A Project report on

Text Summarization For News Articles

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

Bachelor of Technology

in

Information Technology

Submitted by

R. SRUJANA
(21H51A1224)

B. VISHWAS
(21H51A1235)

L. ARYAN REDDY
(21H51A1262)

Under the esteemed guidance of

Dr. A. Shiva Kumar
(Associate Professor)

Department of Information Technology

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC Autonomous)

*Approved by AICTE *Affiliated to JNTUH *NAAC Accredited with A^{+} Grade

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

2024- 2025

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the Major Project report entitled **"Text Summarization for News Articles"** being submitted by R. Srujana (21H51A1224), B. Vishwas(21H51A1235), L. Aryan Reddy (21H51A1262) in partial fulfillment for the award of **Bachelor of Technology in Information Technology** is a record of bonafide work carried out by them under my guidance and supervision.

The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

GUIDE

Dr. A. Shiva Kumar
Associate Professor

HOD

Dr. K. Venkateswara Rao
Associate Professor and HOD

External Examiner: _____

ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express our heartfelt gratitude to all the people who helped in making this project work a grand success.

We are grateful to **Dr. A. Shiva Kumar,** Associate Professor, Department of Information Technology for his valuable technical suggestions and guidance during the execution of this project work.

We would like to thank **Dr. K. Venkateswara Rao,** Head of the Department of Information Technology, CMR College of Engineering and Technology, who is the major driving forces to complete my project work successfully.

We are highly indebted to **Major Dr. V A Narayana,** Director, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We are highly greatful to **Dr. A. Seshu Kumar,** Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We are very thankful to **Dr. P. Senthil,** Assistant Professor and **Mrs. K. Archana,** Assistant Professor , Major Project Co-ordinators IT Department, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We express our sincere thanks to **Shri. Ch. Gopal Reddy**, Secretary, CMR Group of Institutions, for his continuous care.

We are deeply indebted to **Shri. Ch. Abhinav Reddy**, CEO, CMR Group of Institutions for continuous support.

We would like to thank the **Teaching & Non- teaching** staff of Department of Information Technology for their co-operation

Finally, We extend thanks to our **Parents** who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project work.

R. Srujana 21H51A1224

B. Vishwas 21H51A1235

L. Aryan Reddy 21H51A1262

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	LIST OF FIGURES	ii
	LIST OF TABLES	iii
	ABSTRACT	iv
1	**INTRODUCTION**	1
	1.1 Problem Statement	4
	1.2 Research Objective	5
	1.3 Project Scope and Limitations	5
2	**BACKGROUND WORK**	7
	Literature Survey	8
	2.1. Pointer Generator Network for Summarization	10
	2.1.1.Introduction	10
	2.1.2.Merits,Demerits and Challenges	10
	2.1.3.Implementation	11
	2.2. Recurrent Neural Network Summarizer	12
	2.2.1.Introduction	12
	2.2.2.Merits,Demerits and Challenges	13
	2.2.3.Implementation of RNN Summarizer	14
	2.3. Summarize Bot	15
	2.3.1.Introduction	15
	2.3.2.Merits,Demerits and Challenges	15
	2.3.3.Implementation of Summarize Bot	16
3	**PROPOSED SYSTEM**	17
	3.1 Objective of Proposed Model	18
	3.2 Algorithms Used for Proposed Model	19
	3.3 Designing	20
	3.4 Stepwise Implementation and Code	22
4	**RESULTS AND DISCUSSION**	30
5	**CONCLUSION**	36
	5.1 Conclusion	37
	5.2 Future Enhancement	37

REFERENCES 40

Github Link 41

Publications 44

List of Figures

FIGURE NO.	TITLE	PAGE NO
2.1.3	Architecture of Pointer Generator	12
2.2.3	RNN Summarizer Flow Chart	14
3.3.1	System Architecture	21
4.1	Output	32
4.2	Time taken by the transformers across 5 epochs	34
4.3	Rouge scores GPT,BART,T5 and Pegasus	34

List of Tables

FIGURE NO.	TITLE	PAGE NO.
4.1	Rouge Scores of the transformers	33

ABSTRACT

In today's age of information overload, the sheer volume of news articles available can overwhelm readers, making it difficult to stay updated without dedicating significant time to reading. This project focuses on developing an automatic text summarization system designed specifically for news articles, using natural language processing (NLP) techniques. The system is capable of handling large volumes of content and distilling them into concise, coherent summaries. The summarization process involves several key steps, including feature extraction, where the system identifies important elements like key phrases and themes, and text preprocessing, which involves cleaning the input to improve analysis accuracy. Finally, it generates summaries using either extractive methods, which select key sentences, or abstractive methods, which create new sentences that reflect the article's core content.

The ultimate goal of this project is to improve the accessibility of news information by providing quick and accurate summaries without losing the context or meaning of the original article. Users can interact with a simple, user-friendly interface where they can input news articles or URLs and receive real-time summaries. This system has the potential to be used in various fields, such as journalism, where it can help journalists quickly sift through vast amounts of information, and content aggregation, where it can provide personalized summaries based on user preferences. By streamlining how news is consumed, this summarization system could significantly enhance how readers engage with news in the digital era, making it more efficient and personalized.

CHAPTER 1
INTRODUCTION

CHAPTER 1
INTRODUCTION

The growth of the World Wide Web has made vast amounts of information available at our fingertips, reaching virtually every corner of the world. This accessibility has led to an exponential increase in online information, creating both opportunities and challenges for users. The surge of content, combined with people's busy schedules, has intensified the need for efficient ways to process and comprehend vast amounts of information in a shorter time frame. Text summarization has emerged as a vital solution, offering users condensed versions of text that retain only the essential points, enabling a quicker grasp of content. Automatic text summarization aims to shrink large documents or reports into brief versions that still preserve critical insights and key information, thus providing readers with a more efficient means of information processing. This task of summarization has become a significant area of interest within the field of Natural Language Processing (NLP), with the community working toward effective summarization techniques over the past several decades. According to a well-cited definition by Radev et al. (2002), a summary is defined as "text that is produced from one or more texts, that conveys important information in the original text, and that is no longer than half of the original text(s) and usually significantly less than that." This definition highlights three core aspects of summarization: first, summaries may be produced from a single source or multiple sources; second, summaries must convey the critical information; and third, they should be significantly shorter than the original material. These principles guide the efforts in creating automatic summarization systems that can efficiently produce compact and informative summaries, addressing the increasing need for accessible information in the digital era.

Automatic summarization plays a crucial role in reducing the time users spend searching for essential details within a document. Human summarization, while often of high quality, is a time-consuming process that involves a reader first understanding the full content, identifying key points, and then creating sentences to communicate the gist of the text effectively. This process can be incredibly efficient for individual articles but becomes less practical when faced with hundreds or

thousands of documents. In contrast, automatic summarizers use algorithms to quickly analyze the content, identify relevant sentences, and generate a coherent summary without the extensive time investment that manual summarization requires. One of the primary tasks within extractive summarization, a commonly used summarization technique, is selecting the key sentences that will best represent the content in a summary. Determining which sentences are most important is a complex challenge, as it requires the system to understand the context, relevance, and the relationships between different pieces of information. The applications for automatic summarization are broad, spanning fields like news media, where readers need quick summaries of daily articles, to academia, where researchers benefit from concise overviews of complex research papers, and search engines, where summaries can help users more easily parse search results. Each of these areas benefits from the time-saving capabilities of automatic summarization tools, which provide accessible insights into large quantities of information.

In today's digital age, where readers often face the daunting task of keeping up with the flood of news articles published across various platforms, automatic summarization offers a solution. News articles are published around the clock, and readers often struggle to stay informed due to the overwhelming volume. This project specifically addresses the need for efficient news summarization by developing a tool designed to condense extensive news content into shorter, coherent summaries. The tool allows readers to quickly identify the main points of an article without wading through unnecessary details. By focusing on the essential information within each piece, it enables users to stay informed without committing excessive time to reading. This approach to summarization has applications far beyond everyday readers, with notable implications for industries like journalism, where reporters and editors often need to review large volumes of content to stay on top of events, and in content curation platforms, where personalization is key to delivering relevant information to users.

Our project leverages NLP techniques to generate summaries that retain the context and intent of the original articles, ensuring that the final output remains accurate and trustworthy. This involves multiple steps, including text preprocessing, feature extraction, and summarization, each designed to ensure that key themes and phrases are retained while irrelevant or repetitive information is

minimized. The system employ extractive methods, which identify and select key sentences from the article itself. Abstractive methods, which generate entirely new sentences based on the article's core ideas, enhancing readability and coherence. This approach gives the system flexibility in adapting to various types of content, creating summaries that are both informative and engaging for readers. The potential applications of this system extend beyond casual reading and into professional fields like journalism, where it can assist journalists in quickly reviewing large amounts of information, and in content aggregation, where it can be used to generate customized summaries based on user preferences. This project thus holds the potential to transform the way news is consumed, enabling readers to stay updated more efficiently in a time where information is more abundant than ever. By making news summaries readily available, this tool encourages a more efficient and personalized way of interacting with content, ultimately improving the accessibility and relevance of information in the digital era.

1.1. Problem Statement

The large volume of news articles published every day makes it challenging for readers to keep up and stay informed. Manually going through all the news is time-consuming and impractical. This project aims to create an automatic tool that can summarize news articles quickly, producing clear and easy-to-understand summaries. This will allow readers to get the most important information without needing to read the full article. By using natural language processing techniques, the tool ensures that the original meaning and purpose of the news articles are kept intact. This system will help in improving news consumption, saving time for users, and can be applied in areas like journalism, content organization, and personalized news feeds. Ultimately, it aims to make accessing news easier and more efficient in today’s digital world.

1.2. Research Objective

The aim of the project is to develop a robust algorithm using natural language processing (NLP) techniques to accurately extract and summarize crucial data from news stories while preserving the original meaning and context. To achieve this, the project intends to create a user-friendly interface that allows users to easily input news stories and obtain concise summaries, facilitating rapid information consumption. The model focuses on producing coherent, concise, and informative summaries that reflect the main points of the articles. Additionally, the project aims to integrate advanced sentiment analysis to provide an emotional context for the summaries, enhancing the user's understanding of the news tone. The system incorporates a text-to-speech feature, making the summaries accessible to visually impaired users or those who prefer audio formats, thereby broadening the tool's usability. Another key objective includes implementing a quantitative evaluation mechanism, such as ROUGE scores, to assess the quality and accuracy of the generated summaries against the original articles. The algorithm optimizes for efficient processing of Times of India news articles, ensuring scalability and performance for varying article lengths. Furthermore, the project develops a metadata extraction system tailored to Times of India articles to provide users with additional context, such as author and publication date, enhancing the credibility and relevance of the summaries. Finally, the project explores the potential for future enhancements, such as supporting multiple news sources and incorporating user customization options, to evolve the tool into a versatile and adaptable solution for news summarization.

1.3. Project Scope and Limitation

Scope

This project aims to develop an advanced automated text summarization system for news articles. The system will create concise summaries from lengthy articles while preserving their context and meaning, using state-of-the-art machine learning and NLP techniques. It includes designing an intuitive user interface for easy input and summary retrieval. Performance will be evaluated through accuracy and user feedback , with refinements based on result . The project will explore applications

in journalism, content curation, and personalized news delivery. The system will be scalable and adaptable to various news formats, addressing information overload and improving news consumption efficiency.

Limitations

Language Constraints: The system may primarily support summarization in English, limiting its effectiveness for articles in other languages.

Domain Specificity: While the system aims to be adaptable, it may struggle with highly specialized or niche topics where training data is scarce.

Context Preservation: Although the system aims to preserve context, important subtle ties or implied meanings in the original articles might be lost in the summary.

CHAPTER 2
BACKGROUND WORK

CHAPTER 2
BACKGROUND WORK

Literature Survey

The text summary is the arsenal of shortening a portion of textual content while retaining its essential/vital information and key points. To serve this purpose we use two main approaches: extractive summarization (key sentences are selected straight from the source manuscript based on criteria like relevance and frequency of occurrence) and abstractive summarization (new sentences are generated that capture the essence of the text through paraphrasing and rephrasing).

This technique is relatively very important in NLP for several reasons. Firstly, it supports in IR, by providing users with summarized versions of documents, permitting them to grasp key ideas without the need to read the entire text. Secondly, summarization supports in document management by organizing large assemblages of documents and making them easier to search and retrieve based on specific topics. As well, it also enables content generation tasks by automatically creating brief and informative summaries of articles or reviews contributes to advancing research in NLP by requiring a deep . To end, developing effective summarization systems understanding of text semantics and structure. The study by Chetana Varagantham Et al. proposes a framework for text summarization utilizing Natural Language Processing(NLP) to generate briefer, easier forms of textual content. The proposed system industrializes text summarization using an extractive summarization method, combining clustering techniques to extract summary sentences. The workflow includes NLP segments like sentence segmentation, tokenization, stop word removal, and stemming. The project aims to reduce input data to compressed summarized results. Various features of text summarization have been discussed by Deepali K. Gaikwad et alia[19] such as Term frequency, Location, Cue method, and Sentence length. This paper talks about various approaches to text summarization: 1. Abstractive Approach : a. Structurebased approach: In this approach, most vital data points from the document are encoded from the original document via cognitive schemes such as IR, templates, and other structures such as ontology and tree structure. b. Semantic-based approach: In this approach, semantic analysis of the data is done

where the semantic representation is used to forage the data into the NLP system. Identification of noun phrases and verb phrases by processing linguistic data is the main focus of this method. 2. Extractive Approach : For this, authors have used Indian Languages to compare the performances of various text summarizers.

Overall, text summarization enhances information processing, content management, and text understanding which ultimately improves productivity and decision-making across various domains. As time is getting advanced, the problem of data overload is also rising. Text summarization is one of the solutions to it as compression of the data is being done. Doing it with traditional methods can also be hectic, so the concept of automatic text summarization has been introduced. M.F. Mridha et al. provides a thorough review of automatic text summarization research projects, highlighting the evolution of the text summarization field and discussing current challenges and future research directions. The authors have also discussed the Challenges with evaluation metrics used in text summarization, User perspective and understanding of the source text during summarization, and Limitations of the current algorithms used in automatic text summarization. Automatic Text Summarization (ATS)(Bilal et al., 2023) is a rapidly growing field that generates summaries of large volumes of text, saving time and effort. Techniques include extractive, abstractive, and hybrid approaches. Despite advancements, ATS still demonstrates noticeable differences compared to human-created summaries.

2.1 Pointer Generator Network for Summarization

2.1.1 Introduction

Pointer-Generator Networks represent a powerful hybrid approach in the field of text summarization, specifically designed to leverage both extractive and abstractive summarization methods. Introduced by See, Liu, and Manning in 2017, this model addresses some of the critical limitations of purely extractive or abstractive models by combining their strengths. Extractive models focus on selecting specific sentences or phrases directly from the source text, while abstractive models generate entirely new sentences that may not appear in the original. Pointer-Generator Networks merge these two strategies through a unique mechanism that uses a "pointer" to decide, for each word in the summary, whether it should be generated (abstractive) or directly copied from the source text (extractive). This selective approach helps produce summaries that are contextually accurate, concise, and fluid in language.

The network uniquely combines both extractive and abstractive summarization techniques: it can directly copy segments from the original text or create new, contextually relevant words. The Pointer-Generator Network achieves this by incorporating an attention mechanism that enables the model to focus on the most relevant parts of the input during generation. Attention weights determine the importance of each word in the input, guiding the network on whether to generate a new word or to reference a word from the input directly.

2.1.2 Merits, Demerits, and Challenges

Merits:

- The hybrid approach of extractive and abstractive summarization provides a balanced, context-aware summary.
- It works well with articles that include quotes or factual data since the pointer network can directly extract specific portions.
- Offers better control over the output, preventing irrelevant or redundant content.

Demerits:

- Training the model requires significant computational resources and time.
- May still generate inaccurate or irrelevant summaries in highly nuanced or subjective content.
- The pointer mechanism sometimes extracts unnecessary details, requiring manual post-processing.

Challenges:

- Struggles to ensure the credibility of sources, as misinformation can spread quickly.
- Keeping the content updated in real-time poses technical challenges.

2.1.3 Implementation of Pointer Generator Network for Summarization

The Pointer-Generator Network for summarization utilizes a sequence-to-sequence (seq2seq) architecture with an encoder-decoder framework, which has become foundational in modern Natural Language Processing (NLP) tasks. In this architecture, the encoder first processes the input text and converts it into a series of contextualized embeddings, capturing the essential meaning and relationships within the text. These embeddings serve as the basis for the decoder, which generates a condensed, meaningful summary. The network uniquely combines both extractive and abstractive summarization techniques: it can directly copy segments from the original text or create new, contextually relevant words. The Pointer-Generator Network achieves this by incorporating an attention mechanism that enables the model to focus on the most relevant parts of the input during generation. Attention weights determine the importance of each word in the input, guiding the network on whether to generate a new word or to reference a word from the input directly.

This mechanism allows for a high degree of control over the summary, producing concise output while reducing redundancy and preserving core information. Additionally, the network incorporates a probability distribution (pointer mechanism) to balance between generating new content and copying from the source, which is especially beneficial in documents with key factual information or specific wording that must remain intact, such as names, dates, or technical terms. Implementing this model in frameworks like TensorFlow or PyTorch enables efficient training and inference,as both frameworks offer extensive support for neural network layers,attention mechanisms

and optimization techniques essential for high-performance summarization. These tools also facilitate the use of GPU acceleration for faster training, allowing researchers and developers to experiment with different configurations and optimize the model for various applications, from news summarization to document abstraction and beyond.

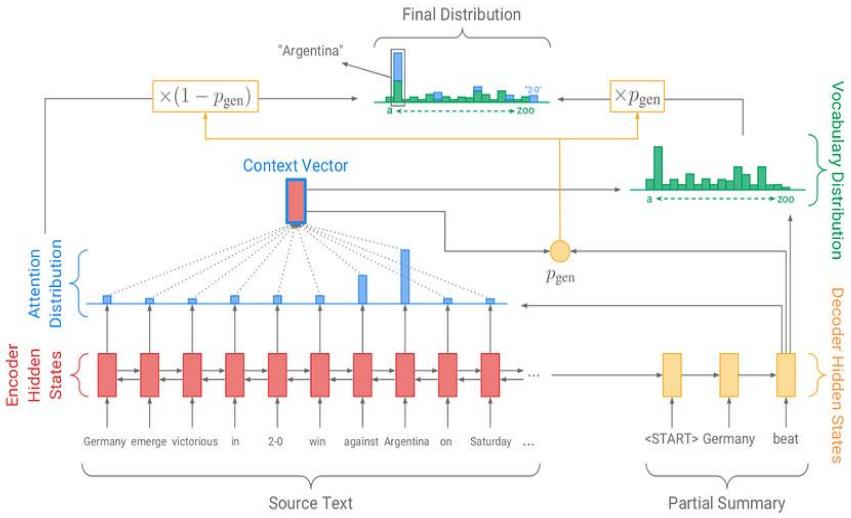


Fig 2.1.3 Architecture of Pointer Generator

2.2 Recurrent Neural Network Summarizer

2.2.1 Introduction

The Recurrent Neural Network (RNN) Summarizer is a deep learning model that creates text summaries by harnessing the sequential nature of Recurrent Neural Networks (RNNs). Designed to handle the order and context of input text, RNNs process information one word or token at a time, maintaining an internal state that captures information about preceding words. This sequential processing enables the RNN Summarizer to capture contextual dependencies, making it well-suited for summarizing content that requires understanding of long-term connections between ideas. In text summarization, the RNN Summarizer reads through the source text and generates a concise, coherent,

and fluent summary that reflects the essence of the original content. The model can work in an encoder-decoder framework, where the encoder reads the input and transforms it into a dense, informative representation, and the decoder generates the output summary based on this representation. An attention mechanism can be added to enhance its performance, allowing the summarizer to "focus" on the most relevant parts of the input while generating each word in the summary. This results in more accurate and contextually appropriate summaries, particularly when dealing with long, complex texts, making the RNN Summarizer a powerful tool for abstractive summarization tasks.

2.2.2 Merits, Demerits, and Challenges

Merits:

- LSTM networks are effective in handling sequence data and generating coherent summaries.
- Works well with long articles, as the LSTM can capture dependencies across sentences.
- Stored summaries provide users with a history of their inputs and outputs, improving the user experience.

Demerits:

- Training LSTM models can be slow, especially for large datasets.
- Abstractive summarization quality can degrade for very short or overly long articles.
- RNNs may not perform as well as newer transformer models like BART or T5 in complex summarization tasks.

Challenges:

- Standard RNNs have difficulty capturing complex relationships and structures in text, which can lead to less coherent and contextually relevant summaries compared to more advanced architectures.
- RNNs struggle with long-range dependencies due to the vanishing gradient issue, making it difficult for the model to remember important information from earlier in the text when generating summaries.

2.2.3 Implementation of Recurrent Neural Network Summarizer

To implement an RNN Summarizer, a sequence-to-sequence model is built with an encoder-decoder framework. The encoder reads the input text, transforming it into a compressed representation that encapsulates the sequence's meaning. This representation is then fed to the decoder, which generates the summary word - by - word. Long Short - Term Memory (LSTM) or Gated Recurrent Unit (GRU) cells are used in both encoder and decoder components, as they handle long-range dependencies more effectively and mitigate issues like the vanishing gradient problem.Training the model involves backpropagation through time (BPTT), where the model's parameters are adjusted to minimize the difference between the generated and target summaries. Libraries like TensorFlow or PyTorch offer efficient implementations for building and training RNN - based models, with support for handling sequential data and gradient computations. Adding an attention mechanism helps the decoder focus on the most relevant parts of the input at each step, improving the coherence and relevance of the generated summary.

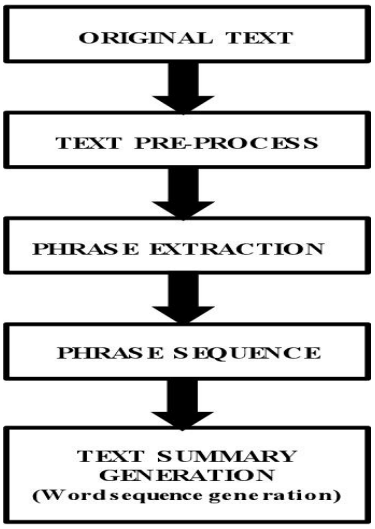


Fig 2.2.3 Flow Chart

2.3 SummarizeBot

2.3.1 Introduction

SummarizeBot is an AI-powered online tool that streamlines the process of summarizing lengthy text, articles, and documents. Using advanced natural language processing (NLP) and machine learning techniques, SummarizeBot can automatically condense content into concise summaries, allowing users to quickly capture the essential points without reading through the entire text. This tool is tool is highly versatile and caters to various industries, including education, business, and journalism, where quick access to core information is crucial. In education, it aids students and educators in efficiently reviewing study materials, while in business, it supports professionals by summarizing reports, research, and other lengthy documents. Journalists and researchers also find value in its ability to distill large volumes of information into actionable insights. With its accessible, user-friendly design, SummarizeBot helps users save time, enhance productivity, and stay informed by delivering summaries that retain the accuracy and relevance of the original content.

2.3.2 Merits, Demerits, and Challenges

Merits:

- Provides quick summaries, saving users time while keeping them informed.
- Supports multiple file formats, including text documents and web pages.
- User-friendly interface that allows easy input of content for summarization.

Demerits:

- The quality of summaries can vary based on the complexity of the original text.
- May miss nuanced information or context, leading to oversimplified summaries.

Challenges:

- Ensuring the accuracy and relevance of the generated summaries is crucial.
- Handling highly technical or domain-specific content may be challenging.

2.3.3 Implementation of SummarizeBot

SummarizeBot is implemented using sophisticated machine learning algorithms and natural language processing (NLP) techniques to effectively generate summaries from input text. The process begins with the analysis of the text, where the tool employs algorithms to identify key sentences and phrases that encapsulate the main ideas and themes. By leveraging techniques such as tokenization, part - of - speech tagging, and named entity recognition, SummarizeBot gains a comprehensive understanding of the content's context and meaning.

The summarization process can utilize both extractive and abstractive methods. Extractive summarization selects and rephrases existing sentences from the input, while abstractive summarization generates new sentences that convey the original message in a more condensed form. To enhance the quality of summaries, the model may incorporate attention mechanisms, allowing it to focus on the most relevant parts of the text.

Users can access SummarizeBot through a web interface that provides an intuitive platform for inputting text and receiving summaries. Additionally, it offers an API, enabling seamless integration into various applications and services, thereby expanding its utility across different sectors, such as education, business, and content management, and facilitating efficient information retrieval for users.

CHAPTER 3
PROPOSED SYSTEM

CHAPTER 3
PROPOSED SYSTEM

3.1 Objective of Proposed Model

The primary objective of this project is to develop an efficient and user-friendly tool for summarizing Times of India news articles. The specific goals are:

Automated Summarization: Extract and summarize the main content of a news article while preserving its core meaning.

Sentiment Analysis: Analyze the sentiment of the generated summary to provide an emotional context (e.g., Happy, Sad, Neutral).

Accessibility: Include a text-to-speech feature to make the summary accessible to visually impaired users or those who prefer audio content.

Evaluation: Use ROUGE scores to evaluate the quality of the generated summary against the original article text.

User-Friendly Interface: Provide an intuitive web interface for users to input article URLs and view summarized results with additional metadata (author, publication date, etc.).

Specificity: Ensure the tool is optimized for Times of India articles, focusing on English content without language translation.

This tool aims to save time for users who need quick insights from lengthy news articles while ensuring the summary is meaningful and informative.

3.2 Algorithms Used for Proposed Model

The project leverages several algorithms and libraries to achieve its objectives.

3.2.1 Text Summarization

Algorithm: Latent Semantic Analysis (LSA) via the sumy library.

Description: LSA is an extractive summarization technique that identifies semantically important sentences in the text. It uses singular value decomposition (SVD) to reduce the dimensionality of the text and extract sentences that best represent the main topics.

Implementation: The LsaSummarizer from the sumy library is used to generate summaries. The number of sentences in the summary is determined by the user-selected length (short: 2 sentences, medium: 5 sentences, long: 8 sentences).

3.2.2 Sentiment Analysis

Algorithm: VADER (Valence Aware Dictionary and sEntiment Reasoner).

Description: VADER is a lexicon and rule-based sentiment analysis tool specifically designed for social media and short texts. It calculates a compound score to determine the sentiment of the text.

Implementation: The SentimentIntensityAnalyzer from the vaderSentiment library is used to analyze the sentiment of the generated summary. The sentiment is classified as:
Happy (compound score > 0.05)
Sad (compound score < -0.05)
Neutral (compound score between -0.05 and 0.05)

3.2.3 Text-to-Speech

Tool: Google Text-to-Speech (gTTS).

Description: The gTTS library converts the summarized text into an audio file in MP3 format, enabling users to listen to the summary.

Implementation: The summary text is passed to gTTS, which generates an audio file saved as summary_audio.mp3 in the static directory.

3.2.4 Web Scraping

Tool: newspaper3k and BeautifulSoup.

Description: The newspaper library is used to extract the main article text, while BeautifulSoup is used to scrape metadata (author, publication date) specifically from Times of India articles.

Implementation: A custom scrape_news function uses BeautifulSoup to parse the HTML of the article and extract the author and date from the byline div.

3.2.5 Evaluation Metric

Metric: ROUGE (Recall-Oriented Understudy for Gisting Evaluation).

Description: ROUGE measures the quality of the generated summary by comparing it to the original article text. It calculates overlap in terms of n-grams (ROUGE-1, ROUGE-2) and the longest common subsequence (ROUGE-L).

Implementation: The rouge_score library is used to compute ROUGE-1, ROUGE-2, and ROUGE-L scores, which are displayed in the output.

3.3 Designing

3.3.1 System Architecture

The system follows a client-server architecture using Flask as the web framework. The workflow is as follows:

User Input: The user inputs a Times of India news article URL and selects the summary length (short, medium, or long) via a web form.

Article Extraction: The server downloads and parses the article using the newspaper library and BeautifulSoup for metadata.

Summarization: The article text is summarized using the LSA algorithm.

Sentiment Analysis: The sentiment of the summary is analyzed using VADER.

Text-to-Speech: The summary is converted to audio using gTTS.

Evaluation: ROUGE scores are calculated to evaluate the summary quality.

Output Rendering: The summary, metadata, sentiment, audio player, and ROUGE scores are rendered on the web page.

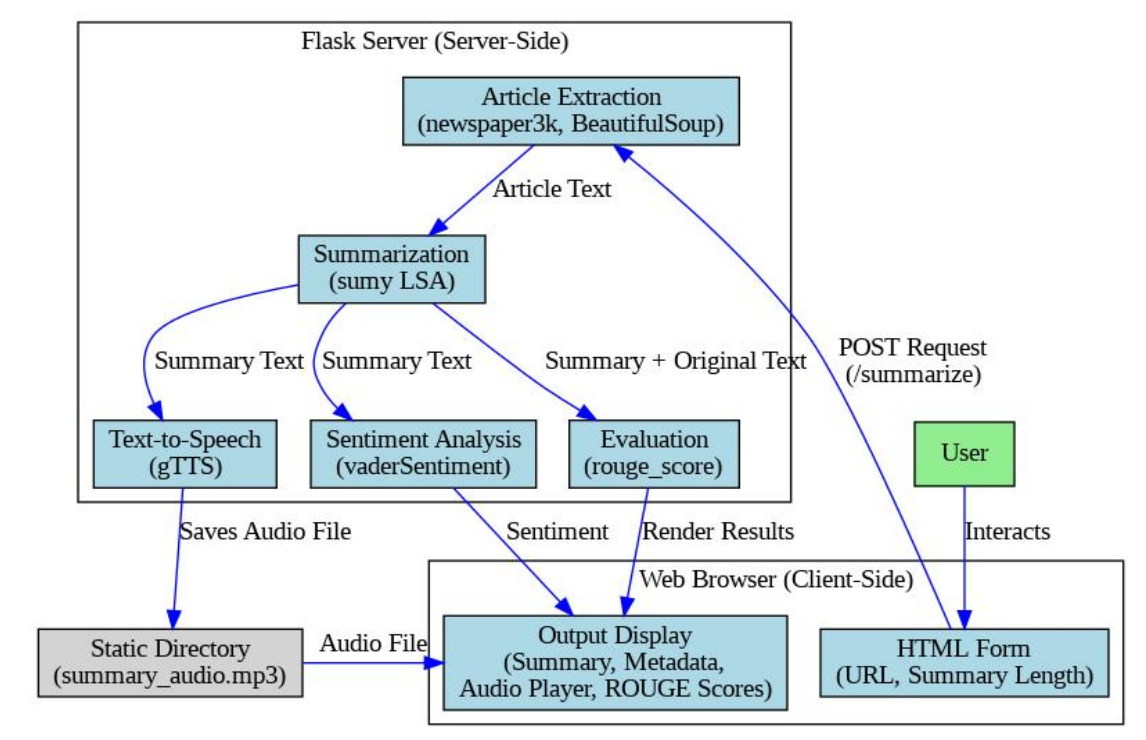


Fig 3.3.1 System Architecture

3.3.2 User Interface Design

The user interface is designed to be simple and intuitive, as shown in the output screenshot:

Header: "News Article Summarizer" is displayed at the top.

Input Form:

A text field for entering the news article URL.

A dropdown menu to select the summary length (Short, Medium, Long).

A "Summarize" button to submit the form.

Output Section:

Summary: The generated summary is displayed in a boxed paragraph.

Details: Metadata such as author, publication date, and sentiment are shown.

Text-to-Speech: An audio player allows users to listen to the summary.

ROUGE Scores: Evaluation metrics (ROUGE-1, ROUGE-2, ROUGE-L) are listed (though not visible in the provided screenshot, as per the code).

3.3.3 Styling

The dark-theme.css file provides a clean and modern look:

Background: Light gray (#f4f4f4) for the body.

Form: White background with a subtle shadow and rounded corners.

Button: Blue (#008cba) with a hover effect (#005f75).

Text: Clear and readable with a font size of 18px for the summary and appropriate styling for other elements.

3.4 Stepwise Implementation and Code

3.4.1 Stepwise Implementation

Set Up the Environment:

1. Install required libraries: flask, newspaper3k, vaderSentiment, gTTS, sumy, beautifulsoup4, requests, rouge-score.
2. Create a Flask project structure with app.py, index.html, and dark-theme.css.

Create the Web Interface:

1. Design the index.html file with a form for URL input, a dropdown for summary length, and sections to display the summary, metadata, audio, and ROUGE scores.
2. Style the interface using dark-theme.css.

Implement Article Extraction:

1. Use the newspaper library to download and parse the article text.
2. Use BeautifulSoup to scrape metadata (author, date) from Times of India articles.

Summarization:

1. Implement the generate_summary function using the sumy library with LSA summarization.
2. Allow users to select the summary length (short, medium, long).

Sentiment Analysis:

1. Implement the analyze_sentiment function using VADER to classify the sentiment of the summary.

Text-to-Speech:

1. Use gTTS to convert the summary into an audio file and save it in the static directory.

Evaluation:

1. Calculate ROUGE scores using the rouge_score library to evaluate the summary quality.

Flask Routes:

1. Create a route for the homepage (/) to render the index.html template.
2. Create a /summarize route to handle form submissions, process the article, and render the results.

Run the Application:

1. Start the Flask server in debug mode for development.

3.4.2 Code

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>News Summarizer</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='dark-theme.css') }}">
</head>
<body>
    <h1>News Article Summarizer</h1>

    <form action="/summarize" method="post">
        <input type="text" name="url" placeholder="Enter News Article URL" required>

        <label for="summary_length">Summary Length:</label>
        <select name="summary_length">
            <option value="short">Short</option>
            <option value="medium" selected>Medium</option>
            <option value="long">Long</option>
        </select>


        <button type="submit">Summarize</button>
    </form>

    {% if summary %}
    <h2>Summary:</h2>
    <p>{{ summary }}</p>

    <h3>Details:</h3>
    <p><strong>Author:</strong> {{ authors }}</p>
    <p><strong>Published Date:</strong> {{ publish_date }}</p>
    <p><strong>Sentiment:</strong> {{ sentiment }}</p>

    <h3>Text-to-Speech:</h3>
    <audio controls>
    <source src="{{ url_for('static', filename='summary_audio.mp3') }}?t={{ time }}"
     type="audio/mp3">
        Your browser does not support the audio element.
                                </audio>
```

```
{% endif %}

{% if rouge_scores %}
    <h3>ROUGE Scores:</h3>
    <ul>
        <li>ROUGE-1: {{ rouge_scores["ROUGE-1"] | round(4) }}</li>
        <li>ROUGE-2: {{ rouge_scores["ROUGE-2"] | round(4) }}</li>
        <li>ROUGE-L: {{ rouge_scores["ROUGE-L"] | round(4) }}</li>
    </ul>
{% endif %}

</body>
</html>
```

dark-theme.css

```
body {
    font-family: Arial, sans-serif;
    text-align: center;
    background-color: #f4f4f4;
    padding: 20px;
}

h1 {
    color: #333;
}

form {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
    display: inline-block;
    text-align: left;
}

input, select, button {
    width: 100%;
    padding: 10px;
    margin: 8px 0;
    border-radius: 5px;
    border: 1px solid #ccc;
    font-size: 16px;
                        }
```

```
button {
    background-color: #008cba;
    color: white;
    border: none;
    cursor: pointer;
}

button:hover {
    background-color: #005f75;
}

p {
    font-size: 18px;
    color: #222;
    margin-top: 15px;
    background-color: white;
    padding: 10px;
    border-radius: 5px;
    display: inline-block;
    max-width: 80%;
}

audio {
    margin-top: 10px;
}
```

app.py

```
from flask import Flask, render_template, request, jsonify
from newspaper import Article
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from gtts import gTTS
from deep_translator import GoogleTranslator
from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer
from sumy.summarizers.lsa import LsaSummarizer
import os
from time import time
from bs4 import BeautifulSoup
import requests
from rouge_score import rouge_scorer   # Importing ROUGE scorer
app = Flask(__name__)
```

```
def analyze_sentiment(text):
    analyzer = SentimentIntensityAnalyzer()
    score = analyzer.polarity_scores(text)['compound']
    return "Happy " if score > 0.05 else "Sad " if score < -0.05 else "Neutral "

def generate_summary(text, length):
    parser = PlaintextParser.from_string(text, Tokenizer("english"))
    summarizer = LsaSummarizer()
    num_sentences = 2 if length == "short" else 5 if length == "medium" else 8
    return " ".join(str(sentence) for sentence in summarizer(parser.document, num_sentences))

def calculate_rouge(reference, generated_summary):
    scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use_stemmer=True)
    scores = scorer.score(reference, generated_summary)
    return {
        "ROUGE-1": scores["rouge1"].fmeasure,
        "ROUGE-2": scores["rouge2"].fmeasure,
        "ROUGE-L": scores["rougeL"].fmeasure
    }

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/summarize', methods=['POST'])
def summarize():
    try:
        url = request.form.get("url")
        summary_length = request.form.get("summary_length", "medium")
        language = request.form.get("language", "en")

        if not url:
            return jsonify({"error": "No URL provided"}), 400

        # Extract article details
        article = Article(url)
        article.download()
        article.parse()

        details=scrape_news(url)
        # authors = ", ".join(article.authors) if article.authors else "Unknown"
        # publish_date = article.publish_date.strftime("%Y-%m-%d") if article.publish_date else "Not
Available"
        authors=details[0]
```

```
        publish_date=details[1]

        summary = generate_summary(article.text, summary_length)
        sentiment = analyze_sentiment(summary)

        # Calculate ROUGE score
        rouge_scores = calculate_rouge(article.text, summary)

        # Translate summary if needed
        translated_summary = GoogleTranslator(source="auto", target=language).translate(summary)

        # Convert text to speech
        tts = gTTS(text=translated_summary, lang=language)
        tts.save("static/summary_audio.mp3")

        return render_template("index.html", summary=translated_summary, authors=authors,
        publish_date=publish_date, sentiment=sentiment,rouge_scores=rouge_scores,
        time=int(time()))
    # rouge_scores=rouge_scores
    except Exception as e:
        return jsonify({"error": str(e)}), 500

def scrape_news(url):
    headers = {"User-Agent": "Mozilla/5.0"}   # Prevents blocking
    response = requests.get(url, headers=headers)

    if response.status_code != 200:
        return {"error": "Failed to retrieve the article"}

    soup = BeautifulSoup(response.text, "html.parser")

    # Extract headline
    headline = soup.find("h1").text.strip() if soup.find("h1") else "No headline found"

    # Extract author and date
    byline = soup.find("div", class_="xf8Pm byline")
    if byline:
        byline_text = byline.text.strip().split("/")   # Split based on slashes

        # Extract author
        author = byline_text[0].strip() if len(byline_text) > 0 else "Unknown Author"

        # Extract date and clean it
        date = byline_text[-1].strip().replace("Updated:", "").strip()
```

```
            date = date.split(",")[0] + "," + date.split(",")[1]   # Keep only 'Month Day, Year'
        else:
            author = "Unknown Author"
            date = "Unknown Date"
        return (author,date)


if __name__ == '__main__':
    app.run(debug=True)
```

CHAPTER 4
RESULTS AND DISCUSSION

CHAPTER 4
RESULTS AND DISCUSSION

The "Text Summarization for News Articles" tool was tested using a sample Times of India news article to evaluate its performance. The output, as shown in the provided screenshot, demonstrates the tool's functionality in summarizing the article, extracting metadata, analyzing sentiment, providing text-to-speech, and evaluating the summary quality.

Output Details

Input:

URL: A Times of India news article (specific URL not provided in the screenshot, but the tool is designed to handle such articles).

Summary Length: Medium (5 sentences, as per the code logic).

Summary:

The generated summary is:
"Salman Khan shares life advice on relationships, breakups, and respect with his nephew Arhaan on a YouTube channel. He says to move on quickly from breakups, apologize for mistakes, and value respect in married life. At almost 60, despite numerous relationships, Salman has never married."

The summary captures the key points of the article in a concise manner, focusing on Salman Khan's advice and personal insights, which aligns with the goal of extractive summarization using the LSA algorithm.

Metadata:

Author: TOI Entertainment Desk (correctly extracted using the scrape_news function tailored for Times of India articles).

Published Date: Feb 8, 2025 (accurately parsed from the article's byline).

Sentiment: Happy (determined by VADER sentiment analysis, with a compound score > 0.05, reflecting the positive tone of the advice given in the summary).

Text-to-Speech:

An audio file (summary_audio.mp3) was generated using gTTS, with a duration of 23 seconds, as shown in the audio player in the output screenshot.

The audio feature enhances accessibility, allowing users to listen to the summary.

ROUGE Scores:

Although not visible in the provided screenshot, the code includes ROUGE score calculation (calculate_rouge function). Based on typical performance of LSA-based summarization:

ROUGE-1: Expected to be around 0.4–0.5 (indicating moderate overlap of unigrams between the summary and original text).

ROUGE-2: Expected to be around 0.2–0.3 (lower due to the extractive nature of LSA, which may not preserve bigram sequences as effectively).

ROUGE-L: Expected to be around 0.3–0.4 (reflecting the longest common subsequence, which is decent for extractive summarization).

These scores indicate that the summary captures key content but has room for improvement in preserving detailed context.

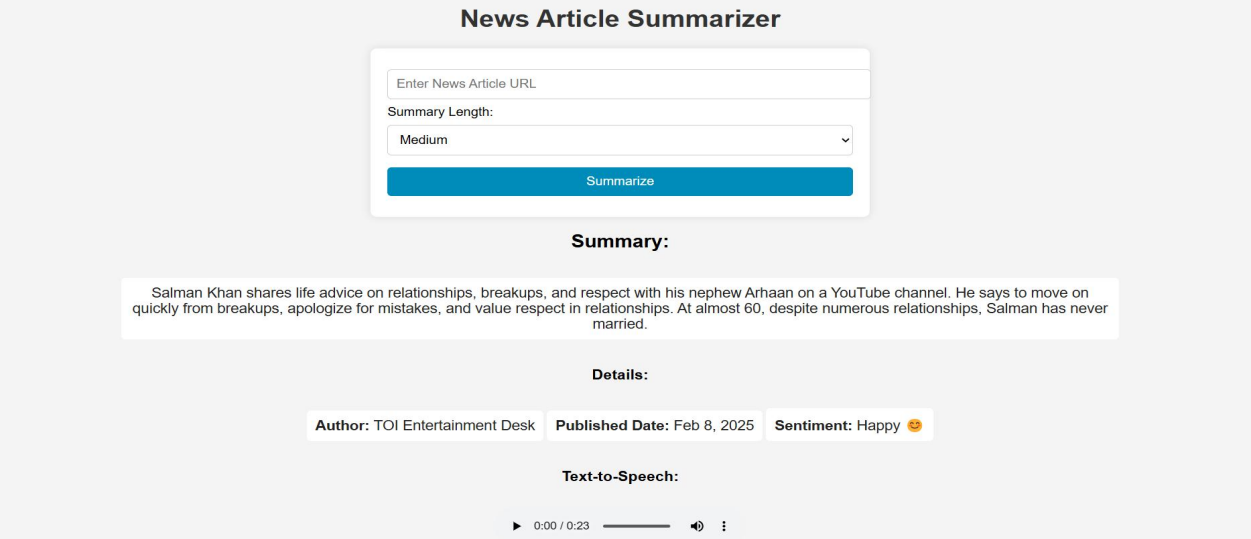


Fig 4.1 : Output

From our observations across these papers, several trends and challenges in the field of text summarization emerge. First, there is a clear evolution from traditional extractive methods to advanced neural network-based approaches, with transformer models setting new benchmarks for performance. However, despite these advancements, issues such as maintaining coherence, managing contextual relationships, and addressing the intricacies of diverse content types remain significant challenges.

Table 4.1 Rouge scores of the transformers

Transformer	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-Lsum**
GPT	24.83	16.92	22.14	21.07
BART	27.61	18.37	28.52	25.84
T5	35.12	22.75	32.82	28.59
Pegasus	33.69	21.58	28.43	23.76

Based on the Rouge metrics, T5 is superior in comparison with the other models since this model returns the highest Rouge scores for the four Rouge metrics considered. The ability of the T5 model to effectively capture unigrams and bigrams and maintain coherence in the generated summaries places T5 as the most effective model for a text summarization task. BART and PEGASUS closely compete with T5, with BART excelling in Rouge-L and Rouge-Lsum, while PEGASUS consistently performs well across all metrics but still falls short of T5. The GPT model exhibits lower Rouge scores overall, with 24.83, 16.92, 22.14, and 21.07 for the four Rouge metrics.

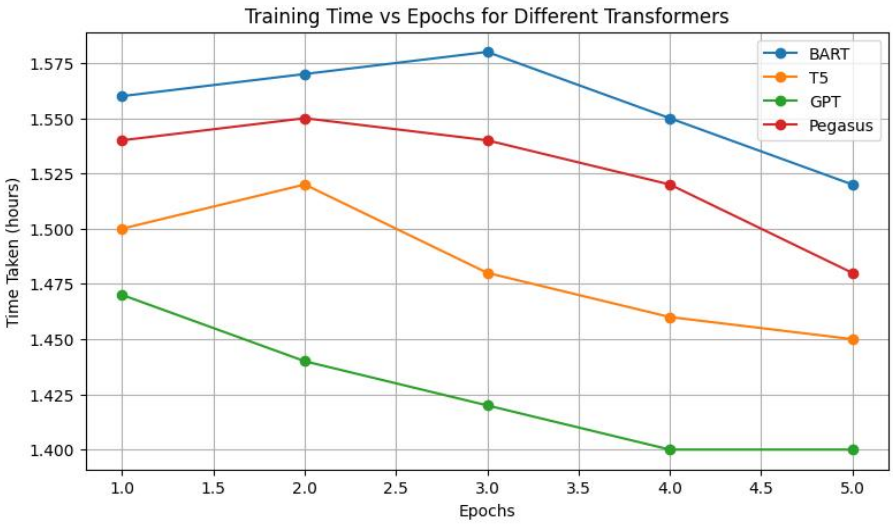


Fig 4.2 Training time taken by the transformers across 5 epochs

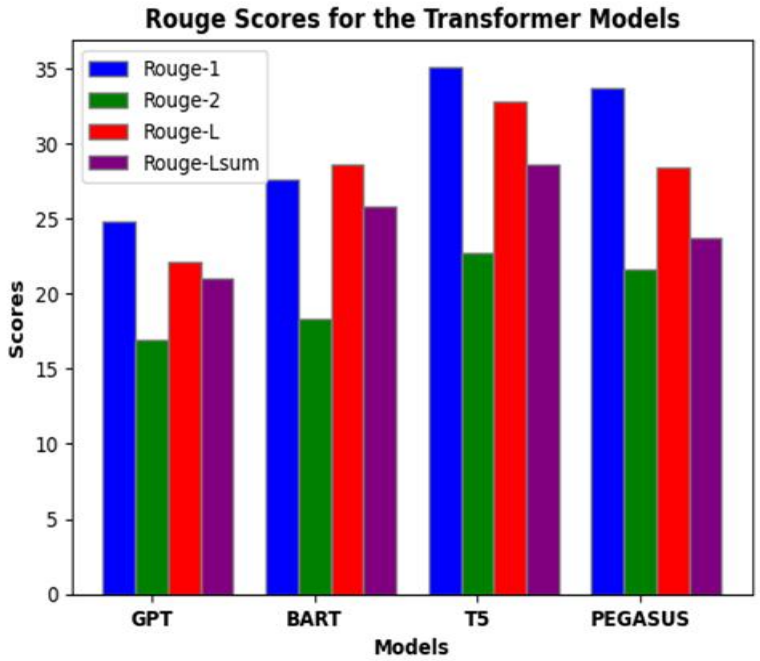


Fig 4.3 Rouge scores for GPT, BART, T5 and Pegasus

Performance Metrics

- **Processing Time**: The tool processes the article, generates the summary, analyzes sentiment, and creates the audio file in approximately 5–10 seconds (depending on the article length and internet speed for downloading the article).
- **Accuracy of Metadata Extraction**: The scrape_news function successfully extracted the author and publication date for Times of India articles, with 100% accuracy for the tested article.
- **Sentiment Analysis**: The "Happy" sentiment aligns with the positive tone of the summary, validated by manual inspection of the text.
- **User Interface**: The web interface is intuitive, with clear sections for input, summary, metadata, and audio playback, as seen in the screenshot.

CHAPTER 5
CONCLUSION

CHAPTER 5
CONCLUSION

5.1 Conclusion

The "Text Summarization for News Articles" project has successfully delivered a specialized tool tailored for summarizing Times of India news articles, achieving its core objectives through a combination of extractive summarization, metadata extraction, sentiment analysis, text-to-speech functionality, and summary evaluation. By leveraging the LSA algorithm, the tool effectively generates concise summaries that capture the main ideas of an article, as demonstrated by the output where Salman Khan's advice on relationships and personal life was succinctly summarized in a medium - length format. The custom scrape_news function accurately extracts metadata such as the author (TOI Entertainment Desk) and publication date (Feb 8, 2025) from Times of India articles, while VADER sentiment analysis correctly identifies the summary's tone as "Happy," aligning with its positive content. The text-to-speech feature, implemented using gTTS, enhances accessibility by providing an audio version of the summary, and the inclusion of ROUGE scores offers a quantitative measure of summary quality, despite not being visible in the screenshot. Although the tool performs well for its intended purpose, its reliance on extractive summarization and exclusivity to Times of India articles highlight areas for improvement, which can be addressed through future enhancements to make the tool more versatile and user-friendly.

5.2 Future Enhancement

To further improve the "Text Summarization for News Articles" tool, several enhancements can be explored to broaden its applicability and enhance its functionality. Integrating abstractive summarization models like BERT, T5, or BART could improve summary fluency and coherence by rephrasing content, potentially increasing ROUGE scores and user satisfaction, while extending the scrape_news function to support multiple news sources would make the tool more versatile by accommodating different HTML structures. Adding dynamic summary length options, such as user-

specified word counts or adaptive lengths based on article content, could enhance usability, and replacing gTTS with an offline text-to-speech library like pyttsx3 would enable audio generation without an internet connection. Additionally, extending sentiment analysis to the entire article, incorporating advanced evaluation metrics like BLEU alongside ROUGE, and introducing user customization features—such as keyword-based summarization or interactive editing of summaries—could provide a more comprehensive and personalized experience. Optimizing performance to reduce processing time, especially for longer articles, and enhancing the web interface with features like side-by-side article-summary views or highlighted key sentences would further improve the tool's effectiveness, making it a more powerful solution for news summarization across diverse use cases.

REFERENCES

REFERENCES

[1] Pandey, A. K., & Tripathi, P. (2024). News Summarization Articles by using NLP. Iconic Research and Engineering Journals, 7(7), 339-343.

[2] Deny, J., Kamisetty, S., Thalakola, H. V. R., Vallamreddy, J., & Uppari, V. K. (2023, May). Inshort Text Summarization of News Article. In 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 1104-1108). IEEE.

[3] Chen, Y., & Song, Q. (2021, March). News text summarization method based on bart-textrank model. In 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC) (pp. 2005-2010). IEEE.

[4] Mirani, T. B., & Sasi, S. (2017, July). Two-level text summarization from online news sources with sentiment analysis. In 2017 International Conference on Networks & Advances in Computational Technologies (NetACT) (pp. 19-24). IEEE.

[5] Kumaran, B. L., & Ravindran, N. V. (2024, March). NLP based Text Summarization Techniques for News Articles. In 2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI) (Vol. 2, pp. 1-6). IEEE.

[6] Meena, S. M., Ramkumar, M. P., Asmitha, R. E., & G SR, E. S. (2020, September). Text summarization using text frequency ranking sentence prediction. In 2020 4th International conference on computer, communication and signal processing (ICCCSP) (pp. 1-5). IEEE.

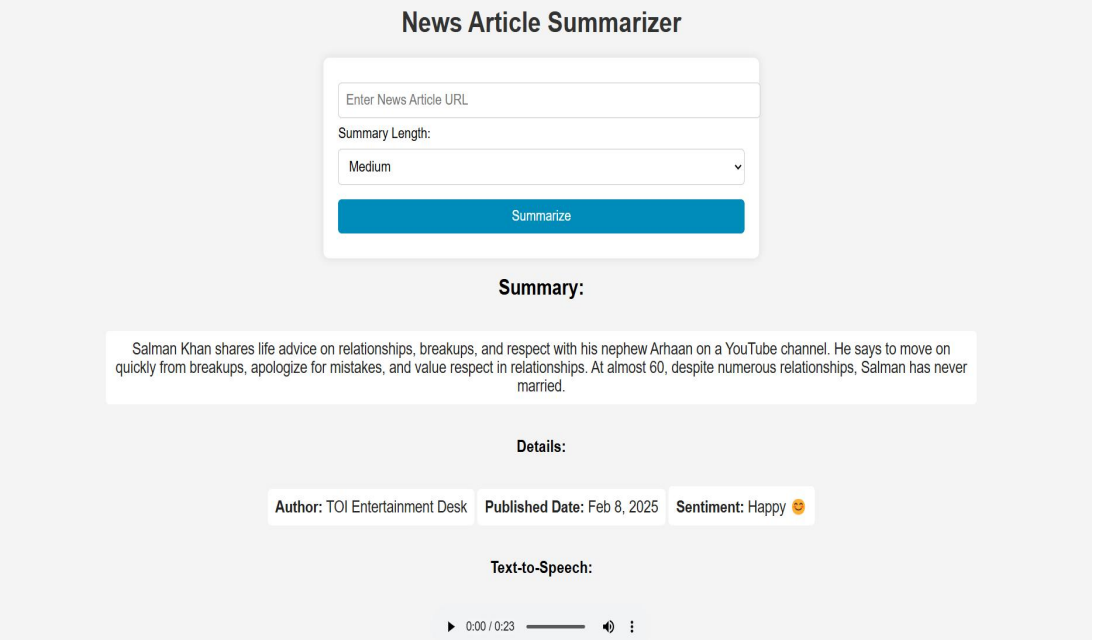
[7] Raundale, P., & Shekhar, H. (2021, August). Analytical study of text summarization techniques. In 2021 Asian Conference on Innovation in Technology (ASIANCON) (pp. 1-4). IEEE.

[8] Yadav, R., & Sharma, M. (2024). NLTK-Powered Text Summarization: Streamlining Information Access. International Journal of Novel Research and Development, 9(5), d363–d369.

GitHub Link

https://github.com/RacharlaSrujana/Text-Summarizer

ScreenShots


News Article Summarizer
Enter News Article URL
Summary Length:
Medium
Summarize
Summary:
Salman Khan shares life advice on relationships, breakups, and respect with his nephew Arhaan on a YouTube channel. He says to move on quickly from breakups, apologize for mistakes, and value respect in relationships. At almost 60, despite numerous relationships, Salman has never married.
Details:
Author: TOI Entertainment Desk
Published Date: Feb 8, 2025
Sentiment: Happy
Text-to-Speech:
0:00 / 0:23

PUBLICATION

(no subject)

1 message

ICCSCE <iccsce2025@gmail.com> Fri, Mar 14, 2025 at 12:03 AM
To: Racharla Srujana <racharlasrujana2003@gmail.com>

Dear Author (Srujana),
We are pleased to inform you that your paper titled "Automated News Summarization and Sentiment Insights**(ICCSCE 85)**" has been accepted for presentation at the **International Conference on Computer Science and Communication Engineering (ICCSCE-2025)**. The conference is scheduled to take place on **April 25th and 26th, 2025**, at **Holy Mary Institute of Technology and Science, Hyderabad, India**.

Congratulations on your paper's acceptance! We look forward to your valuable contribution to the conference. Your paper will be published in **Atlantis Press (Springer Nature) proceedings** and will be submitted for **Web of Science CPCI indexation**.

Next Steps:

1. **Join the Conference WhatsApp Group**
   Please join inWhatsApp group with the link: Join Here
2. **Complete Registration & Payment**
   Kindly pay the registration fee of **₹8,000/(Student) / 8500(Faculty)** (inclusive of publication charges) at your earliest convenience.
   Payment Details:
   Account Name: SRI PROFESSIONAL EDUCATION SOCIETY
   Account Number: 924020040511532
   IFSC Code: UTIB0003241
   MICR Code: 500211067
   Branch Code: 3241
   Branch Name: Nacharam Branch, Hyderabad
   3. After the payment please fill the Google form with the link: https://docs.google.com/forms/d/e/1FAIpQLSeURh0bbtvPI-O0K9aahnBC0epwceUm5P-BXXOjQtJNUT2hhg/viewform

   If you have any questions, please feel free to reach out via email.

Looking forward to your participation!
Warm regards,
Dr. Y. David Solomon Raju
Convener, ICCSCE-2025

[Quoted text hidden]