PROJECT REPORT

CS6370: Natural Language Processing

Deepankar Mishra CS22M041, Nitesh Rakesh Patwa CS22M060

Introduction-

IR System

Information Retrieval is a field of study that deals with the process of retrieving relevant information from a collection of documents. The goal of IR is to provide users with a set of documents that best match their information needs. This process typically involves several steps, including document collection, indexing, query processing, ranking, and presentation.

Document collection involves gathering a set of documents from various sources, such as web pages, scientific papers, or books. Once the documents are collected, they need to be processed and indexed to enable efficient searching. This involves analysing the documents and creating an index that maps each term to the set of documents that contain it. When a user submits a query to an IR system, the query is processed to identify the relevant terms and retrieve the set of documents that match those terms. The system then ranks the documents based on their relevance to the query and presents them to the user in an ordered list. Ranking is a critical step in IR because it determines the order in which documents are presented to the user. The ranking algorithm typically takes into account factors such as the frequency of the query terms in the document, the proximity of the query terms, and the authority of the document. Finally, the results are presented to the user in a format that makes it easy for them to find the information they need. This may involve displaying snippets of text from the documents or providing related searches.

Overall, Information Retrieval is a complex and important field that plays a critical role in enabling users to access and discover relevant information from large collections of documents.

Query-Auto-Complete

Query autocomplete is a popular feature in Information Retrieval (IR) systems that helps users search more efficiently by suggesting relevant queries as they type. This feature has become increasingly common in web search engines like Google and Bing, as well as in other applications like e-commerce and social media platforms. The query autocomplete feature uses machine learning algorithms and statistical models to predict and suggest query terms based on the user's partial input. As the user types, the IR system analyzes the partial query and suggests a list of possible completions that match the user's intent. For example, if a user types "best laptop" into a search engine, the system may suggest "best laptop for gaming" or "best laptop for business" based on the most commonly searched queries for laptops. The user can then select the suggested query that best matches their intent or continue typing to refine their search further. Query autocomplete is beneficial in several

ways. First, it saves users time by providing relevant suggestions that reduce the number of keystrokes required to complete a search. Second, it helps users discover new and relevant topics that they may not have considered before. Finally, it can improve the accuracy and relevance of search results by suggesting more precise and specific queries.

Problem Statement

Dataset : We have been provided with the Cranfield dataset, which is a widely used benchmark dataset for information retrieval. This dataset consists of 1400 documents and 225 queries, each of which is associated with a list of relevant documents along with their corresponding relevance scores. The relevance scores indicate whether a document is highly relevant, partially relevant, or not relevant at all to a given query.

Objective : Our objective is to develop a fully functional information retrieval system that can take queries as input and return k most relevant documents as output. The entire IR system is documented in its database, and we need to implement all the required components, such as preprocessing, indexing, ranking, and evaluation, to build the system end-to-end. Our ultimate goal is to create a robust and accurate IR system that can effectively retrieve the most relevant documents for any given query.

IR SYSTEM

Background

We had previously designed an IR system which used the Cranfield dataset, and was based on the vector space model. The TF-IDF (Term Frequency-Inverse Document Frequency) vector space model is a commonly used technique for information retrieval (IR) systems. It is a way of representing text documents as vectors in a high-dimensional space, with each dimension corresponding to a term in the document collection.

The TF-IDF model is based on the idea that words that appear frequently in a document but rarely in the rest of the collection are good indicators of the document's content. The model uses two main components to calculate the relevance of a document to a user's query:

1. Term Frequency (TF): The number of times a term appears in a document. Documents that contain the query terms more frequently are considered more relevant.
2. Inverse Document Frequency (IDF): The inverse of the frequency of a term across the entire document collection. Terms that appear in many documents are less important than those that appear in a few documents.

To calculate the TF-IDF score for a term in a document, we multiply the term frequency by the inverse document frequency. This produces a score that represents the importance of the term in the document relative to the rest of the collection.

Once we have calculated the TF-IDF scores for all the terms in a document, we can represent the document as a vector in the high-dimensional space, with each dimension corresponding to a term. The length of the vector represents the magnitude of the document in the space, and the cosine similarity between two vectors represents the similarity between the documents. The Vector Space Model (VSM) is a widely used approach for representing text documents in Information Retrieval (IR) systems. It is based on the assumption that documents can be represented as vectors in a high-dimensional space, with each dimension corresponding to a term in the document collection. However, there are several additional assumptions that underlie the VSM approach, including:

1. Independence: The VSM assumes that the terms in a document are independent of each other, and that the presence or absence of one term does not affect the presence or absence of other terms. This assumption may not always hold in practice, as some terms may be related or co-occur frequently in a document.
2. Bag-of-words: The VSM assumes that the order of terms in a document does not matter, and that each document can be represented as a "bag-of-words" with no regard to the syntax or grammar of the language. This assumption ignores the fact that the ordering of words can affect the meaning of a sentence or document.
3. Fixed vocabulary: The VSM assumes that the vocabulary of the document collection is fixed and known in advance. This may not always be the case, as new terms may arise over time or in different contexts.

In this work we address the limitations of Vector space Model and suggest Latent Semantic Analysis as a possible way of improving our evaluation metrics.

We also build a query auto complete feature for the new IR system which can improve the accuracy and relevance of search results by suggesting more precise and specific queries.

Motivation

Latent Semantic Analysis (LSA) is a technique for analyzing and representing the meaning of words and documents in a high-dimensional space. One motivation for using LSA over a baseline VSM model is that it can capture more complex relationships between words and documents by modeling their underlying semantic structure. Whereas VSM models represent words and documents as bags of discrete, independent features, LSA models use a continuous, low-dimensional space to represent them as vectors that encode their semantic similarity. This allows LSA to capture more subtle semantic relationships between words and documents that may not be evident in a simple VSM model.

Another advantage of LSA over VSM is that it can handle synonymy more effectively. Synonymy refers to the fact that different words can have the same or similar meanings, while polysemy refers to the fact that the same word can have multiple meanings. VSM models treat each word as a separate feature, which can lead to sparsity and poor performance when dealing with synonymy and polysemy. LSA, on the other hand, can identify and capture the underlying semantic relationships between words that share similar meanings, even if they are not exact synonyms.

In addition, LSA can help overcome the problem of data sparsity that often arises in NLP tasks. When working with large datasets, it is common for many words to occur only rarely, or even only once. This can make it difficult to build effective models using traditional VSM approaches, which rely on the co-occurrence of words within a fixed context window. LSA can help alleviate this problem by identifying latent, or hidden, relationships between words that may not be apparent from their individual occurrences.

Overall, the motivation to use LSA over a baseline VSM model is to capture more complex semantic relationships between words and documents, handle synonymy and polysemy more effectively, and overcome the problem of data sparsity. While LSA may require more computational resources and may be more complex to implement than a simple VSM model, the potential benefits of using LSA can possibly lead to improved performance and better results in a variety of NLP tasks, including ours at hand, designed an effective IR retrieval system.

Methodology

We have a baseline model, which is what we built in assignment 2, a vector space model based IR system. Our methodology involves addition of LSA to find the similarity scores now, hence our new Model is an LSA based IR system.

The baseline model comprises several steps, outlined below:

1. Sentence Segmentation: The Punkt tokenizer from the NLTK package was employed.
2. Word Tokenization: Penn Treebank Tokenizer was used.
3. Stopword Removal: We eliminated all stopwords specified in the nltk package.
4. Lemmatization: Using the spacy package, we lemmatized words to their root form.
5. Information Retrieval: We transformed the documents into their TF-IDF vector forms. Additionally, we converted the queries to similar TF-IDF vector forms. We then used cosine similarity to rank the documents based on their vector similarity to the query.
6. Evaluation: Our methods were evaluated using various metrics, including Precision, Recall, normalized Discounted Cumulative Gains, F1-score. Only nDCG utilized the relevance scores of the documents corresponding to each query out of all the metrics mentioned.

The proposed LSA based Model (LSA Model) for IR system, outlined below-

1. Sentence Segmentation: The Punkt tokenizer from the NLTK package was employed.
2. Word Tokenization: Penn Treebank Tokenizer was used.
3. Stopword Removal: We eliminated all stopwords specified in the nltk package.
4. Lemmatization: Using the spacy package, we lemmatized words to their root form.
5. Information Retrieval: A TF-IDF cross Document matrix was formed, and then SVD would be done, we choose k topics(The rank approximation) , and then convert each document into into new form in the k-topics space. Similarly every incoming query would be converted to now lie on the k-topics space, and now use the cosine similarity method to find the similarity between the new document and query projection.
6. Evaluation: Our methods would be evaluated using various metrics, including Precision, Recall, normalized Discounted Cumulative Gains, F1-score. Only nDCG utilized the relevance scores of the documents corresponding to each query out of all the metrics mentioned.

Latent Semantic Analysis Implementation-

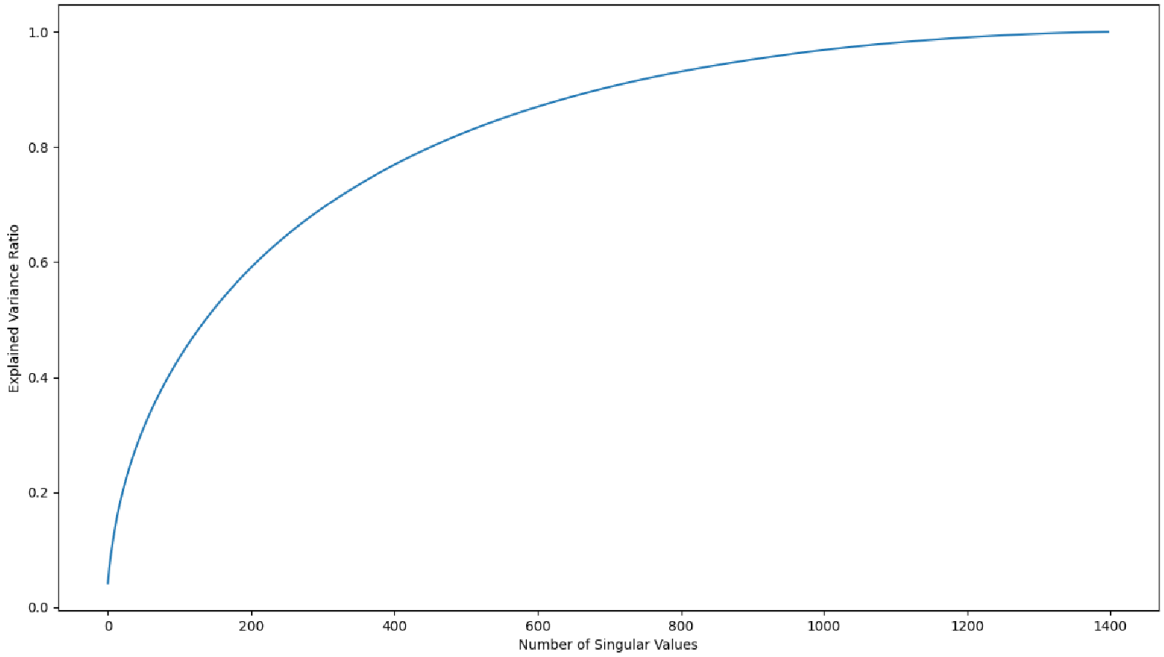
The goal of LSA is to uncover the latent (hidden) relationships between words and concepts in a corpus of text. To implement LSA, we start by creating a term-document matrix that represents the frequency of each term in each document. This matrix is then subjected to singular value decomposition (SVD), a mathematical technique that factors the matrix into three components: U, S, and V.

The resulting U and V matrices represent the semantic structure of the terms and documents, respectively, and the S matrix contains the singular values that measure the importance of each dimension in the decomposition. By reducing the dimensions of the matrix, we can extract the most important underlying concepts and relationships between words.

The S matrix contains the singular values that measure the importance of each dimension in the decomposition. By selecting the top-k singular values, we can approximate the original matrix with a rank-k matrix. The value of k is typically chosen to capture a high percentage of the total explained variance.

For example, if the total sum of squared singular values is 100 and we select the top-3 singular values, which sum up to 75, then we have captured 75% of the total explained variance. If we select the top-10 singular values, which sum up to 90, then we have captured 90% of the total explained variance.

If we want to achieve a high level of accuracy, we may choose a higher value of k. However, this comes at the cost of increased computational complexity and memory requirements. Here at k=500, k explains more than 80% of the total variance.



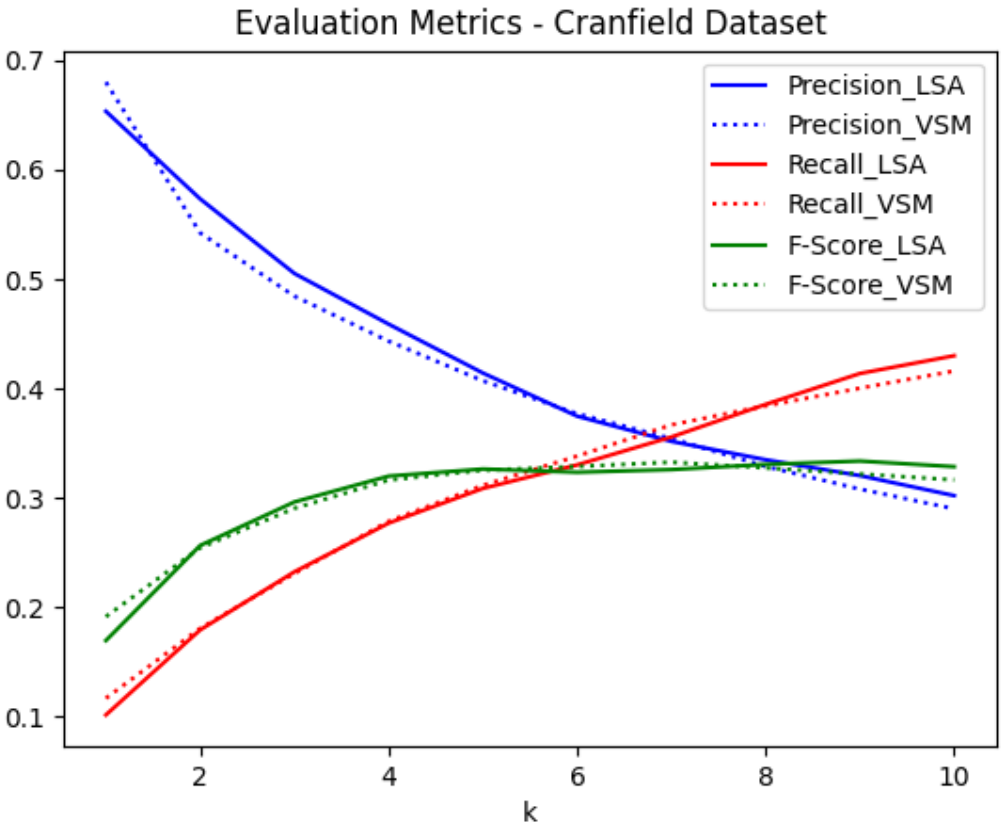
To implement LSA, we first converted the documents into TF-IDF vectors and constructed a matrix with each row representing a document and each column representing a unique term in the corpus. Next, we applied singular value decomposition (SVD) to the TF-IDF matrix to extract the underlying concepts. The resulting document vectors were represented as linear combinations of these concepts, which allows us to capture the semantic relationships between terms in the corpus.

Unlike the traditional TF-IDF vector space model, LSA does not assume that terms are orthogonal to each other. Instead, it leverages the underlying structure in the corpus to capture the relationships between terms, resulting in a more accurate representation of the documents.

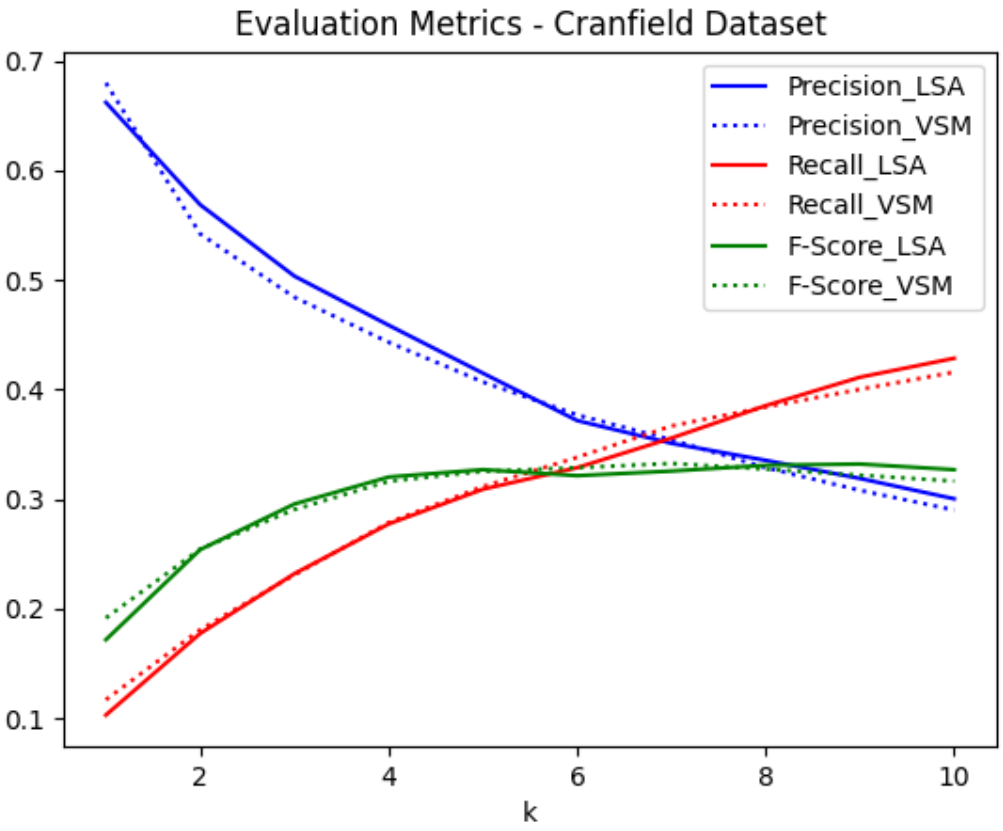
To retrieve relevant documents for a given query, we transformed the query into a vector using the same SVD transformation and computed the cosine similarity between the query vector and the document vectors. This allowed us to rank the documents based on their similarity to the query vector, with the most relevant documents appearing at the top of the list.

Results

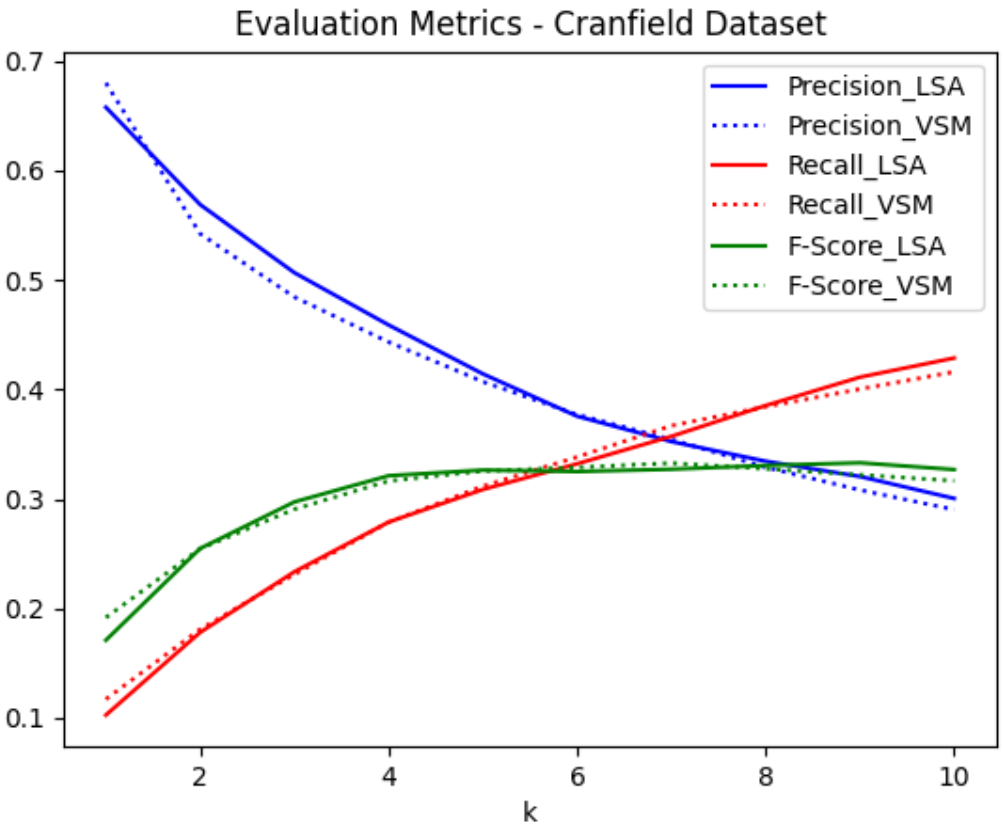
Following are the results of testing our LSA and VSM models on the cranfield dataset, the queries are the ones provided in the qrels dictionary. Below are the graphs of different evaluation metrics (Precision, Recall, F-Score, nDCG etc) against the ranks at which they have been calculated. The total number of concepts are selected to be 500 here, because 500 concepts capture more than 80% of the total variance explained, in the case of LSA.



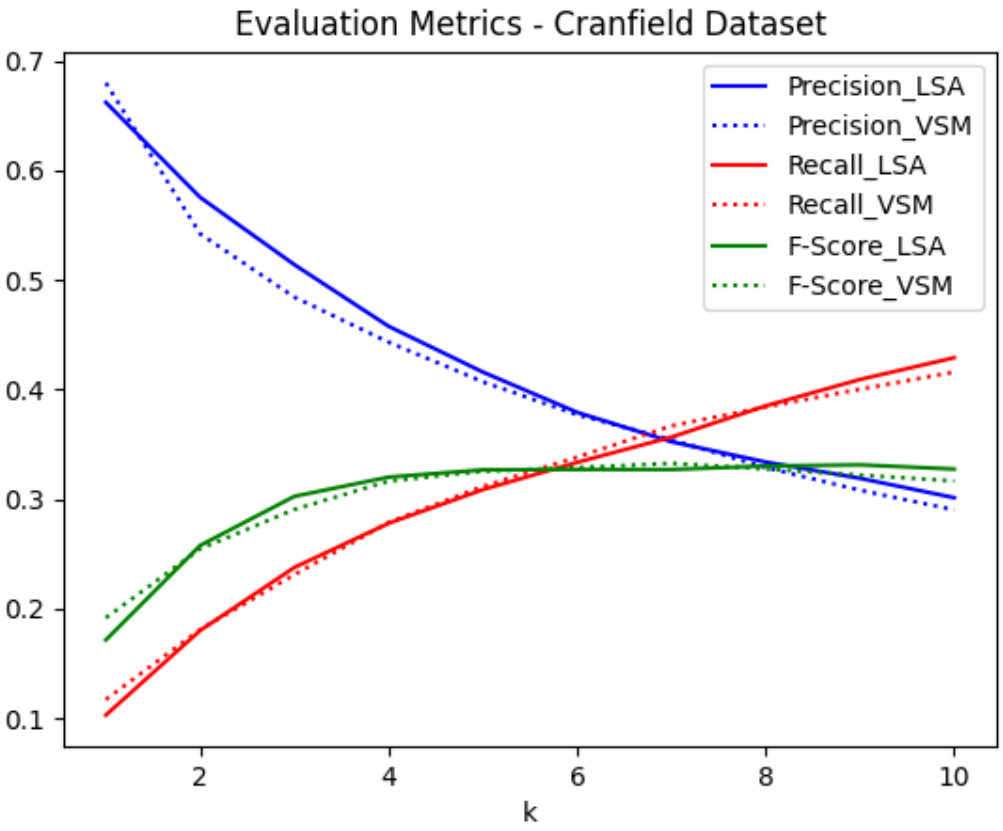
Precision, Recall and F1 score comparison at Rank 450 LSA Model vs VSM Model at retrieval rank ‘k’.



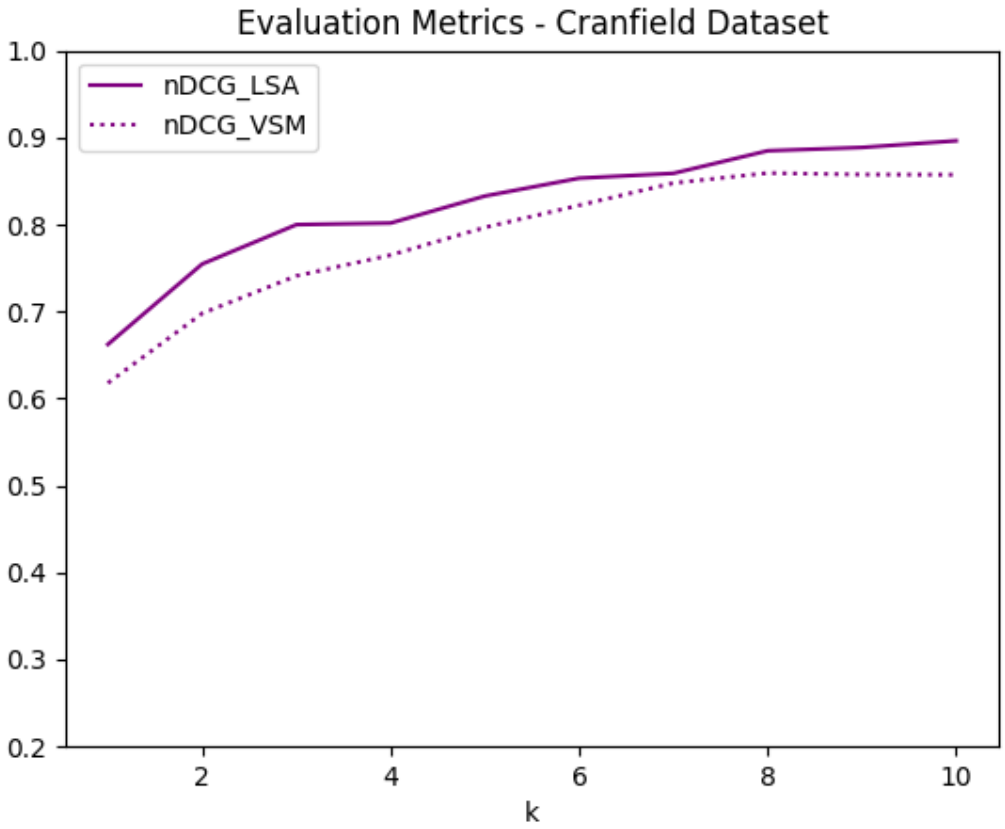
Precision, Recall and F1 score comparison at Rank 500 LSA Model vs VSM Model at retrieval rank ‘k’.



Precision, Recall and F1 score comparison at Rank 550 LSA Model vs VSM Model at retrieval rank ‘k’.


Evaluation Metrics - Cranfield Dataset
Precision_LSA
Precision_VSM
Recall_LSA
Recall_VSM
F-Score_LSA
F-Score_VSM
0.7
0.6
0.5
0.4
0.3
0.2
0.1
2
4
6
8
10
k

Precision, Recall and F1 score comparison at Rank 600 LSA Model vs VSM Model at retrieval rank ‘k’.



nDCG comparison at Rank 500 LSA Model vs VSM Model at retrieval rank ‘k’.

here nDCG values have higher importance because they take into account the level of relevance of a query and the documents. It is clearly visible from the graphs that LSA model gives better performance when compared with the baseline model, in terms of Evaluation metrics such as precision, Recall, F Score and most importantly nDCG. Since relevance for Cranfield queries has levels and is not binary, we employ hypothesis testing based on nDCG values of our proposed model and the baseline model, in the next section.

Hypothesis Testing

A two-sample t-test is a statistical test used to compare the means of two independent samples. It is a parametric test that assumes that the populations from which the samples are drawn are normally distributed and have equal variances.

The null hypothesis for a two-sample t-test is generally that the means of the two variables are equal. The alternative hypothesis is that they are not equal. The test statistic used to

determine the p-value is the t-statistic, which measures the difference between the sample means divided by the standard error of the difference between the means.

The degrees of freedom for the test is calculated by subtracting 1 from the sum of the sample sizes for each group. The p-value for the test is then calculated using a t-distribution table or statistical software.

If the p-value is less than the level of significance, typically set at 0.05, then the null hypothesis is rejected, and it can be concluded that the means of the two variables are statistically different. If the p-value is greater than the level of significance, then the null hypothesis is not rejected, and it can be concluded that there is no significant difference between the means of the two populations.

The two-sample t-test is commonly used in research studies to compare the means of two groups, such as a control group and a treatment group. It can also be used to compare the means of two different populations, such as male and female populations.

Hypothesis testing (based on 500-rank approximation during SVD) on nDCG values of baseline model and proposed new model.

Null hypothesis: There is no significant difference between the mean nDCGs of LSA and VSM.

Alternative hypothesis: The mean nDCG of LSA is greater than the mean nDCG of VSM.

Calculation of the means and standard deviations of the two samples:

Mean nDCG of VSM: 0.7861393288433365

Mean nDCG of LSA: 0.8232457687714791

Standard deviation of nDCG for VSM: 0.08115744504064844

Standard deviation of nDCG for LSA: 0.07247380297424186

t-value: 1.0784303228455874

Degrees of freedom (df): 18

p-value: 0.1475410879667751

So with 80 % confidence we reject the null hypothesis and can claim that LSA performs better than VSM modelling of the IR system, and that the mean nDCG of LSA model of IR system is greater than that of VSM Model (Our baseline Model). We can increase our confidence level as we get more data, that is if we increase the value of ranks at which we calculate the value of nDCG, till, 20,30 and so on, our confidence level with which we reject the null hypothesis will increase.

QUERY AUTOCOMPLETE FEATURE

MOTIVATION:

The motivation behind the query autocomplete feature is to enhance the user experience and simplify the process of finding relevant information quickly. When users are searching for information, they may not always know the exact terms or phrases that would generate the most pertinent results.

The auto-complete feature helps users by predicting the most probable search queries based on their inputs and providing suggestions that can lead them to the most relevant outcomes. Moreover, the auto-complete feature minimizes typing errors and saves time for the user by providing suggestions that align with their intended search query before they complete typing it. This feature is particularly beneficial for users who are browsing on mobile devices or those who have limited typing skills as typing on these devices can be time-consuming and challenging.

Furthermore, the auto-complete feature helps in optimizing the search engine's performance by decreasing the number of irrelevant queries and generating more accurate results. By examining the most common search queries and proposing appropriate keywords and phrases, search engines can comprehend better what users are searching for and enhance their search algorithms accordingly.

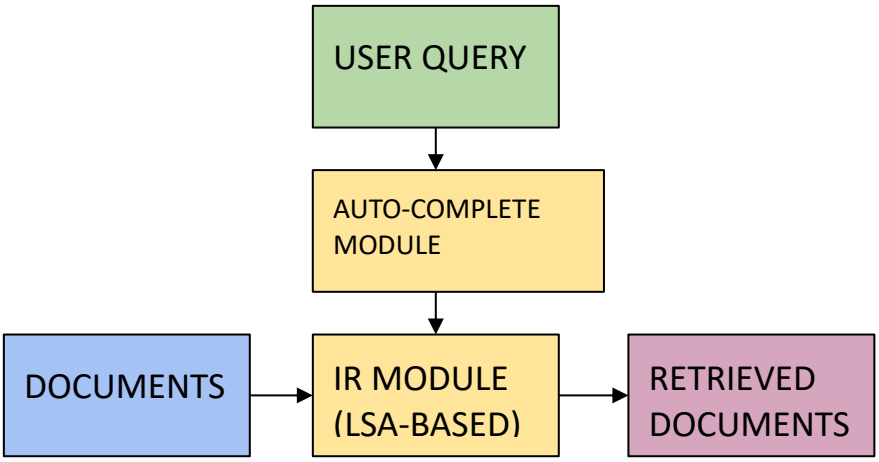
PROBLEM:

The primary problem that the query auto-complete feature aims to solve is the challenge of enabling users to locate the most relevant documents that meet their information needs from

a vast pool of text documents. This problem is particularly difficult because users may utilize different terms or phrases than those found in the documents, and they may also be unsure about what they are searching for.

To overcome this challenge, auto-complete proposes relevant keywords or phrases based on the user's input. These suggestions can help users refine their search queries and find the information they require more quickly and with greater ease.

DESCRIPTION:



Proposed Model Block Diagram

The auto-complete feature of query is an information retrieval technique that anticipates the user's intended search query based on their input and provides suggestions to complete it. In an information retrieval scenario, the user inputs a partial query to the system, and the system retrieves the most appropriate documents from the collection that match the user's information need.

Auto-complete feature commonly employs various methods, such as n-grams, language models, and statistical algorithms, to forecast the most probable words or phrases that the user may enter next based on their partial query. The suggested words or phrases are shown to the user in a list format, and the user can select one of the suggestions or continue typing their query.

CHALLENGES:

The auto-complete feature of query faces several challenges, like -

- *Vocabulary mismatch*: There may be differences between the terms or phrases used in the user query and those found in the documents. Therefore, the system needs to map the user's query to the terms used in the documents and use Laplace smoothing or other techniques to handle out-of-vocabulary words.
- *Contextual suggestion*: Occasionally, the suggested words or phrases provided by the predictive text algorithm may not be related to the context of the partial query. To overcome this issue, the system may use contextual information, such as the user's search history, browsing behavior, or location, to provide more personalized and relevant suggestions.
- *Ranking*: The system must rank the suggested words or phrases according to their relevance to the user's partial query to ensure that the most relevant suggestion is easily identified. One approach is to assign probabilities or scores to each suggested word or phrase, allowing the user to select the most relevant suggestion by selecting the one with the highest score.

OBJECTIVES:

The main goals of the auto-complete feature of query are to provide precise and relevant suggestions that match the user's information needs and to enhance the user experience by minimizing typing errors and saving time. Furthermore, the system should be efficient in retrieving the results and should be capable of handling a vast corpus of text documents.

PROBLEM DEFINITION:

The aim of this code is to develop a language model that utilizes n-grams to predict the probability of a given sequence of words. The model will be trained on the Brown corpus, a dataset comprising written and spoken English samples from various genres.

PROPOSED METHODOLOGY:

The code aims to build a language model using n-grams to predict the probability of a given sequence of words. To achieve this, the following steps will be taken:

1. The Brown corpus will be preprocessed by converting all text to lowercase, removing punctuation, and tokenizing the sentences.

2. The corpus will be split into train, validation, and test sets.

3. A function will be implemented to count the words in the corpus.

4. Another function will be implemented to handle out-of-vocabulary (OOV) words by creating a closed vocabulary of words that appear frequently enough in the training set.

5. An additional function will be implemented to handle unknown tokens by replacing OOV words with a special "<unk>" token.

6. The OOV and unknown token functions will be applied to the training and test sets to further clean the data.

7. A function will be implemented to count n-grams of a given size for a given dataset.

8. A language model will be trained on the training set by counting n-grams of sizes 1 to 4 and using the counts to calculate probabilities of words given their context.

9. The final language model will be tested on the test set, and the results will be reported.

Mathematics of n-gram model used-

1. N-gram model to calculate the Probability of word

Where, P is conditional probability for the word at position 't' in the sentence, given that the words preceding it are Wt-1,…Wt-n**.** And C(...) denotes the number of occurance of the given sequence.

$$P(w_t \mid w_{t-1} \dots w_{t-n}) = \frac{C(w_{t-1} \dots w_{t-n}, w_n)}{C(w_{t-1} \dots w_{t-n})}$$

2. Perplexity calculation

$$PP(W) = \sqrt[N]{\prod_{t=n}^{N-1} \frac{1}{P(w_t | w_{t-n} \ldots w_{t-1})}}$$

Where N = length of the sentence, n = number of words in the n-gram (e.g. 2 for a bigram).

OBSERVATION:

Given a partial sentence by the user, we output the top 4 most relevant words with their probabilities.

- Example1:

```
Partial Sentence: To be or not to

SUGGESTED WORD LIST:
Word: the, Probability: 0.08933742122296032
Word: be, Probability: 0.00289140233013011 33
Word: go, Probability: 0.00017245839441234803
Word: this, Probability: 8.6244070720138e-05
```

- Example2:

```
Partial Sentence: The time has

SUGGESTED WORD LIST:
Word: been, Probability: 0.03685165034457744
Word: come, Probability: 0.00034473842971645267
Word: this, Probability: 8.6244070720138e-05
Word: this, Probability: 8.6244070720138e-05
```

Thus, in example 1, we know the complete sentence can be: “To be or not to be”.

So, its valid candidate words can be: {“the”, “be”, “go”, “this”}; out of which “this” is very likely to occur so it has least probability.

Similarly, in example 2 the candidate words can be: {“been”,”come”, “this”}.

EVALUATION:

Perplexity is a measurement of how well a language model can predict a given sequence of words. Specifically, it is a mathematical measure of how "surprised" or "confused" the language model is when it tries to predict the next word in a sequence.

A lower perplexity score indicates that the language model is more accurate at predicting the next word, while a higher perplexity score indicates that the language model is less accurate.

- Example1

```
Partial Sentence: To be or not to
Sentence: To be or not to the; Perplexity: 666.2967242581356
Sentence: To be or not to be; Perplexity: 500.8579280041194
Sentence: To be or not to go; Perplexity: 667.9986534477612
Sentence: To be or not to this; Perplexity: 645.5418816340233
```

- Example2

```
Partial Sentence: I am happy to see
Sentence: I am happy to see the; Perplexity: 1219.038817608155
Sentence: I am happy to see the; Perplexity: 1219.038817608155
Sentence: I am happy to see me; Perplexity: 973.0059707337538
Sentence: I am happy to see this; Perplexity: 1186.794636602616
```

From above Examples we can observe using manual grammatical rules the relevant word should have least perplexity value, whereas out of the suggested words the irrelevant word has a higher perplexity.

Conclusion

In conclusion, our experimentation with LSA has shown that it outperforms our baseline Vector Space Model (VSM) in the task of Information Retrieval. By leveraging the underlying

semantic relationships between terms, LSA is able to produce more accurate and meaningful document representations compared to the simple frequency-based approach of VSM. Moreover, LSA's ability to handle synonymy and polysemy makes it a more powerful tool for information retrieval.

Through our experiments with the Cranfield dataset, we observed that LSA consistently returned more relevant results for a given query, as compared to the VSM model. This confirms the effectiveness of LSA in capturing the latent structure of the corpus and enabling more sophisticated query processing. Overall, our results demonstrate the superiority of LSA over traditional VSM model and highlight the importance of leveraging advanced techniques for building high-performing Information Retrieval systems.

On implementing an autoquery complete feature , our suggestions would be based on the information we have into our dataset(documents) and hence if the user chooses our suggested query the performance of the IR system(i.e. Precision, Recall, etc) is bound to increase, hence features like auto-query complete are additional ways in which the performance of the IR system can be increased. uery autocomplete feature on our IR system then it can enhance the user experience and simplify the process of finding relevant information quickly.

Also, it can indirectly serve as a spell-check module, thereby increasing the accuracy of the IR system to fetch more relevant documents.

Acknowledgement-

We would like to express our gratitude and thankfulness to Prof. Sutanu Chakraborty for this opportunity to work on this Project. We thank the TA’s for their support for this course. We learnt a lot through this project and hope to come across more knowledge about this field in future.