CS6370: Natural Language Processing Course Project

BE20B031 — Sheth Jenil Samir
CS21B014 — Meghana Banoth
CS21B022 — Himakar Sai Dasari

Indian Institute of Technology, Madras

Abstract. This paper deals with Information Retrieval using a simple search engine. First, we show the results of an IR system implemented with a Vector Space Model (VSM), taken as the baseline model. We then improve over the basic IR system by tackling issues that VSM fails to solve - methods like LSA and query expansion are employed for this purpose. Hyperparameter tuning is done for both approaches and further improvements on these models are explored. The result of spellcheck for custom queries is explored later on.

Keywords: Information Retrieval, Vector Space Model, Latent Semantic analysis, query expansion, spellcheck

1 Introduction

The IR system implemented using the Vector Space Model (VSM) for the first part of this project fails to address many issues one is concerned about when it comes to retrieving the best documents for a query in an ideal order.
VSM represents the documents and queries in a multi-dimensional vector space spanned by the words of the corpus. Each component of the vectors in the space is weighted. Weights can be given based on the TF-IDF values of the word - combining its term frequency within a document and its inverse document frequency - how good it is at differentiating the document it is in from other documents.

Disadvantages of VSM

1. Due to many possible words existing in the corpus, the dimensions of the vectors involved may be very high.
2. It is assumed that the words representing the vector space are orthogonal to each other, which is not the case in natural language.
3. The underlying concepts from which the words may arise from is not captured.

To address these challenges, we explored different methods to construct the IR system, which are briefly explained below.

1.1 Latent Semantic Analysis

Latent Semantic Analysis (LSA) captures the underlying semantic structure of the term-document matrix through a mathematical technique called singular value decomposition. LSA is capable of giving similarity scores between terms, enabling it to deal with synonymy; between documents, and a reduced-rank approximation of the original matrix.

1.2 Query expansion

Query expansion (QE) is a technique that enhances a query by adding words similar to the original query terms. This approach aims to retrieve relevant documents that might otherwise be missed due to variations in word usage. We applied this approach using Word2vec.

1.3 Part of Speech tagging

POS tagging is a process that assigns parts of speech to words, offering valuable information for an IR system. We evaluated the system's performance by incorporating POS tagging into documents and queries while implementing LSA.

1.4 Spellcheck

We implement spellchecking on custom queries by leveraging bigrams from the corpus. When a typo is detected in the query, it is corrected by replacing the misspelled word with the closest matching word from the corpus. This ensures that the query is more accurate and relevant to the content of the corpus.

2 Evaluation Measures

Various metrics are employed to measure the performance of an IRS, each offering unique insights into different aspects of retrieval quality. In this report, we present a comprehensive evaluation of various models using key metrics including `Precision@k`, `Recall@k`, `F0.5Score@k`, `AP@k`(Average Precision), and `nDCG@k` (Normalized Discounted Cumulative Gain @k). By employing these measures, we aim to assess the relevance and effectiveness of various models in retrieving relevant documents for various queries.

2.1 Precision@k:

Precision at k measures the proportion of relevant documents among the top k retrieved documents. It provides insight into the accuracy of the retrieved documents.

Usage:

- Precision is computed for each query individually.
- For each query:
  - Retrieve the top k documents according to the system's ranking.
  - Compare the retrieved documents with the ground truth (relevant documents).
  - Calculate the proportion of relevant documents among the top k retrieved documents.

2.2 Recall@k:

Recall at k measures the proportion of relevant documents that were retrieved among all relevant documents. It provides insight into the system's ability to retrieve all relevant documents.

Usage:

- Recall is computed for each query individually.
- For each query:
  - Retrieve the top k documents according to the system's ranking.
  - Compare the retrieved documents with the ground truth (relevant documents).
  - Calculate the proportion of relevant documents retrieved among all relevant documents.

2.3 F0.5Score@k:

`F0.5Score` at k is the weighted harmonic mean of precision and recall, favoring precision. It balances precision and recall, with a higher weight on precision.

Usage:

- F0.5 Score is computed for each query individually.
- For each query:
  - Compute `Precision` and `Recall` at k.
  - Calculate `F0.5Score` using the formula:

$$\texttt{F}_\beta Score = \frac{(1+\beta^2) * \texttt{Precision} * \texttt{Recall}}{\beta^2 * \texttt{Precision} + \texttt{Recall}}$$

    where $\beta = 0.5$ in our case.

2.4 AP@k (Average Precision@k):

`AP@k` i.e., Average Precision at k is the average of precision values obtained at each relevant document's rank up to k. It considers the precision at different levels of recall.

Usage:

- `AP@k` is computed for each query individually.
- For each query:
  - Compute precision at each relevant document's rank up to k.
  - Average the precision values.

2.5 nDCG@k (Normalized Discounted Cumulative Gain@k):

`nDCG@k` i.e., Normalized Discounted Cumulative Gain at k is a measure of ranking quality, considering both relevance and ranking position of retrieved documents. It provides insight into the ranking effectiveness.

- `nDCG@k` is computed for each query individually.
- For each query:
  - Compute `DCG` (Discounted Cumulative Gain) using the following update formula in the order in which the system has provided the results:

$$\texttt{DCG} = \texttt{DCG}_{\text{old}} + \frac{\text{relevance value of the } i\text{th document}}{\log_2(i+1)}$$

  - Similarly compute `IDCG` (Ideal Discounted Cumulative Gain) at k using the following update formula in the optimal order in which the documents are ordered according to the ground truth:

$$\texttt{IDCG} = \texttt{IDCG}_{\text{old}} + \frac{\text{relevance value of the } i\text{th document}}{\log_2(i+1)}$$

  - Calculate `nDCG` using the following formula:

$$\texttt{nDCG} = \frac{\texttt{DCG}}{\texttt{IDCG}}$$

3 Dataset Used

In our project, we utilized the Cranfield dataset. This dataset is comprised of three main components: queries, documents and query relevance judgements (qrels). The queries represent user search queries, while the documents consist of a collection of texts that are indexed for retrieval. The qrels contain relevance judgements, specifying the positions of various documents in response to specific queries. This serves as the *ground truth* for our evaluation measures.

4 Vector Space Model

4.1 Results

On running the base vector space model, we obtained the below given results

k	Precision	Recall	F-score	MAP	nDCG
1	0.667	0.115	0.188	0.667	0.536
2	0.556	0.184	0.259	0.716	0.499
3	0.489	0.233	0.293	0.717	0.471
4	0.430	0.268	0.305	0.709	0.459
5	0.404	0.310	0.324	0.698	0.460
6	0.376	0.340	0.330	0.686	0.461
7	0.352	0.364	0.331	0.677	0.464
8	0.334	0.389	0.332	0.670	0.470
9	0.311	0.402	0.324	0.663	0.472
10	0.296	0.424	0.322	0.651	0.479

Table 1: Evaluation metrics for base VSM

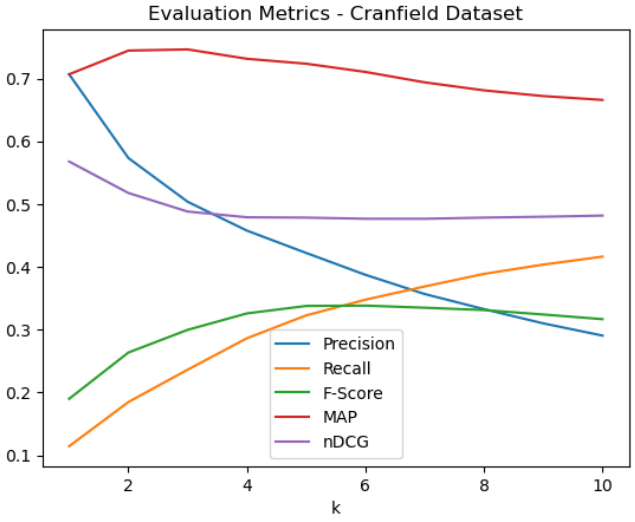

Fig. 1: Plot of the evaluation metrics for base VSM

4.2 Examples of queries where VSM fails

1. **Query** - Location of halted movement for a non sharp object within a supersonic flow
   doc ID needed - 35
   doc IDs obtained - 228 (totally irrelevant), 1300 (bluntness, supersonic are mentioned), 401 (Hypersonic, airflow), 690(Flow, spiked cylinder)
2. **Query** - Some research on high speed flutter exploration into phenomenon of rapid self excited vibrations of a wing of an aircraft
   doc ID needed - 1111
   doc IDs obtained - 100 (vibration, aircraft), 908 (vibration), 51(aircraft), 345(aircraft), 844(vibrations)

3. **Query** - Constraints on structuring for cost effectiveness in missile frameworks
   doc ID needed - 834
   doc IDs obtained - 32(missile), 1217(constraints), 256(totally irrelevant), 834(missile)

5 Query Expansion

Query expansion tries to improve the effectiveness of the system by including words similar to the query words in the query. It tries to encapsulate synonymy and retrieve documents describing concepts similar to the concepts involved in the query. Word2vec was used to obtain words similar to query words. It does so by representing words as vectors in a low dimensional space. There are two variants of Word2vec: Continuous Bag-of-Words (CBOW), which predicts a central word by looking at its context words, and Skipwords, which predicts the context words by looking at a central word. Models for both these approaches were trained. These were used to construct term-document matrices using the expanded queries and **TfidfVectorizer()**.
There is non-determinism involved in the Word2vec in-built function, so the algorithm was run over 5 different iterations with different results for each iteration. The best results are shown in the table below.

k	Precision	Recall	F-score	MAP	nDCG
1	0.671	0.108	0.180	0.671	0.543
2	0.535	0.169	0.242	0.708	0.482
3	0.467	0.213	0.272	0.717	0.455
4	0.414	0.247	0.287	0.713	0.442
5	0.370	0.275	0.292	0.695	0.433
6	0.331	0.293	0.287	0.686	0.427
7	0.307	0.317	0.288	0.670	0.425
8	0.282	0.331	0.288	0.662	0.426
9	0.268	0.349	0.280	0.649	0.430
10	0.251	0.362	0.274	0.638	0.432

Table 2: Evaluation metrics for Query Expansion

Word2vec is observed to perform better for the CBOW variant than the Skipwords variant in general.

5.1 Hyperparameter Tuning

Word2vec involves a number of different hyperparameters. Results were best at window = 3. The sample value was taken to be 6e-5. A higher sample indicates less aggressive downsampling, meaning fewer words will be excluded from the

training data based on frequency. A lower sample value results in more aggressive downsampling - more frequent words are more likely to be excluded from training. The vector size, i.e., the number of components used to represent the vectors for the words in a low dimensional subspace has been experimented with.

6 Latent Semantic Analysis

LSA reduces the dimensionality of the document-term matrix by capturing the underlying structure of the data. This reduction makes the data more manageable and less prone to over fitting. In high-dimensional text data, the document-term matrix tends to be sparse, with many entries being zero. LSA helps alleviate sparsity by transforming the matrix into a lower-dimensional space by using Singular Value Decomposition where the relationships between terms and documents are more densely represented, leading to more robust and reliable similarity measures.

6.1 Hyperparameter Tuning

We only have one hyperparameter, number of latent variables, to test. We plot the MAP values against number of components and choose the one with highest MAP value.

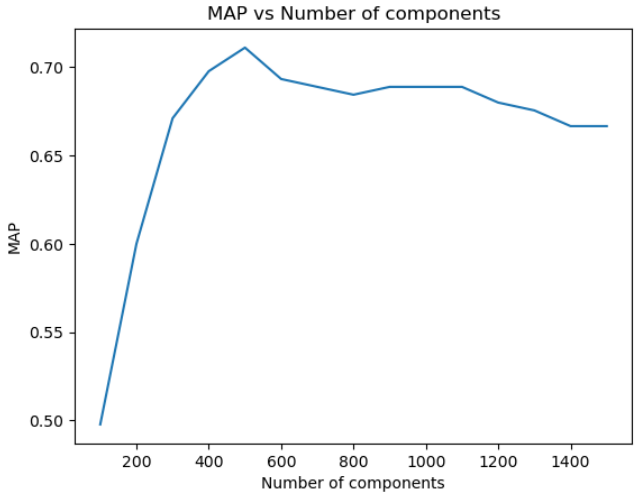

Fig. 2: Hyperparameter tuning for number of components

On analysis, we found that max MAP is obtained at Number of components = 500

k	Precision	Recall	F-score	MAP	nDCG
1	0.707	0.117	0.193	0.707	0.572
2	0.580	0.189	0.268	0.747	0.524
3	0.514	0.245	0.309	0.747	0.496
4	0.462	0.286	0.327	0.732	0.484
5	0.411	0.312	0.328	0.724	0.472
6	0.381	0.343	0.333	0.707	0.475
7	0.357	0.371	0.336	0.697	0.480
8	0.330	0.387	0.329	0.686	0.480
9	0.310	0.403	0.324	0.680	0.483
10	0.290	0.416	0.316	0.671	0.485

Table 3: Evaluation metrics for LSA

6.2 Results

On running our LSA model on cranfield dataset, we obtain the below given results

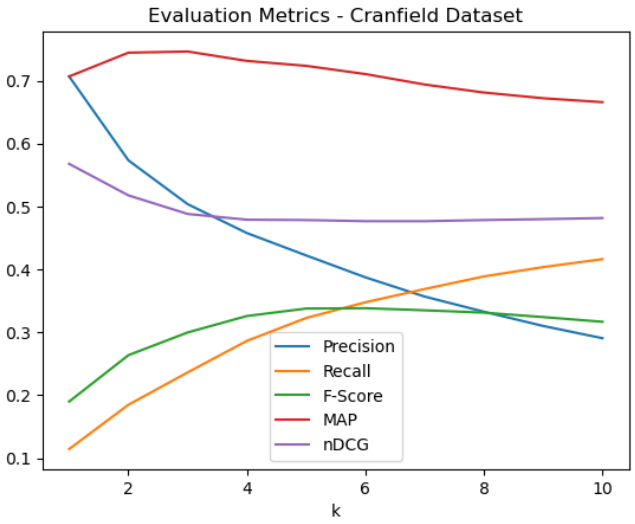

Fig. 3: Plot of the evaluation metrics for LSA

6.3 Hypothesis testing for VSM vs LSA

Hypothesis - On using latent semantic analysis which consists of SVD, we solve the problem of the large and sparse term document matrix to get better retrieved documents

metrics	t	p
Precision	-0.202	0.580
Recall	-0.297	0.617
F-score	-0.284	0.612
nDCG	-0.634	0.738

Table 4: Hypothesis testing for VSM vs LSA

An algorithm LSA is better than VSM with respect to the evaluation measure nDCG in task of Information Retrieval for retrieving relevant documents for given queries

7 LSA with POS tagging

We derive structural knowledge from text with the help of part of speech tagging. LSA, when trained, doesn't differentiate between words used in multiple parts of speech. This lack of distinction can have important semantic consequences. For instance, the word "plane" has distinct meanings as both a verb and a noun. To enhance LSA with structural information, one approach is to enable it to recognize the part of speech for each word during training and when comparing sentences.

7.1 Hyperparameter Tuning

We only have one hyperparameter, number of latent variables, to test. We plot the MAP values against number of components and choose the one with highest MAP value.

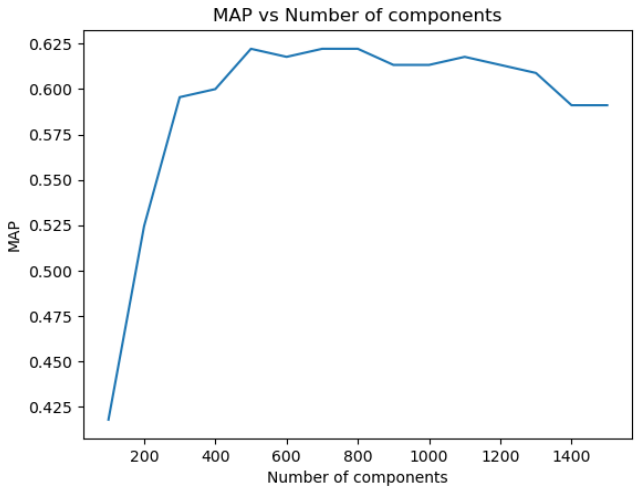

Fig. 4: Hyperparamater tuning for number of components

7.2 Results

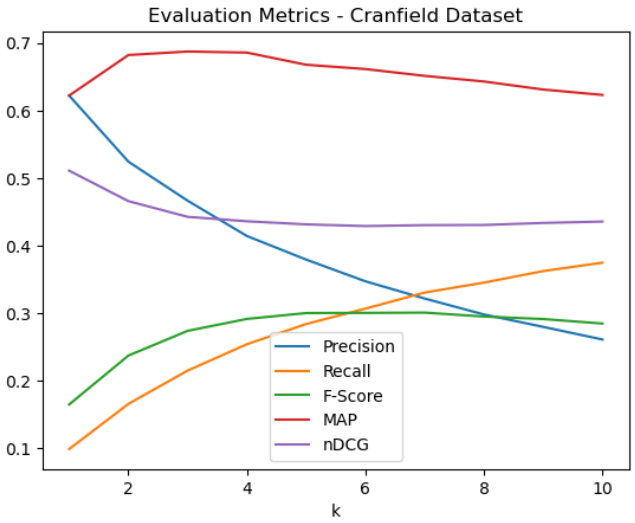

Fig. 5: Plot of the evaluation metrics for LSA+POS tagging

7.3 Hypothesis testing for LSA vs POS tagged LSA

Hypothesis - On tagging words with their parts of speech, we can solve the problem of normal LSA not being able to differentiate between words used in multiple parts of speech.

metrics	t	p
Precision	1.61	0.054
Recall	1.58	0.058
F-score	1.75	0.041
nDCG	1.97	0.025

Table 5: Hypothesis testing for LSA vs LSA+POS tagging

On asumming a predefined significance level of $p = 0.05$, the null hypothesis is rejected w.r.t all the 4 metrics. Thus we can say,
An algorithm LSA is better than LSA+POS tagging with respect to the evaluation measure nDCG in task of Information Retrieval for retrieving relevant documents for given queries.

8 Spellcheck and Spelling Correction

In addition to evaluating the search engine's performance on the Cranfield dataset, we recognized the importance of ensuring its effectiveness with custom queries.

These custom queries may contain typographical errors or misspellings, which could potentially impact the retrieval accuracy of the search engine. To address this concern, we implement a simple spell check logic to enhance the robustness of the search engine. This logic aims to correct misspelled words in the queries by identifying their closest counterparts in the vocabulary.

We began by extracting the vocabulary of the documents of the Cranfield dataset. This can be done by extracting all unique words from the document texts after removing non-alphabetic characters and converting the text to lowercase. This vocabulary forms the basis for identifying correct spellings for misspelled words in the queries.

8.1 Implementation and Evaluation:

The logic we implemented uses bigram distance as a measure of similarity between words. The Levenshtein distance metric is employed to compute the distance between the bigram representation of the misspelled word and those of all the words in the vocabulary. The word with the smallest bigram distance from the misspelled word is identified as the closest match and considered as the corrected spelling. This approach leverages a combination of bigram representation of words and the Levenshtein distance between words.

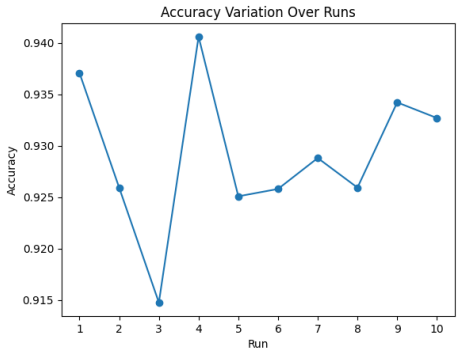

Fig. 6: Spellcheck and correction accuracy variation over 10 runs

To evaluate the effectiveness of the spell check functionality, we generated a set of words with intentional typos to simulate misspellings. For each of such typo, we calculated the corrected word using the logic demonstrated above. We then compared the corrected words with the ground truth to assess the accuracy of the spell check correction. When run on first 100 words of the vocabulary with various cases of typos, the accuracy turned out to be in the range of 90% - 95%.