IRS: Information Retrieval System

Ahmed Shmels (GE22M009)[1]

[1]Department of Data Science & Artificial Intelligence, IIT Madras

March 12, 2025

Abstract

The goal of this project is to develop an efficient information retrieval (IR) system utilizing NLP techniques. Our experiment begins with training a simple Vector Space Model (VSM), which serves as the baseline model. Two baseline models were constructed using different pre-processing techniques. We addressed some limitations of the VSM through techniques such as Latent Semantic Analysis (LSA) and Explicit Semantic Analysis (ESA). Additionally, we explored query expansion by incorporating similar words to the query. The report briefly discusses the improvements of these models and compares them to the baseline model using statistical methods. Furthermore, methods to significantly reduce search time, such as K-Means Clustering and Latent Dirichlet Allocation (LDA) for topic modelling, are discussed. We compared retrieval times with and without clustering using a two-sample t-test. Finally, we implemented a method to enhance user experience during real-time querying, such as query auto-completion using Long Short-Term Memory (LSTM) networks.

Keywords: LDA, K-Means, Query auto-completion, LSA, QE), ESA, LSTM.

1 Introduction

1.1 Vector Space Model (VSM)

The Vector Space Model (VSM) [1] represents documents and queries as vectors in a multi-dimensional space. Each dimension corresponds to a term (word) in the vocabulary. In a basic VSM, the components of these vectors are calculated using the term frequency (TF) within the document and the inverse document frequency (IDF) across the entire corpus [2].

During information retrieval, documents are ranked based on the cosine similarity between their vectors and the query vector. This cosine similarity, which is essentially the cosine of the angle between the two vectors, serves as a measure of relevance.

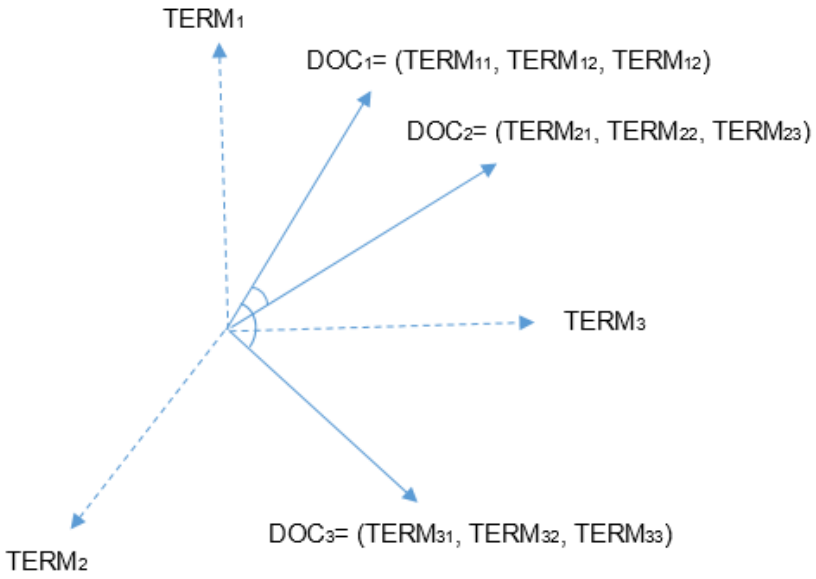


Fig. 1: VSM: Representing docs and terms as vectors in a multi-dimensional space.

1.2 Latent Semantic Analysis (LSA)

Latent Semantic Analysis (LSA), also known as Latent Semantic Indexing (LSI), is a statistical technique in natural language processing (NLP) that uncovers latent (hidden) relationships between words and concepts within a collection of documents [3]. It operates under the principle of distributional semantics, assuming that words with similar meanings tend to occur in similar contexts [4].

LSA leverages singular value decomposition (SVD) to decompose a term-document matrix into a lower-dimensional representation [3]. This reduced representation captures underlying semantic relationships that might not be apparent from the raw term frequencies alone. However, LSA is unable to explicitly capture multiple senses of a word (polysemy), nor does it take into account that every word occurrence is typically intended to refer to only one meaning at a time [5]. Therefore, while LSA can effectively address issues of synonymy (different words with similar meanings), it may not fully resolve the polysemy problem.

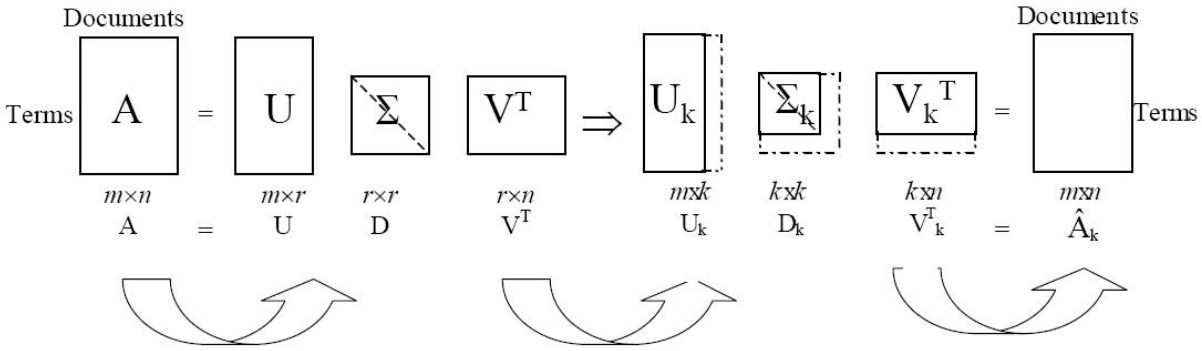


Fig. 2: Illustration of LSA dimensionality reduction.

1.3 Query Expansion (QE)

Query Expansion (QE) is a technique in information retrieval (IR) that enhances a user's initial query Q by adding relevant terms or phrases [6]. This aims to improve retrieval effectiveness by addressing vocabulary mismatch and query ambiguity.

How QE Works:

1. **Term Selection:** Relevant terms $T_0 = \{t_{01}, t_{02}, \ldots, t_{0m}\}$ are identified from sources like thesauruses, top-ranked documents, or user feedback.

2. **Term Weighting:** Added terms are assigned weights based on their importance relative to the original query terms.

3. **Query Reformulation:** The expanded query Q_{exp} combines the original and added terms, often using logical operators (AND, OR):

$$Q_{\text{exp}} = (Q - T_{00}) \cup T_0 = \{t_1, t_2, \ldots, t_i, t_{01}, t_{02}, \ldots, t_{0m}\}$$

   where T_{00} represents removed stop words.

Example: The query "flu symptoms" might be expanded to "flu symptoms fever cough chills."

QE has proven effective in improving retrieval, particularly for short or ambiguous queries [6, 7]. However, its success depends on the quality of expansion terms and chosen strategies.

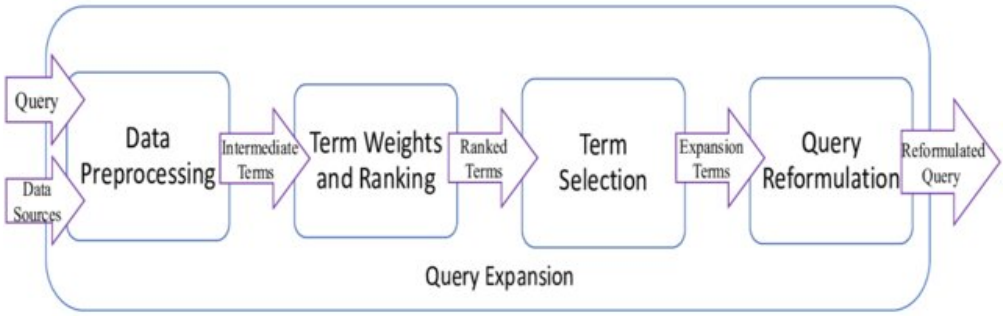


Fig. 3: Illustration of Query Expansion Techniques for Information Retrieval.

With the ever-increasing size of the web, relevant information extraction using a few keywords has become a challenge. QE plays a crucial role in improving Internet searches by reformulating the user's initial query with meaningful terms of similar significance [7]. This technique has been influential in personalized search, question answering, and text retrieval tasks.

1.4 Clustering of Documents

Clustering of documents is a technique used in information retrieval (IR) to group similar documents together based on their content [2]. This can improve retrieval efficiency and effectiveness by reducing the search space and identifying relevant documents more quickly. Two common methods used for document clustering are:

1.4.1 K-Means Clustering

This is a partitioning algorithm that aims to divide a set of documents into K distinct clusters [8]. Each document is assigned to the cluster with the nearest centroid (the mean of the documents in the cluster). The algorithm iteratively refines the cluster assignments and centroids until convergence. K-means is efficient but requires specifying the number of clusters in advance.

1.4.2 Latent Dirichlet Allocation (LDA) Topic Modeling

LDA is a probabilistic generative model that assumes each document is a mixture of latent topics, and each topic is a distribution over words [8]. LDA learns these topic distributions from the corpus and can be used to assign topic probabilities to each document. Documents with similar topic distributions can then be grouped.

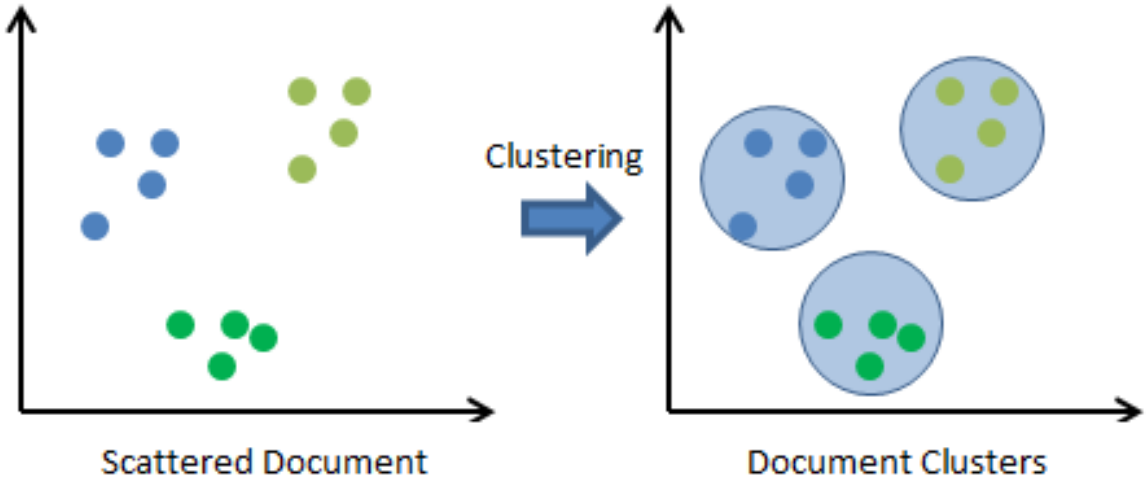


Fig. 4: K-Means: Text Documents Clustering using Algorithm

1.5 Query Auto-Completion with LSTM

Query auto-completion (QAC) enhances user experience by predicting and suggesting the next words or phrases as a user types a query. One effective approach to QAC leverages Long Short-Term Memory (LSTM) neural networks, a type of recurrent neural network (RNN) designed to capture sequential dependencies in data [9].

How LSTM-Based QAC Works:

1. **Training:** An LSTM model is trained on a large dataset of search queries. The model learns the patterns and relationships between words in the queries, essentially predicting the probability of the next word given the preceding context.

2. **Prediction:** As a user types a partial query, the trained LSTM model processes the input sequence and generates probabilities for the most likely next words.

3. **Suggestion:** The top-ranked words with the highest probabilities are presented to the user as auto-completion suggestions.

2 Methodology & Implementation

2.1 Vector Space Model (VSM)

A vector space model (VSM) is created using the Term Frequency-Inverse Document Frequency (TF-IDF) scores of words in the documents. We have implemented two versions of the VSM based on different pre-processing techniques:

- **VSM-1:** Vector Space Model Version
- **VSM-2:** Vector Space Model Version 2

The detailed procedures for each version are provided in the following subsections.

2.1.1 VSM-1

In this model, we adopted pre-processing techniques from NLP assignments. The steps involved in building the model are:

Data Pre-processing

- **Tokenization:** Each document is divided into a list of sentences, and each sentence into a list of words using the Treebank tokenizer from the NLTK package.
- **Stopword Removal:** Common words (stopwords) are removed to reduce the dimensionality of the VSM model.
- **Inflection Reduction (Lemmatization):** Words are reduced to their root forms using lemmatization [6].
- **TF-IDF Matrix Creation:** A TF-IDF matrix is constructed, where each column (representing a document) is normalized to unit length.

Ranking and Evaluation

- **Ranking:** Similar pre-processing steps are applied to queries, yielding query vectors. Cosine similarity is used to measure the similarity between document and query vectors, and documents are ranked based on these similarity scores.
- **Evaluation:** The ranked documents are evaluated using metrics like Mean Precision@k, Mean Recall@k, Mean Average Precision (MAP)@k, normalized Discounted Cumulative Gain (nDCG)@k, and Mean F-Score@k.

2.1.2 VSM-2: Improvements over VSM-1

In this model, we introduce several changes to the data pre-processing techniques of VSM-1, leading to improved performance across all evaluation metrics.

Data Pre-processing

- **Lower casing:** All text is converted to lowercase to ensure consistency and avoid treating words like "Aeroplane" and "aeroplane" as different terms, which would increase the dimensionality of the VSM.
- **Number Removal:** All numbers are removed from the corpus, as they often do not contribute to semantic meaning.
- **Punctuation & Diacritics Removal:** Punctuation marks, accent marks, and other diacritics, along with extra whitespace, are removed. This prevents the VSM from considering punctuation as separate terms, which would negatively impact model performance.
- **Further Pre-processing:** The remaining steps, including tokenization (using the same tokenizer as in VSM-1), stopword removal, lemmatization (preferred over stemming), and TF-IDF matrix construction, are identical to VSM-1.

Ranking and Evaluation

The ranking and evaluation processes are the same as in VSM-1:

- **Ranking:** Queries undergo the same pre-processing steps, and query vectors are obtained. Cosine similarity is used to measure the similarity between document and query vectors, with documents ranked accordingly.
- **Evaluation:** Ranked documents are assessed using metrics such as Mean Precision@k, Mean Recall@k, MAP@k, nDCG@k, and Mean F-Score@k.

2.2 Results & Evaluation

As shown in Table 1, the VSM-2 model outperforms the VSM-1 model across all evaluation metrics.

Table 1 shows that VSM-2 outperforms VSM-1 by 3% to 5% across all metrics. The vocabulary size in VSM-1 was around 8,000 words, whereas in VSM-2 it was reduced to 5,000 words due to improved text pre-processing. Some punctuation was left unremoved in VSM-1, leading to poorer similarity scores and reduced model performance. VSM-2 addressed this by incorporating corpus-specific pre-processing steps.

The distributions of precision, recall, F-score, and nDCG for various queries at $k = 7$ for both models are shown in Figures 5 to 8.

Table 1: Comparison of VSM-1 and VSM-2 Evaluation Results

	VSM-1					VSM-2				
k	**MAP**	**Recall**	**F-score**	**nDCG**	**Prec.**	**MAP**	**Recall**	**F-score**	**nDCG**	**Prec.**
1	0.636	0.110	0.181	0.493	0.636	0.693	0.120	0.193	0.530	0.693
2	0.695	0.180	0.260	0.400	0.538	0.733	0.188	0.266	0.412	0.567
3	0.699	0.230	0.291	0.372	0.467	0.734	0.241	0.302	0.390	0.501
4	0.695	0.264	0.303	0.364	0.411	0.732	0.287	0.328	0.390	0.461
5	0.685	0.294	0.309	0.364	0.372	0.721	0.313	0.328	0.391	0.412
6	0.677	0.323	0.314	0.372	0.343	0.710	0.342	0.331	0.396	0.379
7	0.668	0.347	0.315	0.376	0.320	0.697	0.368	0.334	0.402	0.352
8	0.658	0.364	0.311	0.383	0.301	0.687	0.389	0.330	0.407	0.332
9	0.653	0.382	0.310	0.388	0.285	0.680	0.406	0.325	0.411	0.311
10	0.644	0.400	0.304	0.393	0.267	0.674	0.419	0.318	0.414	0.291

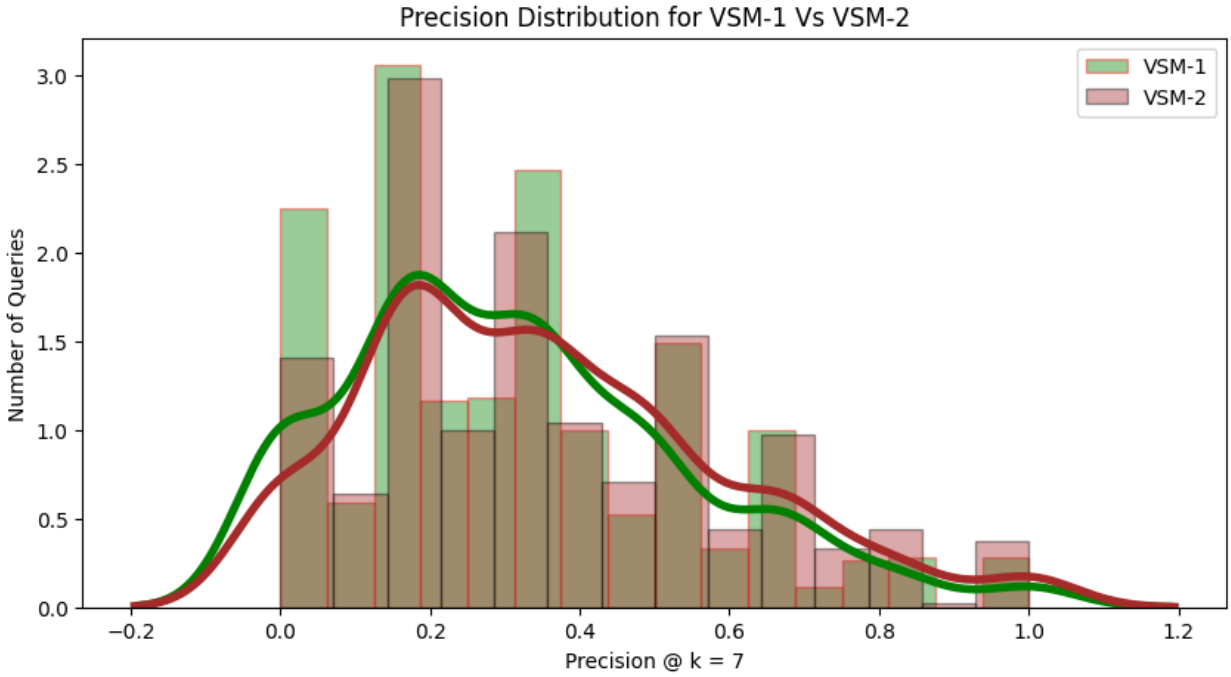


Fig. 5: Precision Distribution for VSM-1 vs. VSM-2 at k=7

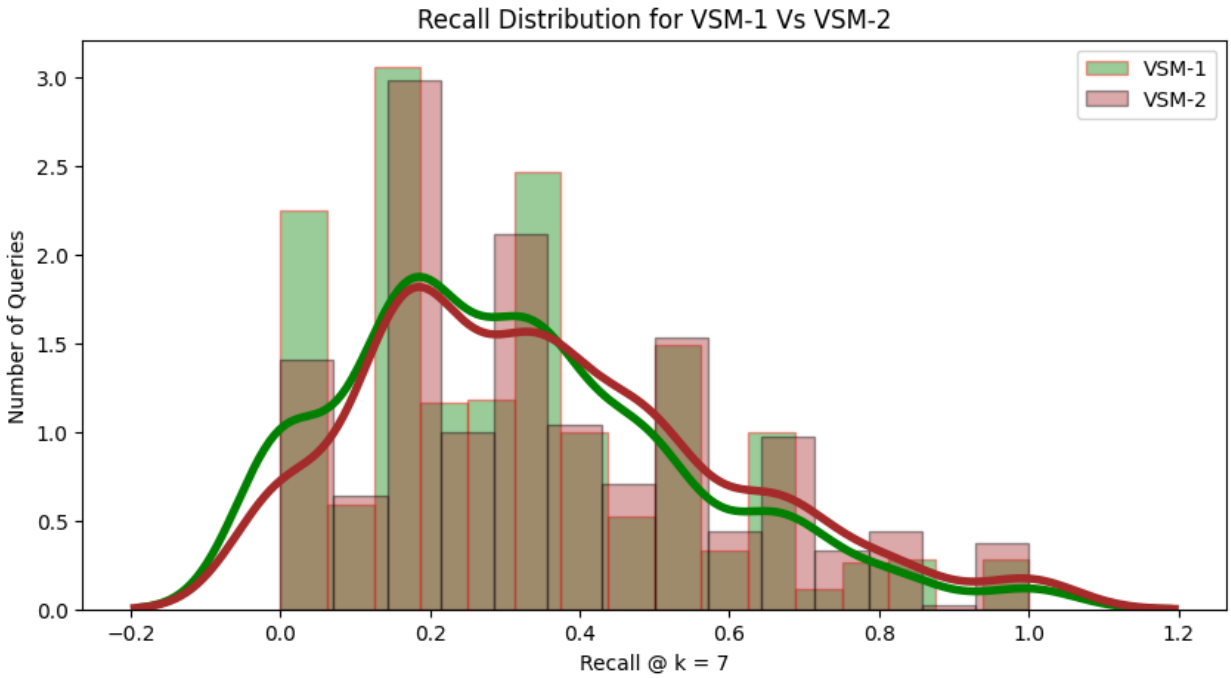


Fig. 6: Recall Distribution for VSM-1 vs. VSM-2 at k=7

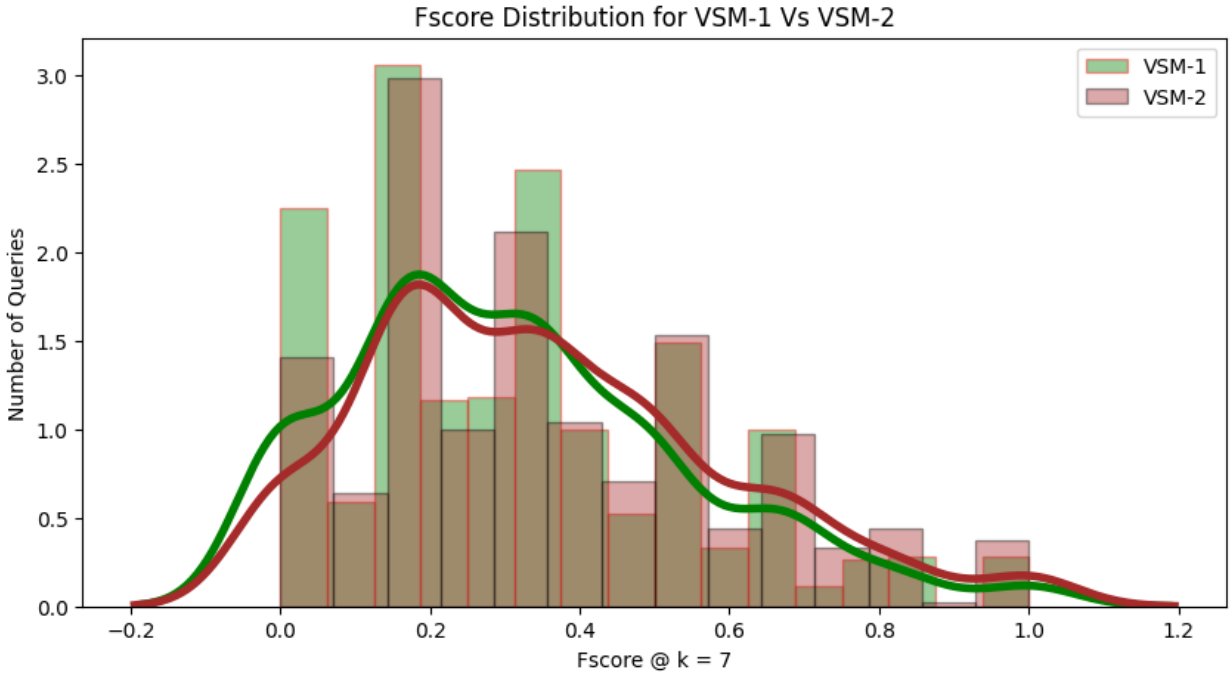


Fig. 7: F-Score Distribution for VSM-1 vs. VSM-2 at k=7

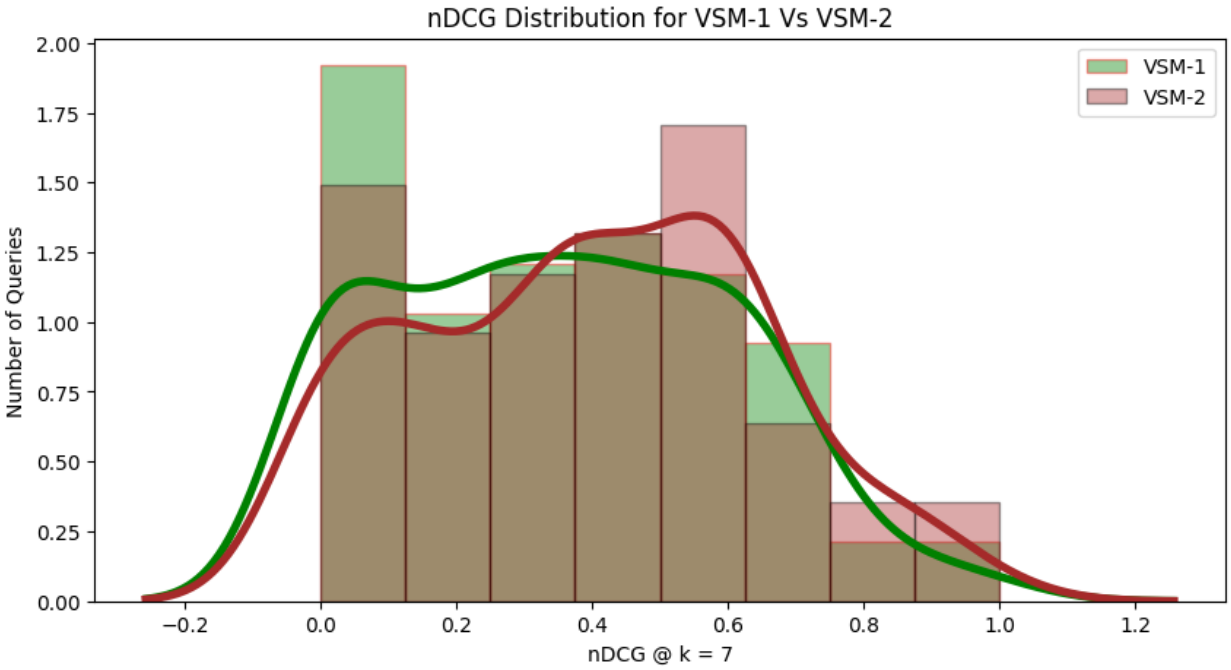
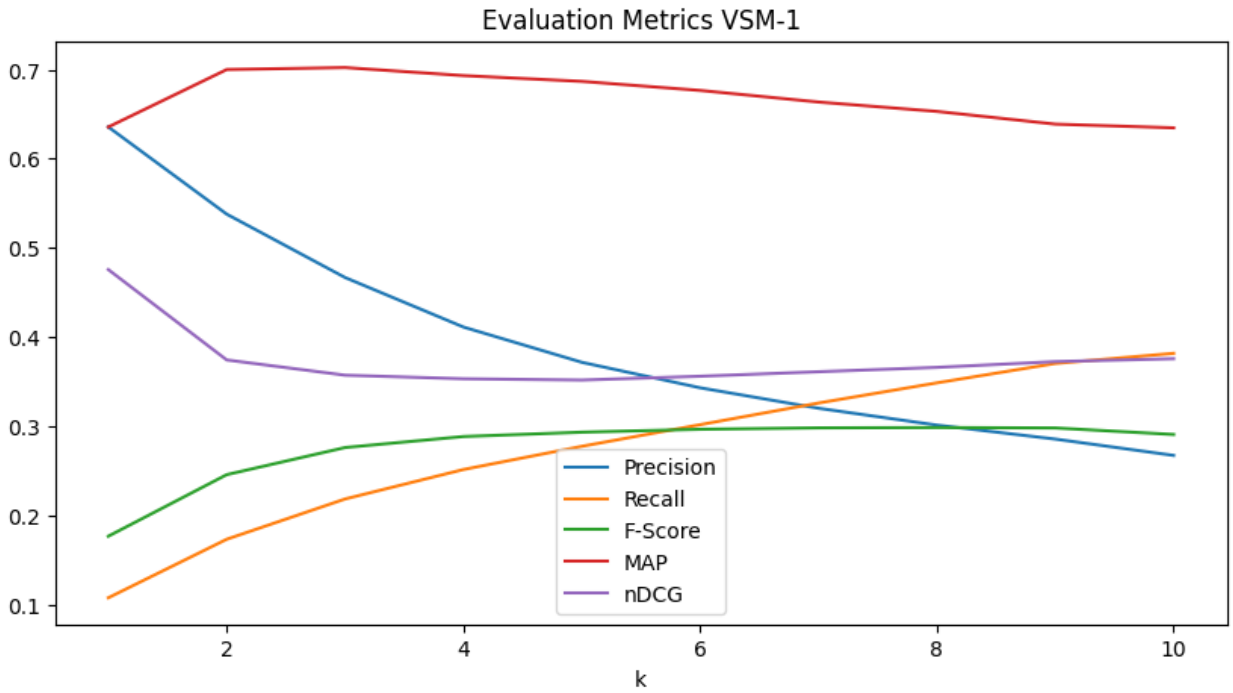


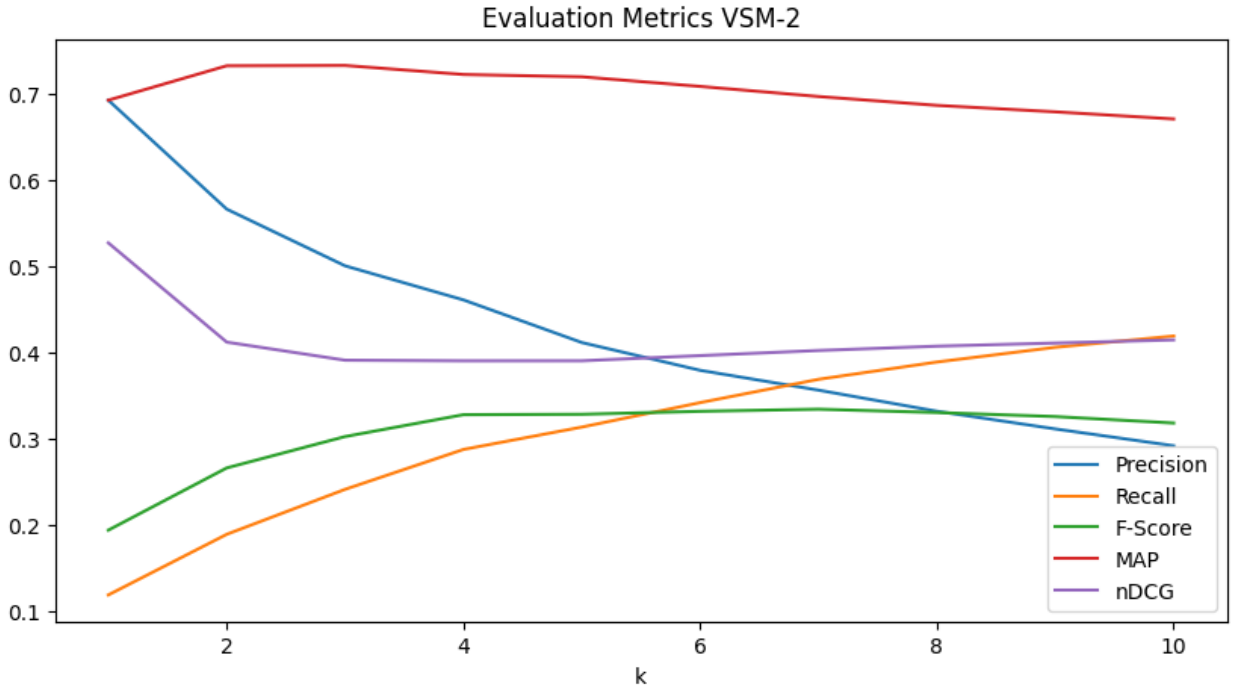
Fig. 8: nDCG Distribution for VSM-1 vs. VSM-2 at k=7

Inference: Figures 5 to 8 clearly illustrate that VSM-2 shifts lower metric values to higher ones, indicating superior performance. Further analysis, such as hypothesis testing, can confirm the statistical significance of this difference.

The general trends of the evaluation measures for all models are as shown in Figure 9.



(a) VSM-1: Results and Evaluation



(b) VSM-2: Results and Evaluation

Fig. 9: Comparison of Evaluation Results for VSM-1 and VSM-2

From now onwards, we use VSM-2 as our Baseline model throughout this report.
Few Examples where the retrieved documents are completely irrelevant.

- **Query ID:** 28
- **Query:** "What application has the linear theory design of curved wings."
- **Documents Retrieved:** [752, 1075, 762, 674, 921, 1051, 247, 451, 1050, 680]
- **Doc 752:** " . . . Slender not-so-thin **wing theory**. A method for making an approximate thickness correction to slender thin-**wing theory** is presented. The method is tested by applying it to cones with rhombic cross-sections and the agreement is found to be good if the cones are not too thick . . . ."
- **Relevant Documents:** [224, 279, 512]
- **Explanation:** The retrieved document by the VSM-2 model has more number of exact matching words to the given query. But for the relevant document, we see there are very few exactly matching words, this is the reason why our VSM-2 model could not retrieve the relevant document.

Few Examples where the retrieved documents are all relevant.

- **Query ID:** 185
- **Query:** "Experimental studies on panel flutter."
- **Documents Retrieved:** [856, 1008, 766, 857, 859, 858, 391, 948, 658, 864]
- **Relevant Documents:** [858, 859, 857, 1008, 856, 15, 285, 894, 766, 948]
- **Doc 856:** " . . . Experimental investigation at Mach numbers 3.0 of the effects of thermal stress and buckling on the flutter of four-bay aluminium alloy **panels** with length-width ratios of 10. Skin-stiffener aluminum alloy panels consisting of four bays, each bay having a length-width ratio of 10, were tested at a Mach number of 3.0 at dynamic pressures ranging from 1,500 psf to 5,000 psf and at stagnation temperatures from 300 F to 655 F, skin-stiffener aluminium alloy **panels** consisting of four bays, each bay having a length-width ratio of 10 . . . "
- **Explanation:** The relevant document contains a lot of exactly matching words similar to the query, so we could retrieve the relevant document.

2.3 Observations

- Precision (Mean precision) decreases and Recall increases as k increases. This implies that more relevant documents are retrieved initially, as most of the top retrieved documents are relevant.
- F-score increases and remains almost constant. This implies a stable trade-off balance between precision and recall. It was low initially due to low recall values.
- MAP forms a smoother curve with less variation compared to the precision curve. It decreases slowly compared to the precision curve.

- For nDCG, it decreases and becomes saturated. This was implemented by changing relevance scores for query positions 1 to 4 (3 to 0, 2 to 1, 1 to 2, 0 to 3). The nDCG curve shows that the IR system could perform better (as the ideal nDCG equals 1), but it shows more or less constant variation as k is increased.

2.4 Hypothesis Testing

- **Null Hypothesis:** VSM-1 and VSM-2 perform similarly in terms of precision, recall, F-score, and n-DCG.
- **Alternative Hypothesis:** VSM-1 and VSM-2 do not perform similarly in terms of precision, recall, F-score, and n-DCG.
- **Approach:** We observed the precision, recall, F-score, and n-DCG for VSM-1 and VSM-2 on 225 queries in the Cranfield queries. Then, we applied a two-sample two-tailed t-test to evaluate the hypothesis on each individual metric.
- **Result of t-test:**

Table 2: Results of t-test

Metric	**t-statistic**	**p-value**
Precision	-1.69241	0.0912622
Recall	-1.76362	0.0784774
FScore	-1.90723	0.0571297
nDCG	-1.75331	0.0802332

- **Conclusion:** All the p-values are greater than 0.05, indicating no significant difference. Thus, we cannot reject the null hypothesis. We conclude that VSM-1 and VSM-2 perform similarly for a given query.

2.5 Limitations of Vector Space Model

- **Curse of Dimensionality:** Long documents are poorly represented because they have poor similarity values (a small scalar product and a large dimensionality).
- **Orthogonality Assumption:** The vector space model assumes the orthogonality of dimensions (words), but it is not true in general.
- **Semantic Sensitivity:** Documents with similar context but different term vocabulary won't be associated.
- **Order Sensitivity:** The order in which the terms appear in the document is lost in the vector space representation.

- **Sparsity:** The document vectors are highly sparse, leading to poor similarity values between documents and also between queries and documents.

From the next section onwards, we will try to address these limitations.

3 Document Representation & Orthogonality (Limitations 1 & 2)

A fundamental assumption in the Vector Space Model (VSM) is that similar words occur in similar documents, and vice versa. However, this creates a circular dependency that can be addressed through the use of latent variables called "concepts." Latent Semantic Analysis (LSA) maps documents into a concept space (analogous to projection onto eigenvectors), allowing for similarity assessment based on higher-order associations [3].

LSA offers two key advantages:

- **Dimensionality Reduction:** By projecting documents onto a lower-dimensional concept space, LSA reduces the storage requirements for representing large document vectors, improving efficiency.
- **Synonymy Mitigation:** LSA partially addresses the orthogonality problem inherent in VSM. It mitigates the challenge of identifying synonymy (words with similar meanings) by merging dimensions associated with these terms, thereby capturing higher-order associations between words.

Despite these improvements, LSA still faces limitations. It does not explicitly handle polysemy (words with multiple meanings), and its concept space may not fully capture all nuances of word relationships.

3.1 Latent Semantic Analysis (LSA)

Latent Semantic Analysis (LSA) is a statistical technique used in natural language processing (NLP) to uncover latent semantic structures in textual data [3]. It assumes that the observed term-document matrix is a noisy representation of an underlying semantic structure, partially obscured by the variability of word choice. LSA utilizes Singular Value Decomposition (SVD), a matrix factorization method, to identify this latent structure and reduce noise [2].

Motivations for Applying LSA

There are several motivations for applying LSA to the term-document matrix:

1. **Dimensionality Reduction:** The original matrix may be too large for efficient computation. LSA provides a lower-rank approximation that captures the essential semantic information while reducing computational complexity.

2. **Noise Reduction:** The original matrix may contain noise due to anecdotal occurrences of terms. LSA can filter out this noise, leading to a "denoised" representation that better reflects the underlying semantic relationships.

3. **Sparsity Mitigation:** The original matrix may be overly sparse, reflecting only the words explicitly present in each document. LSA can help discover latent relationships between documents and terms that share semantic meaning but may not directly co-occur, thus addressing synonymy issues.

LSA Implementations

We implemented two versions of LSA:

1. **LSA Version 1:** Applied to a TF-IDF matrix constructed from the Cranfield document collection.

2. **LSA Version 2:** Applied to a TF-IDF matrix constructed from both the Cranfield and Brown corpora. This expanded corpus was used to potentially capture a wider range of semantic relationships.

Procedure

The general procedure for both LSA versions is as follows:

1. **TF-IDF Matrix Construction:** Create a term-document matrix using TF-IDF weights to represent the importance of terms within documents.

2. **SVD and Rank-k Approximation:** Apply SVD to the term-document matrix and obtain a rank-k approximation. The value of k (number of concepts) is a hyperparameter tuned using grid search cross-validation. The resulting eigenvectors represent the latent semantic concepts.

3. **Transformation and Ranking:** Transform both the query and document vectors into the concept space. Calculate the cosine similarity between the query vector and each document vector in this reduced space. Rank the documents based on their similarity scores.

3.1.1 LSA Hyperparameter Tuning

A critical hyperparameter in LSA is the number of latent dimensions (k), representing the number of concepts extracted from the data. To determine the optimal value of k, we conducted a grid search cross-validation across a range of values, evaluating performance using Mean Average Precision (MAP), recall, normalized Discounted Cumulative Gain (nDCG), and other metrics.

As illustrated in Figure 10, we observed that the optimal performance across evaluation measures was consistently achieved around k = 500. This suggests that 500 latent dimensions effectively capture the underlying semantic structure within the dataset, striking a balance between dimensionality reduction and preserving crucial information.

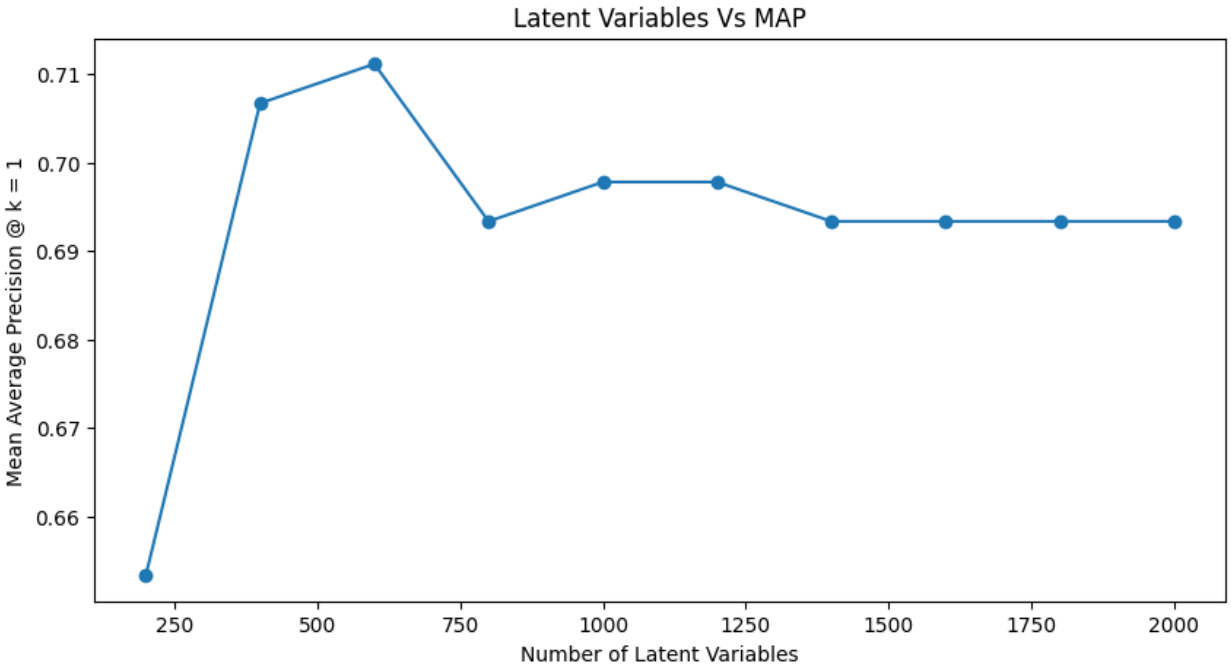


Fig. 10: Mean Average Precision (MAP) as a function of the number of latent variables (k) in LSA.

3.1.2 VSM-2 Vs LSA-500

The comparison of LSA-1(without brown corpus) with VSM-2 is shown in the table

Table 3: LSA-1 Vs VSM-2

	LSA-1					**VSM-2**				
k	**Prec.**	**Recall**	**F-score**	**MAP**	**nDCG**	**Prec.**	**Recall**	**F-score**	**MAP**	**nDCG**
1	0.707	0.119	0.195	0.707	0.532	0.693	0.120	0.193	0.693	0.530
2	0.578	0.190	0.269	0.738	0.414	0.567	0.188	0.266	0.733	0.412
3	0.513	0.244	0.307	0.741	0.394	0.501	0.241	0.302	0.734	0.390
4	0.471	0.294	0.335	0.733	0.397	0.461	0.287	0.328	0.732	0.390
5	0.418	0.317	0.333	0.728	0.395	0.412	0.313	0.328	0.721	0.391
6	0.383	0.346	0.335	0.718	0.400	0.379	0.342	0.331	0.710	0.396
7	0.363	0.378	0.342	0.701	0.409	0.352	0.368	0.334	0.697	0.402
8	0.340	0.399	0.339	0.691	0.413	0.332	0.389	0.330	0.687	0.407
9	0.320	0.418	0.336	0.679	0.417	0.311	0.406	0.325	0.680	0.411
10	0.302	0.433	0.330	0.670	0.422	0.291	0.419	0.318	0.674	0.414

The comparison of LSA-1 (without the Brown corpus) with VSM-2 is shown in Table 3. As we can see, LSA-1 generally performs better than VSM-2, particularly in terms of MAP.

The comparison of LSA-2 (LSA with the Brown corpus) with VSM-2 is shown in Table 4. Adding the Brown corpus does not appear to substantially change the results.

Table 4: LSA-1 Vs VSM-2

	LSA-2					VSM-2				
k	**Prec.**	**Recall**	**F-score**	**MAP**	**nDCG**	**Prec.**	**Recall**	**F-score**	**MAP**	**nDCG**
1	0.702	0.117	0.192	0.702	0.532	0.693	0.120	0.193	0.693	0.530
2	0.573	0.186	0.264	0.733	0.414	0.567	0.188	0.266	0.733	0.412
3	0.511	0.244	0.307	0.738	0.393	0.501	0.241	0.302	0.734	0.390
4	0.458	0.285	0.325	0.733	0.392	0.461	0.287	0.328	0.732	0.390
5	0.420	0.319	0.334	0.716	0.392	0.412	0.313	0.328	0.721	0.391
6	0.379	0.341	0.331	0.710	0.393	0.379	0.342	0.331	0.710	0.396
7	0.357	0.372	0.336	0.696	0.403	0.352	0.368	0.334	0.697	0.402
8	0.336	0.393	0.334	0.684	0.409	0.332	0.389	0.330	0.687	0.407
9	0.315	0.409	0.329	0.673	0.414	0.311	0.406	0.325	0.680	0.411
10	0.298	0.427	0.325	0.662	0.418	0.291	0.419	0.318	0.674	0.414

The general trends of evaluation metrics for LSA-500 are shown in Figure 11.

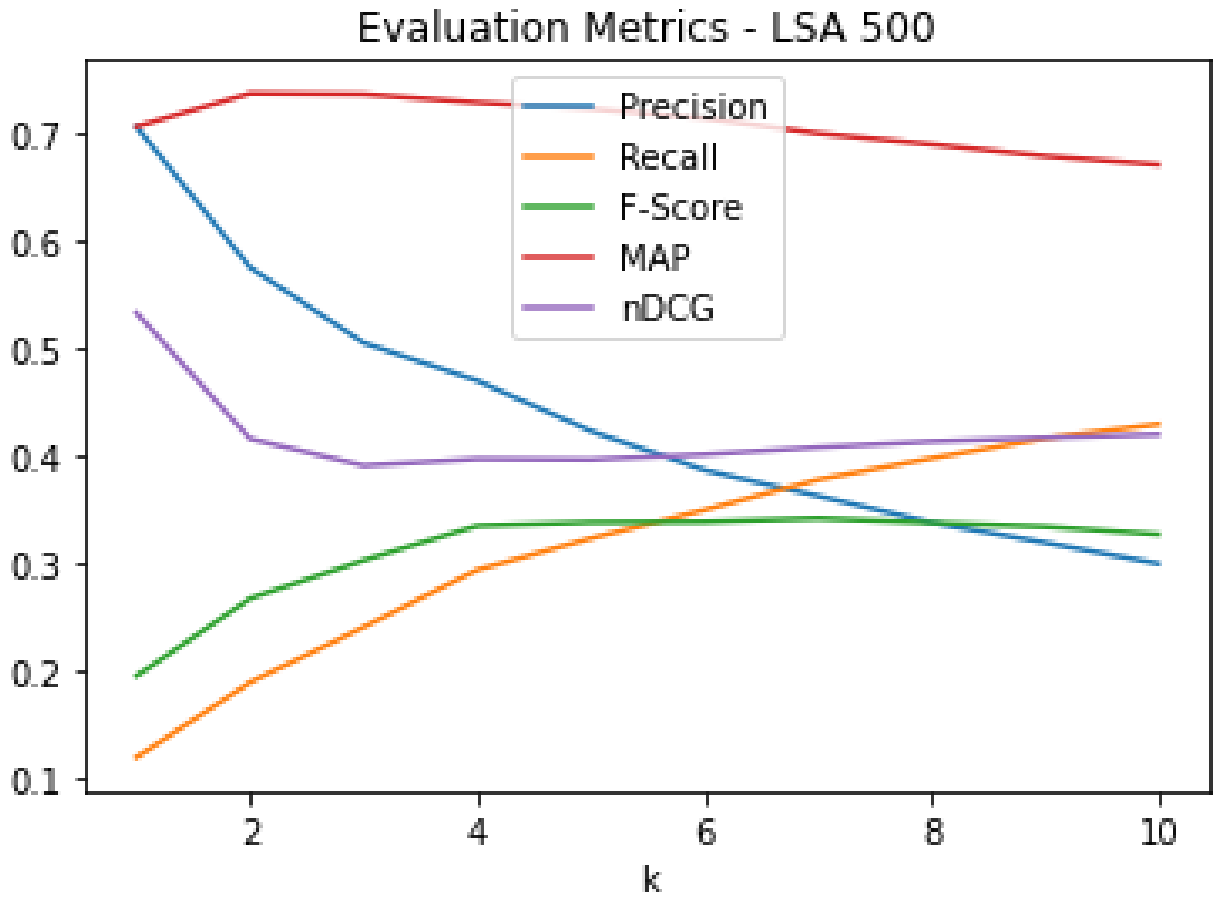


Fig. 11: Mean Average Precision (MAP) as a function of the number of latent variables (k) in LSA.

The distribution statistics of precision, recall, F-score, and nDCG for LSA-500 compared to VSM-2 are shown in Figures 12, 13, 14, and 15 respectively.

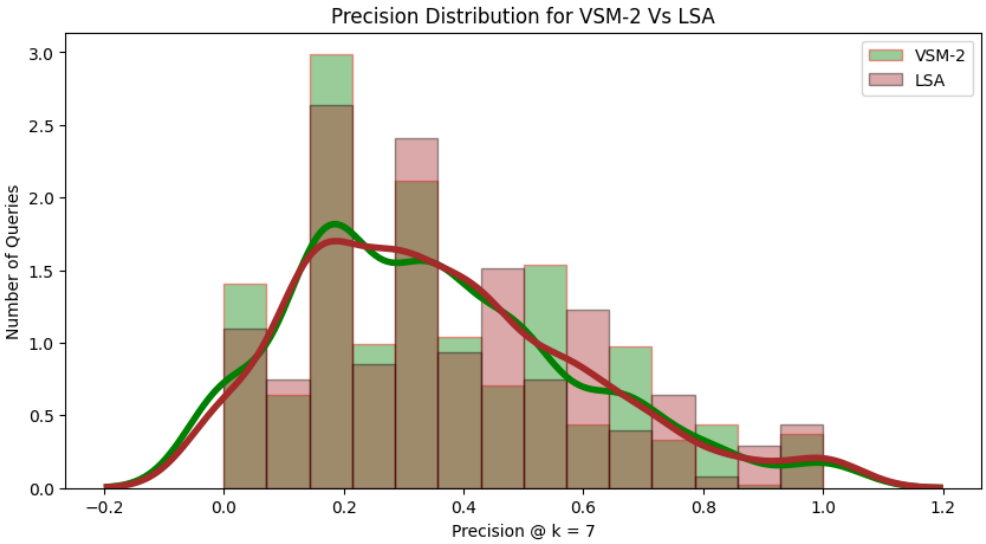


Fig. 12: Precision Distribution for LSA-500 vs. VSM-2 at k=7

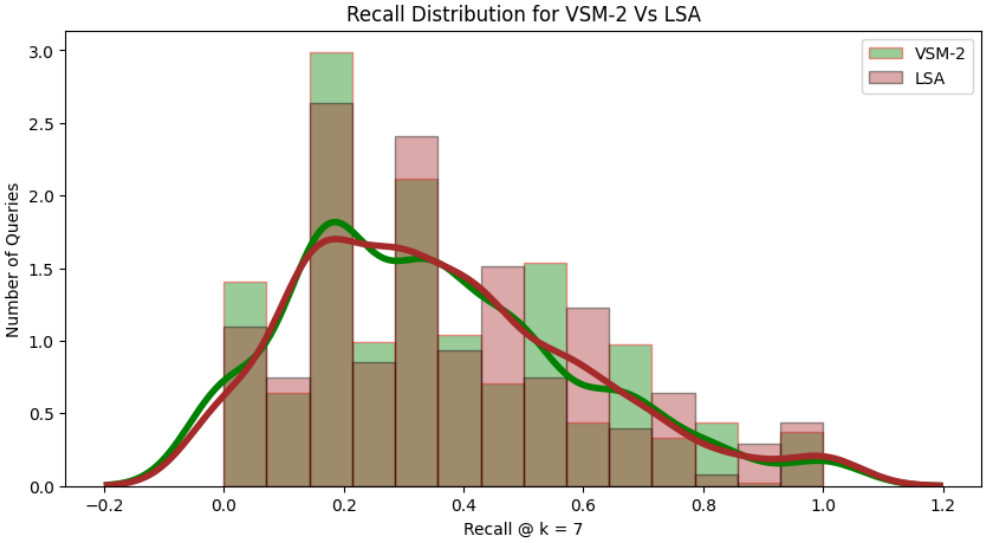


Fig. 13: Recall Distribution for LSA-500 vs. VSM-2 at k=7

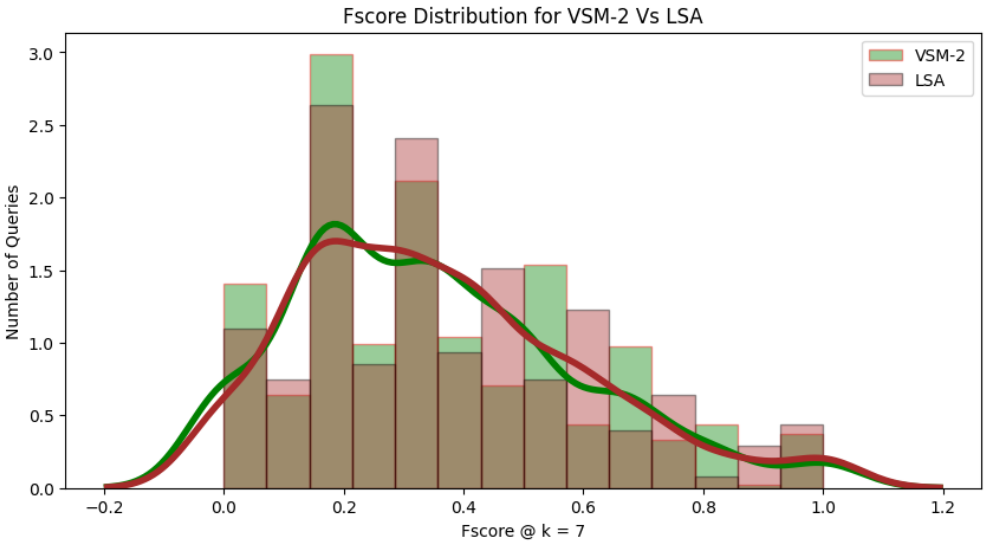


Fig. 14: F-Score Distribution for LSA-500 vs. VSM-2 at k=7

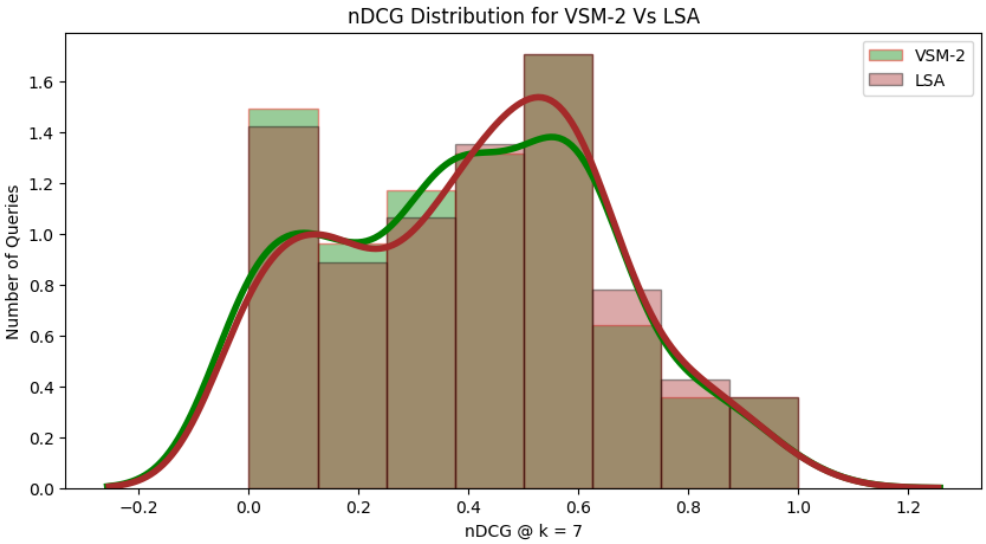


Fig. 15: nDCG Distribution for LSA-500 vs. VSM-2 at k=7

3.2 Observations

From the distributional graphs, we observe that LSA tends to shift the distribution slightly to the right, suggesting an improvement in document representation compared to VSM.

Examples of LSA's Superior Performance

- **Query ID: 109**
  - Query: "panels subjected to aerodynamic heating."

- Relevant Documents: [860, 861, 606, 980, 12, 766]
- VSM-2 Retrieved: [1008, 859, 658, 857, 856, 391, 627, 766, 858, 948]
- LSA-500 Retrieved: [859, 1008, 658, 857, 856, 766, 858]
- LSA Precision: 0.142

Document 766 (retrieved by LSA): "... experimental investigation at mach number of 3.0 of effects of thermal stress and buckling on flutter characteristics..."

- **Query ID: 184**
  - Query: "work on small-oscillation re-entry motions."
  - Relevant Documents: [32, 67, 715, 717, 716, 499, 1379, 639]
  - VSM-2 Retrieved: [207, 281, 1113, 515, 1348, 164, 917, 1346, 715, 658]
  - LSA-500 Retrieved: [207, 1113, 515, 281, 715, 639, 164, 917]
  - LSA Precision: 0.285

  Documents 715 and 639 (retrieved by LSA): "... motion of a ballistic missile... upon entering the atmosphere... effect upon aerodynamic heating..." (715), "... tumbling motion of vehicles entering planetary atmospheres..." (639)

Explanation: LSA outperformed VSM-2 because it captured higher-order associations between words. For example, LSA associated "thermal" and "temperature" with "heating" in query ID 109. VSM-2 relies on direct term matching and thus missed this connection.

3.3 Limitations of LSA

- **Computational Complexity:** The SVD algorithm used in LSA has a computational complexity of $O(n^2 \times k^3)$, where n is the number of documents plus terms and k is the number of latent dimensions. This can be computationally demanding for large datasets.
- **Interpretability:** The reduced-dimension concept space of LSA can be less interpretable than the original term-document matrix.
- **Polysemy:** LSA does not explicitly handle polysemy (words with multiple meanings), potentially leading to inaccuracies in cases where word sense disambiguation is crucial.

3.4 Hypothesis Testing: LSA vs. VSM-2

We conducted a hypothesis test to determine whether there is a statistically significant difference in performance between LSA-500 (with 500 latent dimensions) and VSM-2.

Hypotheses

- **Null Hypothesis (H0):** LSA-500 and VSM-2 perform similarly in terms of precision, recall, F-score, and nDCG.
- **Alternative Hypothesis (H1):** LSA-500 and VSM-2 do not perform similarly in terms of precision, recall, F-score, and nDCG.

Approach

We collected precision, recall, F-score, and nDCG values for both LSA-500 and VSM-2 across the 225 queries in the Cranfield dataset. To assess the significance of the differences, we applied a two-sample, two-tailed t-test to each metric individually.

Results

The results of the t-tests are summarized in Table 5.

Table 5: Results of t-test for LSA-500 vs. VSM-2

Metric	t-statistic	p-value
Precision	-0.176353	0.860097
Recall	-0.252374	0.800860
F-score	-0.263295	0.792445
nDCG	-0.293440	0.769322

Inference

All p-values are much greater than the typical significance level of 0.05. Therefore, we fail to reject the null hypothesis. We conclude that there is not enough evidence to suggest a statistically significant difference in performance between LSA-500 and VSM-2 across the tested metrics for a given query.

4 Addressing Semantic Sensitivity (Limitation 3):

Different words can have the same meaning called synonyms. So using Query Expansion we would like to get the similar words using word2vec.

4.1 Query Expansion (QE) Model

Query expansion (QE) enhances queries by introducing additional tokens or phrases, which the search engine automatically incorporates. For instance, the query "vp marketing" is transformed into "(vp OR "vice president") marketing" [6].

Our implementation involves:

- **Word Embedding Training:** We train a continuous bag-of-words (CBOW) model [**?**] on the corpus to create vector representations (word embeddings) for all words.
- **Similar Term Selection:** For each query term, we identify the most similar word (using cosine similarity) and add it to the query. (See Figure 16)
- **Document Ranking:** We use the expanded query's vector representation to rank documents based on cosine similarity.

```
[282]:  1 res = np.array(model.wv.most_similar(positive=["good"]))[:,0].tolist()
        2 res

[282]: ['agreement',
        'result',
        'experimental',
        'comparison',
        'theoretical',
        'data',
        'compare',
        'satisfactory',
        'prediction',
        'reasonably']
```

Fig. 16: Query expansion model predicting similar words to "good".

Note: Due to the small corpus size, the trained CBOW model may not yield significant improvements in this specific case. However, QE can be highly beneficial with a larger training dataset.

4.2 VSM-2 vs QE

The comparison of the query expansion model and VSM-2 is shown in Table 6.

The general trends of evaluation metrics for Query Expansion (QE) are shown in Figure 17.

Table 6: Comparison of Query Expansion Model and VSM-2 (Base Model)

	Query Expansion					VSM-2				
k	**Prec.**	**Recall**	**F-score**	**MAP**	**nDCG**	**Prec.**	**Recall**	**F-score**	**MAP**	**nDCG**
1	0.568	0.097	0.158	0.568	0.432	0.693	0.120	0.193	0.693	0.530
2	0.449	0.145	0.207	0.609	0.333	0.567	0.188	0.266	0.733	0.412
3	0.394	0.188	0.240	0.622	0.319	0.501	0.241	0.302	0.734	0.390
4	0.358	0.220	0.258	0.621	0.317	0.461	0.287	0.328	0.732	0.390
5	0.334	0.253	0.266	0.616	0.321	0.412	0.313	0.328	0.721	0.391
6	0.312	0.281	0.272	0.605	0.328	0.379	0.342	0.331	0.710	0.396
7	0.290	0.303	0.273	0.598	0.330	0.352	0.368	0.334	0.697	0.402
8	0.275	0.322	0.274	0.585	0.333	0.332	0.389	0.330	0.687	0.407
9	0.262	0.340	0.275	0.578	0.340	0.311	0.406	0.325	0.680	0.411
10	0.247	0.345	0.270	0.570	0.345	0.291	0.419	0.318	0.674	0.414

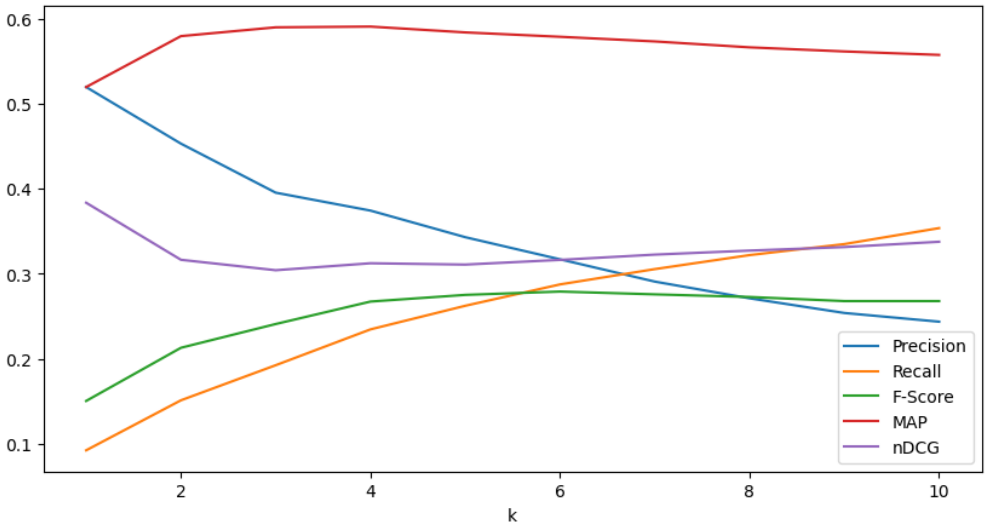


Fig. 17: Evaluation Metrics Trends for Query Expansion (QE)

The distribution statistics of precision, recall, F-score, and nDCG for QE (Query Expansion) compared to VSM-2 are shown in Figures 18, 19, 20, and 21, respectively.

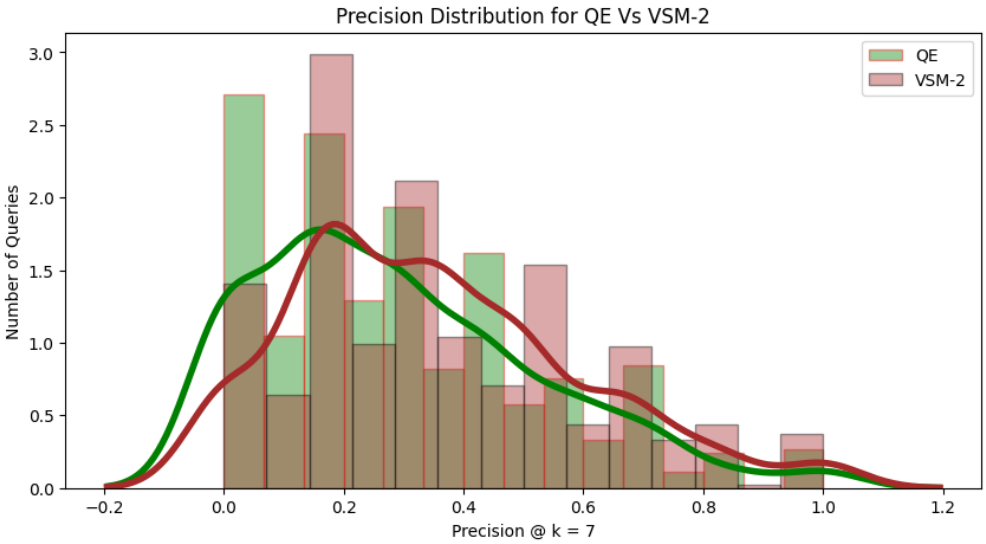


Fig. 18: Precision Distribution for QE vs. VSM-2 at k=7

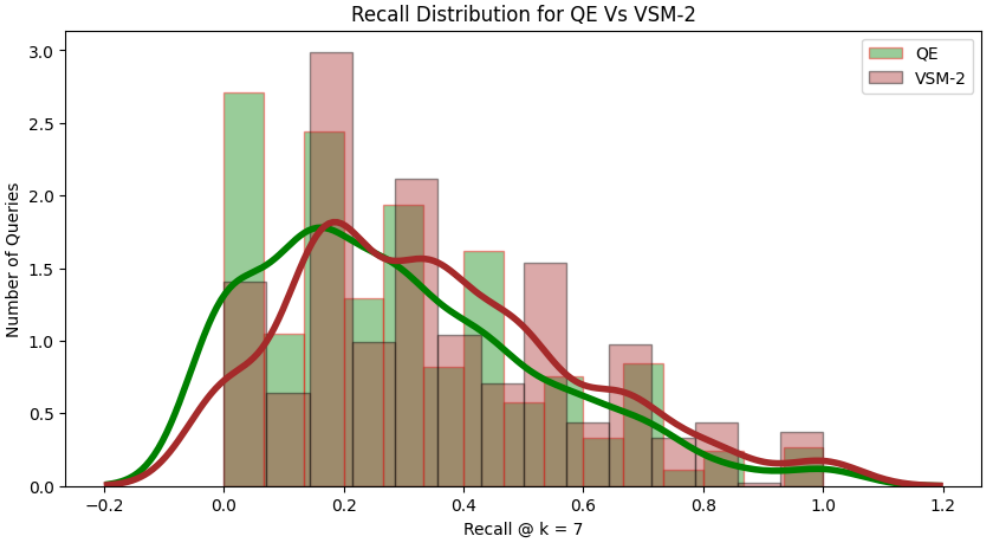


Fig. 19: Recall Distribution for QE vs. VSM-2 at k=7

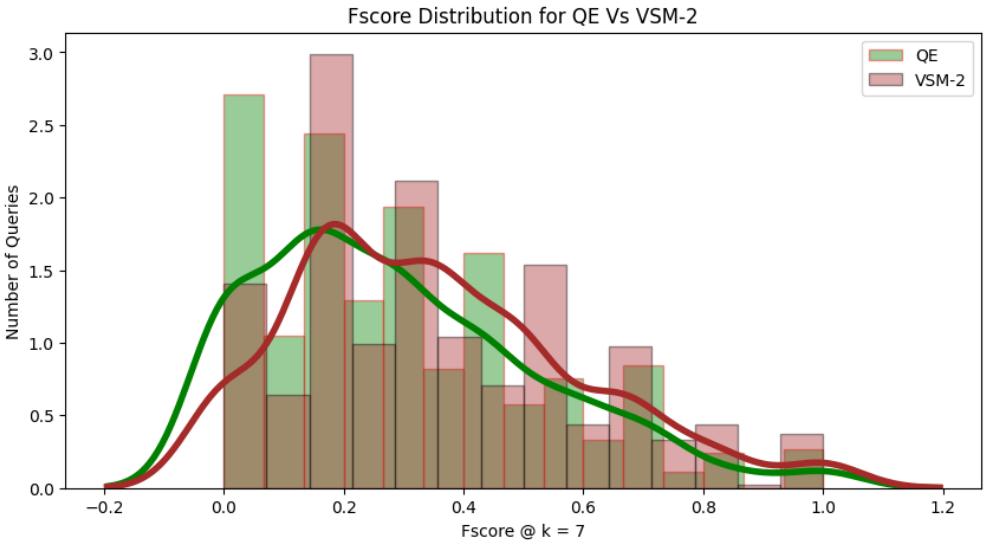


Fig. 20: F-Score Distribution for QE vs. VSM-2 at k=7

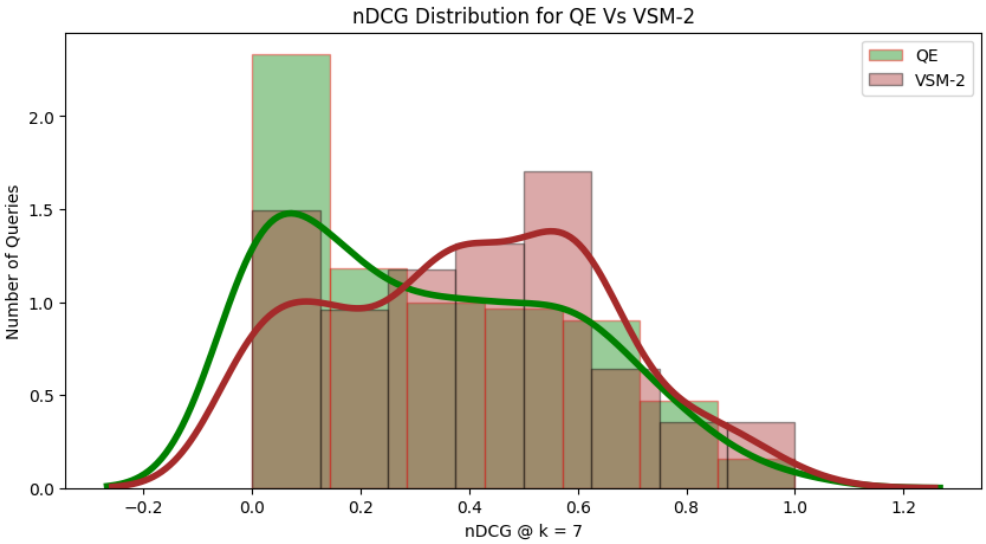


Fig. 21: nDCG Distribution for QE vs. VSM-2 at k=7

4.3 Observations

From the distributional graphs, we observe that the distribution of QE often shifts towards the left, indicating that the QE model underperforms compared to VSM-2. This could be attributed to the poor word2vec representation due to the limited training data.

Examples of QE's Better Performance

Despite its overall lower performance, the QE model shows better results than VSM-2 in certain cases:

- **Query ID: 85**
  - Preprocessed Query: "parameter seriously influence natural transition laminar turbulent flow model wind tunnel."
  - Expanded Query: "... dependence vas delay strengthen roughness layer injection dimensional cover tunnel wind ..."
  - Words Added: [dependence, vas, delay, strengthen, roughness, layer, injection, dimensional, cover, tunnel]
  - VSM-2 Recall: 0.0
  - QE Recall: 0.2
  - Documents Retrieved by QE (Relevant): [710]

  Document 710: "... the smallest height of roughness capable of affecting boundary-layer transition ... low-speed wind tunnel ... critical heights for three types of roughness..."

- **Query ID: 184**
  - Preprocessed Query: "work small oscillation entry motion."
  - Expanded Query: "... limitation upon estimation atmosphere linearize work small oscillation entry motion"
  - Words Added: [limitation, upon, estimation, atmosphere, linearize]
  - VSM-2 Recall: 0.0
  - QE Recall: 0.25
  - Documents Retrieved by QE (Relevant): [715, 716, 639]

  Document 715: "... motion of a ballistic missile angularly misaligned with the flight path upon entering the atmosphere..."

  Document 716: "... study of the oscillatory motion of manned vehicles entering the earth's atmosphere..."

Inference: QE's success in these cases is attributed to the addition of synonymous or co-occurring words. For Query ID 85, QE retrieved document 710 due to the added words "roughness," "tunnel," and "dimensional," which are relevant to the document's content. Similarly, for Query ID 184, the added terms "limitation," "upon," and "atmosphere" helped match relevant documents 715 and 716.

4.4 Hypothesis Testing: VSM-2 vs. QE

We conducted a hypothesis test to determine if there is a statistically significant difference in performance between VSM-2 and the Query Expansion (QE) model.

Hypotheses

- **Null Hypothesis (H0):** VSM-2 and QE perform similarly in terms of precision, recall, F-score, and nDCG.
- **Alternative Hypothesis (H1):** VSM-2 and QE do not perform similarly in terms of precision, recall, F-score, and nDCG.

Approach

We collected precision, recall, F-score, and nDCG values for both VSM-2 and QE across the 225 queries in the Cranfield dataset. A two-tailed t-test was applied to each metric individually to assess the significance of the differences.

Results

The t-test results are presented in Table 7.

Table 7: Results of t-test for VSM-2 vs. QE

Metric	t-statistic	p-value
Precision	-3.111	0.0019836
Recall	-2.753	0.0061363
F-score	-3.185	0.0015464
nDCG	-3.565	0.0004027

Conclusion

We reject the null hypothesis due to the p-values being significantly less than 0.05. Therefore, we conclude that VSM-2 and QE do not perform similarly. While the distribution analysis suggests that QE performs slightly better for some queries, it performs worse for others.

5 Clustering Methods to Reduce Retrieval Time

As the number of documents to search increases, the time taken to retrieve relevant documents for a particular query also increases linearly. To mitigate this issue, we explored two methods to reduce the search time:

1. K-Means Clustering
2. Topic Modeling using Latent Dirichlet Allocation (LDA)

5.1 K-Means Clustering

Clustering is an unsupervised machine learning method for grouping data points based on their similarity. We utilized K-means clustering, an algorithm that partitions a dataset into k distinct clusters [8]. It operates by iteratively assigning each data point to the cluster with the nearest centroid (the mean of the points in that cluster) and then recalculating the centroids until convergence.

Implementation Steps

1. **Construct TF-IDF Matrix:** Represent documents as vectors in a high-dimensional space using TF-IDF weights.
2. **Apply K-Means:** Perform K-means clustering on the TF-IDF matrix, aiming to group documents into **k** clusters.
3. **Determine Optimal k:** The optimal number of clusters, **k**, can be estimated using the elbow method. In our experiments, the best value for **k** was found to be around 6, as illustrated in Figures 22 and 23.

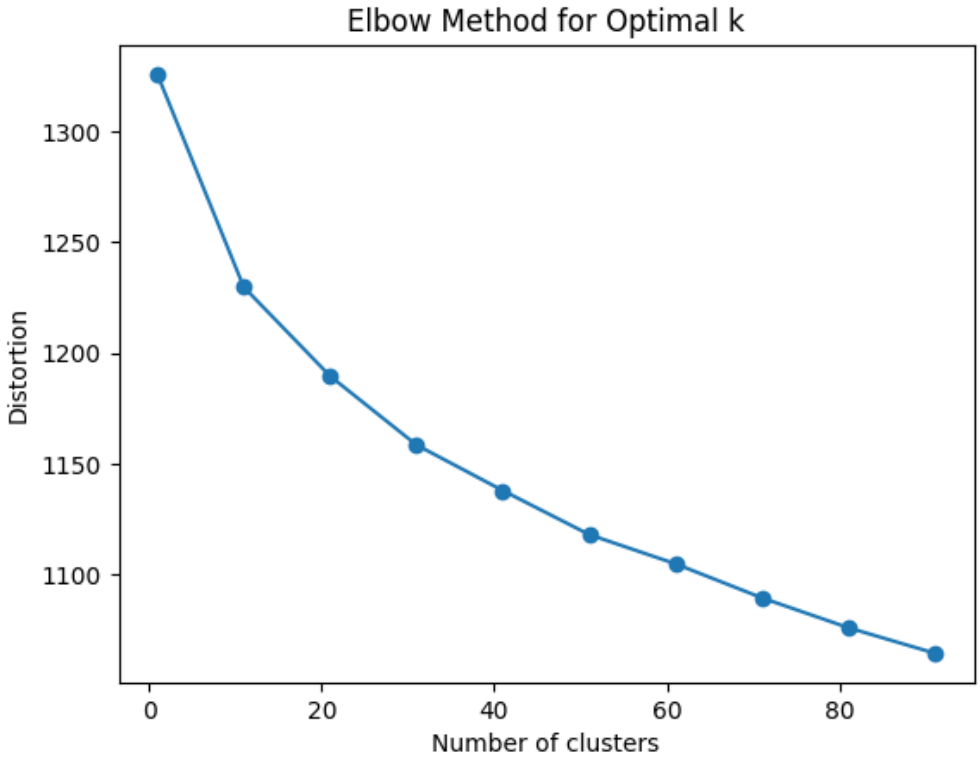


Fig. 22: Elbow plot for determining the optimal number of clusters (k) in K-means.

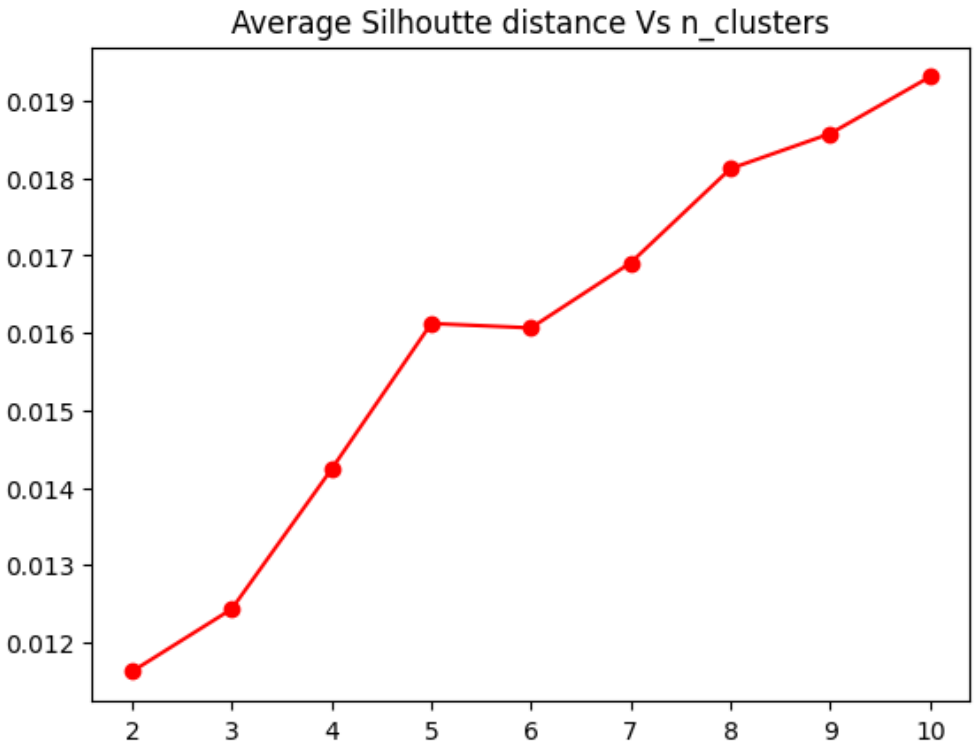


Fig. 23: Visualization of K-means clusters in a 2D projection.

subsectionK-Means Clustering vs. VSM-2

Retrieval Time Comparison

Comparing the K-means clustering model with the VSM-2 base model reveals a substantial improvement in average retrieval time, as demonstrated in Table 8. K-means clustering reduces the retrieval time by a factor of approximately 3.

Table 8: Retrieval Time Comparison: K-means Clustering (k=6) vs. VSM-2

Model	**Avg. Retrieval Time (ms)**
K-means (k=6)	3.7
VSM-2	11.0

Evaluation Metrics Comparison

Table 9 presents a comparison of the K-means clustering model (k=6) and the VSM-2 model in terms of evaluation metrics. The results show a slight decrease (approximately 0.1) in all evaluation measures for the K-means model. This trade-off is expected due to the reduced search space in clustering-based retrieval. However, if retrieval speed is a priority, K-means clustering offers a significant advantage.

Table 9: Evaluation Metrics Comparison: K-Means Clustering (k=6) vs. VSM-2

	K-Means (k=6)					VSM-2				
k	**Prec.**	**Recall**	**F-score**	**MAP**	**nDCG**	**Prec.**	**Recall**	**F-score**	**MAP**	**nDCG**
1	0.573	0.093	0.154	0.573	0.412	0.693	0.120	0.193	0.693	0.530
2	0.458	0.145	0.207	0.609	0.318	0.567	0.188	0.266	0.733	0.412
3	0.407	0.188	0.240	0.611	0.302	0.501	0.241	0.302	0.734	0.390
4	0.369	0.222	0.258	0.602	0.299	0.461	0.287	0.328	0.732	0.390
5	0.340	0.249	0.266	0.597	0.303	0.412	0.313	0.328	0.721	0.391
6	0.307	0.267	0.264	0.594	0.304	0.379	0.342	0.331	0.710	0.396
7	0.290	0.291	0.268	0.583	0.310	0.352	0.368	0.334	0.697	0.402
8	0.270	0.305	0.265	0.576	0.313	0.332	0.389	0.330	0.687	0.407
9	0.253	0.319	0.261	0.570	0.317	0.311	0.406	0.325	0.680	0.411
10	0.237	0.328	0.255	0.562	0.319	0.291	0.419	0.318	0.674	0.414

5.2 Hypothesis Testing: LSA with and without Clustering

We conducted a hypothesis test to determine if there is a statistically significant difference in retrieval time between LSA with and without clustering.

Hypotheses

- **Null Hypothesis (H0):** The mean retrieval time for LSA with clustering is equal to the mean retrieval time for LSA without clustering.
- **Alternative Hypothesis (H1):** The mean retrieval time for LSA with clustering is not equal to the mean retrieval time for LSA without clustering.

Approach

We recorded the retrieval times for both LSA with clustering and LSA without clustering on 200 queries from the Cranfield dataset (Figure 24). A two-tailed t-test was performed to evaluate the statistical significance of the difference.

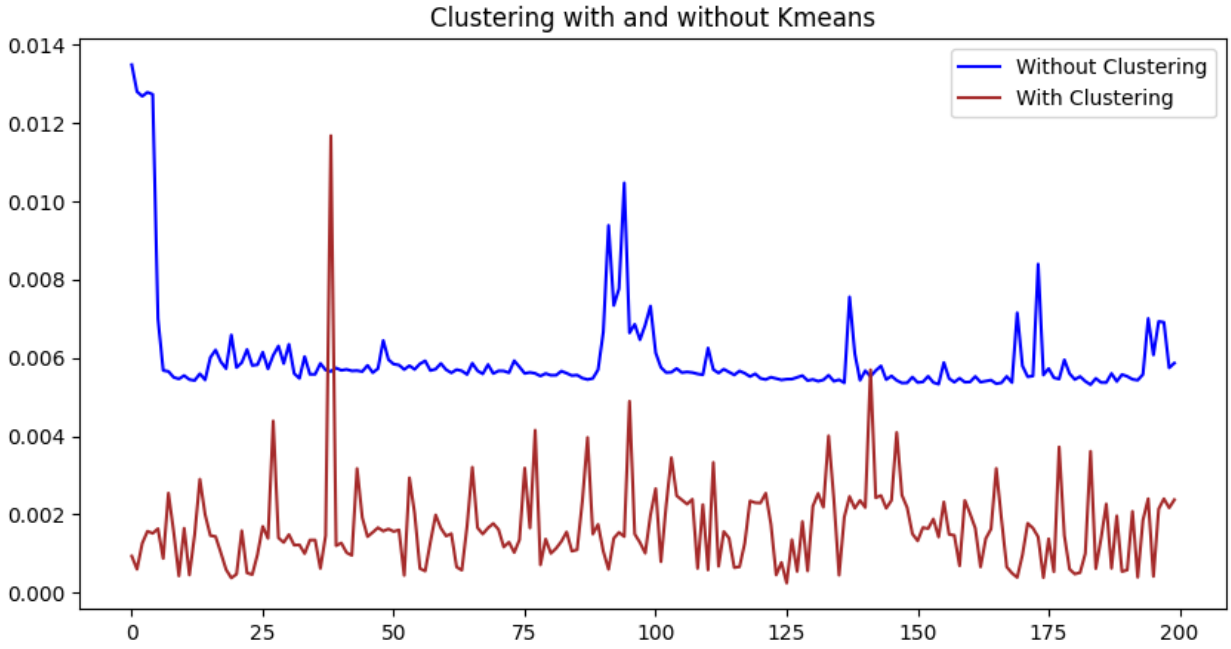


Fig. 24: Retrieval Times for LSA with and without Clustering

Results

The t-test yielded a t-value of 18.24 and a p-value of approximately 1.47e-45, which is significantly less than the chosen significance level ($\alpha = 0.05$).

Conclusion

Based on the results, we reject the null hypothesis. This indicates that there is a statistically significant difference in retrieval time between LSA with clustering and LSA without clustering. As seen in Figure 24, the clustering method significantly reduces the retrieval time for a given query.

5.3 Topic Modeling using Latent Dirichlet Allocation (LDA)

Topic modeling is an unsupervised machine learning technique that automatically identifies latent topics within a collection of documents [5]. Latent Dirichlet Allocation (LDA) is a popular topic modeling algorithm that assumes a generative process where documents are mixtures of topics, and each topic is a distribution over words [5]. The underlying assumption of LDA, similar to LSA, is that documents with similar topics tend to use similar words.

LDA vs. LSA

While both LDA and LSA aim to uncover latent semantic structure, a key difference lies in their modeling assumptions. LSA primarily relies on matrix factorization (SVD) to reduce dimensionality, whereas LDA employs a probabilistic generative model with Dirichlet distributions to represent the relationships between documents, topics, and words [4].

LDA Hyperparameters

LDA has two hyperparameters that influence topic modeling:

1. **Alpha (α):**Controls the distribution of topics within documents. A lower value of alpha encourages sparsity, assigning fewer topics per document, while a higher value promotes a broader distribution of topics across documents.

2. **Beta (β):**Controls the distribution of words within topics. A lower value of beta favors fewer words per topic, making topics more distinct, whereas a higher value leads to broader topic distributions with more overlapping words.

For a detailed discussion of these hyperparameters and their impact on the resulting topic distributions, we referred to [10].

Implementation Steps

Our implementation of LDA-based document clustering follows these steps:

1. **Topic Modeling:** Apply LDA to the corpus to identify latent topics and assign topic probabilities to each document.

2. **Topic Clustering:** Group documents based on their topic distributions. Documents with similar topic profiles are clustered together.

3. **Centroid Calculation:** Compute the centroid (mean vector) for each topic cluster in the document representation space.

4. **Query-Topic Matching:** For a given query, calculate its cosine similarity with each topic centroid.

5. **Targeted Search:** Select the topic cluster(s) most similar to the query and retrieve documents exclusively from those clusters. This can be combined with traditional ranking methods for further refinement [**?**].

5.4 LDA vs. VSM-2

Retrieval Time Comparison

Comparing the LDA model with the VSM-2 base model in terms of average retrieval time, Table 10 shows a slight reduction for LDA, but not as significant as with K-means clustering. This may be due to insufficient data for accurately estimating the alpha and beta parameters of the probability distributions in LDA. Additionally, the relatively high retrieval time (compared to K-means) could result from the inclusion of numerous documents within a single topic.

Table 10: Retrieval Time Comparison: LDA (k=6) vs. VSM-2

Model	Avg. Retrieval Time (ms)
LDA (k=6)	8
VSM-2	11

Evaluation Metrics Comparison

Table 11 presents a comparison of the LDA model and the VSM-2 model in terms of evaluation metrics. The results indicate that the LDA model generally underperforms VSM-2 across most metrics. This could be attributed to the limited dataset size, which may not be sufficient for LDA to effectively learn the underlying topic distributions.

Table 11: Evaluation Metrics Comparison: LDA vs. VSM-2 (Base Model)

	LDA					VSM-2				
k	**Prec.**	**Recall**	**F-score**	**MAP**	**nDCG**	**Prec.**	**Recall**	**F-score**	**MAP**	**nDCG**
1	0.484	0.085	0.139	0.484	0.338	0.693	0.120	0.193	0.693	0.530
2	0.407	0.133	0.188	0.547	0.277	0.567	0.188	0.266	0.733	0.412
3	0.367	0.178	0.223	0.564	0.271	0.501	0.241	0.302	0.734	0.390
4	0.337	0.209	0.239	0.567	0.276	0.461	0.287	0.328	0.732	0.390
5	0.307	0.237	0.247	0.562	0.275	0.412	0.313	0.328	0.721	0.391
6	0.279	0.257	0.247	0.551	0.278	0.379	0.342	0.331	0.710	0.396
7	0.255	0.270	0.241	0.546	0.282	0.352	0.368	0.334	0.697	0.402
8	0.240	0.286	0.241	0.542	0.285	0.332	0.389	0.330	0.687	0.407
9	0.229	0.302	0.240	0.536	0.290	0.311	0.406	0.325	0.680	0.411
10	0.215	0.312	0.235	0.533	0.295	0.291	0.419	0.318	0.674	0.414

5.5 Hypothesis Testing: LSA vs. LDA Retrieval Time

We conducted a hypothesis test to determine if there is a statistically significant difference in retrieval time between LSA and LDA (with 6 topics).

Hypotheses

- **Null Hypothesis (H0):** The mean retrieval time for LSA is equal to the mean retrieval time for LDA.
- **Alternative Hypothesis (H1):** The mean retrieval time for LSA is not equal to the mean retrieval time for LDA.

Approach

We recorded retrieval times for both LSA and LDA (with 6 topics) across 200 queries from the Cranfield dataset, as visualized in Figure 25. A two-tailed t-test was performed to assess

the statistical significance of the observed difference.

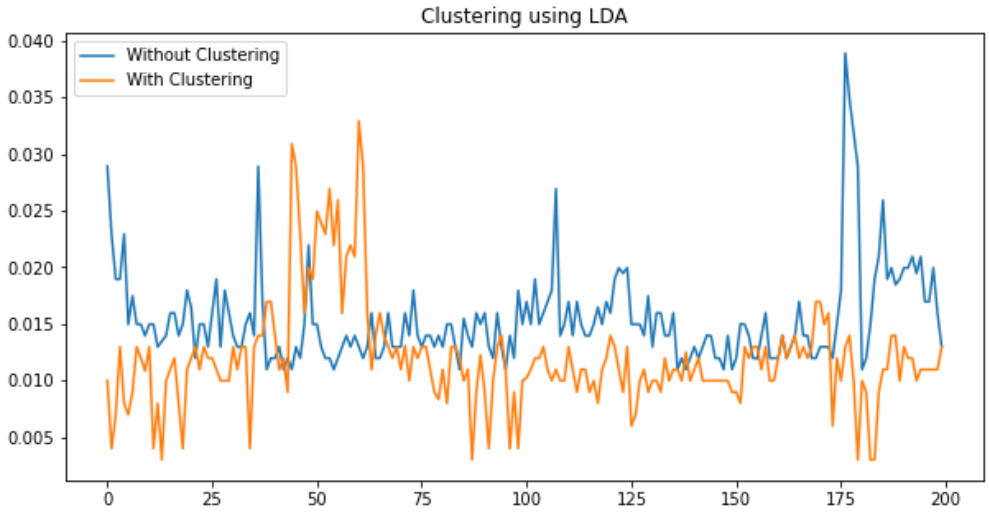


Fig. 25: Retrieval Times for LDA (k=6) vs. LSA

Results

The t-test yielded a t-value of 7.54 and a p-value of approximately 2.74e-11, which is significantly less than the chosen significance level ($\alpha = 0.05$).

Conclusion

Based on the results, we reject the null hypothesis. This indicates a statistically significant difference in retrieval time between LSA and LDA. As shown in Figure 25, LDA outperforms LSA in terms of retrieval time for a given query.

6 Query Auto-Completion

To enhance user interaction with the search engine, we developed a query auto-completion (QAC) model that predicts and suggests the next words or phrases as the user types a query. This feature can be integrated into web browsers to improve the search experience (Figure 26).

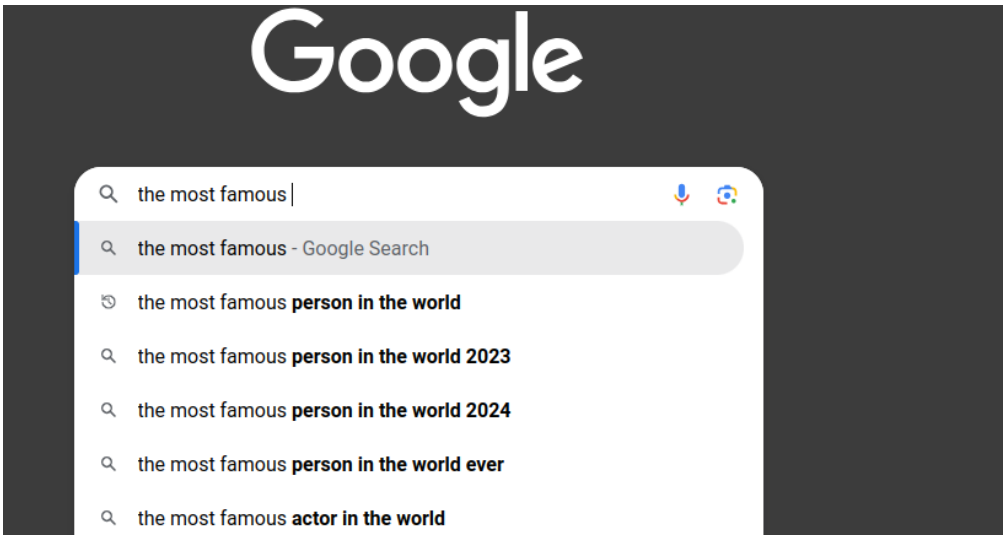


Fig. 26: Query auto-completion integrated with a browser.

6.1 Query Completion Model

Our QAC model is implemented using a shallow neural network architecture:

1. **Embedding Layer:** An embedding layer transforms each word in the query into a 16-dimensional vector representation.
2. **LSTM Layer:** A Long Short-Term Memory (LSTM) layer captures the sequential dependencies and context of the words in the query.
3. **Dense Layer:** A dense layer processes the LSTM output.
4. **Output Layer (Softmax):** The output layer uses a softmax activation function to predict the probability distribution over the vocabulary for the next word.

Training Data Preparation

We preprocessed all sentences in the query dataset to create training data. For each sentence, we extracted sequences of five consecutive words as input and used the following word as the corresponding output (label).

Example: For the sentence "when constructing aeroelastic models of heated high-speed aircraft," the training input sequences would be:

- "when constructing aeroelastic models of"
- "constructing aeroelastic models of heated"
- "aeroelastic models of heated high"
- ...

The corresponding labels would be:

- "heated"
- "high"
- "speed"
- ...

Model Prediction and Evaluation

After training, the model can predict the next n words for an incomplete query (Figure 27a). However, we found that the quality of predictions deteriorates as n increases (Figure 27b). Due to the lack of a suitable baseline model, we relied on manual interpretation of selected queries to evaluate the QAC model's performance.

```
In [53]:  1 complete_query("experimental studies of creep",2)
Out[53]: 'experimental studies of creep buckling either'
```

(a) Query completion predicting the next 2 words.

```
In [14]:  1 complete_query("experimental studies of creep",10)
Out[14]: 'experimental studies of creep buckling or must be developed to the fundamental three speed'
```

(b) Query completion predicting the next 10 words.

Fig. 27: Examples of query completion with varying prediction lengths.

7 Summary

In this work, we developed two baseline vector space models (VSM-1 and VSM-2) using different pre-processing techniques. VSM-2, with its more extensive pre-processing, reduced the dimensionality of the vector space and outperformed VSM-1 in our evaluation.

We then applied Latent Semantic Analysis (LSA) to both pre-processed datasets to enhance document representation. LSA applied to the more extensively pre-processed data (VSM-2) yielded the best overall results, improving the ranking of retrieved documents.

We also explored a query expansion (QE) model, adding semantically similar terms to queries. However, due to the limited training corpus size, this did not significantly improve effectiveness.

Finally, focusing on retrieval efficiency, we implemented document clustering using K-means and LDA. Both methods significantly reduced retrieval time, albeit with a slight trade-off in recall and precision.

Overall, our best model relies on exact query terms, which might not always be present in the corpus. To address this, we implemented a query auto-completion feature, enhancing

the user experience and potentially improving retrieval in real-world scenarios. However, this feature is not directly applicable to the evaluation of our system with a fixed set of test queries.

7.1 Result Summary

Final Result of models @ k=10

Models @ K=10	Precision	Recall	F-Score	MAP	nDCG
VSM-1	0.0075	0.0115	0.0086	0.0136	0.0054
VSM-2	0.291	0.419	0.317	0.671	0.415
LSA	0.3	0.431	0.327	0.67	0.422
ESA	0.155	0.241	0.174	0.403	0.25
W2V	0.048	0.066	0.051	0.115	0.056
QE with TF_IDF	0.292	0.419	0.3179	0.671	0.4146

(a) Comparison of the performance of different models using MAP, Recall, nDCG, and F-score metrics for top 10 documents retrieved.

	Base model					LSA with 500 dimension				
K	precision	recall	F-Score	MAP	nDCG	precision	recall	F-Score	MAP	nDCG
1	0.644	0.109	0.180	0.644	0.494	0.706	0.118	0.194	0.706	0.537
2	0.548	0.183	0.258	0.695	0.396	0.575	0.189	0.267	0.737	0.415
3	0.488	0.230	0.291	0.699	0.372	0.512	0.244	0.307	0.738	0.395
4	0.428	0.263	0.302	0.695	0.365	0.472	0.295	0.336	0.733	0.398
5	0.389	0.293	0.309	0.685	0.364	0.420	0.321	0.336	0.728	0.394
6	0.359	0.323	0.314	0.677	0.371	0.384	0.348	0.337	0.717	0.401
7	0.336	0.346	0.315	0.668	0.377	0.365	0.379	0.343	0.702	0.410
8	0.313	0.363	0.311	0.658	0.383	0.338	0.397	0.338	0.693	0.414
9	0.296	0.382	0.309	0.653	0.653	0.319	0.417	0.334	0.678	0.418
10	0.279	0.399	0.304	0.643	0.393	0.300	0.430	0.327	0.670	0.421

(b) Comparison of Precision, Recall, and F-score between the best LSA and VSM-2 models.

Fig. 28: Summary of Evaluation Results: (a) Overall Model Comparison and (b) LSA vs. VSM-2 Comparison

References

[1] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[2] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge university press, 2008.

[3] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.

[4] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse processes*, vol. 25, no. 2-3, pp. 259–284, 1998.

[5] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 50–57, 1999.

[6] C. Carpineto and G. Romano, "A survey of automatic query expansion in information retrieval," *Acm Computing Surveys (CSUR)*, vol. 44, no. 1, pp. 1–50, 2012.

[7] H. K. Azad and A. Deepak, "Query expansion techniques for information retrieval: a survey," *Information Processing & Management*, vol. 56, no. 5, pp. 1698–1735, 2019.

[8] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[9] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[10] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.