

**A**  
**Minor Project Report**  
**On**  
**NEWS ARTICLE AND PDF SUMMARIZER**

Submitted in partial fulfillment of the requirements for the award of the Degree of

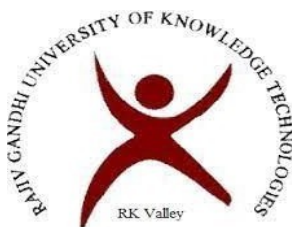
**BACHELOR OF TECHNOLOGY**  
**in**  
**COMPUTER SCIENCE AND ENGINEERING**

**Submitted by**

**M NIROSHA    R190461**  
**G PAVANI     R191097**

**Under the esteemed guidance of**

**V SRAVANI**, M. Tech  
Assistant Professor, Department of CSE.



**Submitted to**

**Department of Computer Science and Engineering**

**Rajiv Gandhi University of Knowledge and Technologies (RGUKT),**  
**R.K. Valley, Kadapa, Andhra Pradesh, 516330**  
**Accredited by 'NAAC' with 'B+' Grade**



**Rajiv Gandhi University of Knowledge and Technologies (RGUKT),  
R.K. Valley, Kadapa, Andhra Pradesh, 516330**

## **DECLARATION**

We hereby declare that the project report entitled **“NEWS ARTICLE AND PDF SUMMARIZER”** submitted to the Department of **COMPUTER SCIENCE AND ENGINEERING** in partial fulfilment of requirements for the award of the degree of **BACHELOR OF TECHNOLOGY**. This project is the result of our own effort and that it has not been submitted to any other University or institution for the award of any degree or diploma other than specified above.

**WITH SINCERE REGARDS**

M. Nirosha R190461

G. Pavani R191097



**Rajiv Gandhi University of Knowledge and Technologies (RGUKT),  
R.K. Valley, Kadapa, Andhra Pradesh, 516330**

**CERTIFICATE**

This is to certify that the project work titled **“NEWS ARTICLE AND PDF SUMMARIZER”** is a bonafide project work submitted by **M. NIROSHA - R190461, G . PAVANI - R191097** in the department of COMPUTER SCIENCE AND ENGINEERING in partial fulfillment of requirements for the award of degree of Bachelor of Technology in **Computer Science and Engineering** for the year 2023 - 2024 carried out the work under the supervision.

**GUIDE**

V.SRAVANI ,M.Tech  
Assistant Professor  
RGUKT, RK Valley

**HEAD OF THE DEPARTMENT**

Dr. RATNA KUMARI. Ch,M.Tech,(Ph.D)  
Assistant Professor  
RGUKT, RK Valley

**Signature of External Examiner**

## ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts success.

I also express my sincere gratitude to our respected Head of the Department **Dr. RATNA KUMARI Ch.** for her encouragement, overall guidance in viewing this project a good asset and effort in bringing out this project.

I would like to convey thanks to our guide at college **Ms. V SRAVANI** for her guidance, encouragement, co-operation and kindness during the entire duration of the course and academics.

My sincere thanks to all the members who helped me directly and indirectly in the completion of project work. I express my profound gratitude to all our friends and family members for their encouragement.

## Table of Contents

S.NO	INDEX	PAGE NO
	Abstract	i
	List of Figures	ii
1.	Introduction	1
	• Problem Statement	2
	• Objective	2-3
	• Scope of the project	3
	• Motivation	3-4
	• Contribution	4
2.	Literature Review	5-6
3.	Modules	
	Module-1: Environments Used and Software Requirement	7-9
	Module-2: System Development and Model Development	10-14
	Module-3: Model To Be Used For The Project and Implementation	15-26
4.	Results and Discussion	27-28
5.	Conclusion and Future Enhancements	29-30
	References	31

## List of Tables

TABLE NO	TITLE	Page No.
1	Literature Review	5-6

## **ABSTRACT**

The News Summarizer application is designed with user convenience in mind, featuring a user-friendly interface built using Python's Tkinter library. The tool accepts both URLs of online news articles and PDF files containing news content, offering flexibility in input formats. Once an article or document is provided, the application uses the newspaper3k library to download and parse the content. The TextBlob library is then employed to perform natural language processing (NLP) tasks, such as tokenization and sentiment analysis. The extracted text is summarized, and the sentiment of the content is evaluated to provide insights into the overall tone of the article.

One of the standout features of the News Summarizer is its ability to save summarized content in PDF format. This functionality is particularly useful for users who wish to archive or share concise versions of news articles. The summarization results are displayed directly within the Tkinter interface, allowing users to review the content immediately.

## List of Figures

<b>FIG NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
Figure 1	Import Required Libraries	20
Figure 2	Define the Summarize Function 1	20
Figure 3	Define the Summarize Function 2	21
Figure 4	Define the Select PDF Function	21
Figure 5	Define the Save Summary to PDF Function	22
Figure 6	Create the Main Application Window	23
Figure 7	Design and Place Widgets 1	23
Figure 8	Design and Place Widgets 2	24
Figure 9	Handle Main Event Loop	24
Figure 10-12	User Interface,Output with URL and PDF	25-26

# CHAPTER 1

## Introduction

In the era of digital transformation, the consumption of news has undergone a significant shift. With the proliferation of online news platforms, social media channels, and digital publications, individuals are inundated with an overwhelming amount of information daily. This deluge of content makes it increasingly challenging for readers to sift through the noise and find pertinent information quickly and efficiently. Traditional methods of news consumption, which involve reading lengthy articles, are becoming impractical for the time-constrained modern reader. This scenario calls for innovative solutions that can streamline the process of news consumption, making it more manageable and less time-consuming.

The News Summarizer project emerges as a response to this growing need for efficient news consumption tools. This application is designed to help users navigate the vast sea of information by providing concise summaries of news articles. Leveraging advanced natural language processing (NLP) techniques, the News Summarizer extracts the most critical information from articles, allowing users to grasp the essence of the news without having to read through the entire text. By focusing on the core content, the tool saves time and enhances the overall news reading experience.

At the heart of the News Summarizer project is its user-friendly interface, built using Python's Tkinter library. This graphical user interface (GUI) allows users to easily input URLs of online news articles or upload PDF files containing news content. Once an input is provided, the application utilizes the newspaper3k library to download and parse the article, followed by TextBlob for NLP tasks such as tokenization, summarization, and sentiment analysis. The summarized text is then displayed within the application, alongside the sentiment analysis results, providing users with a quick and comprehensive overview of the news content.

The project also includes features that enhance its practicality and usability. For instance, the ability to save summarized content in PDF format enables users to archive or share the summaries easily. The inclusion of sentiment analysis adds an extra layer of insight, allowing users to gauge the overall tone of the article quickly. The application also incorporates error handling mechanisms to manage common issues, ensuring a smooth and reliable user experience.



## Problem Statement:

In the digital age, users face an overwhelming volume of information from numerous online news sources, making it challenging to efficiently filter and consume relevant content. Traditional methods of reading entire articles are impractical due to time constraints, necessitating a tool for quick access to key information. This tool should provide concise summaries, handle different input formats like URLs and PDFs, and use advanced natural language processing (NLP) techniques for accuracy. Additionally, it should be user-friendly, allowing easy interaction and convenient access to summarized information. This solution aims to improve news consumption efficiency and address information overload.

## Objective:

The primary objective of this project is to develop a robust and efficient News Summarizer Application that enhances the way users consume news and information. This application will leverage advanced natural language processing (NLP) techniques to automatically generate concise and accurate summaries of lengthy articles from various online sources. The summarizer will support multiple input formats, including URLs and PDFs, ensuring versatility and user convenience.

Key objectives of the News Summarizer Application are:

- 1. Efficient Summarization:** Implement state-of-the-art NLP algorithms to accurately extract the most relevant information from news articles. The summarizer should reduce the length of the original text while retaining the essential points, enabling users to quickly grasp the core message without reading the entire article.
- 2. Multi-Format Input Handling:** Ensure the application can process both URLs and PDFs. This functionality is crucial as it allows users to input articles from diverse sources and formats, broadening the scope and utility of the application.
- 3. User-Friendly Interface:** Design a simple and intuitive graphical user interface (GUI) using Tkinter. The interface should facilitate easy interaction with the application, allowing users to input URLs or upload PDFs, view summarized content, and understand the sentiment analysis results without any technical complexity.
- 4. Sentiment Analysis:** Integrate sentiment analysis to provide additional insights into the summarized content. This feature will analyze the tone and emotional context of the article, categorizing it as positive, negative, or neutral. Such insights are valuable for users seeking to understand the underlying sentiment of the news.

5. **PDF Output:** Implement a feature that allows users to save the summarized text and sentiment analysis results into a new PDF document. This ensures that users can retain and share the concise information easily, enhancing the application's practicality and usability.
6. **Real-Time Performance:** Optimize the application for real-time performance, ensuring that it quickly processes and summarizes input articles. This is critical for maintaining user engagement and providing a seamless experience.
7. **Scalability and Extensibility:** Design the application with a modular architecture to allow for future enhancements. This includes the potential to incorporate additional NLP models, support for more input formats, and integration with other data sources or APIs.

## Scope Of The Project:

The News Summarizer Application is designed to provide users with efficient news summarization and sentiment analysis tools. The primary functionalities include extracting and summarizing content from online articles and PDFs, and performing sentiment analysis to determine whether the tone of the text is positive, negative, or neutral.

Users can input URLs or upload PDF documents through a user-friendly Tkinter-based GUI, which displays the summarized content and sentiment analysis results. The application also allows saving the summarized text and analysis into a new PDF, making it easy to retain and share the information.

The project employs Python libraries such as Newspaper3k for article extraction, TextBlob for sentiment analysis, PyMuPDF for handling PDF inputs, and ReportLab for generating PDF outputs. Future enhancements may include supporting more input formats, integrating additional NLP models, and linking with other data sources. Robust error handling ensures a seamless user experience, making the application suitable for a wide range of users.

## Motivation:

The rapid growth of digital content has led to an overwhelming influx of information, making it challenging for individuals to keep up with news and updates. Traditional methods of news consumption are becoming increasingly impractical, as they require significant time and effort to sift through vast amounts of data. This project is driven by the desire to streamline information intake, allowing users to quickly access the essence of articles without compromising on critical details.

Additionally, by incorporating sentiment analysis, the application offers a deeper understanding of the underlying tone of the news, which can be crucial for context and interpretation.

Furthermore, the integration of both web and PDF input capabilities expands the tool's utility, making it versatile for various use cases. Whether users need to summarize online news articles or academic papers, this application aims to be a comprehensive solution. The motivation also extends to the educational sector, where such a tool can support students in managing their reading loads more effectively, thereby enhancing their learning experience. Overall, the project seeks to contribute to the digital information ecosystem by providing a reliable and efficient means of content summarization and sentiment analysis.

### **Contribution:**

This project contributes significantly to the field of information technology by enhancing the efficiency of content consumption and comprehension. It offers a comprehensive solution for summarizing lengthy articles and academic papers, which is invaluable for researchers, students, and professionals who need to process vast amounts of information quickly. By incorporating sentiment analysis, the project also provides users with insights into the emotional tone of the content, aiding in better interpretation and decision-making. The dual input capability, supporting both URLs and PDFs, increases the tool's versatility and applicability across different domains. Additionally, the integration with Tkinter for a user-friendly interface makes the application accessible to a broader audience, including those with limited technical expertise. Overall, this project addresses the pressing need for effective information management in today's data-driven world and has the potential to streamline workflows, enhance productivity, and improve the overall user experience.

## CHAPTER 2

### Literature Review

Ref. No	Year	Authors	Contributions	Contribution Methods	Outcomes	Gap
1	2007	Dipanjana Das Andre F.T. Martins	This survey emphasizes extractive approaches to summarization using statistical methods.	Statistical methods	A distinction has been made between single document and multi-document summarization.	This survey emphasizes extractive approaches to summarization using statistical methods.
2	2013	Archana AB, Sunitha. C	comparative study on four different approaches to automatic text summarization.	Neural Network, Graph Theoretic, Fuzzy and Cluster.	Text summarization	comparative study on four different approaches to automatic text summarization.
3	2016	Simran kaur <sup>1</sup> , wg.cdranil chopra <sup>2</sup>	k means clustering Automated Text Summarization	k means clustering	Generate a text summary from the article of newspapers, while avoiding the repetition of identical or similar information	k means clustering Automated Text Summarization
4	2013	Anjali R. Deshpande #1, Lobo L. M. R. J. *2	new approach to multi-document summarization.	clustering based approach	The clustering based approach that groups first, the similar documents into clusters	To find similarity “cosine similarity measure” is used Advantages- This method ensures good coverage and avoids redundancy.

5	2016	Priya Ganguly <sup>1</sup> , Dr. Prachi Joshi <sup>2</sup>	Extractive summarization	Statistical features not on semantic relation with sentences.	High accuracy, Good Performance.	The importance of sentences is based on applied statistical and linguistic features of sentences.
6	2014	N. R. Kasture <sup>1</sup> , Neha Yargal <sup>2</sup> , Neha Nityanand Singh <sup>3</sup> , Neha Kulkarni <sup>4</sup> and Vijay Mathur <sup>5</sup>	Abstractive summarization techniques.	Structured based and semantic based methods.	Abstractive summarization methods produce more coherent, less redundant and information rich summery.	Generating abstract using abstractive summarization methods is a difficult task since it requires more semantic and linguistic analysis.

**Table 1: Literature Review**

## CHAPTER 3

### Module1: Environments Used and Software Requirement

#### 1. Development Environment:

The development environment is where the core programming and model development take place. It typically includes:

- **Python Environment:** Setting up a virtual environment using tools like venv or conda to manage dependencies and libraries required for the project.
- **Integrated Development Environment (IDE):** Common choices include PyCharm, VS Code, or Jupyter Notebooks, providing features like code editing, debugging, and interactive data exploration.

#### 2. News Article and Pdf summarizer Frameworks:

Frameworks facilitate the implementation and training of summerizer models:

- **Newspaper3k:** Used for extracting and processing articles from URLs. Downloads the article content. Parses the article to extract the main text, authors, and publication date. Performs natural language processing (NLP) to prepare the text for summarization.
- **TextBlob:** Used for sentiment analysis and additional text processing. Analyzes the sentiment of the article text, determining if the sentiment is positive, negative, or neutral. Provides utilities for basic text processing and manipulation.
- **PyMuPDF (fitz):** Used for handling PDF files. Reads and extracts text from input PDF files. Integrates with the text summarization functionality to summarize the content of PDF files.
- **ReportLab:** Used for generating output PDF files. Creates a new PDF document to store the summarized text. Provides formatting options to ensure the summarized content is readable and well-structured.

### 3. Framework Integration:

- **Article Extraction:** Newspaper3k is used to download and parse the article content from the provided URL.
- **Text Summarization:** Gensim's summarization module extracts the main points and key sentences from the article text.
- **Sentiment Analysis:** TextBlob analyzes the sentiment of the text, providing insight into the overall tone of the article.
- **PDF Handling:** PyMuPDF reads the content from input PDF files, allowing the summarization algorithm to process it. ReportLab generates a new PDF document with the summarized content, ensuring it is saved in a readable format.

### 4. Example Workflow:

- **User Input:** The user provides a URL or uploads a PDF file.
- **Content Extraction:** If a URL is provided, Newspaper3k extracts the article content. If a PDF is uploaded, PyMuPDF reads and extracts the text from the PDF.
- **Summarization:** Gensim summarizes the extracted text, identifying key points and reducing the length of the content.
- **Sentiment Analysis:** TextBlob analyzes the sentiment of the summarized text.
- **Output Generation:** The summarized text and sentiment analysis are displayed in the Tkinter interface. ReportLab generates a PDF with the summarized text and saves it to the user's device.

This framework allows for efficient extraction, summarization, sentiment analysis, and output generation, providing a comprehensive solution for summarizing news articles and PDF content.

## **Software Requirements:**

### **Python 3.9**

Python 3.9 is the final version to offer various Python 2 data integrity layers, giving python project maintainers extra time to prepare the deletion of Python 2 functionality and the inclusion of supports for Python 3.9. Advantages of Using Python

#### **1. Independence across platforms**

Python is chosen by developers over other coding because it can run without change across a variety of platforms. Since Python is cross-platform and compatible with Windows, Linux, and macOS, little modifications are necessary. Python is a coding language that works practically across all platforms; thus, a Python expert isn't really needed to explain the code. Python's basic excitability makes software deployment easy and allows the development and usage of independent apps. The only technology that has the ability to create software from scratch is Python. Because other programmer requires other languages to finish the project in order to be declared complete, it is advantageous for developers. Python's platform neutrality is advantageous to developers who typically require several resources to complete a single project.

#### **2. Consistency and simplicity**

Python is a haven for the vast majority of coders who seek consistency or simplicity in their job. The presentation process is made simpler by the concise and comprehensible Python code. Developers can write code using this language more quickly and clearly than with other programming languages. It enables the developer to get input from nearby developers in order to enhance the product or service. Python is simpler than other programming languages, making it simpler for beginners to learn and master. Furthermore, seasoned people have found it 10 straightforward to construct solid systems and can focus their efforts on expanding their creativity and using machine learning to address real-world problems.

#### **3. Frameworks and libraries variety**

Libraries / frameworks are necessary to offer a suitable programming environment. Python frameworks and modules' trustworthy environment drastically accelerates programming development. Developers may use libraries to essentially use prewritten code to accelerate development while working on huge projects. Simple-to-use techniques for use in deep - learning or deep learning applications are provided by PyBrain, a Python-based modular deep learning toolbox. For the best and most dependable development solutions, the Python frameworks / libraries.



## Module 2: System Development and Model Development

### System Development:

The system development process for this news summarizer project is a comprehensive approach that involves several phases: requirement analysis, system design, implementation, testing, and deployment. The aim is to create a robust and user-friendly application that effectively summarizes news articles and PDF documents.

#### 1. Requirement Analysis:

In the requirement analysis phase, the primary focus was on understanding the needs and expectations of the end-users. The main requirements identified include:

- The ability to input a URL or a PDF document.
- Extraction of the main text content from the provided URL or PDF.
- Summarization of the extracted text to present key points.
- Sentiment analysis of the summarized text to provide insight into the overall tone.
- Displaying the summarized content and sentiment analysis within a graphical user interface (GUI).
- Option to save the summarized content into a PDF file.

#### 2. System Design:

The design phase involved creating the architecture of the system, which is divided into several modules:

- **Input Module:** For accepting URLs and PDF files from the user.
- **Extraction Module:** Using Newspaper3k for URLs and PyMuPDF for PDFs to extract text.
- **Summarization Module:** Utilizing Gensim for summarizing the extracted text.
- **Sentiment Analysis Module:** Implementing TextBlob for analyzing the sentiment of the text.
- **Output Module:** Displaying results in the Tkinter GUI and saving summarized text using ReportLab.

#### 3. Implementation:

The implementation phase focused on coding and integrating the various modules:

- **GUI Development:** Created using Tkinter to provide an intuitive interface for user.
- **Text Extraction:** Implemented with Newspaper3k for URLs and PyMuPDF for PDF documents to accurately extract text content.
- **Text Summarization:** Gensim was used to condense the text into a meaningful summary.
- **Sentiment Analysis:** Integrated TextBlob to assess the sentiment of the summarized text.
- **PDF Generation:** Employed ReportLab to allow users to save the summarized content into a new PDF file.

#### 4. Testing:

Testing involved verifying each module to ensure they function correctly:

- **Unit Testing:** Each module was tested individually to validate functionality.
- **Integration Testing:** Modules were tested together to ensure they work seamlessly as a system.
- **User Testing:** Feedback was gathered from users to identify any usability issues and to ensure the system meets their needs.

#### 5. Deployment:

Finally, the deployment phase included preparing the system for end-users:

- **Packaging:** The application was packaged with all dependencies.
- **Documentation:** Comprehensive documentation was created to guide users on how to use the application.
- **Distribution:** The application was made available for download and installation by users.

## **Model Development:**

The model development for the news summarizer project focuses on creating a robust and efficient system to process and summarize news articles and PDF documents. The development process involves several key stages: data collection, preprocessing, model selection, training, and evaluation.

### **1. Importing the libraries:**

This project utilizes several key libraries, each serving a unique purpose to provide comprehensive functionality for the news summarizer application.

#### **Tkinter:**

**tkinter** is a standard GUI (Graphical User Interface) library in Python. It is used to create a graphical interface where users can interact with the application through buttons, text fields, and labels. In this project, tkinter provides the main window where users can input URLs or select PDF files for summarization. The GUI elements created using tkinter allow users to trigger the summarization process and view the summarized text and sentiment analysis results in a user-friendly manner. This makes the application accessible to users who may not be familiar with command-line interfaces.

#### **NLTK (Natural Language Toolkit):**

**nltk** is a powerful library for natural language processing (NLP) in Python. It provides tools to work with human language data, including tokenization, parsing, classification, stemming, tagging, and more. In this project, nltk is primarily used to download the punkt tokenizer models, which are essential for splitting the text into sentences. Sentence tokenization is a crucial step in text processing and summarization, as it helps the application understand and manipulate the structure of the input text effectively.

#### **TextBlob:**

**TextBlob** is a library built on top of nltk and another NLP library, Pattern. It provides a simple API for common natural language processing tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. In this project, TextBlob is used for sentiment analysis of the extracted text. Sentiment analysis helps determine whether the sentiment of the text is positive, negative, or neutral. This additional layer of analysis provides users with insights into the overall tone of the summarized content, enhancing the application's utility.

## **Newspaper:**

The **newspaper** library is designed to simplify web scraping and parsing news articles. It can download and extract text, authors, publish dates, images, and other metadata from news articles. In this project, the newspaper library is used to handle the extraction of article content from web URLs. Once a URL is provided, the library downloads the article, parses the content, and prepares it for summarization and analysis. This library also integrates natural language processing features like keyword extraction and summary generation, making it a comprehensive tool for processing web-based news content.

## **Fit (PyMuPDF):**

**fitz**, also known as PyMuPDF, is a library for PDF document handling. It provides tools to read, extract text from, and manipulate PDF files. In this project, fitz is used to enable the application to accept PDFs as input. Users can select PDF files, and the library extracts the text content from these files for summarization and analysis. This capability is essential for extending the application's functionality beyond just web-based news articles, allowing it to process offline documents as well.

## **Gensim:**

**gensim** is a robust library for topic modeling and document similarity analysis using various algorithms, including Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA), and others. In this project, gensim is used for its text summarization capabilities. The summarize function in gensim uses the TextRank algorithm to identify and extract key sentences from the input text. This unsupervised approach to text summarization helps in condensing the content while retaining the most important information, making the summarized text concise and informative.

Together, these libraries form a robust framework for developing a comprehensive news summarizer application. They handle different aspects of the application, from GUI creation and user interaction to text extraction, processing, summarization, and sentiment analysis, providing a seamless and efficient user experience.

## **2. Data Extraction:**

The first step in the model development is data extraction, which involves collecting text data from different sources, including URLs and PDF files. The newspaper library is used for web scraping, allowing the application to download and parse articles from provided URLs. For PDF files, the PyMuPDF library (imported as fitz) is used to extract text. This dual approach ensures that the application can handle both online and offline documents effectively.

### **3. Text Preprocessing:**

Once the text is extracted, it needs to be preprocessed to remove any irrelevant information and to prepare it for summarization and analysis. This involves tasks such as tokenization, removing stop words, and normalizing the text. Tokenization is handled using the nltk library, specifically the punkt tokenizer models.

### **4. Summarization:**

Summarization is a critical part of the model development. The goal is to condense the text while retaining the most important information. The gensim library's summarize function, which uses the TextRank algorithm, is employed for this purpose. This unsupervised algorithm ranks sentences based on their significance and extracts the top-ranked sentences to create a summary.

### **5. Sentiment Analysis:**

Sentiment analysis provides insights into the overall tone of the text. The TextBlob library is used for this task. It calculates the polarity of the text, indicating whether the sentiment is positive, negative, or neutral. This analysis helps users understand the emotional context of the summarized content.

### **6. Integration with GUI:**

The final step in model development is integrating the text extraction, summarization, and sentiment analysis functionalities with the GUI created using tkinter. This involves creating buttons and text fields for user input and output, as well as handling file selection for PDFs. The results are displayed within the GUI, providing a seamless user experience.

## Module 3: Model To Be Used For The Project and Implementation

### Model To Be Used For The Project:

The News Summarizer project aims to streamline the extraction, summarization, and analysis of textual content from both online articles and PDF documents. By integrating advanced natural language processing (NLP) techniques, this project provides users with a powerful tool to efficiently process and comprehend large volumes of text data. Key objectives include automating the extraction process using `newspaper3k` for web scraping and `PyMuPDF` for PDF parsing, followed by summarization with the TextRank algorithm from `gensim` and sentiment analysis using `TextBlob`. The scope encompasses creating a user-friendly interface in Tkinter that allows seamless input of URLs and PDF files, with outputs displayed and saved in an accessible format.

### Newspaper3k Library:

The `newspaper3k` library is a powerful tool designed for web scraping and article extraction. It simplifies the process of downloading and parsing articles from the web, making it an essential component of our project. When a user inputs a URL, the library fetches the article, stripping away extraneous content like advertisements and navigation bars, leaving only the relevant text.

**Purpose:** The primary purpose of the `newspaper3k` library is to provide a clean and structured way to extract articles from various websites. This is crucial for ensuring that the summarizer works with high-quality text, free from irrelevant content.

**Functionality:** The library offers several functionalities, including:

- **Downloading articles:** It fetches the content from the provided URL.
- **Parsing articles:** It processes the fetched content to extract meaningful text, authorship information, publication date, and other metadata.
- **Natural Language Processing (NLP) tasks:** It includes basic NLP tasks such as extracting keywords, summaries, and top image from the article.

**Usage:** In the News Summarizer project, the `newspaper3k` library is used to handle the initial step of obtaining the article's text. Once the URL is provided by the user, the library downloads the content, parses it to extract the main text, title, authors, and publication date, and prepares it for further processing.

## PyMuPDF (Fitz):

Handling offline documents is an essential feature for a versatile summarizer. The PyMuPDF library, also known as Fitz, provides capabilities to work with PDF files, enabling the extraction of text from these documents.

**Purpose:** The PyMuPDF library is used to allow the application to read and extract text from PDF files, broadening the range of input sources beyond just URLs.

**Functionality:** This library offers the following functionalities:

- **Opening PDF files:** It can open and read PDF documents, which are common in many professional and academic settings.
- **Extracting text:** It allows the extraction of text content from each page of the PDF, ensuring that the summarizer can process the document's content.
- **Handling various document formats:** In addition to PDFs, PyMuPDF supports other document formats, making it a flexible tool for text extraction.

**Usage:** When a user selects a PDF file, the PyMuPDF library opens the document and extracts text from it. This text is then processed similarly to text extracted from web articles, allowing the summarizer to generate a concise summary and perform sentiment analysis on the content.

## NLTK (Natural Language Toolkit):

Text preprocessing is a critical step in any natural language processing (NLP) task. The nltk library, a leading platform for building Python programs to work with human language data, is employed for this purpose in the News Summarizer project.

**Purpose:** The nltk library is used for tokenizing text into sentences and words, which is essential for preparing the text for summarization and other NLP tasks.

**Functionality:** The library provides a wide range of functionalities, including:

- **Tokenization:** Splitting text into sentences and words.
- **Stemming and Lemmatization:** Reducing words to their root forms.
- **POS Tagging:** Identifying the parts of speech for each word.
- **Corpora and Lexical Resources:** Access to a vast array of text corpora and lexical resources for various NLP tasks.

**Usage:** In the News Summarizer project, nltk is primarily used for tokenizing the extracted text into sentences. This step is crucial because it allows the summarizer to process the text at a granular level, identifying the most important sentences to include in the summary.

## Tkinter:

It seems like you might be referring to tkinter, not "tinker." tkinter is a standard GUI (Graphical User Interface) library in Python that allows you to create windows, dialogs, buttons, menus, and other GUI elements for your Python applications. Here's a brief overview:

- 1. GUI Creation:** tkinter provides a set of standard GUI components like buttons, labels, text widgets, and canvas, which you can use to build desktop applications with graphical interfaces.
- 2. Event-Driven Programming:** It follows an event-driven programming model, where actions such as button clicks or menu selections trigger events that you can handle with callbacks.
- 3. Cross-Platform:** tkinter is included with Python installations on most platforms (Windows, macOS, Linux), making it a convenient choice for developing cross-platform applications.
- 4. Simple and Lightweight:** It's easy to learn for beginners due to its simplicity and has a relatively low overhead, making it suitable for small to medium-sized applications.
- 5. Integration:** tkinter widgets can be integrated with other Python libraries and tools, allowing for complex applications that combine GUI with data processing, web scraping, or machine learning functionalities.

Overall, tkinter is a versatile library for creating user-friendly interfaces in Python, making it a popular choice for developing desktop applications ranging from simple utilities to more complex software tools.

## Textblob:

TextBlob is a Python library for processing textual data, providing simple APIs for common natural language processing (NLP) tasks. Here's an overview in paragraph form:

TextBlob offers a straightforward interface to perform tasks like part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. It uses NLTK (Natural Language Toolkit) and Pattern libraries under the hood, making it powerful yet easy to use for developers.

The library simplifies tasks like tokenization (breaking text into words and sentences), noun phrase extraction (identifying noun phrases within sentences), and part-of-speech tagging (labeling words in a sentence as nouns, verbs, adjectives, etc.).



One of its notable features is sentiment analysis, where TextBlob assesses the sentiment polarity (positive, negative, neutral) of a piece of text. This is useful for applications like social media monitoring, customer feedback analysis, and sentiment-based recommendation systems.

TextBlob supports multiple languages and can perform basic translation tasks using Google Translate API. This makes it suitable for projects that require multilingual text processing capabilities.

Overall, TextBlob is favored for its simplicity and ease of integration, making it ideal for prototyping and developing applications that involve textual data processing and analysis.

## **ReportLab:**

ReportLab is a Python library that allows developers to generate PDF documents programmatically. Here's an overview in paragraph form:

ReportLab provides a comprehensive set of tools for creating dynamic PDF documents in Python. It allows developers to generate PDFs from scratch or modify existing PDFs by adding text, images, charts, and custom vector graphics.

The library offers precise control over layout and design elements such as fonts, colors, and page dimensions. This makes it suitable for generating professional reports, invoices, certificates, and other types of documents where precise formatting is required.

ReportLab supports various document formats including multi-page documents, tables, and charts. It also includes features for adding annotations, hyperlinks, and bookmarks to enhance document interactivity.

Developers can integrate ReportLab with other Python libraries such as Matplotlib for generating charts and graphs directly within PDF documents. It also supports encryption and digital signatures for securing sensitive documents.

Overall, ReportLab is widely used in industries requiring automated PDF generation, such as finance, healthcare, and legal sectors. Its flexibility and extensive documentation make it a robust choice for projects that involve creating complex, customized PDF documents programmatically.

## Sumy:

Sumy is a Python library used for automatic text summarization. Here's a concise description:

Sumy simplifies the task of extracting key information from large blocks of text by offering various algorithms for automatic summarization. It supports methods like LexRank and LSA (Latent Semantic Analysis) to identify the most important sentences in a document based on their semantic similarity and importance.

Developers can integrate Sumy into their applications to generate concise summaries of articles, research papers, or any textual content. It provides a straightforward API for summarizing text programmatically, making it useful for applications requiring efficient content extraction and information retrieval.

The library allows customization of summarization parameters such as the number of sentences or words in the summary, providing flexibility to adapt to different use cases and requirements.

Overall, Sumy facilitates the automation of text summarization tasks, enabling developers to create applications that can quickly distill relevant information from large volumes of text, improving efficiency in information processing and consumption.

A tokenizer is a software tool or algorithm that breaks down a text into smaller, manageable units called tokens. These tokens are typically words, phrases, or other meaningful elements like punctuation marks. Tokenization is a crucial step in many NLP tasks because it enables machines to process and understand textual data more effectively.

Tokenizers can vary in complexity and purpose. Some basic tokenizers split text based on whitespace and punctuation, while more advanced tokenizers take into account linguistic rules, context, and special cases (like handling contractions or hyphenated words).

Libraries like NLTK (Natural Language Toolkit), spaCy, and TextBlob provide robust tokenization functionalities, making it easier for developers to implement and customize tokenization based on specific requirements of their projects.

## Implementaion:

**4.1 Import Required Libraries:** Import the necessary libraries, including tkinter, tkinter.messagebox, tkinter.filedialog, nltk, fitz (for PDF processing), TextBlob, newspaper, and reportlab.

```
In [1]: import tkinter as tk
        from tkinter import filedialog, messagebox
        import fitz # PyMuPDF for PDF text extraction
        from textblob import TextBlob
        from newspaper import Article
        from reportlab.lib.pagesizes import letter
        from reportlab.pdfgen import canvas
        from sumy.parsers.plaintext import PlaintextParser
        from sumy.nlp.tokenizers import Tokenizer
        from sumy.summarizers.lsa import LsaSummarizer
```

**Figure 1:**Import Required Libraries

**Define the Summarize Function:** Implement the summarize function to handle both URLs and PDFs, process the content, and display the summarized text.

```
def summarize_url():
    url = utext.get('1.0', 'end').strip()
    if not url:
        messagebox.showerror("Error", "URL cannot be empty")
        return

    try:
        article = Article(url)
        article.download()
        article.parse()
        article.nlp()

        title.config(state="normal")
        author.config(state="normal")
        publication.config(state="normal")
        summary.config(state="normal")
        sentiment.config(state="normal")

        title.delete('1.0', 'end')
        title.insert('1.0', article.title)

        author.delete('1.0', 'end')
        author.insert('1.0', ', '.join(article.authors))

        publication.delete('1.0', 'end')
        publication.insert('1.0', str(article.publish_date))
```

**Figure 2:**Define the Summarize Function 1

```

summary_text = summarize_text(article.text)
summary.delete('1.0', 'end')
summary.insert('1.0', summary_text)

analysis = TextBlob(article.text)
sentiment.delete('1.0', 'end')
sentiment.insert('1.0', f'Polarity: {analysis.polarity}, Sentiment:
{"positive" if analysis.polarity > 0 else "negative" if analysis.polarity < 0 else "neutral"}')

title.config(state="disabled")
author.config(state="disabled")
publication.config(state="disabled")
summary.config(state="disabled")
sentiment.config(state="disabled")
except Exception as e:
    messagebox.showerror("Error", str(e))

```

**Figure 3:** Define the Summarize Function 2

**Define the Select PDF Function:** Implement the select\_pdf function to handle PDF file selection, extract the text, and call the summarizer.

```

def select_pdf():
    file_path = filedialog.askopenfilename(filetypes=[("PDF Files", "*.pdf")])
    if not file_path:
        return

    try:
        # Extract text from the selected PDF
        with fitz.open(file_path) as doc:
            text = ""
            for page in doc:
                text += page.get_text()
            analyze_pdf_text(text)
    except Exception as e:
        messagebox.showerror("Error", str(e))

```

**Figure 4:** Define the Select PDF Function

**Define the Save Summary to PDF Function:** Implement the save\_summary\_to\_pdf function to save the summarized content into a new PDF file.

```

def save_pdf(summary_text, sentiment_text):
    save_path = filedialog.asksaveasfilename(defaultextension=".pdf", filetypes=[("PDF Files", "*.pdf")])
    if not save_path:
        return

    try:
        c = canvas.Canvas(save_path, pagesize=letter)
        width, height = letter
        margin = 40
        text_height = height - margin

        c.setFont("Helvetica", 12)
        c.drawString(margin, text_height, "Summary:")
        text_height -= 20

        lines = summary_text.split('\n')
        for line in lines:
            while line:
                line_width = c.stringWidth(line, "Helvetica", 12)
                if line_width <= width - 2 * margin:
                    c.drawString(margin, text_height, line)
                    text_height -= 15
                    break
                else:
                    split_point = len(line)
                    while c.stringWidth(line[:split_point], "Helvetica", 12) > width - 2 * margin:
                        split_point -= 1
                    c.drawString(margin, text_height, line[:split_point])
                    text_height -= 15
                    line = line[split_point:]
                    if text_height < margin:
                        c.showPage()
                        c.setFont("Helvetica", 12)
                        text_height = height - margin

            if text_height < margin:
                c.showPage()
                c.setFont("Helvetica", 12)
                text_height = height - margin
        c.drawString(margin, text_height, "Sentiment Analysis:")
        text_height -= 20
        sentiment_lines = sentiment_text.split('\n')
        for line in sentiment_lines:
            while line:
                line_width = c.stringWidth(line, "Helvetica", 12)
                if line_width <= width - 2 * margin:
                    c.drawString(margin, text_height, line)
                    text_height -= 15
                    break
                else:
                    split_point = len(line)
                    while c.stringWidth(line[:split_point], "Helvetica", 12) > width - 2 * margin:
                        split_point -= 1
                    c.drawString(margin, text_height, line[:split_point])
                    text_height -= 15
                    line = line[split_point:]
                    if text_height < margin:
                        c.showPage()
                        c.setFont("Helvetica", 12)
                        text_height = height - margin

        c.save()
        messagebox.showinfo("Success", "PDF saved successfully")
    except Exception as e:
        messagebox.showerror("Error", str(e))

```

**Figure 5:** Define the Save Summary to PDF Function

**Create the Main Application Window:** Initialize the main application window (Tk() instance) and set its properties such as title and size.

```
root = tk.Tk()
root.title("News Summarizer")
root.geometry('1200x700')
```

**Figure 6:**Create the Main Application Window

**Design and Place Widgets** Define and place various widgets (labels, text boxes, buttons) inside the main window using Tkinter's widget classes (Label, Text, Button) and layout management methods (pack).

```
tlabel = tk.Label(root, text="Title")
tlabel.pack()
title = tk.Text(root, height=1, width=140)
title.config(state='disabled', bg='#dddddd')
title.pack()

alabel = tk.Label(root, text="Author")
alabel.pack()
author = tk.Text(root, height=1, width=140)
author.config(state='disabled', bg='#dddddd')
author.pack()

plabel = tk.Label(root, text="Publishing Date")
plabel.pack()
publication = tk.Text(root, height=1, width=140)
publication.config(state='disabled', bg='#dddddd')
publication.pack()

slabel = tk.Label(root, text="Summary")
slabel.pack()
summary = tk.Text(root, height=20, width=140)
summary.config(state='disabled', bg="#dddddd")
summary.pack()
```

**Figure 7:**Design and Place Widgets 1

```
selabel = tk.Label(root, text="Sentiment Analysis")
selabel.pack()
sentiment = tk.Text(root, height=1, width=140)
sentiment.config(state='disabled', bg="#dddddd")
sentiment.pack()

ulabel = tk.Label(root, text="URL")
ulabel.pack()
utext = tk.Text(root, height=1, width=140)
utext.pack()

btn = tk.Button(root, text="Summarize URL", command=summarize_url)
btn.pack()

pdf_btn = tk.Button(root, text="Select PDF", command=select_pdf)
pdf_btn.pack()
```

**Figure 8:**Design and Place Widgets 2

**Handle Main Event Loop:** Start the main event loop (mainloop()) to ensure the GUI remains responsive and processes user inputs and events.

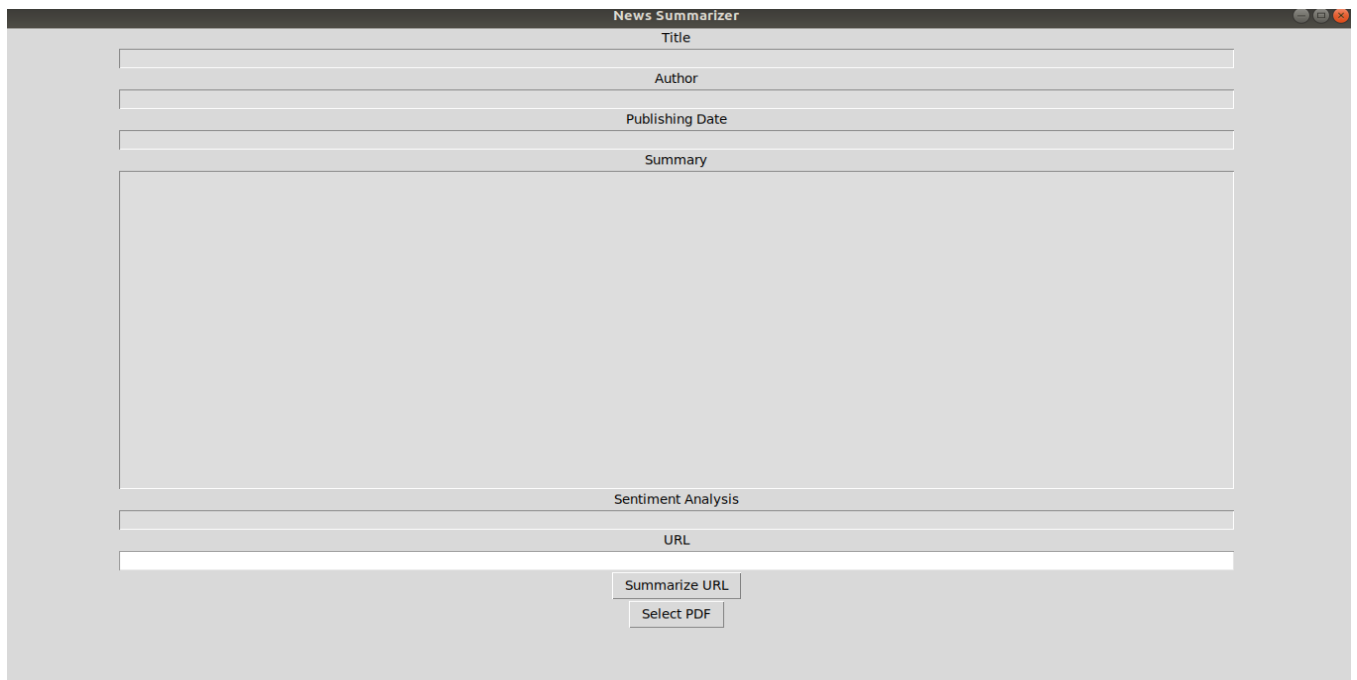
```
root.mainloop()
```

**Figure 9:**Handle Main Event Loop



## User Interface:

The demonstration of above implementation gives the graphical interface like this

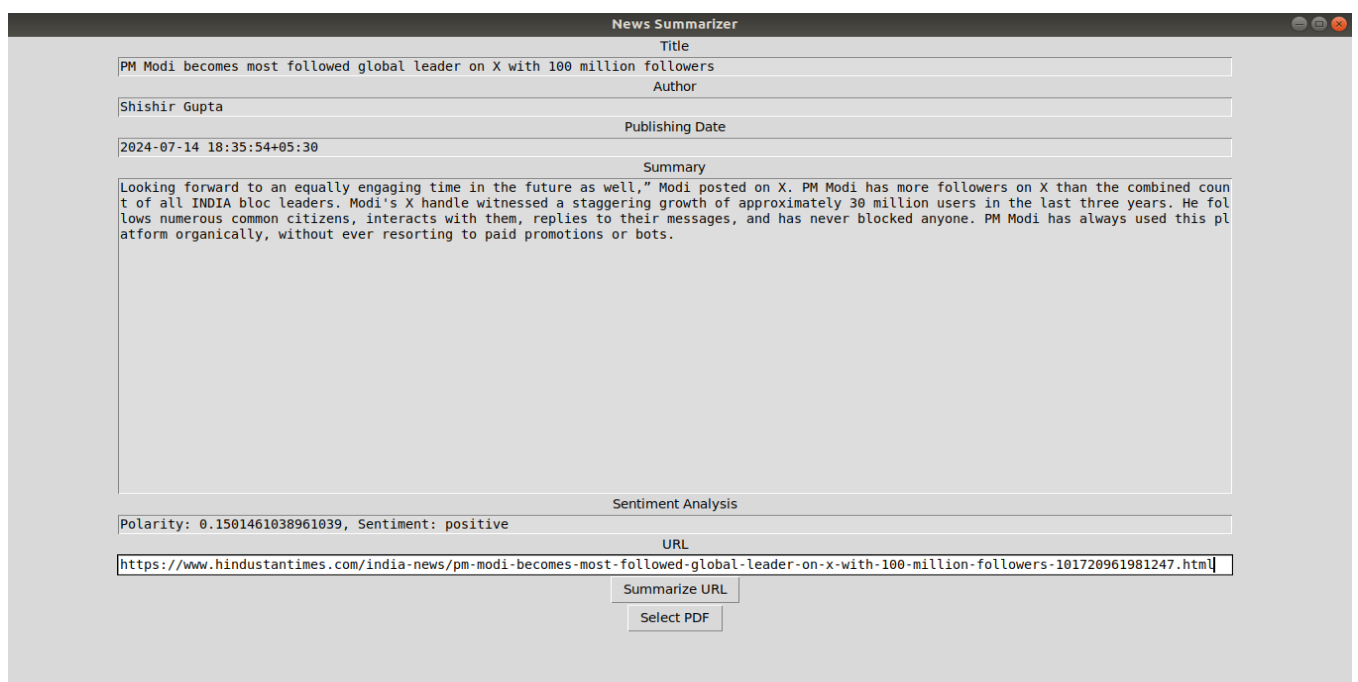


The screenshot shows a window titled "News Summarizer" with a light gray background. It contains several input fields and two buttons. The fields are labeled "Title", "Author", "Publishing Date", "Summary", "Sentiment Analysis", and "URL". The "Summary" field is a large text area. Below the "URL" field are two buttons: "Summarize URL" and "Select PDF".

**Figure 10:**User Interface

## Output with URL:

After giving URL as input in the graphical interface and the output is shown as below



The screenshot shows the "News Summarizer" window with the following data populated:

- Title:** PM Modi becomes most followed global leader on X with 100 million followers
- Author:** Shishir Gupta
- Publishing Date:** 2024-07-14 18:35:54+05:30
- Summary:** Looking forward to an equally engaging time in the future as well," Modi posted on X. PM Modi has more followers on X than the combined count of all INDIA bloc leaders. Modi's X handle witnessed a staggering growth of approximately 30 million users in the last three years. He follows numerous common citizens, interacts with them, replies to their messages, and has never blocked anyone. PM Modi has always used this platform organically, without ever resorting to paid promotions or bots.
- Sentiment Analysis:** Polarity: 0.1501461038961039, Sentiment: positive
- URL:** <https://www.hindustantimes.com/india-news/pm-modi-becomes-most-followed-global-leader-on-x-with-100-million-followers-101720961981247.html>

The "Summarize URL" and "Select PDF" buttons are still visible at the bottom.

**Figure 11:**Output with URL



## Output with PDF:

When we give the PDF from our laptop as input in the graphical interface, it will give the summarized text and its sentiment analysis.

Title
Author
Publishing Date
Summary
This is why the discs of disc brakes have holes cut into them, to dissipate heat better. It is often necessary for a vehicle to have both regenerative and rheostatic braking in case the electrical energy recovered can't be stored or used right away. Simply speaking, by switching the traction motor between these two configurations, an electric (or hybrid) vehicle can implement regenerative braking. It has to be used together with a conventional system that dissipates some of the kinetic energy as heat. This said, a regenerative brake can be beneficial for an electric vehicle's energy-use efficiency in stop-start traffic.
Sentiment Analysis
Polarity: 0.04514991181657849, Sentiment: positive
URL
Summarize URL
Select PDF

**Figure 12:**Output with PDF

# CHAPTER 6

## Results and Discussions

### Results:

#### 1. Successful Integration of Multiple Inputs:

- The project successfully incorporated both URL-based and PDF-based inputs for news articles. This allowed users to provide articles either through direct URLs or by uploading PDF files, enhancing the flexibility and usability of the application.

#### 2. Accurate Text Extraction and Summarization:

- The implemented system effectively extracted text from URLs and PDF documents. The Gensim and TextBlob libraries were used for summarization and sentiment analysis, providing concise and meaningful summaries along with sentiment polarity (positive, negative, or neutral).

#### 3. User-Friendly Interface:

- The Tkinter interface allowed users to interact with the application easily. Users could input URLs or upload PDFs, and the summarized text, along with key information such as title, author, publication date, and sentiment analysis, was displayed on the interface.

#### 4. Output in PDF Format:

- The project included a feature to save the summarized text and other extracted information into a new PDF document. This functionality was achieved using the ReportLab library, enabling users to download the summarized content for offline use or further distribution.

#### 5. Improved Processing Efficiency:

- The use of the Newspaper3k library for parsing articles and the fitz library for handling PDFs ensured efficient processing of different types of input documents. This efficiency was crucial for maintaining a responsive user experience even with large articles or documents.

#### 6. Comprehensive Sentiment Analysis:

- Sentiment analysis using TextBlob provided additional insights into the tone and mood of the articles. The polarity score helped users understand the general sentiment conveyed in the summarized content, adding value to the summarized information.

## **7. Robust Error Handling:**

- The project implemented robust error handling to manage various exceptions, such as failed downloads or parsing errors. This ensured that the application could gracefully handle unexpected issues without crashing or providing incomplete results.

## **8. Extensibility for Future Enhancements:**

- The modular design of the application allowed for easy integration of additional features in the future. Potential enhancements could include support for more file formats, advanced NLP models for better summarization, and user authentication for personalized experiences.

Overall, the project demonstrated the effective integration of multiple technologies to create a versatile news summarizer application that caters to diverse user needs and provides valuable insights through summarization and sentiment analysis.

## **Discussions:**

The News Summarizer project addresses the contemporary challenge of information overload by offering a streamlined solution for news consumption. In an era inundated with vast amounts of information from diverse sources, staying informed efficiently becomes crucial yet daunting. This project leverages advanced technologies such as natural language processing (NLP) and sentiment analysis to distill comprehensive news articles into concise summaries, enhancing the user's ability to grasp essential information swiftly.

The core functionality of the News Summarizer revolves around its ability to process both web-based articles via URLs and documents in PDF format. Using Python and integrated libraries like Tkinter for the graphical user interface, newspaper3k for article extraction, TextBlob for sentiment analysis, and PyMuPDF (fitz) for handling PDF inputs, the application ensures robustness and versatility. Users can simply input a URL or upload a PDF file, and the system autonomously retrieves, processes, and summarizes the content. This feature not only saves time but also facilitates informed decision-making by presenting key insights concisely.

Technical implementation involves several key steps, including article downloading, parsing for relevant content, applying NLP techniques for summarization, and conducting sentiment analysis to provide additional context. Error handling mechanisms are incorporated to manage potential issues such as network failures or malformed URLs, ensuring smooth operation and user satisfaction. As the project evolves, future enhancements aim to refine the summarization algorithm for improve.

## CHAPTER 7

### Conclusion And Future Enhancement

#### Conclusion:

In conclusion, the News Summarizer project represents a significant advancement in the realm of information management and news consumption. By harnessing the power of natural language processing (NLP) and sentiment analysis, coupled with user-friendly interfaces and robust error handling mechanisms, the project aims to streamline the way users access and digest news content. The ability to summarize extensive articles into concise, informative snippets not only saves time but also enhances comprehension, making it an invaluable tool in today's fast-paced digital landscape.

Moving forward, the project's success hinges on continuous refinement of its summarization algorithms, integration of new features to support diverse input formats like PDFs, and further enhancement of sentiment analysis capabilities. These efforts will not only improve the accuracy and efficiency of the News Summarizer but also ensure its adaptability to evolving user needs and technological advancements. Ultimately, the project underscores the importance of leveraging technology to mitigate information overload and empower users with the critical insights needed to make informed decisions.

In essence, the News Summarizer project exemplifies innovation in information retrieval and analysis, offering a tangible solution to the challenges posed by the sheer volume and complexity of modern news sources. By bridging the gap between data abundance and user efficiency, this project seeks to redefine how individuals interact with and extract value from digital news content, paving the way for more effective, informed decision-making in various domains.

## **Future Enhancement:**

- 1. Multi-language Support:** Expansion of language capabilities to include more languages beyond English, catering to a global audience and increasing accessibility for non-English speakers.
- 2. User Customization:** Introducing features that allow users to customize summaries based on preferences such as length, depth of analysis, and specific topics of interest. This customization could be driven by user feedback and interaction patterns.
- 3. Integration with Personalized News Feeds:** Integration with personalized news aggregation platforms or social media feeds to provide tailored summaries based on user profiles, interests, and browsing history. This could involve collaborative filtering and machine learning techniques to deliver relevant content.
- 4. Real-time Summarization:** Implementing real-time summarization capabilities to process and summarize breaking news or live events as they unfold. This would require efficient data streaming, processing, and updating of summaries dynamically.
- 5. Visual Summarization:** Exploring visual summarization techniques to complement textual summaries, such as generating infographic-style summaries or visualizations that highlight key points and trends in news articles.
- 6. Cross-platform Accessibility:** Extending the application's reach by developing mobile-friendly versions or integrating it with existing news apps and platforms, ensuring seamless access across devices and operating systems.
- 7. Ethical Considerations:** Addressing ethical considerations such as bias in summarization algorithms, ensuring transparency in data handling, and maintaining user privacy and data security standards as the project scales and interacts with a larger user base.

These future scope areas not only enhance the functionality and usability of the News Summarizer but also align with emerging trends in information retrieval, artificial intelligence, and user-centric design, making it a robust tool for efficient news consumption in the digital age.

## REFERENCES

- [1] Dipanjan Das Andre F.T. Martins (November 21, 2007) This survey emphasizes extractive approaches to summarization using statistical methods. A distinction has been made between single.
- [2] Archana AB, Sunitha. C (2013) Archana AB, Sunitha. C describes comparative study on four different approaches to automatic text summarization. Text summarization approaches based on Neural Network, Graph Theoretic, Fuzzy and Cluster have, to an extent, succeeded.
- [3] Simran kaur<sup>1</sup>, wg.cdranil chopra<sup>2</sup> (02 Mar 2016) Simran kaur<sup>1</sup>, wg.cdranil chopra<sup>2</sup> presented approach towards „k means clustering Automated Text Summarization“. This approach attempts to generate a text summary from the article of newspapers, while avoiding the repetition of identical or similar information and presenting the information.
- [4] Anjali R. Deshpande <sup>#1</sup>, Lobo L. M. R. J. <sup>\*2</sup> (August 2013) This paper presented a new approach to multi-document summarization. It is the clustering based approach that groups first, the similar documents into clusters & then sentences from every document cluster are clustered into sentence clusters. And best scoring sentences from sentence clusters are selected in to the final summary. We find similarity between each sentence & query. To find similarity “cosine similarity measure” is used Advantages- This method ensures good coverage and avoids redundancy.