CS6370 Project Report

Sanket Pramod Bhure[1], Advait Amit Kisar[2], and Netheti Devi Varaprasad[3]

[1]Aerospace Engineering, IIT Madras, ae20b108@smail.iitm.ac.in
[2]Aerospace Engineering, IIT Madras, ae20b007@smail.iitm.ac.in
[3]Mechanical Engineering, IIT Madras, me20b120@smail.iitm.ac.in

Abstract

This project aims to create an effective Information Retrieval System using NLP techniques. We started with the Vector Space Model (VSM) as our baseline and developed two different versions using various pre-processing methods. We then explored alternatives like LSA and ESA to improve upon VSM's limitations. Additionally, we investigated methods like Clustering (K-means) to speed up searches and compared retrieval times with and without clustering. Later on this, we also considered query expansion using FastText by adding similar words to the query. Lastly, we experimented with Query Auto-completion using LSTMs to enhance user experience. Through thorough analysis and statistical methods, we evaluated the performance of these models against the baseline.

Keywords: Auto-completion, baseline model, Clustering, FastText, LSA, LSTMs, VSM-1, VSM-2.

1. Introduction

The **Vector Space Model (VSM)** is a method that represents documents and queries as vectors of numbers. In a basic VSM, these vectors are created by considering the term frequencies (word count) and the inverse document frequency in the corpus. When retrieving information, the documents are ranked based on the cosine similarity between their vectors and the query vector.

Latent Semantic Analysis (LSA) is a technique used in Natural Language Processing (NLP) to analyze the relationships between a set of documents and the terms they contain. LSA generates a set of concepts associated with the documents and the terms. It takes advantage of the hidden structure of how terms are related to documents. LSA is expected to address issues related to synonymy and polysemy partially.

In **Document Clustering**, we utilize K-means to group similar documents together. When a query is provided, we measure its similarity with all cluster centers and retrieve the documents only from the cluster that shows a high similarity. This approach effectively reduces the number of documents that need to be searched for a given query, resulting in a faster retrieval. Hence, we streamline the search process and optimize the efficiency of document retrieval.

Query Expansion is a technique used in information retrieval to improve search results by adding synonyms or related terms to the original search query. FastText is a word embedding model developed by Facebook AI Research that is known for its efficiency in handling large amounts of text data and capturing semantic relationships between words. By utilizing the FastText model, we can effectively expand a query by finding similar words or synonyms that may enhance the search results.

Query Auto-Completion, a feature aimed at streamlining search processes, predicts and suggests the next words or phrases as users type queries. Our approach employs an LSTM-based algorithm trained on extensive datasets to anticipate the next 'k' words accurately. By leveraging contextual patterns from past queries, our system enhances search efficiency and user experience by offering real-time suggestions tailored to individual search contexts.

2. Vector Space Model (VSM)

A Vector Space Model (VSM) is constructed by utilizing TF-IDF scores, which represent the importance of words in the documents. In our project, we have implemented two versions of the VSM, each incorporating different pre-processing techniques.

1. **VSM-1 (Vector Space Model Version 1):** This version follows a specific set of procedures to preprocess the data and construct the VSM.
2. **VSM-2 (Vector Space Model Version 2):** In this version, an alternative approach to pre-processing the data is employed.

The specific techniques and procedures used in respective versions are described in separate subsections.

2.1 VSM-1

In this model, we used techniques from NLP assignments to process the data. Here are the steps we followed to build the model:

- **Data Pre-processing:**

  - **Tokenization:** Documents are segmented into sentences and then words using the **"Treebank tokenizer"** from the NLTK package.
  - **Stopword Removal:** Common words (stopwords) like "a", "an", and "the" are eliminated from the word list. While this step is optional, excluding stopwords reduces the vocabulary size in the VSM model.
  - **Inflection Reduction: Lemmatization** is employed to reduce words to their base form. This approach is preferred over stemming for its accuracy in extracting root words.
  - **TF-IDF Matrix:** The TF-IDF matrix is computed as the product of Term Frequency (TF) and Inverse Document Frequency (IDF). Each column in the matrix is then normalized to unit length for the documents.

- **Ranking and Evaluation:**
  - **Ranking:** Similar pre-processing is applied to queries, and query vectors are generated. The document-query similarity is measured using cosine similarity. Documents are ranked based on the resulting similarity scores, where higher cosine values indicate greater similarity.
  - **Evaluation:** Ranked documents undergo evaluation using various metrics such as Mean Precision@k, Mean Recall@k, MAP@k, nDCG@k, and Mean F-Score@k to assess retrieval performance.

2.2 VSM-2

In this model, we modified how we preprocess the data in VSM-1. This subsection provides a detailed description of the methodology we adopted, and we will also discuss the improvements achieved in VSM-2 as compared to VSM-1. VSM-2 shows enhancements across all metrics when compared to VSM-1.

- **Data Pre-processing:**
  - **Including the Title of Documents:** In addition to the body of documents, we have incorporated the titles of documents to enhance information retrieval. Recognizing the significant value titles can provide, often surpassing that of the body content, we have amplified their importance by including them three times in the dataset. To reflect this emphasis, we recalculated the TF-IDF matrix, giving additional weightage to the titles.
  - **Uniform Decapitilization:** All letters are converted to lowercase to ensure uniformity in the text. This step is crucial to prevent the duplication of words due to case variations, thereby reducing the dimensionality of vectors. For example: If we don't change words like **'Airplane'** to lowercase, the model

treats **'Airplane'** and **'airplane'** as different, making our vectors bigger, a problem known as the **'Curse of Dimensionality'**.

– **Removing Numbers in the Text:** Numbers are removed from the corpus to focus solely on textual content, avoiding potential noise introduced by numerical values.

– **Removing Punctuations and Other Diacritics:** Special characters and extra white spaces are removed to maintain text cleanliness. Failure to do so may result in the VSM model considering punctuation as additional words, leading to inflated vocabulary size and dimensionality.

– **Common Pre-processing:** The same tokenizer and stopwords list as VSM-1 are utilized. Lemmatization, rather than stemming, is employed to maintain accuracy in reducing words to their base form. TF-IDF matrices are constructed for both documents and queries, following the same methodology as VSM-1.

- **Ranking and Evaluation:**

  – **Ranking:** Queries undergo similar pre-processing steps, resulting in query vectors. Document-query similarity is established using cosine similarity, and documents are ranked based on the resulting similarity scores.

  – **Evaluation:** Ranked documents are evaluated using metrics such as Mean Precision@k, Mean Recall@k, MAP@k, nDCG@k, and Mean F-Score@k to assess retrieval performance and compare it with VSM-1.

2.3 Results and Evaluation

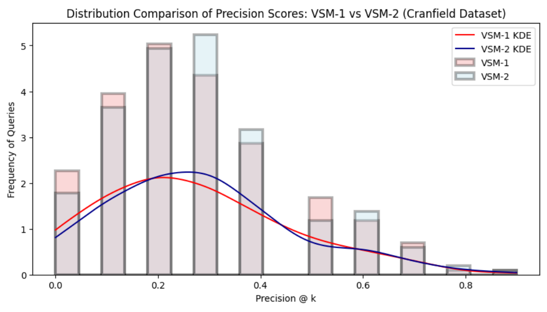
The comparison table for both the models i.e. VSM-1 and VSM-2 is shown in Table. 1

	VSM-1					**VSM-2**				
K	**MAP**	**n-DCG**	**f-score**	**Recall**	**Precision**	**MAP**	**n-DCG**	**f-score**	**Recall**	**Precision**
1	0.1019	0.4237	0.1685	0.1019	0.6133	0.1114	0.4560	0.1839	0.1114	0.6711
2	0.1616	0.4140	0.2392	0.1684	0.5177	0.1733	0.4390	0.2565	0.1793	0.5577
3	0.1965	0.4170	0.2734	0.2145	0.4637	0.2094	0.4332	0.2904	0.2274	0.4874
4	0.2205	0.4168	0.2924	0.2528	0.4200	0.2323	0.4304	0.3053	0.2635	0.4344
5	0.2371	0.4204	0.2998	0.2830	0.3804	0.2518	0.4345	0.3152	0.2964	0.3973
6	0.2510	0.4245	0.3039	0.3110	0.3496	0.2662	0.4386	0.3158	0.3206	0.3630
7	0.2621	0.4311	0.3040	0.3334	0.3250	0.2771	0.4425	0.3169	0.3459	0.3365
8	0.2696	0.4350	0.2997	0.3494	0.3027	0.2870	0.4517	0.3168	0.3696	0.3166
9	0.2771	0.4406	0.2972	0.3685	0.2854	0.2958	0.4585	0.3140	0.3895	0.2982
10	0.2842	0.4481	0.2964	0.3882	0.2724	0.3025	0.4647	0.3104	0.4076	0.2822

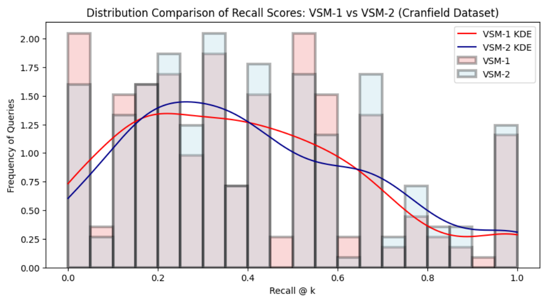
Table 1: Evaluation Metrics

From Table 1, it's evident that **Version-2** consistently outperforms Version-1 across all metrics by a margin of **2%** to **8%**. Version-1 had a vocabulary of approximately **9600** words, whereas Version-2 utilized a streamlined vocabulary of **8800** words, owing to enhanced text preprocessing techniques implemented in Version-2. The retention of punctuation in Version-1 had a detrimental effect on its similarity scores, thereby undermining its overall performance. Additionally, the incorporation of document titles played a pivotal role in the improvement observed from VSM-1 to VSM-2.

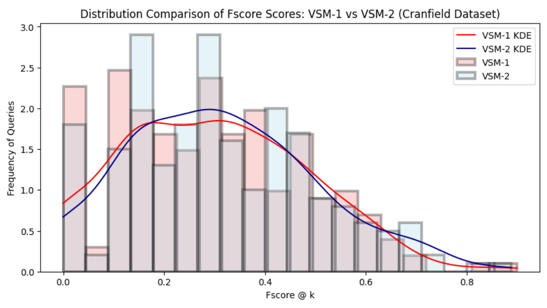
The distribution statistics of precision, recall, f-score, and nDCG of various queries @k = 10 for VSM-1 vs. VSM-2 is shown in figures 1a, 1b, 1c and 1d.



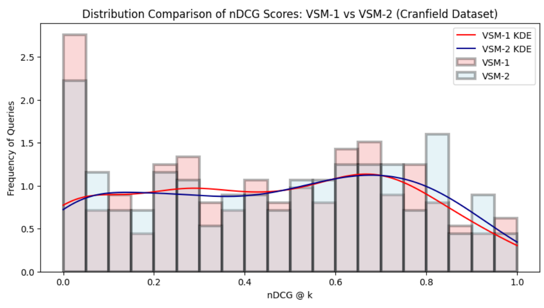
(a) Precision distribution for VSM-1 vs VSM-2 at $k = 10$



(b) Recall distribution for VSM-1 vs VSM-2 at $k = 10$



(c) F-score distribution for VSM-1 vs VSM-2 at $k = 10$

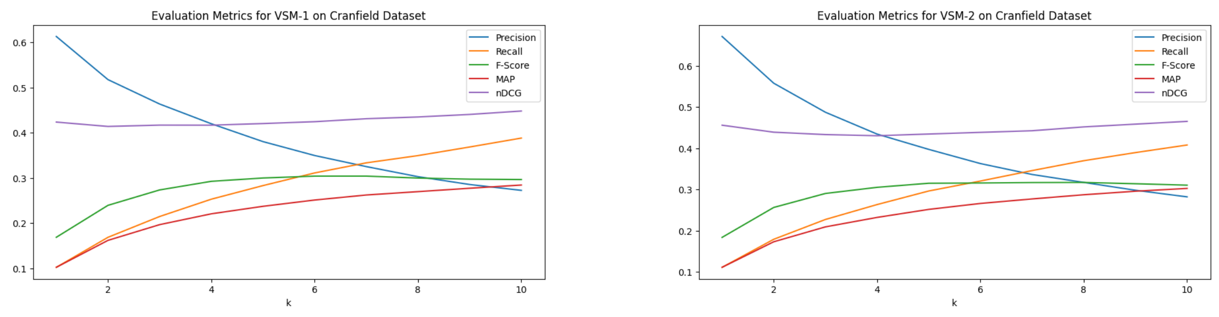


(d) nDCG distribution for VSM-1 vs VSM-2 at $k = 10$

Figure 1: Distribution statistics for Evaluation metrics

Results: The distributions for VSM-1 and VSM-2 indicate that the latter consistently exhibits higher values across all metrics, including precision, recall, f-score, and nDCG. This suggests that the **VSM-2** model performs better than VSM-1. Subsequently, hypothesis testing will be employed to further validate this observation, aiming to confirm that VSM-2 and VSM-1 do not perform similarly.

The general trends of the evaluation measures for VSM-1 and VSM-2, were as shown in Figure 2



(a) VSM-1 (b) VSM-2

Figure 2: Evaluation Metrics

Starting now, we'll be using **VSM-2** as our Baseline model for the rest of this report.

2.4 Example Testing on Cranfield Dataset

We will be comparing an example of a query from the Cranfield dataset on VSM-2 to test its retrieval by analyzing the first retrieved document and checking whether is it relevant to the query or not.

The query we are using as an example is 2nd query from the Cranfield Dataset.
Query: "what are the structural and aeroelastic problems associated with flight of high speed aircraft ."
Documents Retrieved: 12, 51, 746, 884, 1169

First Retrieved Document: "some structural and aeroelastic considerations of high speed flight . the dominating factors in structural design of high-speed aircraft are thermal and aeroelastic in origin . the subject matter is concerned largely with a discussion of these factors and their interrelation with one another . a summary is presented of some of the analytical and experimental tools available to aeronautical engineers to meet the demands of high-speed flight upon aircraft structures . the state of the art with respect to heat transfer from the boundary layer into the structure, modes of failure under combined load as well as thermal inputs and acrothermoelasticity is discussed . methods of attacking and alleviating structural and aeroelastic problems of high-speed flight are summarized . finally, some avenues of fundamental research are suggested ."

Observation: The query is related to the structural and aeroelastic problems encountered by an aircraft when flying at high speeds. As it can be observed in the first retrieved document, the document contains terms like structural, aeroelastic, flight, high speed, etc. which shows that the VSM-2 is working well on a lexical basis. There is a good enough overlap of words between the query and this document.

2.5 Observations

- As we increase k, precision tends to **drop**, while recall **rises**. This suggests that initially, the top retrieved documents are mostly relevant.
- The F-score shows an increasing trend and **stabilizes**, indicating a consistent balance between precision and recall. Initially, the F-score might be lower due to low recall.
- MAP exhibits a smoother curve with less fluctuation compared to precision, gradually increasing.
- nDCG shows **stability** across different k values, reflecting consistent IR system performance. This indicates effective document ranking based on relevance, irrespective of specific k values.

2.6 Hypothesis Testing

Null Hypothesis: We wanted to see if two different models, VSM-1 and VSM-2, give similar results regarding precision, recall, f-score, and n-DCG.

Alternate Hypothesis: The alternate hypothesis proposes that VSM-1 and VSM-2 do not perform similarly in terms of precision, recall, f-score, and n-DCG.

Strategy: We compared VSM-1 and VSM-2 using data from 225 queries in the Cranfield dataset. Then, we used statistical tests to check if they performed similarly across these metrics.

Results: After analyzing Table 2, we found that the p-values (which indicate how likely our results are due to chance) were all greater than **0.05**. This means there wasn't enough evidence to reject the idea that VSM-1 and VSM-2 perform similarly for the given queries.

Metric	t-statistic	p-value
Precision	-0.5630	0.5736
Recall	-0.7642	0.4451
Fscore	-0.7988	0.4247
nDCG	-0.5948	0.5522

Table 2: VSM-1 vs VSM-2 Hypothesis Testing

Conclusion: Based on our analysis, VSM-1 and VSM-2 give almost the same results for the queries we tested.

2.7 Limitations of Vector Space Model

1. **Curse of Dimensionality**: Lengthy documents suffer from poor representation and similarity values due to the high-dimensional space, making it challenging to

capture subtle similarities effectively.

2. **Assumption of Orthogonality**: The model assumes words are orthogonal, but in reality, their relationships are more complex, leading to oversimplification and potential inaccuracies.

3. **Semantic Sensitivity**: Documents with similar contexts but different terminologies aren't properly related, as the model relies solely on term occurrences rather than semantic equivalence.

4. **Loss of Term Order**: Sequential term arrangement in documents is ignored, resulting in the loss of valuable contextual information crucial for understanding document semantics.

5. **High Sparsity of Document Vectors**: Document vectors often exhibit high sparsity, leading to suboptimal similarity assessments and ineffective information retrieval.

6. **Problem with Synonyms**: As the model works on textual representation and not on semantics, it is unable to capture the context when synonyms are used. Hence, different results are obtained for synonyms of the same word.

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

3. Improving Document Representation

One of the issues with VSM-2 was the inherent assumption of orthogonality and the inefficient retrieval because of the curse of dimensionality. These issues are directly dependent on the representation of documents, which in the case of VSM-2 was the TF-IDF matrix. Another issue is about circularity between words and documents. We often infer that two documents are similar when the words are similar. On the other hand, to determine the similarity of words, we use their occurrence in similar documents. To solve this problem, we need to incorporate techniques like Factor Mode Analysis or its variants which help to use a different space to represent documents and words to eliminate or reduce the issues to a certain extent. We have used **Latent Semantic Analysis (LSA)** to address these issues. The detailed analysis of this method has been explained in the next subsections.

3.1 Latent Semantic Analysis (LSA)

In the representation used in VSM-2, the orthogonality and dimensionality issues can be eliminated by representing the documents as well as words in a concept space. This concept space is a space that has subtler or deeper information between the documents and words, which the textual representation itself doesn't carry. This technique is called

Latent Semantic Analysis (LSA) which represents the words and documents in the concept space to address these issues.

LSA decomposes a given term-document matrix containing the TF-IDF values using Singular Value Decomposition (SVD). The larger singular values can be used to construct a concept space to represent the terms and documents in this concept space.

Consider a $m \times n$ dimensional term-document matrix A, which represents m terms or words and n documents. Each row of the matrix A is a term represented in document space and each column is the document represented in the term space. Now, we perform SVD on this matrix as follows:-

$$A_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^T_{n\times n} \tag{1}$$

After performing SVD on this matrix, we obtain three matrices, namely, $m \times m$ dimensional U, $m \times n$ dimensional Σ, and $n \times n$ dimensional V. The column orthogonal matrices U and V represent the terms and documents in l dimensional vector space where $l = \min(m, n)$. The singular values in Σ are sorted in descending order and the largest k singular values are chosen for forming the new k dimensional concept space, where $k \leq l$. Hence, we can reconstruct the term-document matrix using the new concept space where the first k columns of U and V matrices are used. This can be explained as follows:-

$$A_{m\times n} = U_{m\times k}\Sigma_{k\times k}V^T_{k\times n} \tag{2}$$

Here, Σ is a diagonal matrix and U & V are representations of terms and documents respectively in k dimensional concept space which extracts the higher level associations between the terms and documents.

LSA solves the problem of circularity using Factor Analysis i.e. representing the terms and documents in a new concept space. Moreover, the problem of orthogonality is solved by reconstructing the term-document matrix only using a few singular values. LSA provides us with a good k-rank approximation of the original matrix. Additionally, the size of the data for processing is reduced. For the reconstruction of A, we can store the k-rank approximations of U, Σ, and V. It also solves the problem of sparsity of document vectors because all words will have a non-zero component in the document vector because of the subtle information captured by the concept space representation.

For this model, we need to tune the hyperparameter k in the implementation of LSA so that the loss incurred from this reconstruction is minimal and the performance of the retrieval system increases. This study is done in the next subsection.

3.2 LSA Hyperparameter Tuning

For tuning the hyperparameter k used in LSA, we plot the Mean Average Precision as a function of k to find the optimal value which increases the performance of the retrieval system.

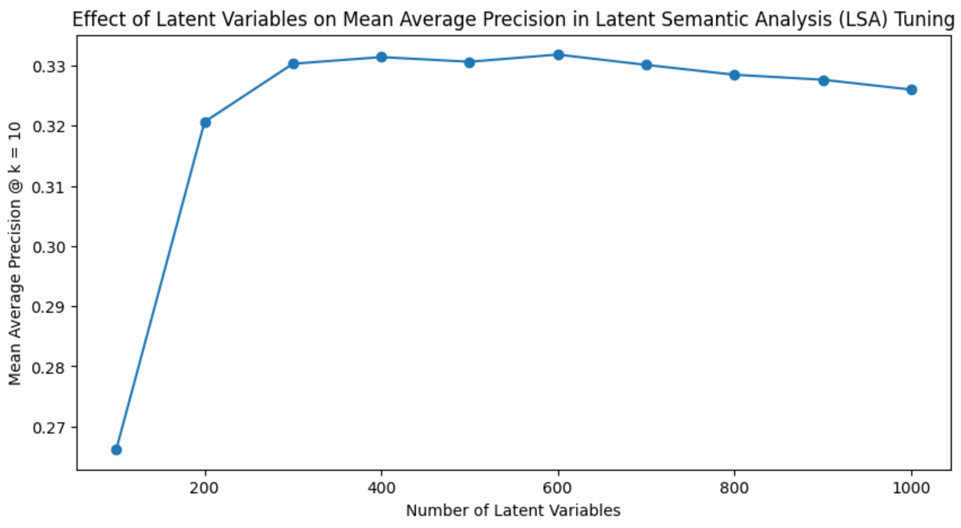


Figure 3: Variation of MAP with k

As observed in this plot, the optimal value of k for LSA is **600** with a maximum **Mean Average Precision** of **0.3318**. In other words, we will use 600-dimensional concept space to represent the words and 1400 documents in the Cranfield dataset to reconstruct the term-document matrix and retrieve documents.

3.3 VSM-2 vs. LSA

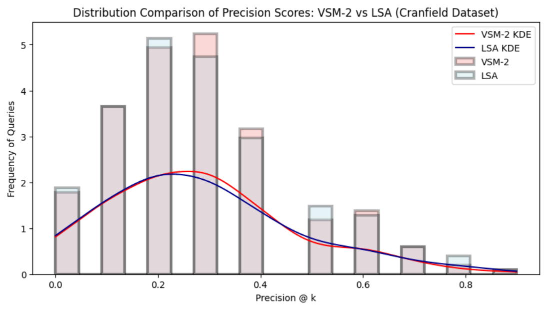
The comparison table for VSM-2 and the model using LSA on top of VSM-2 is shown in Table. 3

	Baseline Model (VSM-2)					**LSA with 600 latent variables**				
K	**MAP**	**n-DCG**	**f-score**	**Recall**	**Precision**	**MAP**	**n-DCG**	**f-score**	**Recall**	**Precision**
1	0.1114	0.4560	0.1839	0.1114	0.6711	0.1193	0.4927	0.1954	0.1193	0.7067
2	0.1733	0.4390	0.2565	0.1793	0.5577	0.1838	0.4671	0.2655	0.1879	0.5733
3	0.2094	0.4332	0.2904	0.2274	0.4874	0.2297	0.4618	0.3094	0.2461	0.5141
4	0.2323	0.4304	0.3053	0.2635	0.4344	0.2621	0.4658	0.3320	0.2916	0.4667
5	0.2518	0.4345	0.3152	0.2964	0.3973	0.2795	0.4697	0.3365	0.3219	0.4204
6	0.2662	0.4386	0.3158	0.3206	0.3630	0.2947	0.4761	0.3374	0.3490	0.3844
7	0.2771	0.4425	0.3169	0.3459	0.3365	0.3071	0.4849	0.3404	0.3768	0.3613
8	0.2870	0.4517	0.3168	0.3696	0.3166	0.3169	0.4908	0.3379	0.3983	0.3383
9	0.2958	0.4585	0.3140	0.3895	0.2982	0.3261	0.4957	0.3354	0.4179	0.3205
10	0.3025	0.4647	0.3104	0.4076	0.2822	0.3261	0.4982	0.3288	0.4321	0.3013

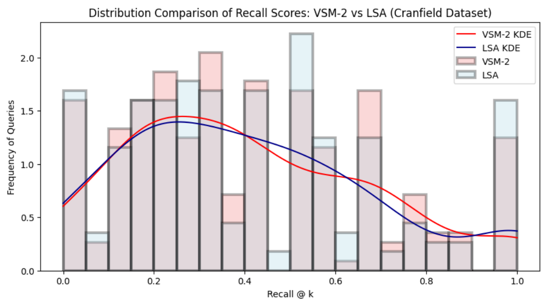
Table 3: Evaluation Metrics

From Table 3, it's evident that the model using **LSA** consistently outperforms VSM-2 across all metrics by a margin of **2.80%** to **12.83%** and the average improvement is **7.62%**. However, the improvement in terms of performance is not that significant. A possible reason could be that the words and documents in the Cranfield dataset could be independent enough so that the orthogonality used in VSM-2 makes sense. However, the small fraction of inter-dependency has been eliminated by incorporating LSA.

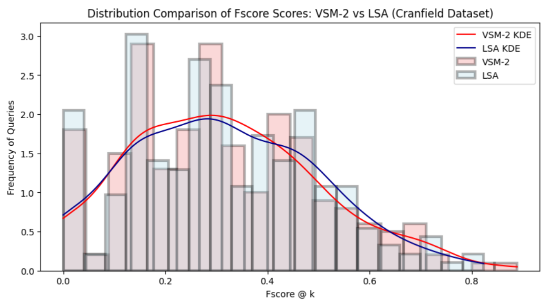
The distribution statistics of precision, recall, f-score, and nDCG of various queries @k = 10 for VSM-2 vs. LSA is shown in figures 4a, 4b, 4c and 4d.



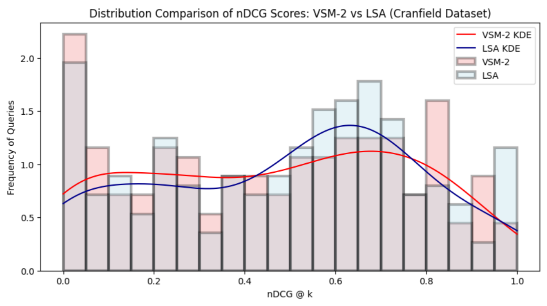
(a) Precision distribution for VSM-2 vs LSA at $k = 10$



(b) Recall distribution for VSM-2 vs LSA at $k = 10$



(c) F-score distribution for VSM-2 vs LSA at $k = 10$



(d) nDCG distribution for VSM-2 vs LSA at $k = 10$

Figure 4: Distribution statistics for Evaluation metrics

Results: From the distributional graphs, we observe that LSA shifts the distribution slightly towards the right. This implies that Document Representation is improved using Latent Semantic Analysis. Subsequently, hypothesis testing will be employed to validate this observation further, aiming to confirm that VSM-2 and LSA **do not perform similarly**.

The general trends of the evaluation measures for LSA, were as shown in Figure 5

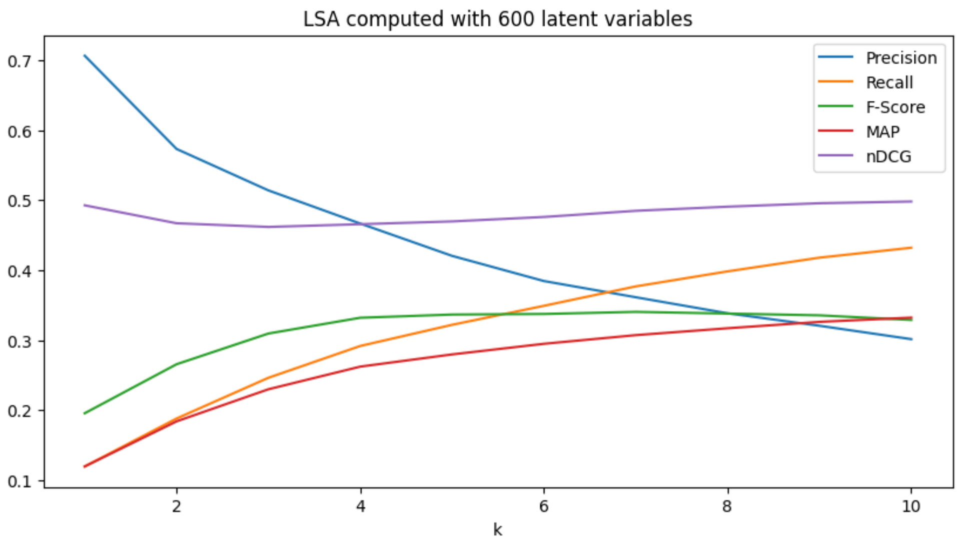


Figure 5: LSA

3.4 Observations

We will see some examples and analyze how LSA performed better than our baseline model.

Query ID: 109

Query: panels subjected to aerodynamic heating.

Retrrieval by VSM-2 : [31, 51, 859, 627, 864, 1008, 142, 285, 982, 856]

Retrieval by LSA : [859, 1008, 658, 857, 856, 627, 391, 766, 858, 864]

Document - 627 (Retrieved by VSM-2): "flutter analysis of circular panels . the flutter problem of flat circular panels with edges elastically restrained against rotation has been formulated in terms of small-deflection plate theory . the panel is subjected to uniform all-round tension or compression in its middle plane, in addition to the supersonic compressible flow passing over its upper surface with still air below . linear piston theory is employed to predict the aerodynamic load on the vibrating panel . the problem is investigated by a rayleigh-type analysis involving chosen modes of the panel as degrees of freedom . in order to investigate the convergence of the solution, the flutter-mode shape of the clamped-edge panel has been expressed in a series form in powers of r cos o . the results of three-, four-, and five-term approximations have displayed oscillatory behavior with apparently rapid convergence of the solution."

Document - 766 (Retrieved by LSA): "experimental investigation at mach number of 3. 0 of effects of thermal stress and buckling on flutter characteristics of flat single-bay panels of length-width ratio 0. 96 . flat, single-bay, skin stiffener panels with length-width ratios of 0.96 were tested at a mach number of 3.0, at dynamic pressures ranging from 1,500 to stagnation temperatures from 300 f to effects of thermal stress and buckling on the flutter of such panels . the panel supporting structure allowed partial thermal expansion of the skins in both the longitudinal and lateral directions . panel skin material and skin thickness were varied . a boundary faired through the experimental flutter points consisted of a flat-panel portion, a buckled-panel portion, and a transition point, at the intersection of the two boundaries, where a panel is most susceptible to flutter . the flutter region consisted of two fairly distinct sections, a large-amplitude flutter region and a small-amplitude flutter region . the results show that an increase in panel skin temperature flutter . the flutter trend for buckled panels is reversed . use of a modified temperature parameter, which approximately accounts for the effects of differential pressure and variations in panel skin material and skin thickness, reduced the scatter in the data which resulted when these effects were neglected . the results are compared with an exact theory for clamped panels for the condition of zero midplane stress . in addition, a two-mode /transtability/ solution for clamped panels is compared with the experimentally determined transition point."

Observations:

- **Higher Order Associations:** LSA demonstrated superior performance by capturing higher order associations between words. For instance, in document 766, LSA successfully identified the relevance of terms like "thermal" and "temperature" to the concept of "heating" in the query. This ability to understand contextual relationships enabled LSA to retrieve the relevant document accurately. In contrast, VSM-2 struggled to retrieve documents based on such higher-order associations.

- **Non-Exact Word Matching:** Unlike VSM-2, which relies heavily on exact word matching, LSA exhibited a more nuanced approach. For instance, LSA successfully captured the semantic similarity between terms like "**flat-panel**" and "**buckled-panel**" even when the query contained only the term "panels". This flexibility in understanding semantic relationships beyond exact word matches contributed to LSA's effectiveness in retrieving relevant documents.

3.5 Hypothesis Testing LSA vs. VSM-2

Null Hypothesis: We wanted to see if two different models, VSM-2 and LSA, give similar results regarding precision, recall, f-score, and n-DCG.

Alternate Hypothesis: The alternate hypothesis proposes that VSM-2 and LSA do not perform similarly in terms of precision, recall, f-score, and n-DCG.

Strategy: We compared VSM-2 and LSA using data from 225 queries in the Cranfield dataset. Then, we used statistical tests (two sample two-tailed t-test) to check if they performed similarly across these metrics.

Results: After analyzing Table 4, we found that the p-values (which indicate how likely our results are due to chance) were all greater than **0.05**. This means there wasn't enough evidence to reject the idea that VSM-2 and LSA perform similarly for the given queries.

Metric	**t-statistic**	**p-value**
Precision	-1.0891	0.2769
Recall	-0.9368	0.3494
Fscore	-1.0390	0.2993
nDCG	-1.2007	0.2305

Table 4: VSM-2 vs LSA Hypothesis Testing

3.6 Limitations of LSA

1. **Computational Complexity:** The primary limitation of LSA is the substantial computational effort required by the Singular Value Decomposition (SVD) algorithm. The computational complexity of SVD is $O(n^2 \times k^3)$, where n represents the combined count of documents and terms, and k denotes the number of embedded latent dimensions.

2. **Loss of Interpretability:** LSA sacrifices interpretability to some extent. The transformed representation of documents and terms in the latent semantic space may not be readily interpretable by humans, hindering the ability to extract meaningful insights directly from the model.

3. **Semantic Ambiguity:** LSA partially addresses the problem of Synonymy, which refers to words with similar meanings. However, it does not effectively handle Polysemy, which occurs when the same word has different meanings in different contexts.

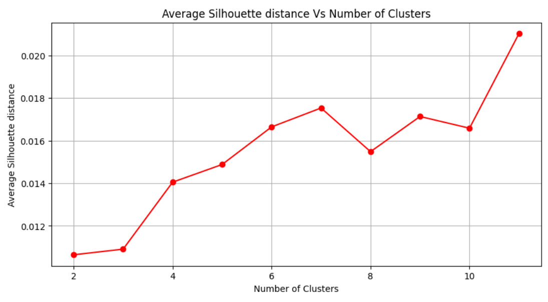
4. Optimizing Retrieval Time with Clustering Method

To speed up finding documents as our collection grows, we're using **K-means clustering**, one of the most renowned clustering techniques. This method groups similar documents based on their content. So, when we search, we're not looking through every single document. Instead, we're searching within these groups, which makes finding what we need much faster.

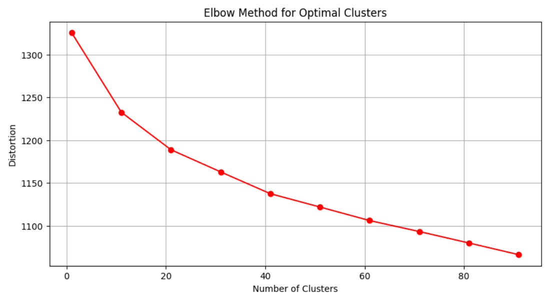
4.1 K-means Clustering

Clustering is an unsupervised method of grouping data. In our approach, we employ K-means clustering, which is an application of the Expectation-Maximization (EM) algorithm. The implementation steps are outlined as follows:

1. Apply K-means clustering on the TF-IDF matrix of documents, grouping them into 'k' clusters. The value of 'k' is determined using an elbow plot, with the optimal value typically around **9**. See Figure 6.
2. Given a query, calculate the cosine similarity between the query and the 'k' cluster centroids. Based on these similarity scores, assign the query to the most relevant cluster.
3. Calculate the cosine similarity between the query and all documents within the assigned cluster, ranking these documents accordingly.



(a) Silhouette Scores vs Number of Clusters



(b) Elbow Method Analysis

Figure 6: Cluster Optimization for Document Retrieval

4.2 Observations

4.2.1 Comparing Retrieval Time

We compare the retrieval time between the Clustering model and the Base model to assess their efficiency in document retrieval. A comparison of average retrieval times between the two models is presented in Table 5. The analysis reveals that the retrieval time in the Clustering model is reduced by a factor of **3** compared to the Base model.

Comparision	Clustering(number of clusters is 9)	VSM-2
Mean Retrieval Time	4 ms	12 ms

Table 5: Retrieval Time Comparision

4.3 Hypothesis Testing LSA without clustering vs. LSA with clustering

Null Hypothesis: The mean retrieval time of LSA with and without clustering techniques is identical.

Alternate Hypothesis: The mean retrieval time of LSA with and without clustering techniques varies.

Strategy: We observed the retrieval time for LSA with and without clustering across **200** queries sourced from the Cranfield collection. A two-tailed t-test was performed with a significance level (α) of **0.05** to assess the hypotheses.

Results: Our statistical analysis revealed a significant **t-value** of 18 and an extremely small **p-value** of 2.968e-52, nearly 0. The p-value is much less than the significance level ($\alpha = 0.05$), indicating strong evidence against the null hypothesis. Therefore, we reject the **null hypothesis** in favor of the alternative hypothesis.

Conclusion: Based on our findings, we conclude that clustering significantly reduces document retrieval time for queries. This conclusion is supported by the observed improvement in the performance of clustering over non-clustering, as evident from Figure 7.

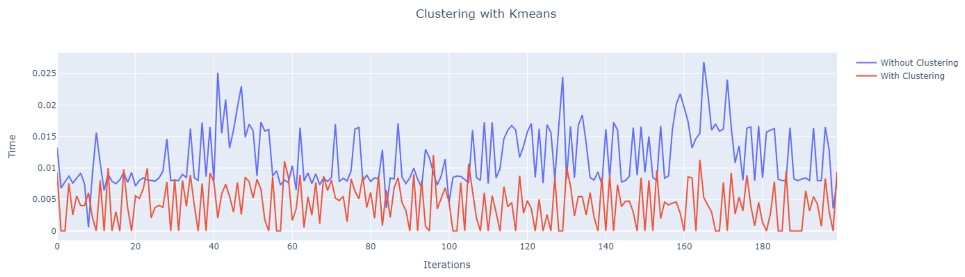


Figure 7: The graph provides a visual comparison of retrieval times for LSA with and without clustering across the 200 queries.

5. Overcoming Synonym Ambiguity with Query Expansion

The motivation behind query expansion lies in the inherent ambiguity and variability of human language. Words can have multiple meanings and interpretations, leading to potential mismatches between the user's query and the desired information. By incorporating similar words into the query, query expansion helps to overcome this challenge by capturing additional facets of the user's intent.

To achieve effective query expansion, a reliable model is required to identify words that share similar meanings. One such model is **FastText**, a robust word embedding framework developed by Facebook's AI Research (FAIR). FastText represents words as dense vectors in a continuous multi-dimensional space, capturing both semantic and syntactic information. This enables the model to understand the contextual meaning of words and establish meaningful relationships between them.

5.1 Query Expansion (FastText Model)

Query Expansion is a technique used to broaden search queries by introducing additional tokens or phrases. In this approach, the search engine automatically modifies the query to include these expanded terms. Consider a search query for "smartphone reviews". Through query expansion, this query could be broadened to include related terms, such as "mobile phone review" or "cell phone reviews". Here's how it works:

1. We begin by training a pre-trained FastText model on a corpus of text, which allows us to generate vector representations (also known as word embeddings) for all the words in the corpus.

2. Next, for each word in the original query, we identify the top most similar word based on cosine similarity with the query word and append it to the query. For example:
   Query: aircraft
   Top most similar words: ['craft', 'hovercraft', 'erode', 'vtol', 'damage', 'spacecraft', 'operating', 'design', 'modest', 'role']

3. Now we take the query's vector representation and rank the documents as per cosine similarity values.

5.2 Query Expansion vs VSM-2

The comparison of the Query Expansion model is shown in the table 6

	Baseline Model (VSM-2)					Query Expansion (FastText Model)				
K	**MAP**	**n-DCG**	**f-score**	**Recall**	**Precision**	**MAP**	**n-DCG**	**f-score**	**Recall**	**Precision**
1	0.1114	0.4560	0.1839	0.1114	0.6711	0.1042	0.4355	0.1707	0.1042	0.6044
2	0.1733	0.4390	0.2565	0.1793	0.5577	0.1608	0.4345	0.2442	0.1718	0.5288
3	0.2094	0.4332	0.2904	0.2274	0.4874	0.1969	0.4250	0.2765	0.2198	0.4607
4	0.2323	0.4304	0.3053	0.2635	0.4344	0.2180	0.4235	0.2869	0.2515	0.4033
5	0.2518	0.4345	0.3152	0.2964	0.3973	0.2357	0.4270	0.2967	0.2824	0.3706
6	0.2662	0.4386	0.3158	0.3206	0.3630	0.2475	0.4304	0.2974	0.3057	0.34
7	0.2771	0.4425	0.3169	0.3459	0.3365	0.2576	0.4354	0.2964	0.3279	0.3136
8	0.2870	0.4517	0.3168	0.3696	0.3166	0.2684	0.4419	0.2987	0.3507	0.2983
9	0.2958	0.4585	0.3140	0.3895	0.2982	0.2773	0.4496	0.2998	0.3758	0.2844
10	0.3025	0.4647	0.3104	0.4076	0.2822	0.2847	0.4564	0.2964	0.3926	0.2702

Table 6: Evaluation Metrics

The general trends of evaluation metrics for Query Expansion is shown in figure 8

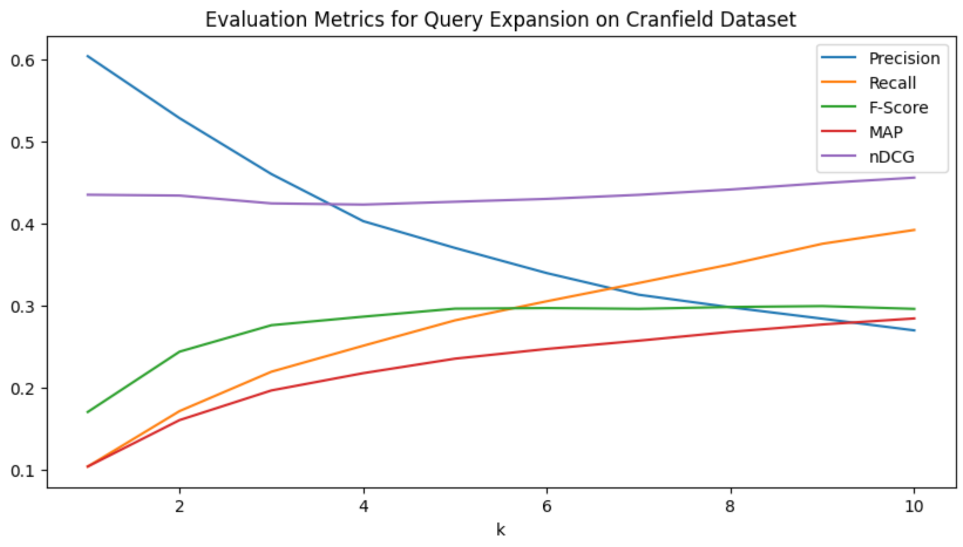
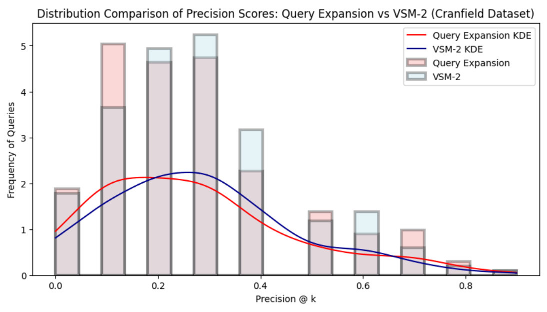
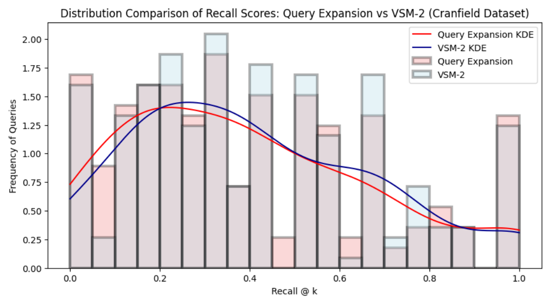


Figure 8: Query Expansion Evaluation Metrics

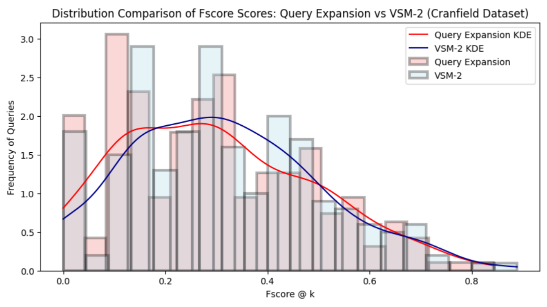
The distribution statistics of precision, recall, f-score, and nDCG of various queries @k = 10 for VSM-2 vs. Query Expansion is shown in figures 9a, 9b, 9c and 9d



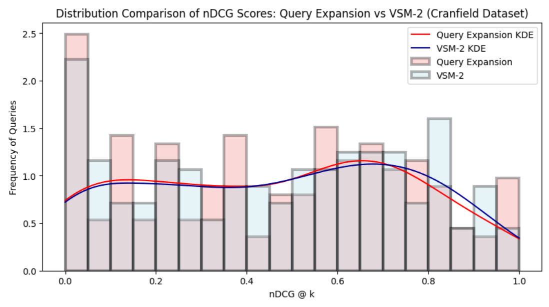
(a) Precision distribution for QE vs VSM-2 at $k = 10$



(b) Recall distribution for QE vs VSM-2 at $k = 10$



(c) F-score distribution for QE vs VSM-2 at $k = 10$



(d) nDCG distribution for QE vs VSM-2 at $k = 10$

Figure 9: Distribution statistics for Evaluation metrics

5.3 Observations

The leftward shift in the distribution of Query Expansion (QE) suggests poor performance, possibly due to limited FastText representation stemming from a small training dataset.

5.4 Hypothesis Testing VSM-2 Vs Query Expansion

Null Hypothesis: We wanted to see if two different models, VSM-2 and Query Expansion, give similar results regarding precision, recall, f-score, and n-DCG.

Alternate Hypothesis: The alternate hypothesis proposes that VSM-2 and Query Expansion do not perform similarly in terms of precision, recall, f-score, and n-DCG.

Strategy: We compared VSM-2 and Query Expansion using data from 225 queries in the Cranfield dataset. Then, we used statistical tests (two sample two-tailed t-test) to check if they performed similarly across these metrics.

Results: After analyzing Table 7, we found that the p-values (which indicate how likely our results are due to chance) were all smaller than **0.05**. This means there was enough evidence to reject the idea that VSM-2 and Query Expansion perform similarly for the given queries.

Metric	t-statistic	p-value
Precision	-2.5670	0.0023
Recall	-1.7534	0.00456
Fscore	-2.4835	0.00328
nDCG	-2.8348	0.00087

Table 7: VSM-2 vs Query Expansion Hypothesis Testing

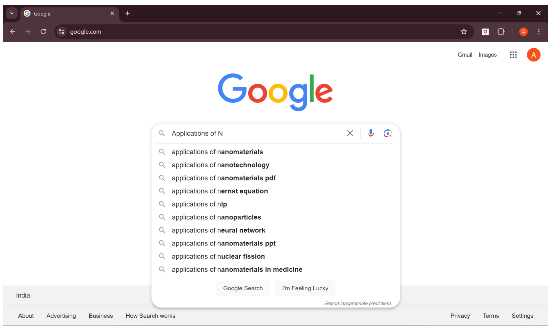
Conclusion: We can confidently reject the null hypothesis, indicating that Query Expansion (QE) and VSM-2 perform differently for a given query. Upon examining the distributions, it's evident that the Query Expansion model showed slight improvements for some queries but performed worse for others.

6. Autocompletion

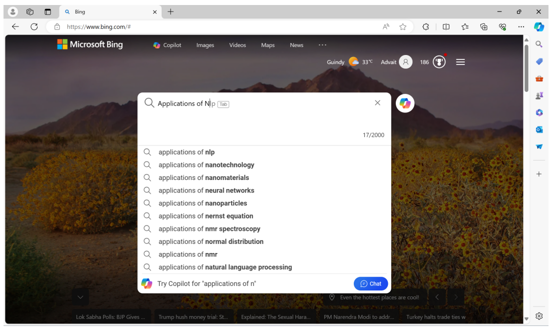
Autocompletion suggests a completed version of a query when a few initial words are typed. These suggestions are closer to the user's intent and can help him/her search documents with minimized efforts.

6.1 Introduction

Autocompletion is widely used in web and local search engines to complete the query. It is also extensively used in writing email drafts and on messaging platforms like WhatsApp, etc. Due to the development of AI, various grammar correction applications like Grammarly predict the entire query before we complete it.


Google
Applications of N
applications of nanomaterials
applications of nanotechnology
applications of nanomaterials pdf
applications of nernst equation
applications of nlp
applications of nanoparticles
applications of neural network
applications of nanomaterials ppt
applications of nuclear fission
applications of nanomaterials in medicine
Google Search
I'm Feeling Lucky
Report inappropriate predictions
How Search works

(a) Autocompletion from Google


Microsoft Bing
Copilot
Images
Videos
Maps
News
Guindy
33°C
Advait
Applications of Nlp
Tab
17/2000
applications of nlp
applications of nanotechnology
applications of nanomaterials
applications of neural networks
applications of nanoparticles
applications of nernst equation
applications of nmr spectroscopy
applications of normal distribution
applications of nmr
applications of natural language processing
Try Copilot for "applications of n"
Chat
Even the hottest places are cool!

(b) Autocompletion from Bing

Figure 10: Autocompletion for the incomplete query - "Applications of N"

In the above example, the query we intend to write is "Applications of NLP", but when the above incomplete query is typed, the search engine shows different examples that are quite relevant to our intent.

6.2 Approach for Autocompletion

The approach used in making this functionality of autocompletion is explained as follows:-

1. The first step is to import the JSON queries and their IDs from the file 'cran-queries.json'.
2. After merging all queries with punctuation marks, the tokenizer from Tensorflow is used to tokenize the dataset.
3. Input sequences are generated by making sequences using original queries, which are further padded to ensure uniform length of all samples.
4. The next step is to create one hot label using the no. of words and labels obtained from padded queries.
5. The model we use consists of a layer creating embeddings followed by an LSTM layer with 50 nodes followed by a softmax layer with no. of nodes equal to the no. of words.
6. The model was trained using Adam optimizer with Categorical Cross Entropy as the loss function for 200 epochs at a learning rate of 0.001.
7. Finally, a few examples are tested with the incomplete query and no. of words to be predicted as the input, and the output is the complete query.

The **training accuracy** of the model reached after training was **92.60 %.** The model can capture the context of the queries, and this has been demonstrated in the examples given in the next subsection.

6.3 Examples

The model was trained on the Cranfield dataset's queries and it was tested on a few of the queries as follows:-

1. **Incorrect Query:** "to find an approximate correction for"
   Predicted Complete Query: "to find an approximate correction for thickness in slender thin wing theory"
   Original Complete Query: "to find an approximate correction for thickness in slender thin wing theory ."

2. **Incorrect Query:** "why does the compressibility transformation fail to correlate the high"
   Predicted Complete Query: "why does the compressibility transformation fail to correlate the high speed data for helium and air"

Original Complete Query: "why does the compressibility transformation fail to correlate the high speed data for helium and air ."

3. **Incorrect Query:** "how is the heat transfer downstream of the mass transfer region effected by mass transfer at"
   Predicted Complete Query: "how is the heat transfer downstream of the mass transfer region effected by mass transfer at the nose of a blunted cone"
   Original Complete Query: "how is the heat transfer downstream of the mass transfer region effected by mass transfer at the nose of a blunted cone ."

4. **Incorrect Query:** "does transition in the hypersonic wake"
   Predicted Complete Query: "does transition in the hypersonic wake depend on body geometry and size"
   Original Complete Query: "does transition in the hypersonic wake depend on body geometry and size"

5. **Incorrect Query:** "what is a criterion that the transonic flow around an airfoil with a round leading edge be validly analyzed"
   Predicted Complete Query: "what is a criterion that the transonic flow around an airfoil with a round leading edge be validly analyzed by the linearized transonic flow theory"
   Original Complete Query: "what is a criterion that the transonic flow around an airfoil with a round leading edge be validly analyzed by the linearized transonic flow theory ."

Observation:- It can be observed that this autocorrection model is completing the queries highly accurately.

7. Summary

In this study, we constructed two baseline models, Baseline-1 and Baseline-2, operating as vector space models. The distinction lies in their preprocessing techniques, with Baseline-2 undergoing more extensive preprocessing, resulting in a reduction of dimensions within the vector space model. Empirical results indicated a marginal performance enhancement of **Baseline-2** over Baseline-1.

Subsequently, we applied Latent Semantic Analysis (LSA) to both preprocessed datasets to obtain improved representations of the term-document matrix. Notably, LSA applied to extensively preprocessed data yielded the most favorable outcomes among all models, primarily by refining the rank ordering of retrieved documents.

Moving on to Section 4, we aimed to make our model work better by improving precision, recall, and other important measures. We also kept in mind that Information

Retrieval (IR) systems need to be fast and efficient. So, we focused on making document retrieval faster. We used document clustering techniques for this purpose and found that it indeed made a big difference in **retrieval time**. However, there was a trade-off - while retrieval time improved, the precision and recall went down a bit, which we expected.

In Section 5, we introduced a new approach termed the "Query Expansion Model," aimed at enhancing the query by adding the top-k similar words to it, subsequently utilizing this expanded query for the Information Retrieval (IR) system. Surprisingly, this method did not yield improved efficacy. This could potentially be attributed to the limited size of the training corpus.

Our best-performing model heavily relies on exact query words, which may occasionally be absent from the corpus. Consequently, in Section 6, we implemented a Query Auto-Completion feature to assist users in formulating appropriate queries. It is worth noting that while this auto-completion feature holds practical utility in real-world deployment scenarios, its relevance diminishes when evaluating systems using a predefined set of test queries.

References

[1] 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.

[2] Navigli, R. (2009). Word Sense Disambiguation: A survey. ACM computing surveys (CSUR), 41(2), 1-69.

[3] Piotr Bojanowski, Armand Joulin, Tomas Mikolov, fastText - Meta Research, 2016

[4] Analytics Vidhya, What is LSTM? Introduction to Long Short-Term Memory

[5] GeeksforGeeks, Latent Semantic Analysis

[6] Wikipedia, Latent Semantic Analysis

[7] GeeksforGeeks, K-means Clustering

[8] Wikipedia, K-means Clustering

[9] Wikipedia, Polysemy

[10] Wikipedia, Word Sense Disambiguation