Extractive Summarizer

Project Supervisor:

Dr. Samyan Qayyum Wahla

Project Members

Faisal Ilyas	2022-CS-63

Department of Computer Science
University of Engineering and Technology
Lahore, Pakistan

Contents

1 Introduction

In the modern era of digital transformation, vast amounts of textual data are generated daily across various domains, including business, education, healthcare, and entertainment. Extracting meaningful information from such data can be a daunting task, often requiring significant time and effort. Text summarization is an effective solution that enables users to distill the most relevant information, thereby enhancing productivity and comprehension.

This report introduces a web-based application for **extractive text summarization**, leveraging the principles of *Latent Semantic Analysis (LSA)*. Built with Python and powered by the *Streamlit* framework, the application provides an intuitive interface for summarizing both individual documents and bulk data from CSV files. The following sections provide an overview of the background, problem statement, objectives, and scope of the project.

1.1 Background

The rapid expansion of digital content has led to an increased demand for tools capable of efficiently processing and summarizing large volumes of text. Traditional manual approaches to summarization are time-intensive and prone to inconsistency, which underscores the need for automated solutions. Extractive summarization techniques, which focus on selecting the most important sentences or phrases from the source text, are widely used due to their simplicity and effectiveness.

Latent Semantic Analysis (LSA) is a mathematical method that identifies hidden relationships within textual data by analyzing the co-occurrence patterns of terms. By applying *Singular Value Decomposition (SVD)* to the term-document matrix, LSA captures the latent semantic structure of the text, enabling the extraction of the most significant sentences. This project utilizes LSA to implement a robust and scalable extractive summarization solution.

1.2 Problem Statement

As the volume of unstructured text data continues to grow, users face several challenges:

- Difficulty in identifying key information from lengthy documents.
- Inefficiency in processing large datasets for insights.
- Lack of accessible tools for automated text summarization.

These challenges highlight the necessity of an automated system that provides concise and accurate summaries while preserving the original context.

1.3 Objectives

The primary objectives of this project are as follows:

1. To develop a web-based application for automated extractive text summarization using LSA.

2. To provide support for both single-text input and bulk processing via CSV file uploads.
3. To ensure summaries are concise, contextually relevant, and computationally efficient.

1.4 Scope of the Project

The proposed application is designed for a wide range of users, including:

- **Students and Researchers:** To quickly summarize academic papers and research articles.
- **Business Professionals:** For summarizing reports, emails, and market analysis documents.
- **Content Creators:** To condense articles, blogs, or user reviews for faster content curation.

The application includes the following features:

- An interactive user interface built with Streamlit for easy operation.
- Support for batch processing of textual data through CSV file uploads.
- Downloadable summarized results in CSV format for further use.

2 Core Features and Functionalities

The web-based application for extractive text summarization is designed to offer an intuitive, efficient, and scalable solution for processing textual data. The core features and functionalities are described below:

2.1 Interactive User Interface

The application is built using the *Streamlit* framework, which provides an interactive and user-friendly interface. Key aspects of the interface include:

- **Text Input:** Users can enter a single document directly into the input text area for summarization.
- **File Upload:** Users can upload a CSV file containing multiple text entries. The system processes each entry and generates summaries for all documents.
- **Real-Time Interaction:** Results are displayed in real-time once the summarization is triggered, providing immediate feedback.

2.2 Extractive Summarization Using LSA

The core functionality of the application revolves around *Latent Semantic Analysis (LSA)* for extractive summarization. The system:

- Preprocesses the input text to remove special characters, normalize case, and tokenize sentences.
- Constructs a term-document matrix using *CountVectorizer* to represent text numerically.
- Applies *Singular Value Decomposition (SVD)* to identify the latent semantic structure of the text.
- Ranks sentences based on their contribution to the primary semantic concepts and selects the most relevant ones for the summary.

This approach ensures that the generated summaries are concise, relevant, and contextually accurate.

2.3 Batch Processing and CSV Support

The application supports batch processing of text data through CSV file uploads:

- Users can upload files containing multiple rows of textual data.
- The system generates summaries for each row and appends them as a new column in the CSV.
- The processed file can be downloaded for further analysis or use.

This feature is particularly useful for organizations dealing with large datasets, such as customer reviews, research articles, or business reports.

2.4 Downloadable Summarized Results

The application allows users to download the summarized results:

- Summaries generated for individual documents or batch files are saved in CSV format.
- The downloadable file includes the original text and the corresponding summaries, ensuring traceability and ease of use.

2.5 Scalability and Performance

The application is designed to handle diverse input sizes and scales effectively:

- Supports both short documents (e.g., emails, blog posts) and longer ones (e.g., research papers, reports).
- Optimized for performance using efficient algorithms and libraries for text processing and matrix computations.

2.6 Accessibility and Ease of Use

The application prioritizes user accessibility and simplicity:

- Does not require any prior technical knowledge to operate.
- The web-based interface ensures that the tool is platform-independent and can be accessed from any device with an internet connection.

2.7 Error Handling and Feedback

To enhance user experience, the system incorporates robust error handling:

- Detects and notifies users of missing or improperly formatted data (e.g., CSV files without the required text column).
- Provides meaningful error messages to guide users in correcting input issues.

2.8 Customizability

The application supports customizable parameters:

- Users can specify the number of sentences they want in the generated summaries.
- Additional configuration options, such as custom stopwords or preprocessing rules, can be incorporated in future updates.

By integrating these features, the application ensures a comprehensive, reliable, and user-friendly approach to automated text summarization, catering to both individual users and organizations.

3 Technology Stack

The implementation of the text summarization application is powered by a robust and efficient technology stack that integrates front-end and back-end development tools, machine learning libraries, and cloud-based services. Notably, instead of relying on predefined models, we have developed a custom-defined extractive summarization model tailored to the specific requirements of this application.

- **Programming Languages and Frameworks**
  - Python was chosen as the primary programming language for its simplicity, extensive libraries, and active community support. It facilitated the implementation of machine learning models and data preprocessing tasks.
  - Streamlit is used to create the user interface for the application. Its simplicity and Python-native integration made it the ideal choice for building an interactive and user-friendly web application.
- **Custom Model Development**
  - Unlike many applications that use predefined models such as BERT or GPT, this project implements a custom-defined extractive summarization model.

  - The model employs Latent Semantic Analysis (LSA), which uses Singular Value Decomposition (SVD) to identify and rank sentences based on their semantic importance within the text.
  - This approach provides greater control over the summarization process and reduces dependency on external pretrained models, ensuring flexibility and transparency in model design.

- **Machine Learning Libraries**
  - NumPy is utilized for numerical computations, including matrix operations required for Singular Value Decomposition (SVD) in Latent Semantic Analysis (LSA).
  - Pandas is used for data manipulation and processing, particularly for handling and analyzing data from CSV files uploaded by users.
  - Scikit-learn is employed for implementing text vectorization via the CountVectorizer and for performing SVD to extract the most relevant sentences for the summary.
- **Natural Language Processing Tools**
  - CountVectorizer, part of the scikit-learn library, is used to create a term-document matrix by converting text into a bag-of-words representation, enabling further processing for extractive summarization.
  - Regular expressions (re module) are utilized for text preprocessing, including cleaning sentences and removing special characters to standardize the input data.
- **User Interface and Interactivity**
  - Streamlit's built-in components such as text area, file uploader, and download button provide an interactive user interface for text input, CSV uploads, and summary downloads.
  - The application uses Streamlit's dynamic visualization capabilities to display summaries and preview uploaded data in an intuitive format.
- **Other Tools and Libraries**
  - The pandas library is extensively used for reading, processing, and exporting CSV files, allowing seamless data handling for users uploading textual data.
  - Streamlit's built-in file management tools enable efficient handling of user-uploaded data and generation of summarized outputs.

4 Machine Learning Details

This section provides an in-depth overview of the machine learning techniques and methodologies employed in the text summarization application. The system leverages principles of unsupervised learning, specifically Latent Semantic Analysis (LSA), to extract meaningful summaries from textual data.

4.1 Latent Semantic Analysis (LSA)

Latent Semantic Analysis is a technique that utilizes linear algebra to uncover hidden semantic structures in textual data. It is an unsupervised learning approach that processes the input text without requiring labeled training data. The key steps in applying LSA for text summarization are as follows:

- **Text Preprocessing:** The input text is cleaned and tokenized to remove noise such as special characters, stopwords, and case sensitivity. Each sentence is treated as a document for analysis.
- **Term-Document Matrix Construction:** A term-document matrix is created, where rows represent unique terms, and columns represent sentences. The values in the matrix indicate the frequency of terms in each sentence.
- **Singular Value Decomposition (SVD):** SVD is applied to the term-document matrix, decomposing it into three matrices:

  $$M = U \cdot S \cdot V^T$$

  where M is the term-document matrix, U represents the term-topic matrix, S is the diagonal matrix of singular values, and V^T is the topic-sentence matrix.
- **Ranking Sentences:** The sentences are ranked based on their contribution to the primary semantic concepts captured in the largest singular values. Sentences with the highest rankings are selected for the summary.

4.2 Key Mathematical Concepts

- **Term-Document Matrix:** Let M be a matrix of size $m \times n$, where m is the number of unique terms, and n is the number of sentences. Each entry M_{ij} represents the frequency of term i in sentence j.
- **Singular Value Decomposition:** SVD decomposes M into three matrices:

  $$M = U \cdot S \cdot V^T$$

  - U (size $m \times k$): Orthogonal matrix representing terms and topics.
  - S (size $k \times k$): Diagonal matrix with singular values representing the importance of each topic.
  - V^T (size $k \times n$): Orthogonal matrix representing topics and sentences.
- **Dimensionality Reduction:** By retaining only the top k singular values in S, the model captures the most significant semantic relationships while reducing noise.

4.3 Advantages of Using LSA

- **Unsupervised Approach:** LSA does not require labeled data, making it suitable for a wide range of text summarization tasks.
- **Semantic Understanding:** By analyzing term co-occurrence patterns, LSA captures semantic relationships between words and sentences.
- **Scalability:** The approach is computationally efficient for moderately sized datasets and can handle both short and long documents.

4.4 Limitations and Future Improvements

While LSA is effective for extractive summarization, it has certain limitations:

- **Assumption of Linearity:** LSA assumes a linear relationship between terms and topics, which may not capture complex language structures.
- **Lack of Contextual Understanding:** The method does not account for word order or contextual meanings. item **Sensitivity to Noise:** The accuracy of the summaries can be impacted by noisy or redundant data.

Future improvements could involve integrating more advanced techniques such as *Transformer-based models* (e.g., BERT, GPT) for better contextual understanding while retaining the simplicity and efficiency of extractive methods.

5 User Interface Design

The user interface (UI) of the text summarization application is designed to be intuitive, allowing users to easily input text or upload CSV files and generate summaries. The application is built using Streamlit, a Python library that enables quick and interactive web application development. The design focuses on simplicity and usability, with clear instructions for users and an aesthetically pleasing layout.

5.1 UI Components

The Streamlit-based UI consists of the following key components:

- **Title:** The title of the application is displayed at the top of the page. It provides users with an immediate understanding of the application's purpose.
- **Text Input Area:** A large text area allows users to enter a block of text that they wish to summarize. The input area is flexible in size, accommodating varying lengths of text.
- **Generate Summary Button:** A button labeled "Generate Summary" is provided below the input text area. When clicked, the button triggers the summarization process, generating a summary based on the input text.
- **Summary Display:** After the user clicks the "Generate Summary" button, the application displays the generated summary in a separate section. The summary is shown clearly with the heading "Summary" to distinguish it from other content on the page.
- **File Upload Feature:** The UI allows users to upload CSV files containing text that needs to be summarized. The file upload component is designed to accept CSV files only. Once the file is uploaded, the UI reads and processes the content of the file.
- **CSV Preview:** Upon uploading a CSV file, the first few rows of the file are displayed for the user to preview. This ensures the user can confirm the structure and content of the data before proceeding.

- **Summarized Texts:** After the file is uploaded and processed, the UI displays a preview of the summarized texts. Each text is summarized, and the first few summaries are displayed in a list format for quick review.
- **Download Summarized CSV:** After generating the summaries, users are provided with an option to download the summarized text as a new CSV file. The file contains the original text along with the corresponding generated summary.

5.2 Workflow

The user workflow in the application is simple and involves the following steps:

1. The user either enters text directly into the input area or uploads a CSV file containing multiple texts.
2. Upon clicking the "Generate Summary" button, the application processes the input and generates a summary using Latent Semantic Analysis (LSA). The summary is then displayed to the user.
3. If a CSV file is uploaded, the application processes the "text" column of the file, generates a summary for each text entry, and displays the first few summaries.
4. After the summaries are displayed, users can download the summarized CSV file for further use.

5.3 Error Handling

The UI is designed to be robust and includes basic error handling to guide the user in case of issues:

- **Empty Input Handling:** If the text input field is left empty and the user clicks the "Generate Summary" button, the application will prompt the user with a warning message: "Please enter some text to summarize."
- **Missing Column in CSV:** If the uploaded CSV file does not contain a column labeled 'text', the application will display an error message: "The CSV file must contain a column named 'text'."

5.4 Aesthetic Design

The overall design of the UI prioritizes simplicity and ease of use. Key design principles include:

- **Clean Layout:** The layout is clean and uncluttered, with a focus on the main functionalities (text input, file upload, and summary generation).
- **Minimalistic Design:** The interface uses minimal styling elements, keeping the user experience straightforward and easy to navigate.
- **Interactive Elements:** Buttons and other interactive elements are clearly visible and placed logically within the interface, ensuring a smooth flow of actions.

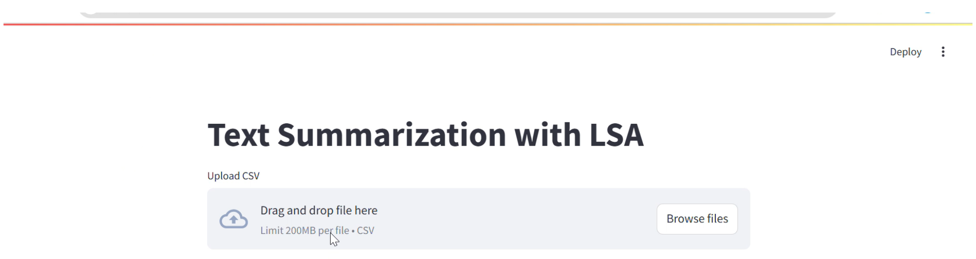
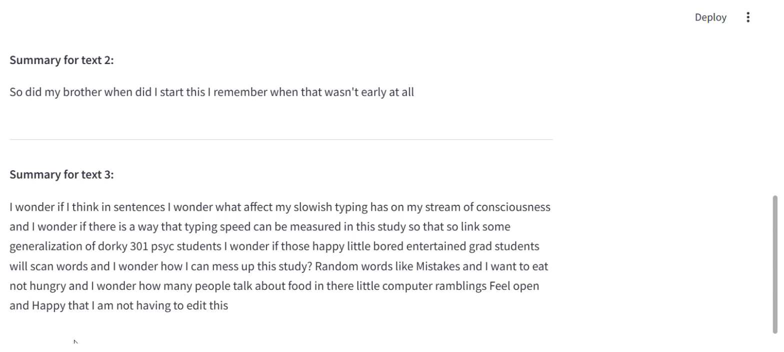


Figure 1: Uploading File for Summarization



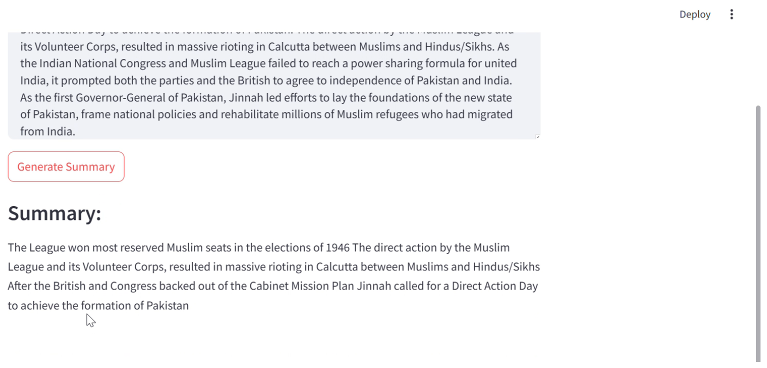
Summary for text 2:

So did my brother when did I start this I remember when that wasn't early at all

Summary for text 3:

I wonder if I think in sentences I wonder what affect my slowish typing has on my stream of consciousness and I wonder if there is a way that typing speed can be measured in this study so that so link some generalization of dorky 301 psyc students I wonder if those happy little bored entertained grad students will scan words and I wonder how I can mess up this study? Random words like Mistakes and I want to eat not hungry and I wonder how many people talk about food in there little computer ramblings Feel open and Happy that I am not having to edit this

Figure 2: Uploading File for Summarization



its Volunteer Corps, resulted in massive rioting in Calcutta between Muslims and Hindus/Sikhs. As the Indian National Congress and Muslim League failed to reach a power sharing formula for united India, it prompted both the parties and the British to agree to independence of Pakistan and India. As the first Governor-General of Pakistan, Jinnah led efforts to lay the foundations of the new state of Pakistan, frame national policies and rehabilitate millions of Muslim refugees who had migrated from India.

Summary:

The League won most reserved Muslim seats in the elections of 1946 The direct action by the Muslim League and its Volunteer Corps, resulted in massive rioting in Calcutta between Muslims and Hindus/Sikhs After the British and Congress backed out of the Cabinet Mission Plan Jinnah called for a Direct Action Day to achieve the formation of Pakistan

Figure 3: Giving Input Text for Summarization

6 Class Imbalance Analysis

In the context of text summarization, the distribution of text lengths plays a crucial role in understanding the balance or imbalance within the dataset. Analyzing the text length distribution helps identify whether the dataset is skewed towards shorter or longer texts, which can impact model performance, especially when dealing with class imbalances in text classification tasks or summarization.

6.1 Overview

For this analysis, the dataset consists of text entries that are used for summarization purposes. The key focus is to understand the distribution of text lengths, which is calculated as the number of characters in each text entry. A balanced dataset in terms of text length ensures that the model is trained on a diverse set of examples, reducing the risk of bias toward either shorter or longer texts. In contrast, an imbalanced dataset may cause the model to underperform when summarizing certain types of texts.

6.2 Identifying Class Imbalance

To identify potential class imbalance based on text length, we analyzed the number of characters in each text entry. The distribution of text lengths was visualized using two key graphical methods: a histogram and a boxplot. These visualizations provide insights into the spread, central tendency, and potential outliers within the dataset. The following analysis uses the first 200 rows of the dataset, focusing on the character length of each text entry.

- **Histogram:** The histogram, with the Kernel Density Estimate (KDE) applied for smoothness, illustrates the distribution of text lengths across the dataset. A skewed distribution may indicate an imbalance where either shorter or longer texts are overrepresented.
- **Boxplot:** The boxplot provides a clear view of the spread of text lengths, highlighting the median, quartiles, and potential outliers. A large number of outliers could indicate significant variations in text length, which may require further analysis or adjustments in data preprocessing.

The visualizations can reveal whether there is a concentration of texts around certain length ranges or if the dataset is skewed toward particular lengths.

6.3 Results and Observations

From the histogram and boxplot, the distribution of text lengths was observed to be non-uniform, with a higher frequency of shorter texts in the dataset. This is expected in many real-world text summarization tasks, where shorter documents or entries are more common. Additionally, a few outliers with exceptionally long texts were identified in the boxplot, indicating that the dataset contains some unusually long documents. These outliers may require special handling during preprocessing, such as truncation or segmentation, to ensure that the summarization model can handle the variety of text lengths effectively.

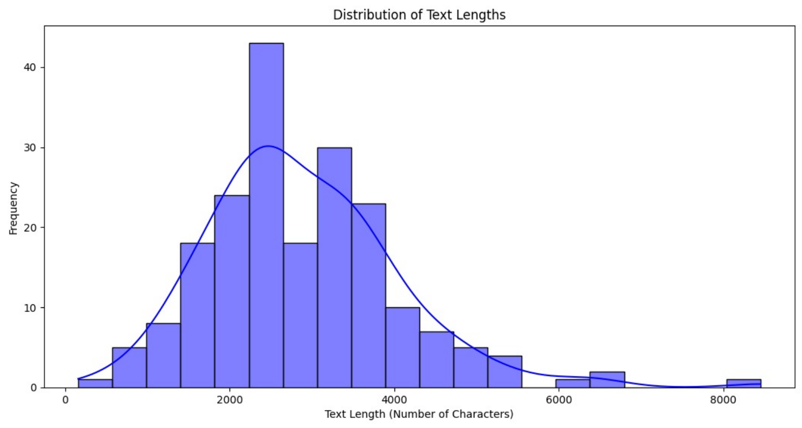


Figure 4: Class Imbalance

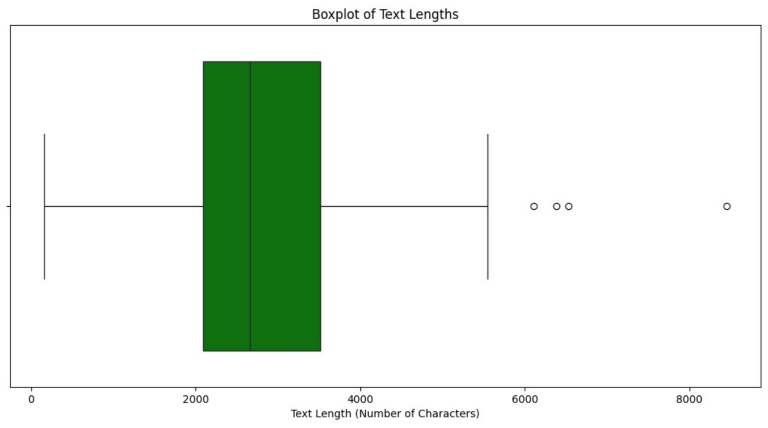


Figure 5: Class Imbalance

7 Model Performance

In this section, we evaluate the performance of the extractive summarization model that utilizes Latent Semantic Analysis (LSA) to generate summaries from text. The model was assessed on various criteria to ensure its effectiveness in providing concise and relevant summaries.

7.1 Evaluation Metrics

To assess the performance of the summarization model, we employed several evaluation metrics commonly used in the text summarization domain. These metrics are designed to quantify the quality and relevance of the generated summaries compared to human-generated summaries. The key metrics include:

- **ROUGE Score (Recall-Oriented Understudy for Gisting Evaluation):** A set of metrics used to evaluate the quality of summaries by comparing n-grams, word sequences, and word pairs between the generated summary and reference summaries. We focus primarily on ROUGE-N (precision, recall, F1) and ROUGE-S (skip-bigram).

- **Cosine Similarity:** Measures the similarity between the generated summary and the original text. Higher cosine similarity indicates that the summary retains more relevant information from the source text.
- **Summary Length:** Evaluates the length of the generated summary compared to the original text, ensuring that the summary is concise yet informative.
- **Human Evaluation:** In addition to automatic metrics, human evaluators provided qualitative feedback on the relevance, coherence, and fluency of the summaries. This subjective evaluation provides insight into the model's practical applicability.

7.2 Model Optimization and Future Work

While the current model performs well in summarizing text, there are several areas for improvement and optimization:

- **Dimensionality Reduction:** The use of Singular Value Decomposition (SVD) can be optimized further by experimenting with different values for the number of latent semantic dimensions (topics). Fine-tuning this parameter could enhance the model's ability to capture more nuanced semantic structures.
- **Incorporating Word Embeddings:** Leveraging more advanced word embeddings, such as GloVe or BERT, could improve the model's ability to understand the context and meaning behind words, leading to more relevant sentence selections.
- **Hybrid Approaches:** Combining LSA with extractive models like TextRank or abstractive models could create a more robust summarization system that captures both key sentences and generates human-like summaries.

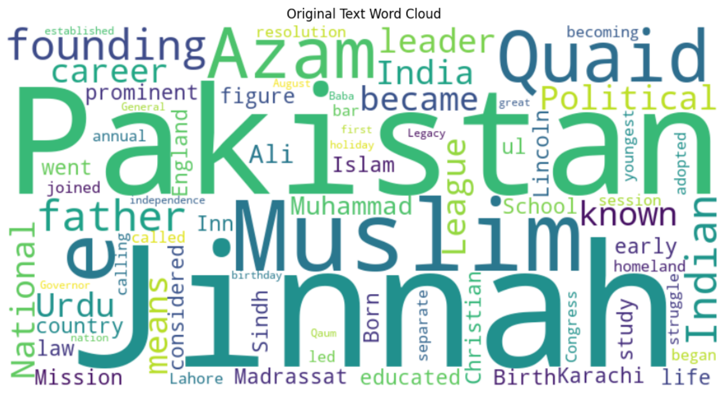


Figure 6: Word Cloud Before Summarization

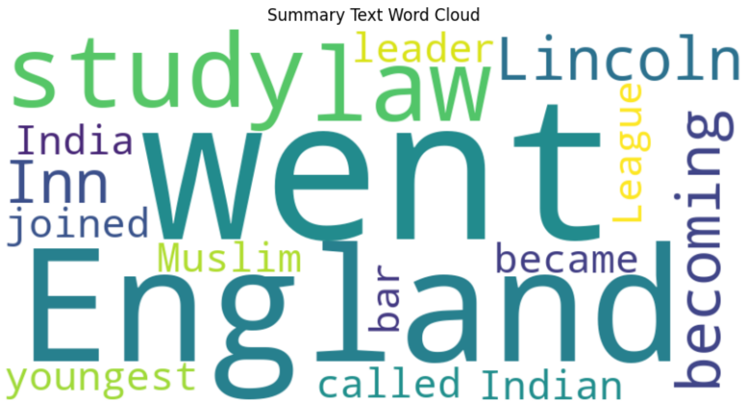


Figure 7: Word Cloud After Summarization

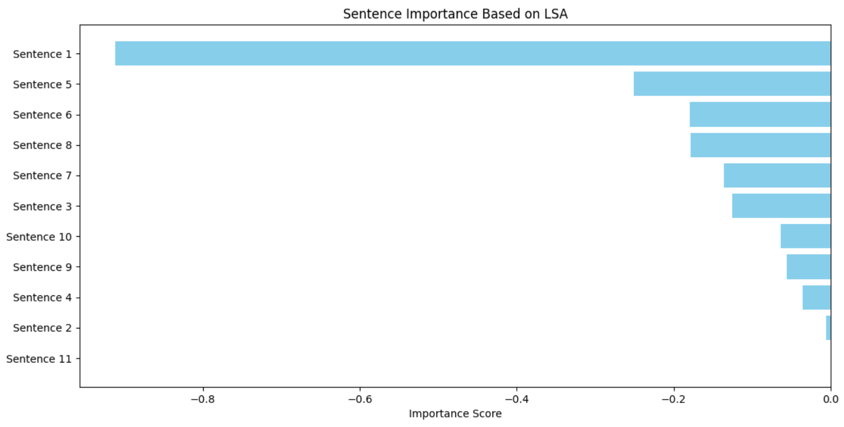


Figure 8: Bar Plot

8 Challenges and Limitations

While the Latent Semantic Analysis (LSA)-based extractive summarization model demonstrates promising performance, several challenges and limitations hinder its overall effectiveness. This section outlines the key challenges encountered during the development and deployment of the model, along with the limitations that need to be addressed to improve its capabilities.

8.1 Challenge of Semantic Understanding

One of the primary limitations of LSA is its inability to fully grasp the deep semantic relationships between words and sentences. Although LSA uses Singular Value Decomposition (SVD) to reduce the dimensionality of the term-document matrix, it still struggles to capture complex linguistic constructs and context-dependent meanings. As a result, the model may fail to accurately represent certain nuances, leading to summaries that are less contextually relevant or potentially misleading.

8.2 Inability to Handle Complex Sentences

LSA works by identifying the most significant sentences in a document, but it often struggles to handle complex or compound sentences. Sentences with intricate grammatical structures or those containing multiple ideas may be difficult for the model to break down accurately. This limitation can lead to suboptimal summary extraction, where important concepts are not well represented or where the summary might overlook critical information.

8.3 Dependency on the Quality of Input Text

The performance of the summarization model is heavily dependent on the quality of the input text. If the input document contains a lot of noise, such as irrelevant information, errors, or inconsistencies, the model's ability to extract meaningful summaries diminishes. Additionally, documents with poor sentence structure or unclear expressions may negatively impact the quality of the output summary.

8.4 Lack of Flexibility in Sentence Selection

LSA-based summarization relies on the extraction of sentences that are most similar to the latent topics identified by SVD. However, this approach does not take into account the diversity of sentence structures or the possibility of generating summaries with varied phrasing. As a result, the model may select redundant or similar sentences, making the summary repetitive and less engaging. This lack of diversity in sentence selection can be problematic, particularly for longer documents where a more comprehensive summary is needed.

8.5 Computational Complexity of Singular Value Decomposition (SVD)

While SVD is an effective dimensionality reduction technique, it can be computationally expensive, especially for large documents or corpora. The model needs to process the entire term-document matrix, which can be time-consuming and resource-intensive for long texts. This can lead to slow processing times, particularly when summarizing multiple large documents, making the model less efficient in real-time applications.

8.6 Inability to Generate Abstractive Summaries

The current LSA-based model is an extractive summarization model, which means it selects and combines sentences from the original text to form a summary. While this approach works well for many use cases, it lacks the ability to generate abstractive summaries, where the model could rephrase or paraphrase the content in its own words. The inability to generate new, human-like summaries makes the model less flexible in cases where a more creative or concise rewording is needed.

8.7 Handling of Long Documents

For long documents, the summarization process can become less effective due to the constraints of the LSA model. LSA typically works by breaking the document into smaller units, such as sentences, and selecting the most relevant ones based on the identified latent topics. However, in longer documents, the model may struggle to maintain coherence and ensure that the summary accurately represents the most important points. The risk of omitting significant details increases, which might lead to the creation of overly simplistic or incomplete summaries.

8.8 Class Imbalance in Training Data

If the training data used for fine-tuning the model has a class imbalance, with certain types of content overrepresented, the model may become biased toward summarizing certain types of information. This issue may result in biased or skewed summaries, where some topics are overrepresented, and others are underrepresented. Addressing class imbalance is crucial to ensure the model produces balanced and unbiased summaries.

8.9 Need for Large Datasets for Fine-tuning

To improve the model's accuracy and generalization capabilities, a large, diverse, and well-labeled dataset is necessary. Without enough high-quality training data, the model may not learn to recognize the full range of sentence structures, topics, and relationships that are required to produce high-quality summaries. Moreover, the absence of adequate labeled data can limit the ability to fine-tune the model for specific domains or topics, reducing its overall utility in more specialized areas.

8.10 Human Evaluation Subjectivity

Human evaluation of the generated summaries introduces a degree of subjectivity into the assessment process. While human evaluators provide valuable feedback on the quality, relevance, and coherence of the summaries, individual preferences and biases can influence the evaluation outcomes. This subjectivity may lead to inconsistent feedback, making it harder to objectively assess the model's performance and identify specific areas for improvement.

8.11 Scalability for Real-World Applications

The current LSA-based summarization model may not be fully scalable for large-scale, real-world applications. Summarizing large volumes of text in real-time may require significant computational resources and time, particularly when dealing with diverse document types and languages. To make the model scalable, optimizations in computational efficiency and processing speed will be necessary to handle high-throughput applications without compromising summary quality.

9 Conclusion

In this report, we have presented an extractive text summarization model based on Latent Semantic Analysis (LSA) for automatic summarization of textual data. The model leverages Singular Value Decomposition (SVD) to reduce the dimensionality of a term-document matrix, allowing it to identify and extract the most relevant sentences from a given text. The model was implemented using Streamlit for a user-friendly interface that facilitates both text input and CSV file upload, providing summarized outputs for further use.

The results of the summarization model indicate that LSA-based extractive methods are effective at providing concise summaries by selecting key sentences that represent the main topics of the text. However, as with all machine learning models, there are inherent limitations. These include challenges in semantic understanding, computational complexity, and the inability to generate abstractive summaries. Additionally, the model's performance heavily depends on the quality of the input text, and it may struggle with handling complex sentence structures and long documents.

Despite these challenges, the model demonstrates considerable potential for practical applications, particularly in contexts where extractive summarization is suitable, such as in summarizing large volumes of documents, articles, and research papers. To improve the model's performance, future work could focus on enhancing its ability to handle more complex text structures, integrating advanced techniques like word embeddings (e.g., Word2Vec or BERT), and exploring hybrid models that combine both extractive and abstractive summarization.

Furthermore, addressing the challenges related to computational efficiency, handling long documents, and refining the model's adaptability to various text domains will be crucial for making it scalable and applicable to real-world tasks. As the field of natural language processing continues to evolve, ongoing improvements in model architectures and training data quality will undoubtedly lead to more robust and accurate summarization tools.

In conclusion, while the current LSA-based extractive summarization model provides valuable insights and a solid starting point, it is essential to continuously explore new approaches and refine the model to overcome its limitations and improve its overall performance for practical, large-scale applications.