Information Retrieval

Aadit Mahajan, bs21b001@smail.iitm.ac.in
Anirudh Rao, be21b004@smail.iitm.ac.in
Shahista Afreen, na21b050@smail.iitm.ac.in
Shreya Rajagopalan, be21b038@smail.iitm.ac.in
Sidharthan SC, be21b039@smail.iitm.ac.in

Indian Institute of Technology, Madras, Chennai 600036, TN, India

Abstract. This project investigates the limitations of traditional Information Retrieval (IR) systems based on the Vector Space Model (VSM), which often fail to capture semantic relationships between terms due to their reliance on lexical similarity. To address these issues, we explored enhancements using Latent Semantic Analysis (LSA) and Explicit Semantic Analysis (ESA), both of which enrich document and query representations by incorporating latent or concept-based semantics. In addition, we experimented with spreading activation using WordNet-based semantic similarity and neural embeddings to further improve retrieval quality. All approaches were evaluated on the Cranfield dataset using standard IR metrics including Precision, Recall, F-score, Average Precision, and nDCG. Despite their theoretical promise, the alternative methods did not yield significant performance improvements over the baseline VSM, suggesting that simple term-based models can remain competitive in controlled settings with limited vocabulary and context diversity.

Keywords: Information retrieval · Vector space model · Explicit semantic analysis · Latent semantic analysis

1 Introduction

1.1 The Problem of Information Retrieval

The task of an information retrieval (IR) system, or search engine, is to return relevant documents from a larger collection of documents, given a query by a user. This query arises from some tacit intent of the user and is constrained by the nature of the queries that the information retrieval system can accept (for example, the system may only accept image-based queries). The system returns documents whose contents are relevant to the query. In some informational retrieval systems, like conversational information retrieval systems, the user provides relevance feedback on the relevance of documents returned by a search query. This information is then used to refine the query and produce more relevant results in an iterative process. On the other hand, there is no such iteration in single-shot information retrieval systems. The basic model implemented here is the Vector Space Model (VSM)

1.2 Vector Space Model

In this model, documents are represented in a high-dimensional vector space where each dimension corresponds to a term / word. For each document, a "weight" is assigned to each term in the inverted

index and a vector is constructed. A similar vector is constructed for the query. A term-document matrix containing the weights can be constructed to represent the vector space
The similarity of the query to the documents can be measured using the **cosine similarity** between the corresponding vectors.

$$\text{sim}(\vec{Q}, \vec{D}_i) = \cos\theta = \frac{\vec{Q} \cdot \vec{D}_i}{|\vec{Q}||\vec{D}_i|}$$

The **term frequency** (TF) of a term i can be defined as the number of times the term i occurs in a document. This accounts for local information. The **document frequency** (DF) of a term i can be defined as the number of documents in which the term i occurs.
Let D denote the total number of documents. We define the **inverse document frequency** (IDF) of a term i as

$$\text{IDF}_i = \log\left(\frac{D}{\text{DF}_i}\right)$$

The ratio $\frac{D}{\text{DF}_i}$ is the reciprocal of the probability of selecting a document containing a queried term from the collection of documents. This can be viewed as a global probability over the entire collection.
The most commonly used term weight is the **term frequency-inverse document frequency** (TF-IDF).

$$\text{TF-IDF}_{i,j} = \text{TF}_{i,j} \times \log\left(\frac{D}{\text{DF}_i}\right)$$

The VSM implementation here consists of the following steps:

Tokenization: Text is split into sentences and then further into words using the Treebank tokenizer provided by the NLTK package.
Stopword Removal: Commonly used words such as "a", "an", and "the" are excluded from the token list to minimize the vocabulary size, which can improve the performance of the Vector Space Model (VSM).
Inflection Reduction: Stemming is performed using PorterStemmer from NLTK library.
TF-IDF Matrix: The Term Frequency-Inverse Document Frequency (TF-IDF) matrix is constructed by multiplying the term frequency with the inverse document frequency. Each document vector is then normalized to unit length to standardize the representation.
Ranking: Queries undergo the same preprocessing steps as the documents. Query vectors are generated and cosine similarity is used to compute the similarity between the query and document vectors. Documents are ranked in descending order based on their similarity scores.
Evaluation: Retrieval performance is evaluated using a range of metrics, including Mean Precision@k, Mean Recall@k, Mean F-Score@k, Mean Average Precision (MAP@k), and Normalized Discounted Cumulative Gain (nDCG@k).

1.3 Limitations of VSM

The Vector Space Model is a simple model for IR but it comes with the following disadvantages:

- The model assumes that terms are orthogonal to each other meaning they are not related to each other, which ignores contextual relationships

- VSM does not take into account semantic relationships. It relies on exact term matching and cannot handle cases of synonymy or polysemy effectively.
- Exact term matching also means that if a query contains terms that are not present in relevant documents, those documents will not be retrieved even if they are relevant. And similarly, even irrelevant documents may be retrieved if the query contains some matching terms.
- It does not take word order into consideration.
- The model produces high-dimensional, sparse vector representation of documents, which can be computationally expensive.

The following examples from the implementation on Cranfield dataset display the limitations of VSM. These are few queries where the model failed to retrieve even one relevant document among the top 10 retrieved documents.

- Example 1:
  - Query 28: "what application has the linear theory design of curved wings."
  - Top 10 documents: [1051, 1075, 680, 923, 921, 920, 762, 247, 674, 470]
  - Relevant documents: [224, 279, 512]
  - Observation: Model takes only direct words like "curved wings" and "linear theory" into consideration. It does not learn the hidden meaning to answer the relevant query.

- Example 2:
  - Query 63: "where can i find pressure data on surfaces of swept cylinders ."
  - Top 10 documents: [738, 1045, 839, 843, 1046, 678, 891, 491, 1051, 1121]
  - Relevant documents: [567, 564, 566, 539]
  - Observation: Some are completely unrelated (about some polynomial equation). Others contain keywords present in the query like "pressure" and "cylinder"

- Example 3:
  - Query 216: "what investigations have been made of the wave system created by a static pressure distribution over a liquid surface ."
  - Top 10 documents: [958, 175, 764, 1156, 407, 971, 367, 64, 1225, 1220]
  - Relevant documents: [156, 506]
  - Observation: The relevant documents are about "shallow-water wave resistance". The query does not mention the specific term. The model is unable to learn that wave resistance is related to the query.

These observations suggest that a number of improvements can be made on this basic IR model such as incorporating semantic relationships for better performance. A number of techniques have been attempted for the same, and the better performing ones have been tested for significance. These methods have been detailed in the rest of the paper.

2 Methods

2.1 Dataset

The Cranfield dataset is used for the experiments in this paper. It is a collection of documents and queries in the aerospace engineering domain. The documents contain abstracts of scientific publications in this domain. This dataset contains 1400 documents (cran_docs.json), 225 queries (cran_queries.json), and query-document relevance judgements (cran_qrels.json).

The positions of the reference documents for each query (in cran_qrels.json) indicate the following judgements:

1 - References which are a complete answer to the question.
2 - References of a high degree of relevance, the lack of which either would have made the research impracticable or would have resulted in a considerable amount of extra work.
3 - References which were useful, either as general background to the work or as suggesting methods of tackling certain aspects of the work.
4 - References of minimum interest, for example, those that have been included from an historical viewpoint.

Query-Reference pairs in which the reference is of no interest to the query are excluded from the relevance file.

2.2 Techniques

2.2.1 Explicit Semantic Analysis

Selection of Concepts for the Corpus: Due to the large size of the full Wikipedia corpus and the limited computational resources available, we opted to use a subset of Wikipedia articles to define the concept documents for ESA. This subset was selected by extracting keywords directly from the Cranfield dataset, which consists of research paper abstracts primarily in aerospace and mechanical engineering. By creating a domain-specific concept corpus that aligns with the keyword space of the Cranfield dataset, we aimed to generate more meaningful and dense ESA vectors. Keyword extraction was performed using KeyBERT, a pretrained BERT-based model that ranks keywords based on semantic similarity. We used KeyBERT to extract and rank keywords from the Cranfield texts, selecting the top n keywords to generate corpora of varying sizes. These keyword sets were saved to separate files, enabling experimentation with different corpus sizes during ESA vector generation.

Corpus Generation: Corpora were generated by querying the Wikipedia API for summaries corresponding to each set of top k keywords. This was done using Python's requests library. Prior to querying, all keywords were preprocessed with the WordNetLemmatizer from the NLTK library to normalize terms and improve API match quality. To speed up the process, API requests were executed in parallel. Each keyword set produced a separate corpus, which was stored in individual JSON files. These JSON files were later used as concept bases for computing ESA embeddings.

ESA Vector Generation: The ESA pipeline begins by preprocessing both the concept corpus and input text using tokenization, stopword removal, and lemmatization. A TF-IDF vectorizer is then trained on the processed Wikipedia summaries of top-ranked keywords to construct a concept-document matrix. Input text is tokenized into sentences, each transformed into TF-IDF vectors, and their cosine similarity to the concept matrix is computed. The final ESA vector is the mean similarity score across all sentences, with each dimension corresponding to a concept keyword. To optimize performance, TF-IDF models and matrices are cached and reused unless missing. The implementation returns both the raw ESA vector and a dictionary mapping each concept label to its semantic relevance score.

2.2.2 Spreading Activation

Spreading activation simulates how information or relevance propagates through a network, with the network's nodes representing words and the edges indicating their semantic relationships. In the context of information retrieval, this technique allows for the expansion of a query's relevance to related terms in the document corpus, leading to more accurate and contextually relevant results.

The process begins by preprocessing both the query and the documents to extract meaningful, lemmatized terms. This involves removing non-alphabetic characters, eliminating common stop-words, and reducing words to their base forms using lemmatization. After this preprocessing, the query is represented as a sparse vector using the TF-IDF method, which measures the importance of words within the context of the entire corpus. Each document in the collection is similarly converted into a TF-IDF vector.

To capture the semantic relationships between words, we utilize WordNet, a lexical database that groups words into sets of synonyms, known as synsets, and establishes hierarchical relationships between them. For each word in the query and the documents, we compute its similarity to other words based on their shared synsets in WordNet. This semantic similarity is calculated using path similarity, which measures how closely related two words are within the WordNet hierarchy. The resulting similarity scores form a matrix representing the semantic relationships between words across the entire corpus.

Once the semantic similarity matrix is constructed, the spreading activation process is applied. The initial relevance of words in the query (their TF-IDF scores in the query) is propagated through the similarity network. The relevance of words in the query is transferred to related words in the vocabulary, with the transfer strength determined by the semantic similarity between the words. This spreading of activation allows the relevance of the query to affect not only the words directly contained within it but also the terms that are semantically related to them.

The final step involves computing the overall relevance of each document by combining the spread activation with the initial document relevance, which is represented by the TF-IDF matrix. The resulting document relevance scores are used to rank the documents, with the most relevant documents appearing at the top.

2.2.3 Neural Embeddings

Using *neural embeddings* for IR specifically for this use case involves two major processes - fine-tuning the GloVe Embeddings for ranking and integrating the fine-tuned embeddings into an IR system. The former is implemented by creating a training dataset from Cranfield, building a simple ranking model that uses an embedding layer, and training the model with a pairwise ranking loss. The latter involves using the fine-tuned model to generate representations for both queries and documents, then computing cosine similarities between query and document vectors to generate a ranking. Finally, standard IR evaluation metrics (precision, recall, MAP, nDCG, etc.) are computed. The implementation follows the following steps:

Data Preparation:

Cranfield documents (cran_docs): A JSON file contains the Cranfield documents (cran_docs). Typically, every document has an author, a body, and an ID. The body of each document is lemmatized, converted to lowercase, filtered to eliminate stopwords and non-alphabetic tokens, and tokenized using NLTK's word tokenizer. A list of tokens for every document is produced as a result.

Cranfield Queries (cran_queries): In JSON, queries are similarly stored. A query number and the query text are included in every entry.The query text is subjected to the same tokenization and cleaning procedures.

Relevance Assessments (cran_qrels): Using a query_num field and a document id field, these relevance judgments provide a list of the document IDs deemed pertinent for every query. In addition to being used for evaluation, the qrels are the foundation for creating training samples.

Building the Vocabulary: A vocabulary (word2idx) is created by processing the documents as well as the queries. Tokens are mapped to numerical indices by the vocabulary, which also sets aside an index (usually 0) for padding. Because it aligns text tokens with pre-trained GloVe embeddings, this vocabulary is essential.

Creating the Embedding Matrix and Loading the Pre-Trained GloVe:

Embeddings for GloVe: A GloVe file that has already been trained, like glove.6B.200d.txt, is read. If there is a GloVe vector that matches each word in your vocabulary, it is added to an embedding matrix (of shape [vocab_size, embed_dim]). Random initialization can be applied to non-appearing words. The weights of an embedding layer are initialized using this matrix.

Producing Training Samples (Triples):

Formation of Triplets: For every query, a triplet is created in order to train the ranking model. Directly taken from the Cranfield queries is the query text. A document that shows up in the query's query queries is referred to as a positive document. Negative Documents are those selected at random from documents that were not deemed pertinent to the inquiry. The triplets' combined list serves as the fine-tuning training dataset.

Defining the Neural Ranking Model:

Embedding Layer: It is initialized with the pre-trained (and later fine-tuned) GloVe embedding matrix and set as trainable so that backpropagation adjusts the embeddings.

Representation Construction: For each input (query or document), after converting token sequences into embeddings, the model performs an average pooling along the sequence dimension to produce a single vector. A linear transformation (followed by a tanh activation) is then applied. The purpose of the transformation is to adjust the representations further and to better separate relevant from non-relevant items.

Fine-Tuning with a Ranking Loss:

Pairwise Hinge Ranking Loss function encourages the model to produce a representation for the query that is more similar (in cosine similarity) to the positive document representation than to the negative document by a margin. Finally, the model is trained on mini-batches of these triplets.

Model integration:

In Phase 2 , the fine-tuned model is integrated into the IR pipeline by defining a class (e.g., FineTunedInformationRetrieval) that computes vector representations for documents and queries. First, the Cranfield corpus (cran_docs.json) is preprocessed and tokenized using the same cleaning functions applied during training. Each tokenized document is then converted into a fixed-length sequence of indices using a helper function, with a max_len parameter (e.g., 100) controlling the sequence length—shorter sequences are padded, while longer ones are truncated. These sequences are fed into the fine-tuned model (in evaluation mode) to compute document embeddings, which are stored along with their corresponding IDs to serve as the searchable index. When a new query is processed, it undergoes the same preprocessing, indexing, and padding/truncating steps before being passed to the model to compute its vector representation. Cosine similarity is then calculated between the query vector and each document vector, and the documents are ranked based on these similarity scores to return the most relevant results. The evaluation is done same as how it was done to previous methods

2.2.4 Latent Semantic Analysis

Implementation Overview: Latent Semantic Analysis (LSA) provides a sophisticated approach to information retrieval by transforming textual data into a semantic vector space. The implementation explores multiple variations to capture semantic relationships beyond traditional keyword-based methods.

Variations Explored: Four distinct LSA implementations were investigated to compare semantic representation effectiveness. These include Basic LSA, LSA with Lemmatization, Non-Negative Matrix Factorization (NMF), and Singular Value Decomposition (SVD). Each approach offers unique insights into semantic space construction and document representation.

Data Preprocessing: Text normalization forms the critical first step in our analysis. The preprocessing pipeline encompasses multiple strategies: Basic LSA relies on standard tokenization and minimal preprocessing. LSA with Lemmatization introduces advanced linguistic normalization, reducing words to their base or dictionary form. This approach aims to consolidate variant word forms, potentially improving semantic matching.

Vectorization Techniques: Two primary vectorization methods were employed to convert preprocessed text into numerical representations. TF-IDF (Term Frequency-Inverse Document Frequency) weighting provides a nuanced representation by emphasizing distinctive terms. Count Vectorization offers an alternative baseline representation through raw term frequency matrices.

Dimensionality Reduction Approaches: The core of LSA lies in transforming high-dimensional text representations: Singular Value Decomposition (SVD) represents the classical LSA approach, decomposing the term-document matrix into orthogonal components. Non-Negative Matrix Factorization (NMF) offers an alternative method, enforcing non-negativity constraints that can produce more interpretable semantic representations.

Semantic Space Construction: Each implementation creates a unique approach to semantic mapping. SVD-based LSA decomposes the initial term-document matrix into three fundamental components: a term-concept matrix, a diagonal matrix of singular values, and a concept-document matrix. NMF approaches generate topic-based representations by identifying non-negative components that capture semantic patterns.

Search Mechanism: The search process transforms queries into the same semantic space as document representations. Cosine similarity serves as the primary metric for measuring vector proximity, computing semantic relatedness between query and document vectors. Each LSA variation provides a distinct perspective on semantic matching.

Performance Evaluation: Two primary metrics assess the search implementations: Mean Average Precision (MAP) evaluates ranking quality by computing average precision across multiple queries. Normalized Discounted Cumulative Gain (nDCG) measures ranking effectiveness with a logarithmic penalty for relevant documents appearing lower in results.

Dimensionality Optimization: Selecting the optimal number of dimensions represents a critical implementation challenge. The approach involves systematically testing multiple dimensional configurations, evaluating performance across different LSA variations. This process identifies the most effective dimensional representation that balances information preservation with noise reduction.

Comparative Analysis: Each LSA implementation reveals unique strengths: Basic LSA provides a foundational semantic representation. Lemmatization-enhanced LSA offers improved linguistic normalization. SVD demonstrates powerful linear algebraic decomposition. NMF introduces non-negative topic modeling approaches.

Implementation Characteristics: The LSA approaches offer significant advantages, including semantic understanding beyond keyword matching and efficient document representation. Limitations include static document representations and sensitivity to preprocessing parameters.

Conclusion: Our comprehensive exploration of Latent Semantic Analysis demonstrates the power of advanced semantic representation techniques. By capturing underlying conceptual relationships through multiple approaches, LSA transcends traditional search limitations, offering sophisticated methods for semantic document matching and retrieval.

2.3 Evaluation Metrics

The following standard metrics were used to evaluate the effectiveness of the models.

- **Precision** measures the fraction of retrieved documents that are relevant:

$$\text{Precision} = \frac{|\text{Relevant} \cap \text{Retrieved}|}{|\text{Retrieved}|}$$

- **Recall** quantifies the fraction of relevant documents that are retrieved:

$$\text{Recall} = \frac{|\text{Relevant} \cap \text{Retrieved}|}{|\text{Relevant}|}$$

– **F-score** is the harmonic mean of precision and recall, providing a balance between them:

$$\text{F-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

– **Normalized Discounted Cumulative Gain (nDCG)** evaluates the ranking quality of retrieved documents, giving higher weight to relevant documents appearing earlier in the ranked list:

$$\text{DCG}_p = \sum_{i=1}^{p} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}, \quad \text{nDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p}$$

where rel_i is the graded relevance of the result at position i, and IDCG_p is the ideal DCG (maximum possible DCG) up to position p.

All metrics are reported as mean $\pm$ standard deviation across multiple queries to account for variation in retrieval performance.

2.4 Hypothesis Testing

Raw observations from plots suggested that Vanilla VSM, ESA and LSA gave comparable results. Therefore we conducted hypothesis tesing to compare the performance of these models. A one-tailed independent two-sample t-test assuming is performed for effective comparison to test if one method is better than another. The comparison can broadly be stated as follows:

An algorithm A1 is better than A2 with respect to the evaluation measure E in task T on a specific domain D under certain assumptions A.

Here,

Algorithms (A1, A2) in comparison are (ESA, VSM), (LSA, VSM) and (ESA, LSA), one each for each test.

Evaluation measures under consideration are Precision, Recall, F score and nDCG. These metrics capture different aspects of effectiveness of the retrieval task.

Task T can be defined as the task of *ad hoc document retrieval* i.e., return the most relevant documents from a corpus, given a query.

Domain D can be defined as that of aerospace engineering academic documents as the dataset used for training and testing is the Cranfield Dataset.

A number of reasonable assumptions A are made to simplify the analysis without diluting the reliability of the test.

Assumptions

– The evaluation metric scores for each query are paired across models.
– The differences between paired scores are approximately normally distributed.

- The number of queries is sufficiently large for the Central Limit Theorem to be valid.
- One-tailed paired T-test holds good for these samples
- All the metrics are taken at rank 10

Hence, a total of 12 tests are performed across 3 methods and 4 metrics.

2.4.1 ESA vs VSM

Null Hypothesis H_0 ESA does not perform better than VSM across evaluation measures E on task T on domain D under assumptions A. i.e., the mean difference in metric scores $(ESA - VSM)$ is ≤ 0.

Alternate Hypothesis H_1 ESA performs significantly better than VSM across evaluation measures E on task T on domain D under assumptions A. i.e., the mean difference in metric scores $(ESA - VSM)$ is > 0.

2.4.2 LSA vs VSM

Null Hypothesis H_0 LSA does not perform better than VSM across evaluation measures E on task T on domain D under assumptions A. i.e., the mean difference in metric scores $(LSA - VSM)$ is ≤ 0.

Alternate Hypothesis H_1 LSA performs significantly better than VSM across evaluation measures E on task T on domain D under assumptions A. i.e., the mean difference in metric scores $(LSA - VSM)$ is > 0.

2.4.3 ESA vs LSA

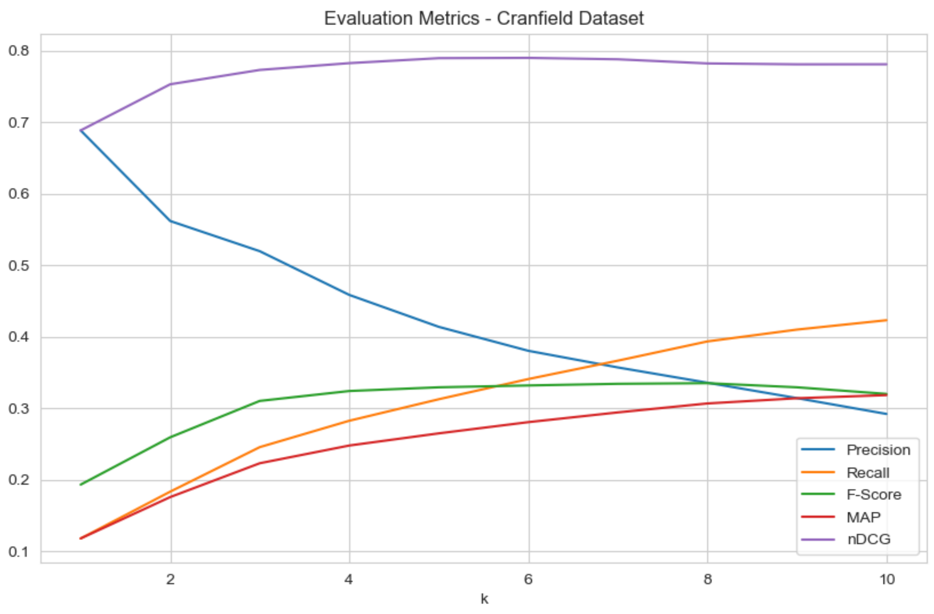
Null Hypothesis H_0 ESA does not perform better than LSA across evaluation measures E on task T on domain D under assumptions A. i.e., the mean difference in metric scores $(ESA - LSA)$ is ≤ 0.

Alternate Hypothesis H_1 ESA performs significantly better than LSA across evaluation measures E on task T on domain D under assumptions A. i.e., the mean difference in metric scores $(ESA - LSA)$ is > 0.

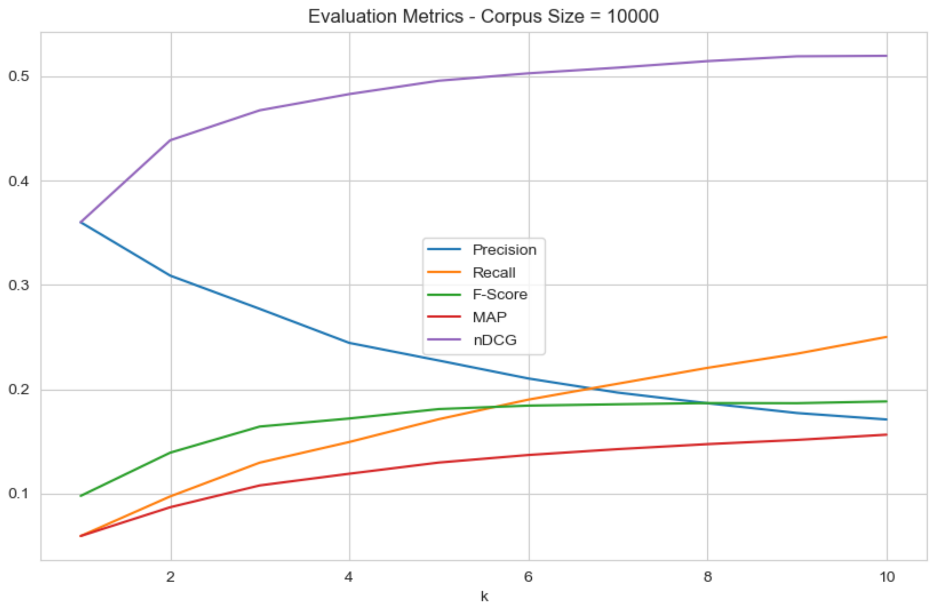
3 Results

Among the attempted techniques as improvements to VSM, the better performing ones were LSA and ESA. Hence, these two have been discussed in detail in this section.

3.1 Individual model performance

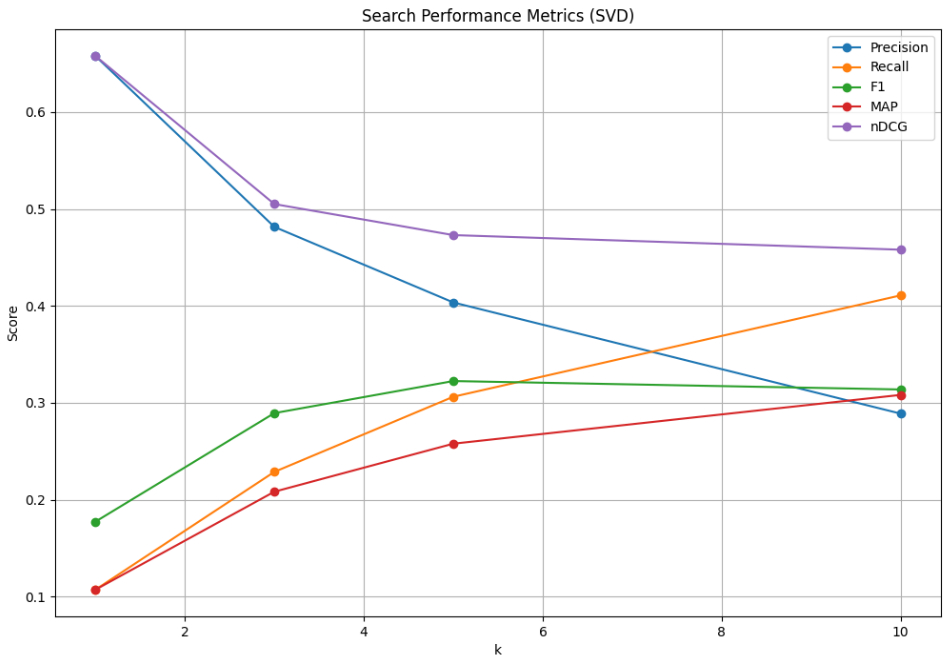


(a) Vector Space Model



(b) Explicit Semantic Analysis with $n = 10000$ (Corpus Size)

Fig. 1: Performance plots for VSM and ESA models



(a) Latent Semantic Analysis with SVD

Fig. 2: Performance plot for the LSA model

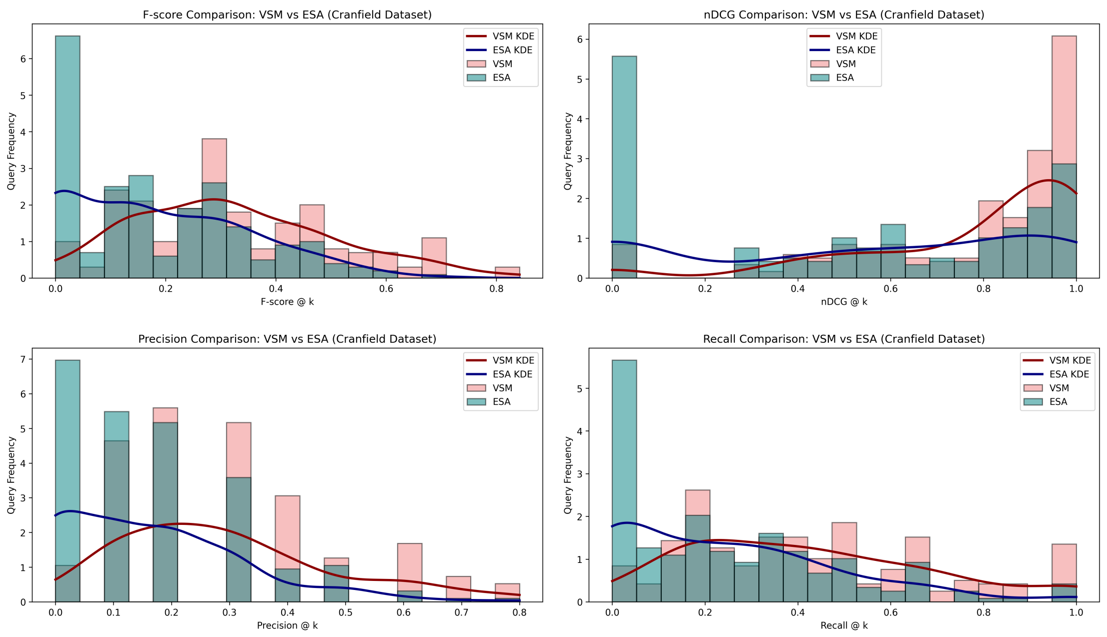
The metrics for individual model performance can be summarised as follows:

Model	**Precision**	**Recall**	**F-score**	**nDCG**
VSM	0.2924 ± 0.1857	0.4235 ± 0.2672	0.3204 ± 0.1815	0.7813 ± 0.2559
ESA	0.1649 ± 0.1574	0.2478 ± 0.2431	0.1836 ± 0.1644	0.5175 ± 0.3836
LSA	0.3049 ± 0.2016	0.4366 ± 0.2821	0.3330 ± 0.1969	0.7609 ± 0.2838

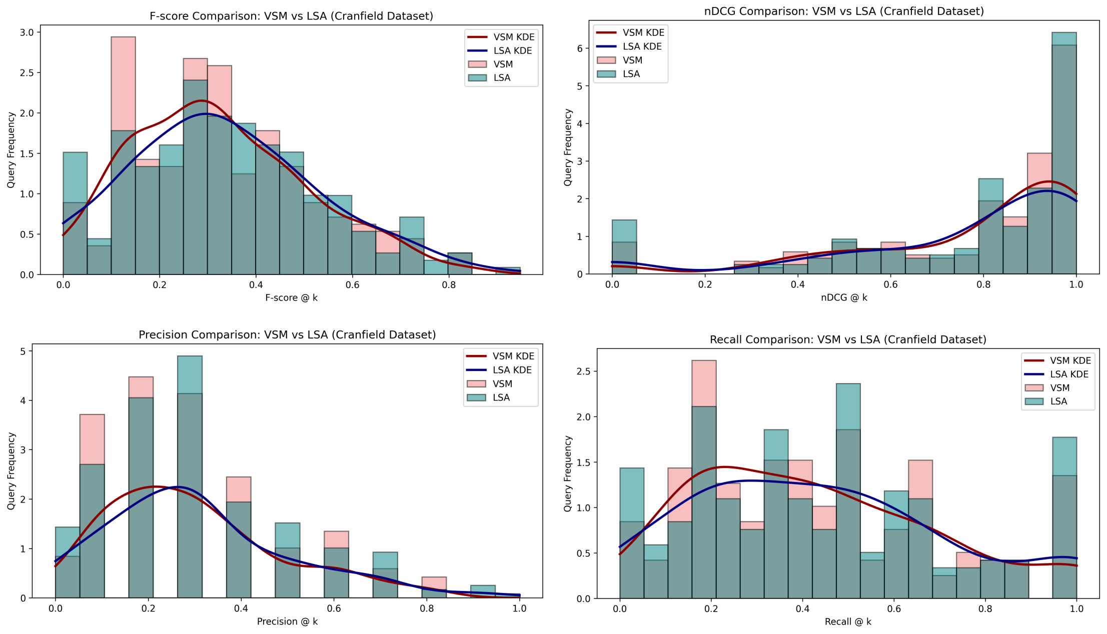
Table 1: Performance summary (mean ± std) of VSM, ESA, and LSA models.

3.2 Comparative model performance

Figures 3 and 4 show the plots for the comparative analysis done between each pair of methods that were implemented as part of this project.

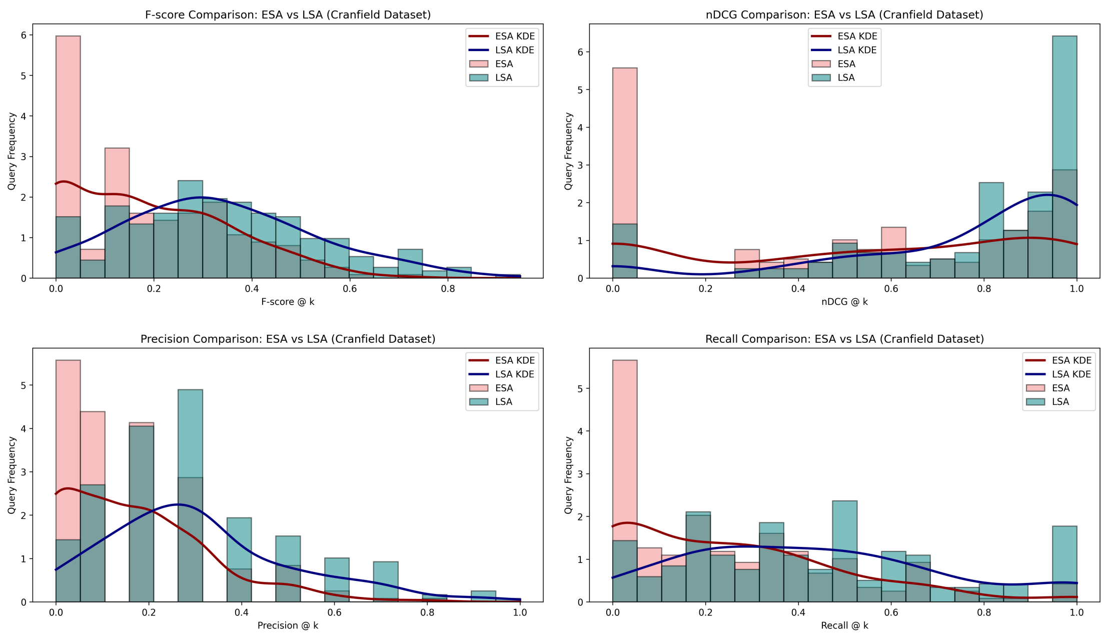


(a) VSM vs ESA comparison



(b) VSM vs LSA comparison

Fig. 3: Comparative analysis: VSM vs ESA and VSM vs LSA



(a) ESA vs LSA comparison

Fig. 4: Comparative analysis: ESA vs LSA

3.3 Hypothesis Test

Tables 2, 3 and 4 summarise the results of the hypothesis tests:

Metric	**t-statistic**	**p-value**	**Significant?**	**Better model**
Precision	7.8434	0.0000	Yes	VSM
Recall	7.2759	0.0000	Yes	VSM
F-score	8.3611	0.0000	Yes	VSM
nDCG	8.5657	0.0000	Yes	VSM

Table 2: Statistical comparison between VSM and ESA (one-tailed t-test).

4 Conclusion

Our evaluation demonstrates that the Vector Space Model (VSM) and Latent Semantic Analysis (LSA) perform comparably across standard retrieval metrics, showing solid baseline effectiveness. While we experimented with more advanced methods such as Spreading Activation and neural embedding-based approaches, these did not yield significant improvements in our current setup.

Metric	**t-statistic**	**p-value**	**Significant?**	**Better model**
Precision	-0.6796	0.7514	No	-
Recall	-0.5054	0.6932	No	-
F-score	-0.7018	0.7584	No	-
nDCG	0.8023	0.2114	No	-

Table 3: Statistical comparison between VSM and LSA (one-tailed t-test).

Metric	**t-statistic**	**p-value**	**Significant?**	**Better model**
Precision	-8.1928	1.0000	No	-
Recall	-7.5861	1.0000	No	-
F-score	-8.7164	1.0000	No	-
nDCG	-7.6351	1.0000	No	-

Table 4: Statistical comparison between ESA and LSA (one-tailed t-test).

Looking ahead, several enhancements could further strengthen the system, including incorporating spell checking and auto-complete mechanisms to better handle user query variability. Additionally, although Explicit Semantic Analysis (ESA) did not outperform in our current limited configuration, we anticipate that leveraging the full Wikipedia corpus would substantially boost its performance, making it a promising direction for future work.

5 Resources

The complete code and plots can be found on the following GitHub Repository:
https://github.com/petrichor247/Information-Retrieval-Systems

References

1. Grootendorst, M. (2020). KeyBERT: Minimal keyword extraction with BERT (Version v0.3.0) [Software]. Zenodo.
2. Gabrilovich, E., & Markovitch, S. (2007). Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07) (pp. 1606–1611). Morgan Kaufmann Publishers.
3. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41(6), 391–407.
4. Ngo, V. M., Cao, T. H., & Le, T. M. V. (2010). Combining named entities with WordNet and using query-oriented spreading activation for semantic text search. In 2010 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF) (pp. 1–6). IEEE.
5. Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).