4.1.1 Singular Value Decomposition (SVD):

Singular Value Decomposition (SVD) of a matrix A is computed by:

$$A = U\Sigma V^T$$

where A is an M x N matrix, U is an M x M, V is an N x N, and Σ is an M x N matrix; U (a column orthogonal matrix) and V (a row orthogonal matrix) are unitary matrices and Σ contains the singular values arranged in descending order.
U represents the concepts, whereas V^T consists of concepts represented in terms of documents. M corresponds to the total number of individual words and N corresponds to the total number of documents; In this case, M = 8824 and N = 1400

The dimensionality is reduced by selecting the k largest eigenvalues in Σ and by using LSA we can capture the relation between terms and it can handle synonymy (by assuming words that occur in similar context have similar meaning).

Computing similarity: The queries are transformed into vectors and the cosine similarity is computed with the reduced-dimensional matrices (U' and V'), generated after truncating the smaller singular values (as they are considered less important).

Based on the above-computed similarity scores, the relevant documents are retrieved.

4.2 Bi-gram score

One of the drawbacks of the vector space model is that it considers each word as an individual entity. So phrases will not be recognized and will not have enough significance in the model. But phrases tend to occur a lot in natural language. We can solve this problem by using "N-grams".

We model N consecutive words as a single type called an N-gram. So we can construct a vector space model for these N-grams and rank the documents according to the existence of these N-grams in the corresponding queries, The baseline vector space model is basically a 1-gram or a unigram model.

Although we could extend this to as high a value of N as we wanted, we hit the point of diminishing returns very quickly. This is because the number of phrases decreases rapidly as N increases. For this project we are only considering 2-grams or bigrams. We add the bigram scores for a document with the unigram scores and then rank the documents accordingly.

4.3 Query Spell Checking

Queries are processed to check if there are any spelling errors in them; We used the **"correction(w)"** function in the **"pyspellchecker"** library, which returns the most probable correct spelling for an input word.

- Spell check is performed only on the custom queries as the queries in the Cranfield Dataset have all the words spelled correctly.
- Cranfield Dataset, Words from nltk have been used to determine the most probabale words.

5 Experimentation and Results

In LSA, after performing SVD, we have **compared the evaluation-scores for K=10** (number of documents to be retrieved) to arrive at the optimal number of the singular valuations to be chosen; after the mentioned process, **n = 800 was the optimal number of values** to be chosen from the 1400 values.

For the Bigrams model, we constructed an index of bigrams similar to the index we constructed for unigrams in the original vector space model. After constructing the index, we match the bigrams against the bigrams in the incoming queries and provide a score for each of the documents. We then added these scores to the scores obtained from the original unigram model and obtained the final scores for the documents. Using these scores, we generate a ranked list of relevant documents for each of the documents.

The spelling error detection is being done only on the custom queries, as it is taking significantly high time when compared to other evaluations; Also the queries in the Cranfield Dataset, all the queries are correctly spelt and there are chances of spelling errors only in the custom queries.

We have also evaluated the scores by varying the number of documents to be retrieved and the results, for various models, have been documented and plotted below:

5.1 VSM Model (Baseline):

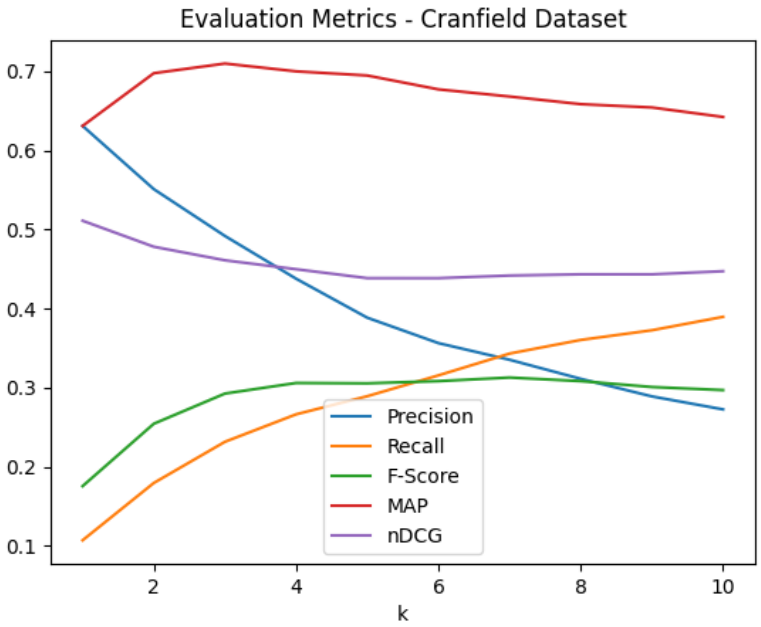

Fig 1: Scores for Baseline Model

K	Precision	Recall	F-Score	MAP	nDCG
1	0.6311	0.1069	0.1754	0.6311	0.5111
2	0.5511	0.1794	0.2543	0.6977	0.4782
3	0.4919	0.2315	0.2925	0.7100	0.4611
4	0.4377	0.2662	0.3059	0.7001	0.4498
5	0.3884	0.2891	0.3054	0.6949	0.4385

Table 1: Scores for VSM Model

5.2 Bi-Gram:

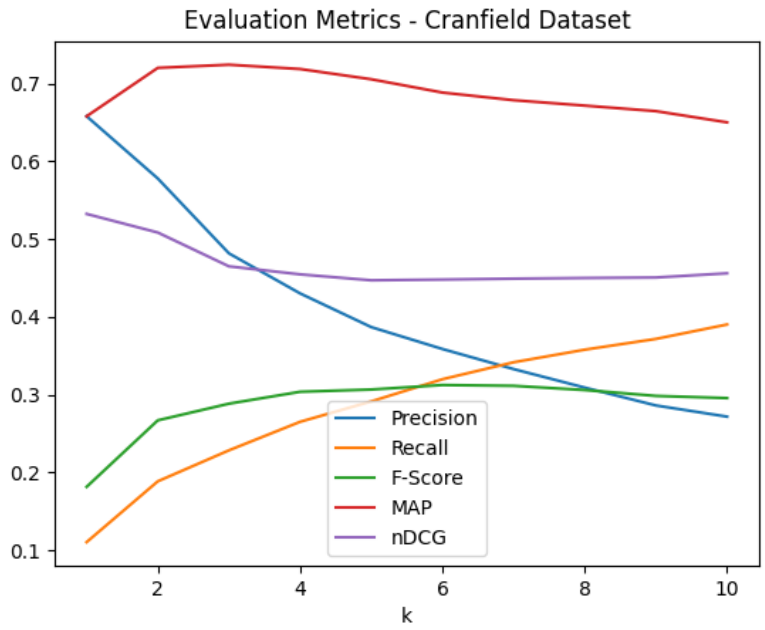

Fig 2: Scores for Bi-Gram Model

K	Precision	Recall	F-Score	MAP	nDCG
1	0.6577	0.1102	0.1813	0.6577	0.532
2	5777	0.1884	0.2668	0.72	0.5083
3	0.4815	0.2282	0.2882	0.7241	0.4648
4	0.43	0.2649	0.3035	0.7185	0.4545
5	0.3866	0.2911	0.3064	0.7056	0.4469

Table 2: Scores for Bi-Gram Model

5.3 LSA:

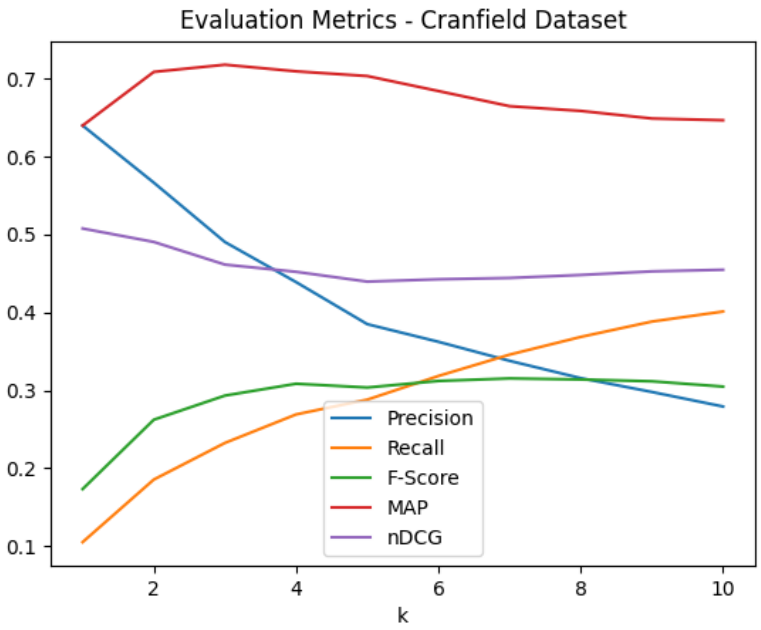

Fig 3: Scores for LSA Model

K	Precision	Recall	F-Score	MAP	nDCG
1	0.64	0.1049	0.1731	0.64	0.5077
2	0.5666	0.1853	0.2621	0.7088	0.4905
3	0.4904	0.2323	0.2931	0.7181	0.4614
4	0.4389	0.2689	0.3083	0.7096	0.4521
5	0.3848	0.2879	0.3035	0.7036	0.4395

Table 3: Scores for LSA Model

6 Conclusion

From the above tables and graphs, we can see the improvement in performance from the baseline model, as the scores have increased.

Hence we can conclude:

Latent Semantic Analysis (LSA) performs better compared to the **Vector Space Model** (Baseline Model), for **Retrieving relevant documents in ranked order** on **Cranfield Dataset** under assumptions

Similarly,
Bi-Gram Model performs better compared to the **Vector Space Model** (Baseline Model), for **Retrieving relevant documents in ranked order** on **Cranfield Dataset** under assumptions

7 Bibliography

1. Vector Space Model - GeeksForGeeks https://www.geeksforgeeks.org/web-information-retrieval-vector-space-model/
2. Latent Semantic Analysis - GeeksforGeeks https://www.geeksforgeeks.org/latent-semantic-analysis/
3. pyspellchecker for spellcheck https://pypi.org/project/pyspellchecker/
4. NLTK for Punkt tokenizer, Wordnet lemmatizer and stopwords https://www.nltk.org/
5. Evaluation measures for IR system https://www.pinecone.io/learn/offline-evaluation/
6. Latent Semantic Analysis — Deduce the hidden topic from the document https://towardsdatascience.com/latent-semantic-analysis-deduce-the-hidden-topic-from-the-document-f360e8c0614b