VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", MACHHE, BELAGAVI-590018

ML Mini Project Report

on

TEXT SUMMARIZATION USING GPT MODELS

Submitted in partial fulfillment of the requirements for the VI semester

Bachelor of Engineering

in

Artificial Intelligence & Machine Learning

of

Visvesvaraya Technological University, Belagavi

By

Bheemambika B(1CD21AI006)

Inchara V (1CD21AI019)

Under the Guidance

of

Dr. Varalatchoumy.M,

Prof. Syed Hayath,

Dept. of AI&ML

Department of Artificial Intelligence & Machine Learning

CAMBRIDGE INSTITUTE OF TECHNOLOGY, BANGALORE-560 036

2023-2024

CAMBRIDGE INSTITUTE OF TECHNOLOGY

K.R. Puram, Bangalore-560 036

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

CERTIFICATE

Certified that **Ms. Bheemambika** bearing USN **1CD21AI006** and **Ms. Inchara** bearing USN **1CD21AI019,** a bonafide student of **Cambridge Institute of Technology,** has successfully completed the ML Mini Project entitled **"TEXT SUMMARIZATION USING GPT MODELS"** in partial fulfillment of the requirements for VI semester **Bachelor of Engineering** in **Artificial Intelligence & Machine Learning** of **Visvesvaraya Technological University, Belagavi** during academic year 2023-24. It is certified that all Corrections/Suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The ML Mini Project report has been approved as it satisfies the academic requirements prescribed for the Bachelor of Engineering degree.

Mini Project Guides,	**Head of the Department,**
	Dr. Varalatchoumy. M
	Dept. of AI&ML, CITech
Dr. Varalatchoumy.M	
Prof. Syed Hayath	
Dept. of AI&ML, CITech	

DECLARATION

We, Bheemambika and Inchara of VI semester BE, Artificial Intelligence & Machine Learning, Cambridge Institute of Technology, hereby declare that the mini project entitled **"TEXT SUMMARIZATION USING GPT MODELS"** has been carried out by us and submitted in partial fulfillment of the course requirements of VI semester **Bachelor of Engineering** in **Artificial Intelligence & Machine Learning** as prescribed b**y Visvesvaraya Technological University, Belagavi**, during the academic year 2023-2024.

We also declare that, to the best of our knowledge and belief, the work reported here does not form part of any other report on the basis of which a degree or award was conferred on an earlier occasion on this by any other student.

Date:
Place: Bangalore

Bheemambika B
1CD21AI006
Inchara V
1CD21AI019

ACKNOWLEDGEMENT

We would like to place on record of our deep sense of gratitude to **Shri. D. K. Mohan,** Chairman, Cambridge Group of Institutions, Bangalore, India for providing excellent Infrastructure and Academic Environment at CITech without which this work would not have been possible**.**

We are extremely thankful to **Dr. G. Indumathi,** Principal, CITech, Bangalore, for providing us the academic ambience and everlasting motivation to carry out this work and shaping our careers.

We express our sincere gratitude to **Dr. Varalatchoumy M.,** Prof. & Head, Dept. of Artificial Intelligence & Machine Learning, CITech, Bangalore, for her stimulating guidance, continuous encouragement and motivation throughout the course of present work.

We also wish to extend our thanks to Mini Project Guides, **Dr. Varalatchoumy M.,** Prof & Head and **Prof. Syed Hayath** Assistant Professors, Dept. of AI&ML, CITech, Bangalore for the critical, insightful comments, guidance and constructive suggestions to improve the quality of this work.

Finally, to all our friends, classmates who always stood by us in difficult situations also helped us in some technical aspects and last but not the least, we wish to express deepest sense of gratitude to our parents who were a constant source of encouragement and stood by us as pillar of strength for completing this work successfully.

Bheemambika B

Inchara V

ABSTRACT

“Text Summarization Using GPT Models” is a critical task in natural language processing (NLP) that involves condensing a long text into a shorter version while preserving its main ideas and key information. This project explores the use of Generative Pre-trained Transformer (GPT) models for text summarization. GPT models, particularly those based on the GPT-3 and GPT-4 architectures, have shown remarkable capabilities in generating human-like text and understanding context, making them well-suited for summarization tasks. The project begins with an overview of the evolution of text summarization techniques, from extractive methods that select important sentences directly from the source text to abstractive methods that generate new sentences based on the source content. Extractive methods, while straightforward, often fail to produce coherent summaries. Abstractive methods, on the other hand, can generate more natural and concise summaries but pose greater challenges in maintaining factual accuracy and coherence. The core of this project is the application of GPT models for abstractive summarization. GPT models are fine-tuned on a diverse range of text summarization datasets to enhance their ability to generate high-quality summaries. Techniques such as beam search, nucleus sampling, and reinforcement learning are employed to improve the generation process. The project also addresses the challenges of handling long documents, maintaining summary coherence, and reducing redundancy. Evaluation of the summarization models is conducted using standard metrics like ROUGE (Recall-Oriented Understudy for Gisting Evaluation), BLEU (Bilingual Evaluation Understudy), and human judgment.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND REQUIREMENTS

1.1 Background

The "Text Summarization Using GPT Models" project leverages advanced generative AI to automate the process of summarizing lengthy texts into concise and informative summaries. With the increasing volume of digital information, this tool aims to enhance information accessibility and usability by extracting key information from extensive documents. The project utilizes GPT models, known for their ability to generate human-like text, ensuring that the summaries are coherent and informative. The advancement of machine learning and NLP, particularly with the advent of Generative Pre-trained Transformers (GPT), has revolutionized text summarization. GPT models, developed by OpenAI, have demonstrated remarkable capabilities in understanding and generating human-like text. These models leverage vast amounts of data and sophisticated neural network architectures to generate coherent, contextually relevant, and concise summaries. Text summarization has been an active area of research in the field of natural language processing (NLP) for several decades. The primary goal of text summarization is to reduce the length of a text document while preserving its key information and overall meaning. This task is essential in today's information-rich world, where the ability to quickly digest large volumes of text is highly valuable.

1.2 Why

The explosion of digital content has led to an overwhelming amount of information available online and in various digital repositories. This abundance of information makes it challenging for individuals and organizations to quickly access and digest the key points of lengthy documents, articles, and reports. Text summarization, which aims to condense long texts into shorter versions that still convey the main ideas, has emerged as a crucial solution to this problem. The project of using GPT models for text summarization addresses several needs and leverages the latest advancements in natural language processing (NLP) to improve the summarization process.

The "Text Summarization Using GPT Models" project leverages advanced generative AI to automate the process of summarizing lengthy texts into concise and informative summaries. With the increasing volume of digital information, this tool aims to enhance information accessibility and usability by extracting key information from extensive documents. The project utilizes GPT models, known for their ability to generate human-like text, ensuring that the summaries are coherent and informative. The advancement of machine learning and NLP, particularly with the advent of Generative Pre-trained Transformers (GPT), has revolutionized text summarization. GPT models, developed by OpenAI, have demonstrated remarkable capabilities in understanding and generating human-like text. These models leverage vast amounts of data and sophisticated neural network architectures to generate coherent, contextually relevant, and concise summaries.

1.3 Problem Statement

“Develop an advanced text summarization tool leveraging GPT models to condense large texts into brief, accurate summaries.”

1.4 Objectives

1. Develop Automated Summaries:
    - Utilize GPT models to generate concise, coherent, and informative summaries that retain essential information from lengthy texts.
2. Create an Intuitive User Interface:
    - Implement a user-friendly Streamlit-based web application that allows easy interaction for inputting text, uploading PDFs, and selecting summarization styles.
3. Offer Multiple Summarization Styles:
    - Provide users with various summarization options, including Brief, Detailed, and Key Points, to cater to different needs and preferences.
4. Enable Interactive Querying:
    - Allow users to ask questions about the summarized text and receive accurate, contextually relevant answers, enhancing information accessibility and usability.

CHAPTER 2

LITERATURE SURVEY

2.1 Building Customized Chatbots for Document Summarization and Question Answering using Large Language Models using a Framework with OpenAI, LangChain, and Streamlit

Authors: Sangita Pokhrel, Swathi Ganesan, Tasnim Akther, Lakmali Shashika Karunarathne Mapa Senavige

Journal: Journal of Information Technology and Digital World

Publication Year: 2024

Summary: The paper presents a comprehensive framework for building customized chatbots powered by large language models (LLMs) to perform document summarization and question-answering tasks. The framework leverages technologies such as OpenAI, LangChain, and Streamlit to help users efficiently extract insights from lengthy documents. The study details the framework's architecture, implementation, and practical applications, providing a step-by-step guide for developers to create end-to-end document summarization and question-answering applications.

Key Points:

1. **Framework Components:**
2. OpenAI: Utilized for its GPT models, providing the core language understanding capabilities necessary for summarization and question-answering.
3. LangChain: Enhances the GPT models' capabilities by offering a framework for performing linguistic processing tasks efficiently.
4. Streamlit: Serves as the user interface foundation, allowing developers to create intuitive and interactive chatbot interfaces.

2. Challenges Addressed:

- Information overload due to the exponential growth of digital information.
- The need for rapid extraction of important insights from large amounts of text.

3. Applications:

- Custom chatbots for various domains such as customer service and educational assistance.
- Enhancing productivity and facilitating information retrieval.

4.Methodology:

- Step-by-step guide for creating chatbots.
- Integration of OpenAI, LangChain, and Streamlit to streamline the development process.

Relevance to the Project:

This paper is highly relevant to the project on text summarization using GPT models as it provides a practical framework for developing chatbots that can effectively summarize documents and answer questions. The use of OpenAI's GPT models aligns with the project's goals, offering insights into integrating these models with other tools like LangChain and Streamlit to enhance functionality and user interaction. The detailed methodology and practical applications discussed in the paper can serve as a valuable guide for implementing similar solutions in the project.

2.2 Text Summarization using NLP Technique

Authors: Balaji N., Megha N., Deepa Kumari, Sunil Kumar P., Bhavatarini N., Shikah Rai A.
Journal: Conference Paper at International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)
Publication Year: 2022

Summary: The paper discusses automated text summarization techniques using natural language processing (NLP). The rapid increase in online text data necessitates efficient methods for generating summaries, which can be done manually or automatically. The study focuses on automated summarization due to its efficiency and accuracy advantages. It evaluates two primary methods: extractive and abstractive summarization. Extractive summarization involves selecting significant sentences directly from the source text, while abstractive summarization interprets and generates a new summary that conveys the core message of the original text. The paper utilizes the BBC news dataset to compare these techniques using transformer-based pre-trained models.

Key Points:

1.Increasing Need for Summarization: The proliferation of online text data requires efficient summarization techniques.

2.Summarization Methods:

- Extractive Summarization: Directly selects important sentences from the text.
- Abstractive Summarization: Generates new sentences that convey the main ideas.

3.Transformer Models: Utilizes pre-trained models based on transformer architecture for generating summaries.

4.Dataset: Uses the BBC news dataset for evaluation.

5.Evaluation and Results: Compares the effectiveness of extractive and abstractive methods using various NLP and deep learning techniques.

Relevance to the Project:

This paper is highly relevant to projects focused on text summarization using GPT models due to the transformer Architecture the study emphasizes the use of transformer models, which are the foundation of GPT (Generative Pre-trained Transformer) models. It provides a comparative analysis of different summarization methods, which can guide the choice of techniques and models in GPT-based projects. Highlights the importance and effectiveness of pre-trained models in summarization tasks, aligning with the use of GPT models for generating high-quality summaries.

CHAPTER 3

METHODOLOGY

The methodology of the Generative AI Text Summarization Blog encompasses various stages, including data collection, data preprocessing, model training, and feature extraction. Each stage is crucial to ensure the final product is robust, efficient, and user-friendly. Below is a detailed description of each stage, including the processes and techniques employed. The steps needed for summarization are data collection, preprocessing, selecting an appropriate summarization algorithm, training the summarization model, evaluating the performance of the model, and deploying the model in a production environment to generate summaries for new input text. It is important to note that text summarization is a complex task, and the quality of the generated summaries depends on the quality of the data and the chosen algorithm. Therefore, careful attention should be paid to each step of the summarization process to ensure accurate and coherent summaries.

3.1 Data Collection

- ➢ Clearly state the objectives of the text summarization task (e.g., summarizing news articles, scientific papers).
- ➢ Collect diverse text data for training and fine-tuning the GPT model.
- ➢ Preprocess the data by cleaning and standardizing it to ensure consistency and quality
- ➢ Data collection is the foundation of any AI-based application. For the Generative AI Text Summarization Blog, the primary data source is text from PDF documents and user-provided text input. The data collection process involves:

1. **User Input:** Users can manually input text into a text area provided on the Streamlit interface. This text can come from various sources, including articles, reports, academic papers, and more.
2. **PDF Uploads:** Users can upload PDF documents, which are then processed to extract text content. This feature is particularly useful for those who need to summarize lengthy documents without manually copying the text.

Source Identification:

Identify and select appropriate sources from where text data will be collected:

1. Web Scraping: Utilize tools like BeautifulSoup for extracting text from websites.

2. API Integration: Access data through APIs such as news APIs, academic databases, or other relevant sources.

3. Manual Collection: Curate datasets manually if specific sources or datasets are required.

In a broader context, training a generative AI model like Google's requires vast amounts of textual data. This data typically comes from a diverse range of sources, such as books, websites, research papers, and other digital text repositories. The model must be exposed to various writing styles, topics, and formats to ensure its summaries are accurate and coherent across different types of text.

3.2 Data Preprocessing:

Once the data is collected, it undergoes preprocessing to prepare it for summarization. Data preprocessing involves several steps to ensure the text is clean and suitable for AI model input. The main steps include:

1. **Text Extraction:** For PDF documents, the PyPDF2 library is used to extract text from each page of the uploaded PDF. The extracted text is concatenated to form a single continuous string, which is then processed further.

2. **Cleaning and Tokenization:**

- **Cleaning:** This step involves removing unnecessary characters, such as special symbols, extra spaces, and non-textual elements that may be present in the text. This ensures the text is clean and free of noise that could affect the model's performance.

- **Tokenization:** The text is split into smaller units called tokens (words or phrases). Tokenization is essential for understanding the structure of the text and preparing it for feature extraction and model input.

3. **Normalization:**

- **Lowercasing:** Converting all text to lowercase to maintain consistency.
- **Removing Stop Words:** Commonly used words (e.g., "the", "is", "in") that do not contribute significantly to the meaning of the text are removed. This helps in focusing on the more meaningful parts of the text.
- **Lemmatization/Stemming:** Reducing words to their base or root form (e.g., "running" to "run"). This helps in standardizing the text and reducing variability.

3.3 Model Training:

Training a generative AI model like the one used in this project involves several stages, including feature extraction, model architecture design, and iterative training with optimization. Here's a breakdown of the model training process:

1. **Feature Extraction:**

- **N-grams:** Extracting contiguous sequences of words (n-grams) from the text. This helps in capturing the context and meaning of the text.
- **TF-IDF:** Term Frequency-Inverse Document Frequency (TF-IDF) scores help in identifying important words in the text based on their frequency and significance.
- **Word Embeddings:** Techniques like Word2Vec, GloVe, or BERT are used to convert words into vector representations. These embeddings capture semantic meanings and relationships between words.

2. **Model Architecture:**

- The generative model architecture is designed based on Transformer networks, which are state-of-the-art in natural language processing (NLP). Transformer models, like BERT or GPT, use attention mechanisms to understand the context and relationships within the text.

- Encoder-Decoder Architecture: This is commonly used in text summarization tasks. The encoder processes the input text and generates a context-rich representation, while the decoder generates the summary based on this representation.

3. **Training Process:**

- **Dataset Preparation:** Large-scale text datasets are prepared for training the model. These datasets are often divided into training, validation, and test sets.
- **Supervised Learning:** The model is trained using supervised learning techniques, where the input text and corresponding summaries are provided. The model learns to generate summaries by minimizing the difference between its output and the actual summaries.
- **Fine-Tuning:** The model is fine-tuned on specific datasets to improve its performance on particular types of text or domains. This involves adjusting the model's parameters to optimize its output.
- **Evaluation:** The model's performance is evaluated using metrics like ROUGE (Recall-Oriented Understudy for Gisting Evaluation), which compares the overlap of n-grams between the generated summary and the reference summary.

3.4 Feature Extraction:

Feature extraction is crucial in transforming raw text into meaningful representations that the model can understand and process. In the context of the Generative AI Text Summarization Blog, feature extraction involves:

1. Text Representation:

- **Bag-of-Words (BoW):** Representing text as a collection of words, ignoring grammar and order. This method captures word frequency but lacks context.
- **TF-IDF Vectors:** Converting text into TF-IDF vectors helps in identifying important words by considering their frequency and inverse document frequency.

- **Word Embeddings:** Using pre-trained word embeddings like Word2Vec, GloVe, or BERT to convert words into dense vectors. These embeddings capture semantic meanings and relationships between words.

2. Contextual Embeddings:

- **BERT (Bidirectional Encoder Representations from Transformers):** BERT embeddings provide context-aware representations of words by considering the entire sentence. This helps in understanding the meaning of words in different contexts.
- **Transformer Models:** Utilizing the Transformer architecture for feature extraction, which involves attention mechanisms to capture dependencies and relationships between words.

3. Sequence Representation:

- Positional Encoding: Adding positional information to the embeddings to maintain the order of words in the text. This is crucial for models like Transformers that process entire sequences.
- Sentence Embeddings: Representing entire sentences or paragraphs as vectors using models like Sentence-BERT. This helps in capturing the overall meaning and context of longer text.

The study aimed to evaluate the performance of different text summarization models in generating coherent and informative summaries from long texts. To achieve this, three popular text summarization models were implemented and evaluated, including the Bert model, Pegasus model and Simple T5 model. The models were tested on a samsum dataset collected from online sources. The evaluation of the models was based on the ROUGE metric, which is commonly used to measure the quality of summaries in text summarization tasks. The ROUGE scores were calculated for the generated summaries and compared with the reference summaries to assess the performance of each model

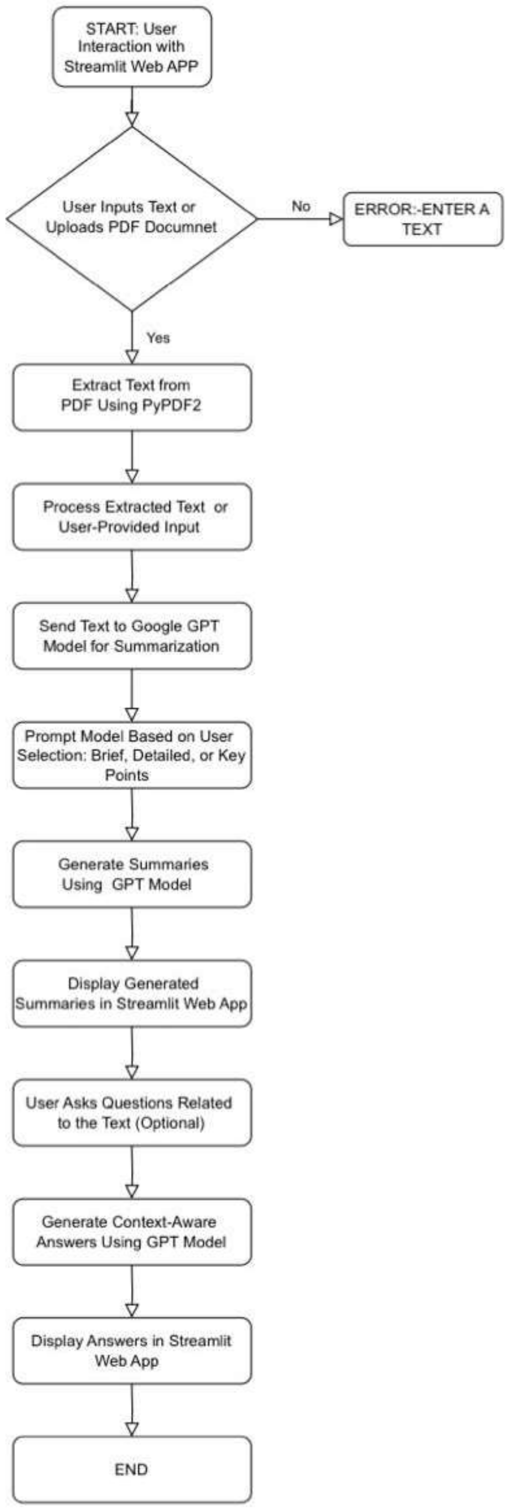

START: User Interaction with Streamlit Web APP
User Inputs Text or Uploads PDF Documnet
No
ERROR:-ENTER A TEXT
Yes
Extract Text from PDF Using PyPDF2
Process Extracted Text or User-Provided Input
Send Text to Google GPT Model for Summarization
Prompt Model Based on User Selection: Brief, Detailed, or Key Points
Generate Summaries Using GPT Model
Display Generated Summaries in Streamlit Web App
User Asks Questions Related to the Text (Optional)
Generate Context-Aware Answers Using GPT Model
Display Answers in Streamlit Web App
END

FIG 3.1: METHODOLOGY

3.5 System Architecture

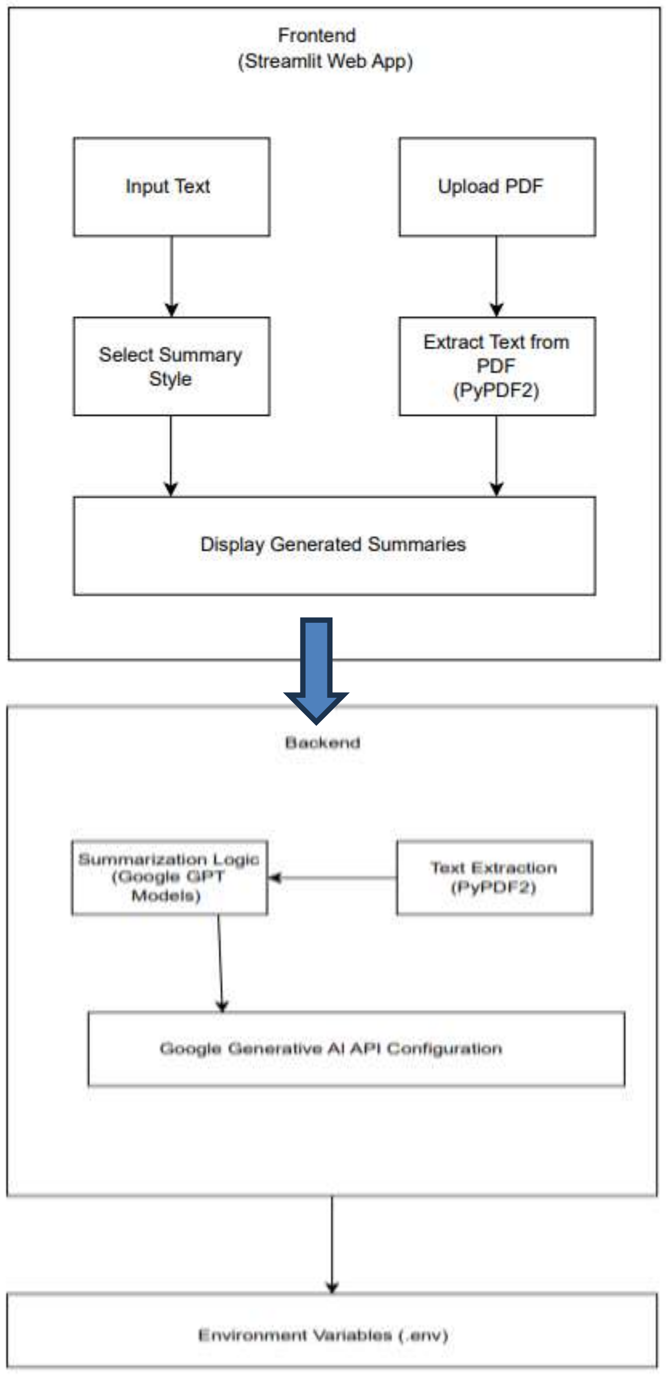

Frontend
(Streamlit Web App)
Input Text
Upload PDF
Select Summary Style
Extract Text from PDF (PyPDF2)
Display Generated Summaries
Backend
Summarization Logic (Google GPT Models)
Text Extraction (PyPDF2)
Google Generative AI API Configuration
Environment Variables (.env)

FIG 3.2: SYSTEM ARCHITECTURE

The system architecture of the Generative AI Text Summarization Blog involves several components that work together to provide a seamless user experience. These components include the user interface, backend processing, AI model integration, and data storage. The architecture ensures efficient handling of user inputs, text extraction, summarization, and query-based interactions.

1. User Interface

The user interface (UI) is built using Streamlit, which provides an interactive and user-friendly web application. The UI components include:

- **Text Area**: Allows users to input text directly.
- **PDF Uploader**: Enables users to upload PDF documents for text extraction.
- **Buttons**: For selecting the type of summarization (Brief, Detailed, Key Points) and initiating the summarization process.
- **Query Input**: A text input field for users to ask questions about the text.
- **Output Display**: Sections to display extracted text, generated summaries, and answers to user queries.

2. Backend Processing

The backend processing involves several steps to handle user inputs, extract text, and communicate with the AI model. The main components include:

- **Text Extraction Module**:
- **PyPDF2**: Used to extract text from uploaded PDF documents.
- **Text Cleaning and Tokenization**: Preprocesses the extracted text to remove noise and tokenize it for further processing.
- **Summarization and Query Processing**:
- **Summarization Module**: Communicates with the AI model to generate summaries based on user-selected styles.

- **Query Processing Module**: Handles user queries and generates relevant answers based on the content.

3. AI Model Integration

The AI model integration is a crucial part of the system architecture, leveraging Google Generative AI to perform text summarization and query-based interactions. The components include:

- **Google Generative AI API**: Configured using API keys to interact with the AI model.
- **Generative Model**: Utilizes the model (e.g., Gemini-pro) to generate content, including summaries and answers to queries.
- **Prompt Engineering**: Constructs prompts to guide the AI model in generating the desired outputs based on the input text and user preferences.

4. Data Storage and Management

Data storage and management ensure that user inputs, extracted text, and generated summaries are efficiently handled. The components include:

- **Temporary Storage**: Stores user inputs and extracted text during the session.
- **Session Management**: Manages user sessions to maintain continuity and handle multiple users concurrently.

5. System Workflow

The overall workflow of the system is as follows:

1. **User Interaction**:
    - Users input text directly or upload a PDF document via the Streamlit UI.
    - Users select the type of summarization or input a query.
2. **Text Extraction**:
    - If a PDF is uploaded, the PyPDF2 module extracts text from the document.
    - The extracted text is preprocessed (cleaned and tokenized) for further processing.

3. **Summarization/Query Processing**:
   - ➢ The user-selected summarization style or query is sent to the respective module.
   - ➢ The module constructs a prompt and sends it to the Google Generative AI API.
   - ➢ The AI model generates the summary or answers the query.
4. **Output Display**:
   - ➢ The generated summary or answer is displayed on the Streamlit UI.
   - ➢ Users can view the results and interact further if needed.

3.6 Tools and Technologies

The development of the Generative AI Text Summarization Blog involves various tools and technologies that contribute to different aspects of the system. These tools and technologies are selected to ensure efficiency, robustness, and ease of use. Below is a detailed list of the tools and technologies used:

1. Programming Languages

- ➢ **Python**: The primary programming language used for developing the application. Python is known for its simplicity, readability, and extensive libraries for data processing and machine learning.

2. Frameworks and Libraries

- ➢ **Streamlit**: A framework used to create the interactive web application. Streamlit is chosen for its simplicity and ability to quickly turn Python scripts into web apps.
  - **Streamlit Components**: Used for building the UI elements like text areas, file uploaders, buttons, and displaying the results.
- ➢ **PyPDF2**: A library used for extracting text from PDF files. It allows reading and parsing PDF documents, which is essential for handling PDF uploads.
  - **PdfReader**: The specific module within PyPDF2 that reads and extracts text from PDF pages.

3. Artificial Intelligence and Machine Learning

- **Google Generative AI**: The AI model used for generating text summaries and answering queries.
    - **Google Generative AI API**: The API used to interact with the Google Generative AI models. It requires proper configuration with an API key.
    - **Generative Model (e.g., Gemini-pro)**: The specific generative AI model used for summarization and query-based interactions.
- **NLP Techniques**:
    - **Tokenization**: Breaking down text into individual tokens (words or phrases).
    - **TF-IDF (Term Frequency-Inverse Document Frequency)**: A statistical measure used to evaluate the importance of a word in a document.
    - **Word Embeddings (Word2Vec, GloVe, BERT)**: Methods for converting words into vector representations that capture semantic meanings.
    - **Transformer Models (BERT, GPT)**: State-of-the-art models used for understanding and generating human-like text.

4. Environment Management

- **dotenv**: A library used to load environment variables from a .env file. This is essential for securely managing API keys and configuration settings.

5. Text Processing and Analysis

- **Natural Language Processing (NLP)**: Techniques and libraries for preprocessing, analyzing, and understanding text data.
    - **Text Cleaning and Normalization**: Removing noise, converting text to lowercase, removing stop words, and lemmatization/stemming.
    - **Feature Extraction**: Extracting meaningful features from text using methods like n-grams, TF-IDF, and embeddings.

6. Development and Deployment

- **Integrated Development Environment (IDE)**: Tools like PyCharm, VS Code, or Jupyter Notebooks for developing and testing the application.
- **Version Control (Git)**: Used for version control and collaboration. Platforms like GitHub or GitLab can be used to manage the codebase.

7. System Integration and APIs

- **HTTP Requests**: Used to interact with external APIs (e.g., Google Generative AI API) for sending and receiving data.
- **API Key Management**: Secure handling of API keys and tokens, typically managed using environment variables loaded by the dotenv library.

8. Styling and User Experience

- **CSS**: Custom styling for the Streamlit app to enhance the user interface and user experience.
    - **Custom CSS Styles**: Defined within the Streamlit app to style headers, footers, buttons, and content sections.

9. Data Storage and Management

- **Temporary Storage**: Handling temporary data storage within the session for user inputs and extracted text.
- **Session Management**: Managing user sessions to maintain continuity and handle multiple users concurrently.

Summary of Tools and Technologies

1. **Programming Language**: Python
2. **Frameworks and Libraries**: Streamlit, PyPDF2, dotenv
3. **AI and ML**: Google Generative AI, Generative Model (e.g., Gemini-pro), NLP techniques (Tokenization, TF-IDF, Word Embeddings, Transformer Models)
4. **Environment Management**: dotenv

5. **Text Processing and Analysis**: NLP techniques (Text Cleaning, Normalization, Feature Extraction)
6. **Development and Deployment**: IDE (PyCharm, VS Code), Version Control (Git)
7. **System Integration and APIs**: HTTP Requests, API Key Management
8. **Styling and User Experience**: CSS
9. **Data Storage and Management**: Temporary Storage, Session Management

<u>Hardware Requirement Specification:</u>

- Processor: a minimum dual-core processor.
- Storage: minimum 256 GB SSD.
- RAM: 8GB to 32GB RAM.
- Internet Connection: Required for accessing the Google Generative AI API.

<u>Software Requirement Specification:</u>

- Streamlit : Framework for building the web application.
- Python : Python 3(Programming language for development).
- PyPDF2: Library for PDF text extraction.
- Google Generative AI: API for accessing GPT models.
- Dotenv: Library for managing environment variables

CHAPTER 4

IMPLEMENTATION

4.1 Steps Followed

1. Setup and Import Libraries:

- Import necessary libraries: dotenv, streamlit, google.generativeai, os, PyPDF2.
- Load environment variables using load_dotenv() to get API keys or other sensitive information from a .env file.

2. Streamlit Configuration:

- Configure the Streamlit app with a title, icon, and layout using st.set_page_config().
- Apply custom CSS for styling various elements such as the body, header, footer, buttons, and blog content.

3. Google Generative AI Configuration:

- Configure Google Generative AI by setting the API key using genai.configure(api_key=os.getenv("AIzaSyAKMl5-ElJKJObdImX8_h7N6prUB5vBecY")).
- Instantiate the Generative Model using genai.GenerativeModel('gemini-pro').

4. Function Definitions:

- **Summarize Function**: Define a function summarize(text, style) that generates summaries based on the selected style (Brief, Detailed, Key Points) by creating a prompt and passing it to the generative model to get a response.
- **Extract Text from PDF Function**: Define a function extract_text_from_pdf(file) that uses PyPDF2.PdfReader to read and extract text from a PDF file.

5. Streamlit App Layout:

- **Header**: Create a header using HTML and CSS within st.markdown().

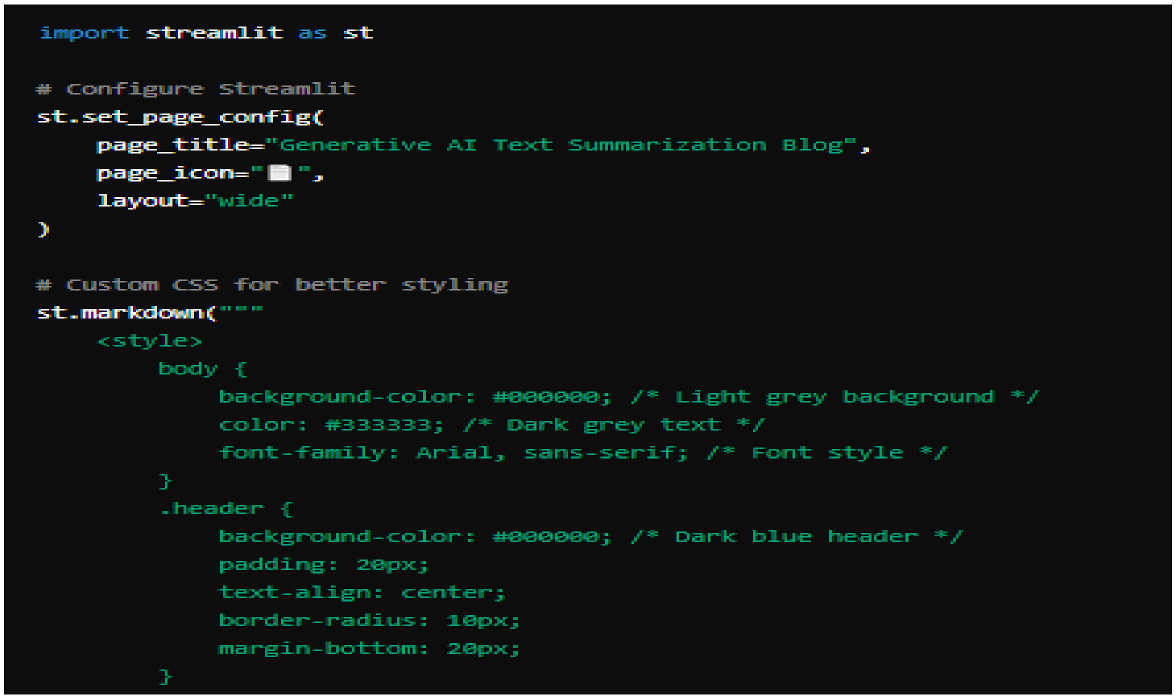
- **Introduction Section**: Add an introductory section to explain the purpose and functionality of the app.
- **Text Input**: Create a large text area for users to input text using st.text_area().
- **File Uploader**: Add a file uploader for PDF files using st.file_uploader(). If a PDF is uploaded, extract its text and display it.
- **Summary Buttons**: Create three buttons for different summary styles (Brief, Detailed, Key Points) using st.button(). When a button is pressed, call the summarize() function and display the generated summary.
- **Query Input**: Add a text input field for users to ask questions related to the uploaded or input text. Generate answers using the generative model and display them.

6. Footer:

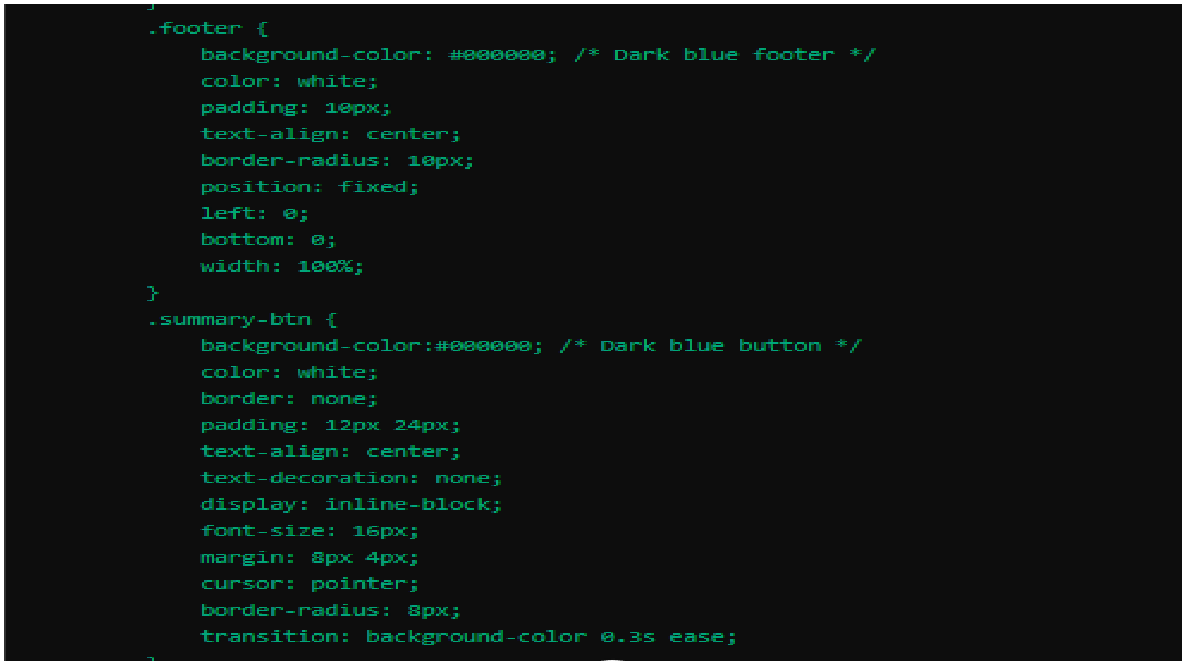
- Create a footer to credit the creator using HTML and CSS within st.markdown().

4.2 Code Snippet

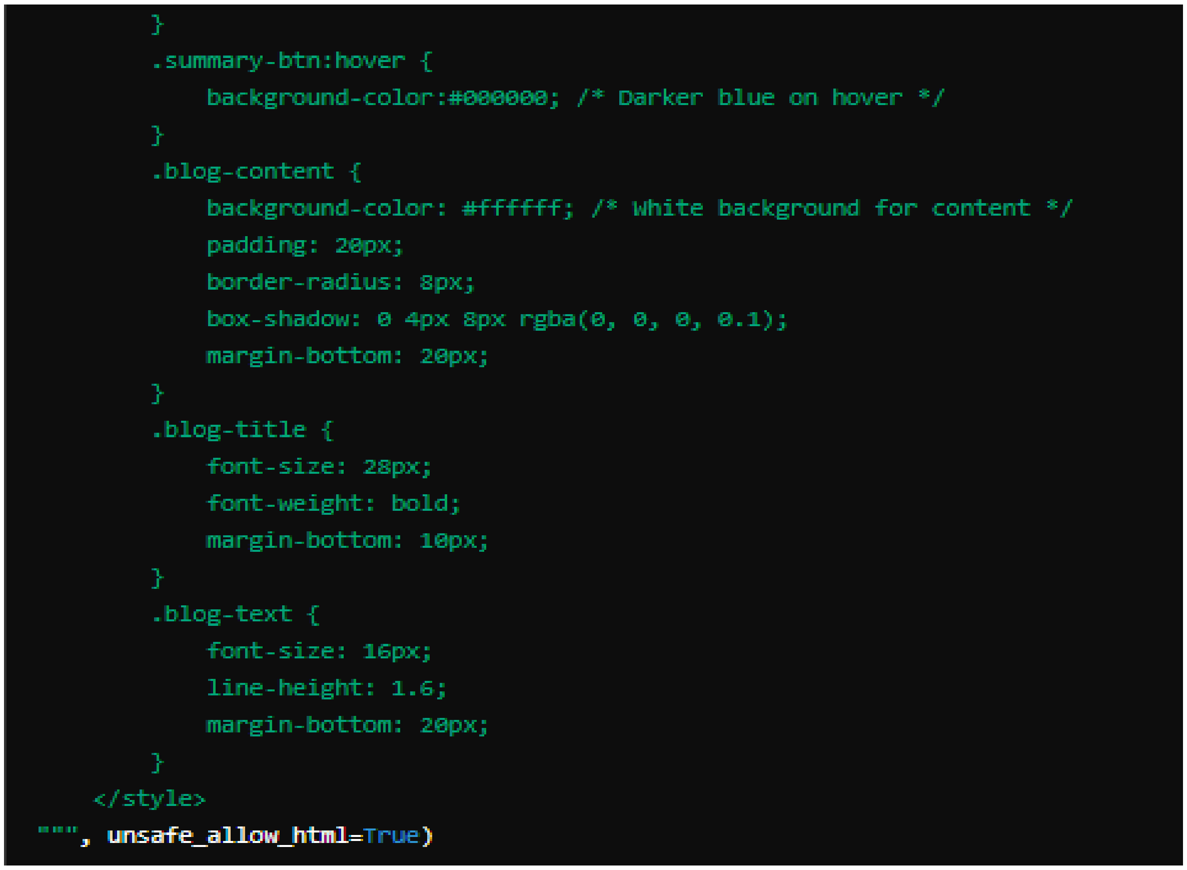
Front End:



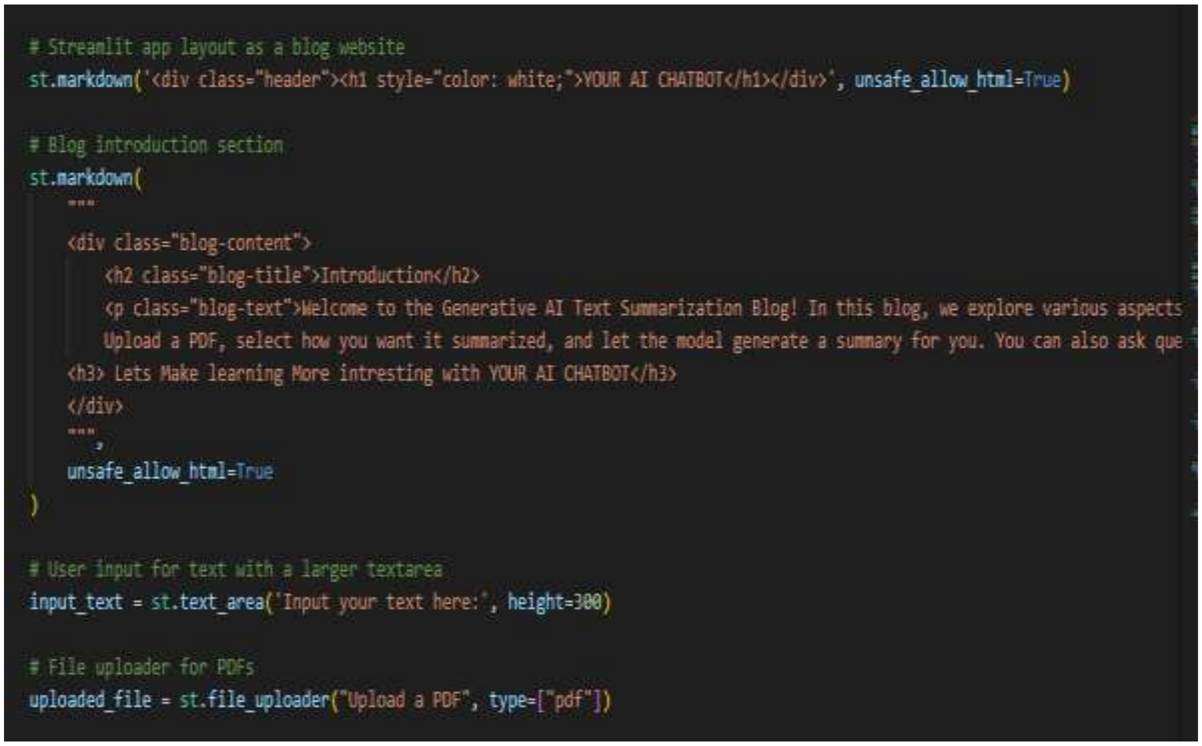
```
import streamlit as st

# Configure Streamlit
st.set_page_config(
    page_title="Generative AI Text Summarization Blog",
    page_icon="📄",
    layout="wide"
)

# Custom CSS for better styling
st.markdown("""
    <style>
        body {
            background-color: #000000; /* Light grey background */
            color: #333333; /* Dark grey text */
            font-family: Arial, sans-serif; /* Font style */
        }
        .header {
            background-color: #000000; /* Dark blue header */
            padding: 20px;
            text-align: center;
            border-radius: 10px;
            margin-bottom: 20px;
        }
```



```
        .footer {
            background-color: #000000; /* Dark blue footer */
            color: white;
            padding: 10px;
            text-align: center;
            border-radius: 10px;
            position: fixed;
            left: 0;
            bottom: 0;
            width: 100%;
        }
        .summary-btn {
            background-color:#000000; /* Dark blue button */
            color: white;
            border: none;
            padding: 12px 24px;
            text-align: center;
            text-decoration: none;
            display: inline-block;
            font-size: 16px;
            margin: 8px 4px;
            cursor: pointer;
            border-radius: 8px;
            transition: background-color 0.3s ease;
```

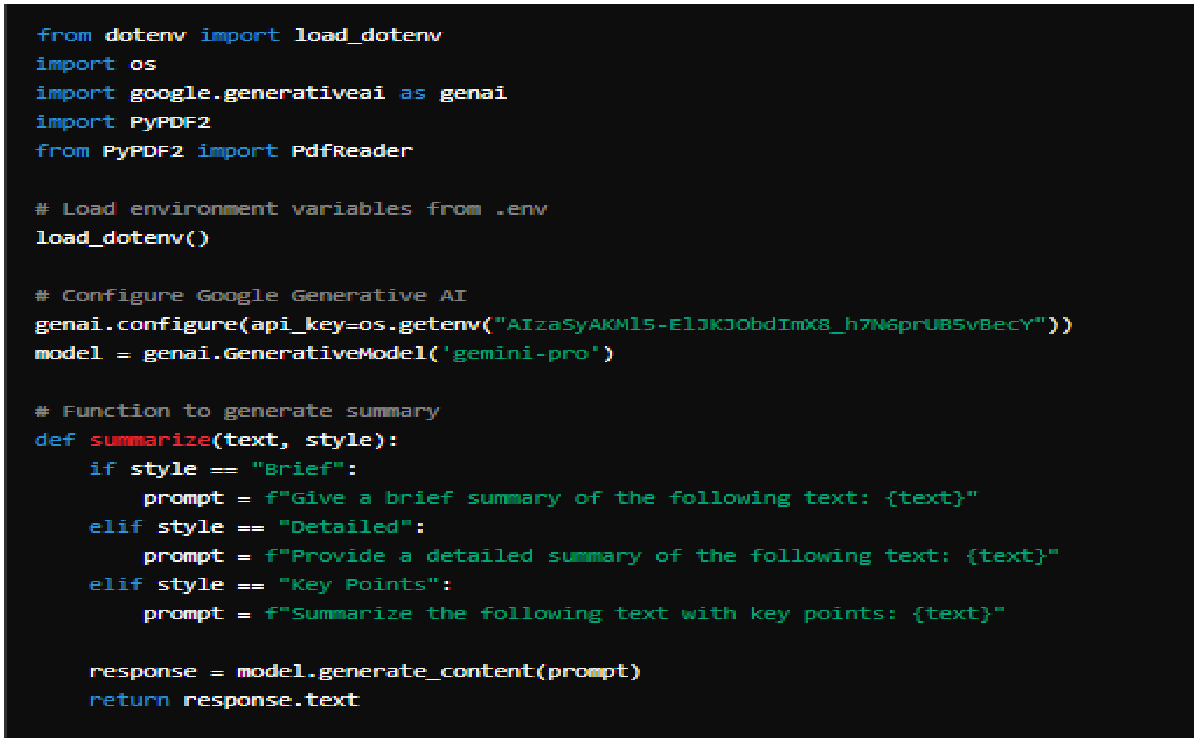


```
        }
        .summary-btn:hover {
            background-color:#000000; /* Darker blue on hover */
        }
        .blog-content {
            background-color: #ffffff; /* White background for content */
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            margin-bottom: 20px;
        }
        .blog-title {
            font-size: 28px;
            font-weight: bold;
            margin-bottom: 10px;
        }
        .blog-text {
            font-size: 16px;
            line-height: 1.6;
            margin-bottom: 20px;
        }
    </style>
""", unsafe_allow_html=True)
```

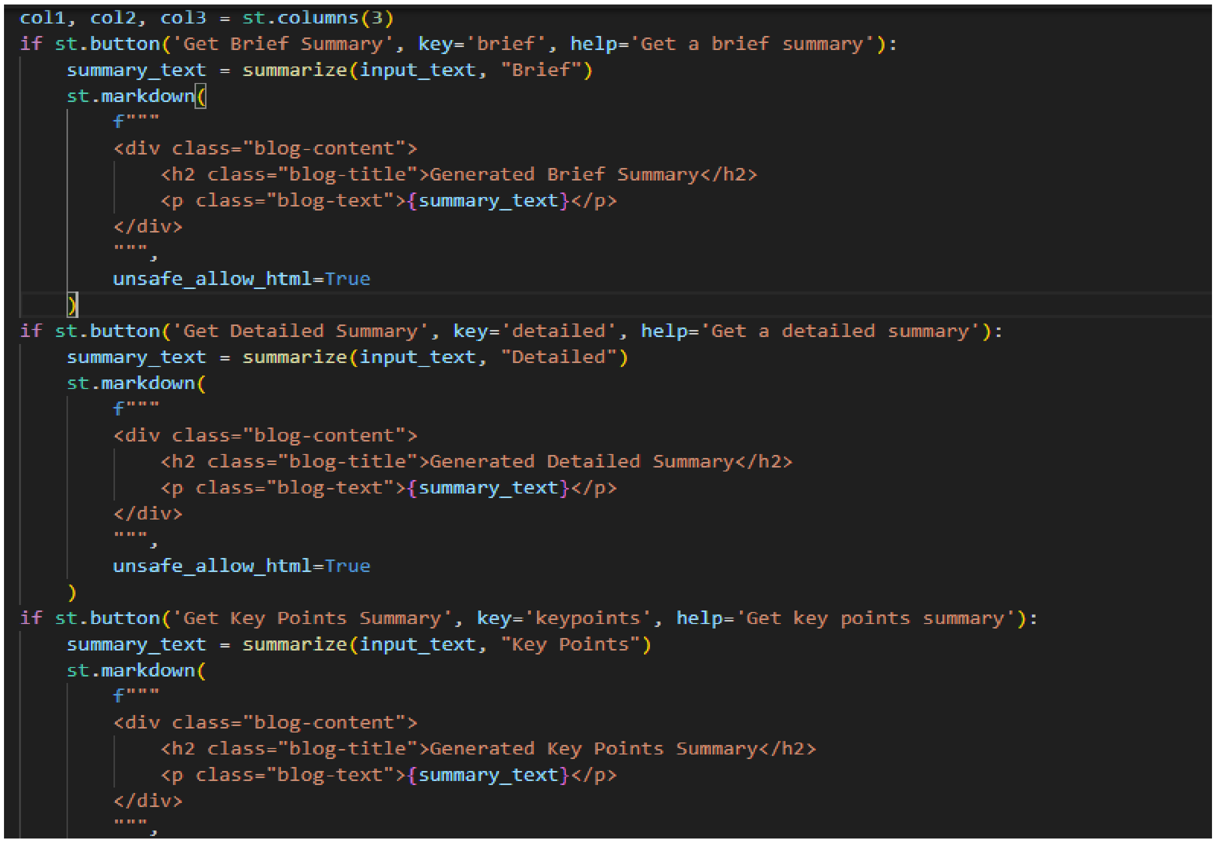


```
# Streamlit app layout as a blog website
st.markdown('<div class="header"><h1 style="color: white;">YOUR AI CHATBOT</h1></div>', unsafe_allow_html=True)

# Blog introduction section
st.markdown(
    """
    <div class="blog-content">
        <h2 class="blog-title">Introduction</h2>
        <p class="blog-text">Welcome to the Generative AI Text Summarization Blog! In this blog, we explore various aspects
        Upload a PDF, select how you want it summarized, and let the model generate a summary for you. You can also ask que
    <h3> Lets Make learning More intresting with YOUR AI CHATBOT</h3>
    </div>
    """,
    unsafe_allow_html=True
)

# User input for text with a larger textarea
input_text = st.text_area('Input your text here:', height=300)

# File uploader for PDFs
uploaded_file = st.file_uploader("Upload a PDF", type=["pdf"])
```

Back End:



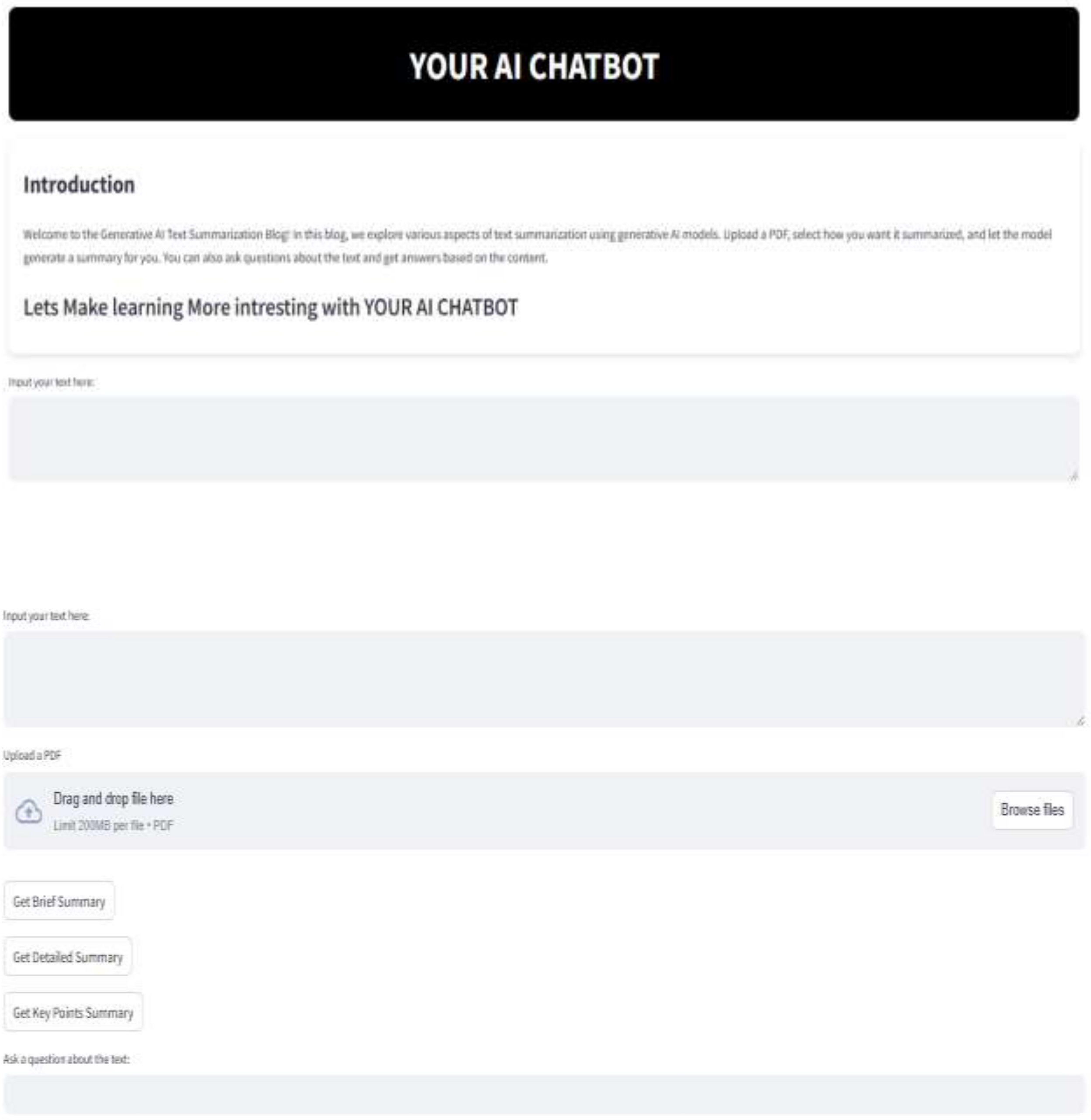
```
from dotenv import load_dotenv
import os
import google.generativeai as genai
import PyPDF2
from PyPDF2 import PdfReader

# Load environment variables from .env
load_dotenv()

# Configure Google Generative AI
genai.configure(api_key=os.getenv("AIzaSyAKMl5-ElJKJObdImX8_h7N6prUB5vBecY"))
model = genai.GenerativeModel('gemini-pro')

# Function to generate summary
def summarize(text, style):
    if style == "Brief":
        prompt = f"Give a brief summary of the following text: {text}"
    elif style == "Detailed":
        prompt = f"Provide a detailed summary of the following text: {text}"
    elif style == "Key Points":
        prompt = f"Summarize the following text with key points: {text}"

    response = model.generate_content(prompt)
    return response.text
```



```
col1, col2, col3 = st.columns(3)
if st.button('Get Brief Summary', key='brief', help='Get a brief summary'):
    summary_text = summarize(input_text, "Brief")
    st.markdown(
        f"""
        <div class="blog-content">
            <h2 class="blog-title">Generated Brief Summary</h2>
            <p class="blog-text">{summary_text}</p>
        </div>
        """,
        unsafe_allow_html=True
    )
if st.button('Get Detailed Summary', key='detailed', help='Get a detailed summary'):
    summary_text = summarize(input_text, "Detailed")
    st.markdown(
        f"""
        <div class="blog-content">
            <h2 class="blog-title">Generated Detailed Summary</h2>
            <p class="blog-text">{summary_text}</p>
        </div>
        """,
        unsafe_allow_html=True
    )
if st.button('Get Key Points Summary', key='keypoints', help='Get key points summary'):
    summary_text = summarize(input_text, "Key Points")
    st.markdown(
        f"""
        <div class="blog-content">
            <h2 class="blog-title">Generated Key Points Summary</h2>
            <p class="blog-text">{summary_text}</p>
        </div>
        """,
```

CHAPTER 5

RESULT



YOUR AI CHATBOT

Introduction

Welcome to the Generative AI Text Summarization Blog! In this blog, we explore various aspects of text summarization using generative AI models. Upload a PDF, select how you want it summarized, and let the model generate a summary for you. You can also ask questions about the text and get answers based on the content.

Lets Make learning More intresting with YOUR AI CHATBOT

Input your text here:

Input your text here:

Upload a PDF

Drag and drop file here

Limit 200MB per file • PDF

Browse files

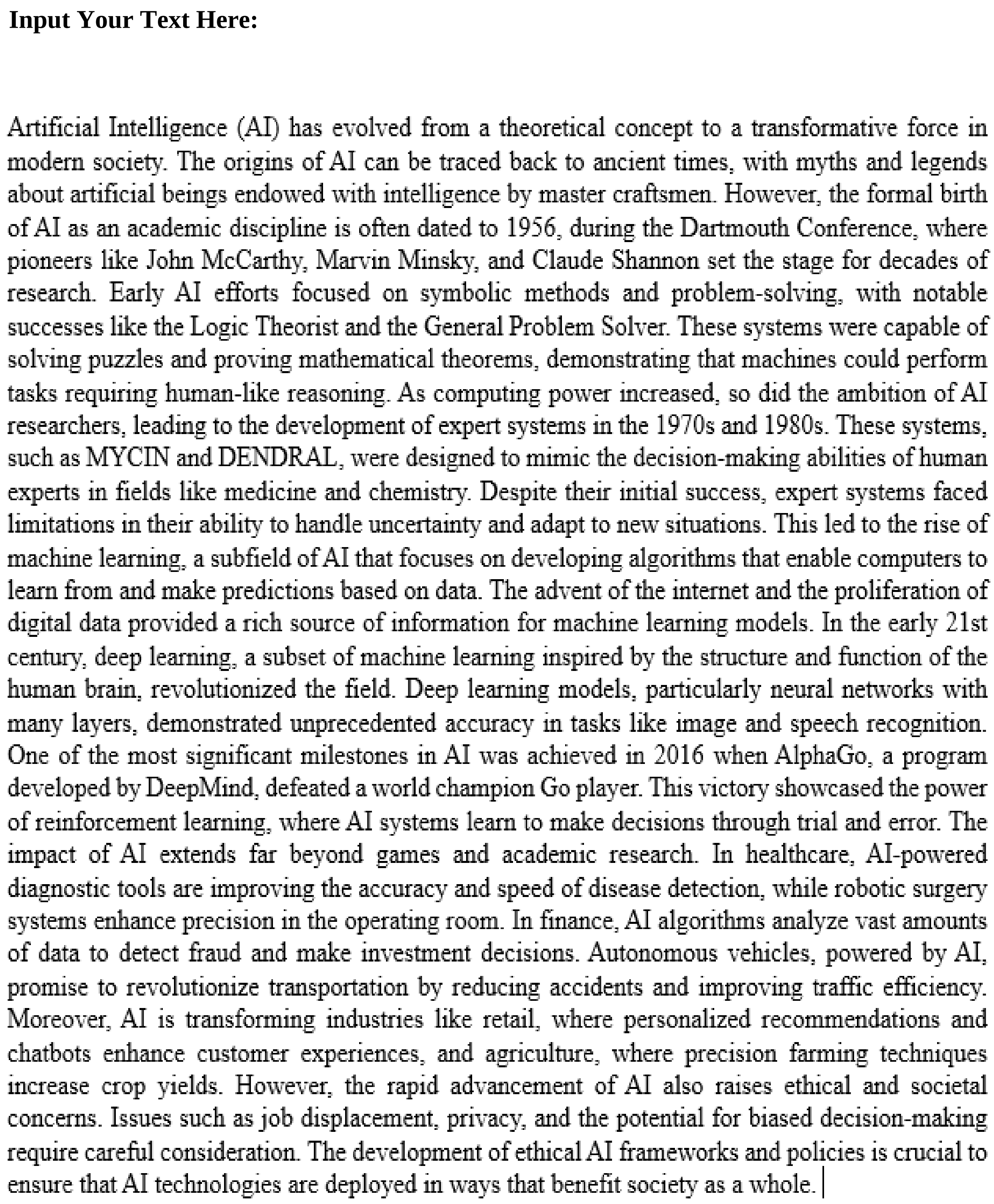
Get Brief Summary

Get Detailed Summary

Get Key Points Summary

Ask a question about the text:

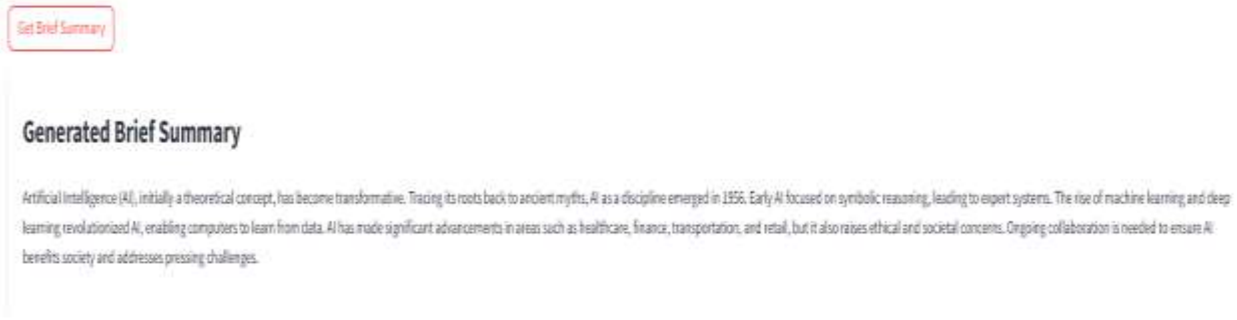
FIG 5.1: FRONT END



Input Your Text Here:

Artificial Intelligence (AI) has evolved from a theoretical concept to a transformative force in modern society. The origins of AI can be traced back to ancient times, with myths and legends about artificial beings endowed with intelligence by master craftsmen. However, the formal birth of AI as an academic discipline is often dated to 1956, during the Dartmouth Conference, where pioneers like John McCarthy, Marvin Minsky, and Claude Shannon set the stage for decades of research. Early AI efforts focused on symbolic methods and problem-solving, with notable successes like the Logic Theorist and the General Problem Solver. These systems were capable of solving puzzles and proving mathematical theorems, demonstrating that machines could perform tasks requiring human-like reasoning. As computing power increased, so did the ambition of AI researchers, leading to the development of expert systems in the 1970s and 1980s. These systems, such as MYCIN and DENDRAL, were designed to mimic the decision-making abilities of human experts in fields like medicine and chemistry. Despite their initial success, expert systems faced limitations in their ability to handle uncertainty and adapt to new situations. This led to the rise of machine learning, a subfield of AI that focuses on developing algorithms that enable computers to learn from and make predictions based on data. The advent of the internet and the proliferation of digital data provided a rich source of information for machine learning models. In the early 21st century, deep learning, a subset of machine learning inspired by the structure and function of the human brain, revolutionized the field. Deep learning models, particularly neural networks with many layers, demonstrated unprecedented accuracy in tasks like image and speech recognition. One of the most significant milestones in AI was achieved in 2016 when AlphaGo, a program developed by DeepMind, defeated a world champion Go player. This victory showcased the power of reinforcement learning, where AI systems learn to make decisions through trial and error. The impact of AI extends far beyond games and academic research. In healthcare, AI-powered diagnostic tools are improving the accuracy and speed of disease detection, while robotic surgery systems enhance precision in the operating room. In finance, AI algorithms analyze vast amounts of data to detect fraud and make investment decisions. Autonomous vehicles, powered by AI, promise to revolutionize transportation by reducing accidents and improving traffic efficiency. Moreover, AI is transforming industries like retail, where personalized recommendations and chatbots enhance customer experiences, and agriculture, where precision farming techniques increase crop yields. However, the rapid advancement of AI also raises ethical and societal concerns. Issues such as job displacement, privacy, and the potential for biased decision-making require careful consideration. The development of ethical AI frameworks and policies is crucial to ensure that AI technologies are deployed in ways that benefit society as a whole.

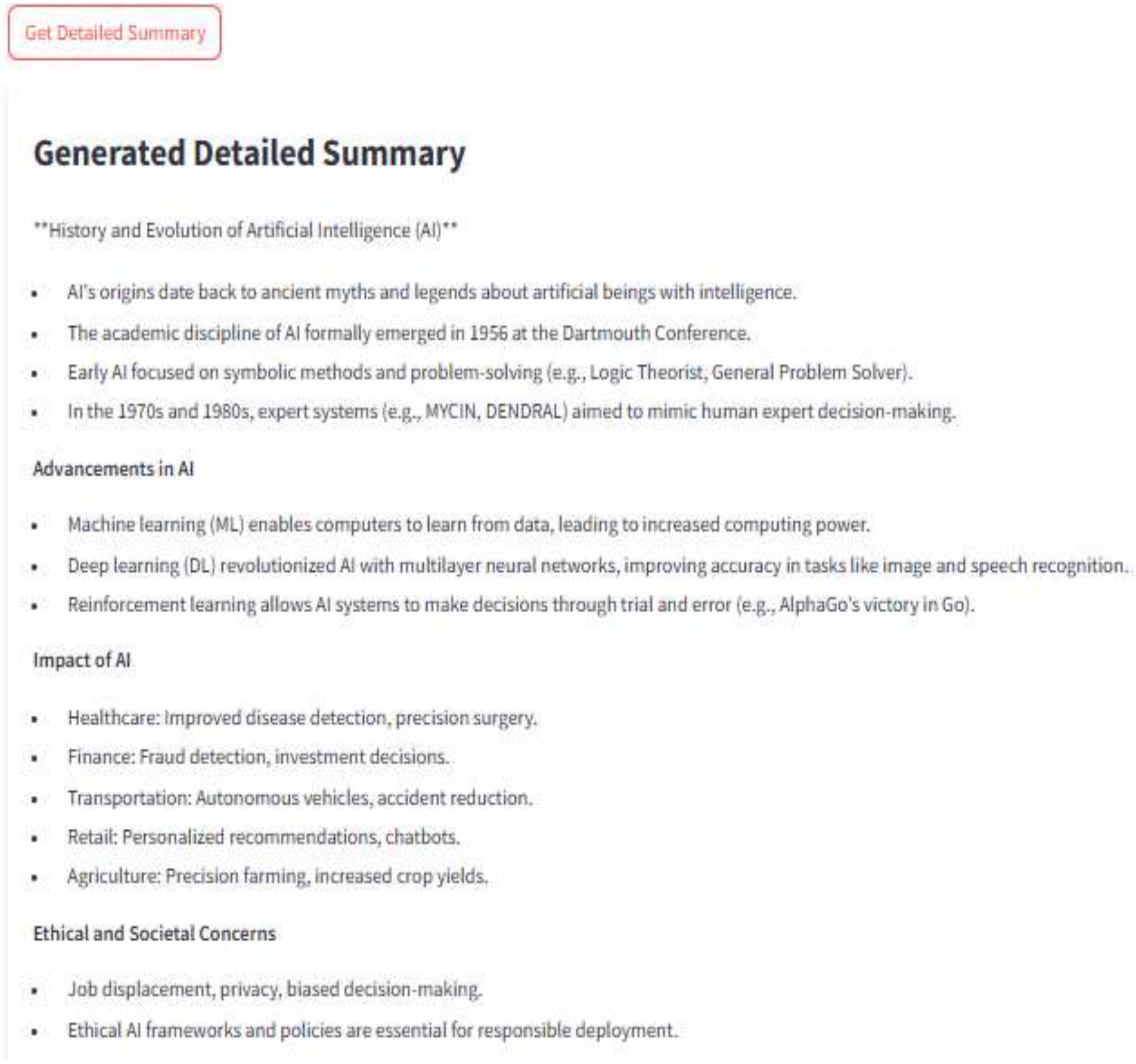
FIG 5.2: INPUT TEXT



Generated Brief Summary

Artificial Intelligence (AI), initially a theoretical concept, has become transformative. Tracing its roots back to ancient myths, AI as a discipline emerged in 1956. Early AI focused on symbolic reasoning, leading to expert systems. The rise of machine learning and deep learning revolutionized AI, enabling computers to learn from data. AI has made significant advancements in areas such as healthcare, finance, transportation, and retail, but it also raises ethical and societal concerns. Ongoing collaboration is needed to ensure AI benefits society and addresses pressing challenges.

FIG 5.3: GENERATED BRIEF SUMMARY



Generated Detailed Summary

History and Evolution of Artificial Intelligence (AI)

- AI's origins date back to ancient myths and legends about artificial beings with intelligence.
- The academic discipline of AI formally emerged in 1956 at the Dartmouth Conference.
- Early AI focused on symbolic methods and problem-solving (e.g., Logic Theorist, General Problem Solver).
- In the 1970s and 1980s, expert systems (e.g., MYCIN, DENDRAL) aimed to mimic human expert decision-making.

Advancements in AI

- Machine learning (ML) enables computers to learn from data, leading to increased computing power.
- Deep learning (DL) revolutionized AI with multilayer neural networks, improving accuracy in tasks like image and speech recognition.
- Reinforcement learning allows AI systems to make decisions through trial and error (e.g., AlphaGo's victory in Go).

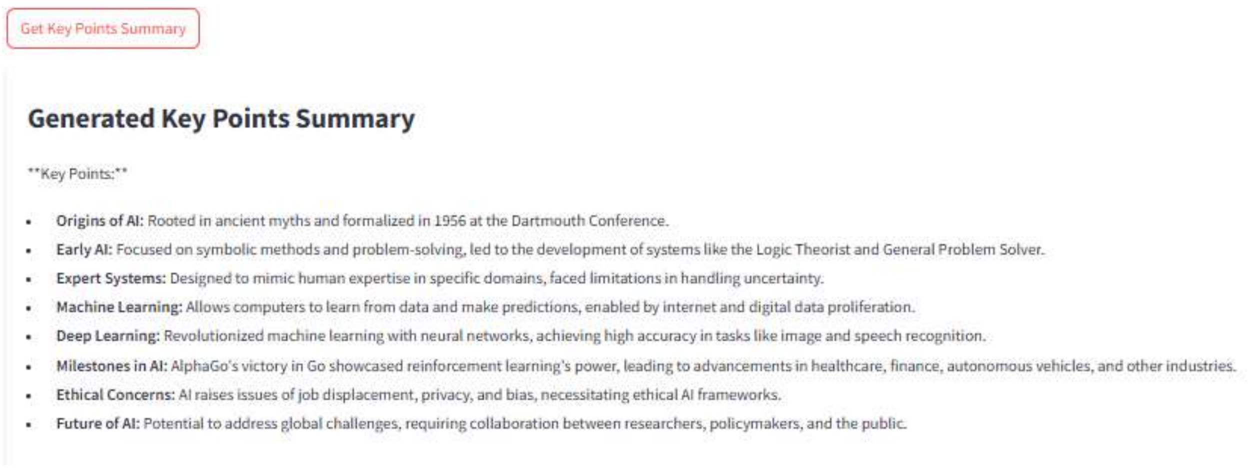
Impact of AI

- Healthcare: Improved disease detection, precision surgery.
- Finance: Fraud detection, investment decisions.
- Transportation: Autonomous vehicles, accident reduction.
- Retail: Personalized recommendations, chatbots.
- Agriculture: Precision farming, increased crop yields.

Ethical and Societal Concerns

- Job displacement, privacy, biased decision-making.
- Ethical AI frameworks and policies are essential for responsible deployment.

FIG 5.4: GENERATED DETAILED SUMMARY



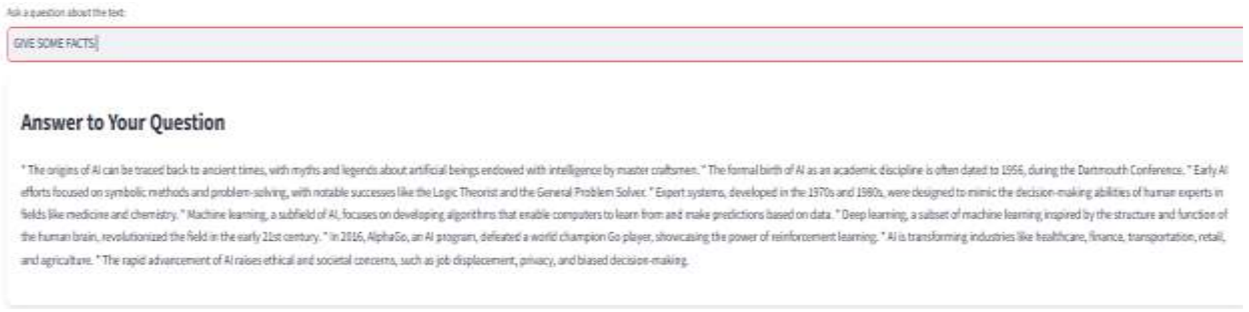
Get Key Points Summary

Generated Key Points Summary

Key Points:

- **Origins of AI:** Rooted in ancient myths and formalized in 1956 at the Dartmouth Conference.
- **Early AI:** Focused on symbolic methods and problem-solving, led to the development of systems like the Logic Theorist and General Problem Solver.
- **Expert Systems:** Designed to mimic human expertise in specific domains, faced limitations in handling uncertainty.
- **Machine Learning:** Allows computers to learn from data and make predictions, enabled by internet and digital data proliferation.
- **Deep Learning:** Revolutionized machine learning with neural networks, achieving high accuracy in tasks like image and speech recognition.
- **Milestones in AI:** AlphaGo's victory in Go showcased reinforcement learning's power, leading to advancements in healthcare, finance, autonomous vehicles, and other industries.
- **Ethical Concerns:** AI raises issues of job displacement, privacy, and bias, necessitating ethical AI frameworks.
- **Future of AI:** Potential to address global challenges, requiring collaboration between researchers, policymakers, and the public.

FIG 5.5: GENERATED KEY POINTS SUMMARY



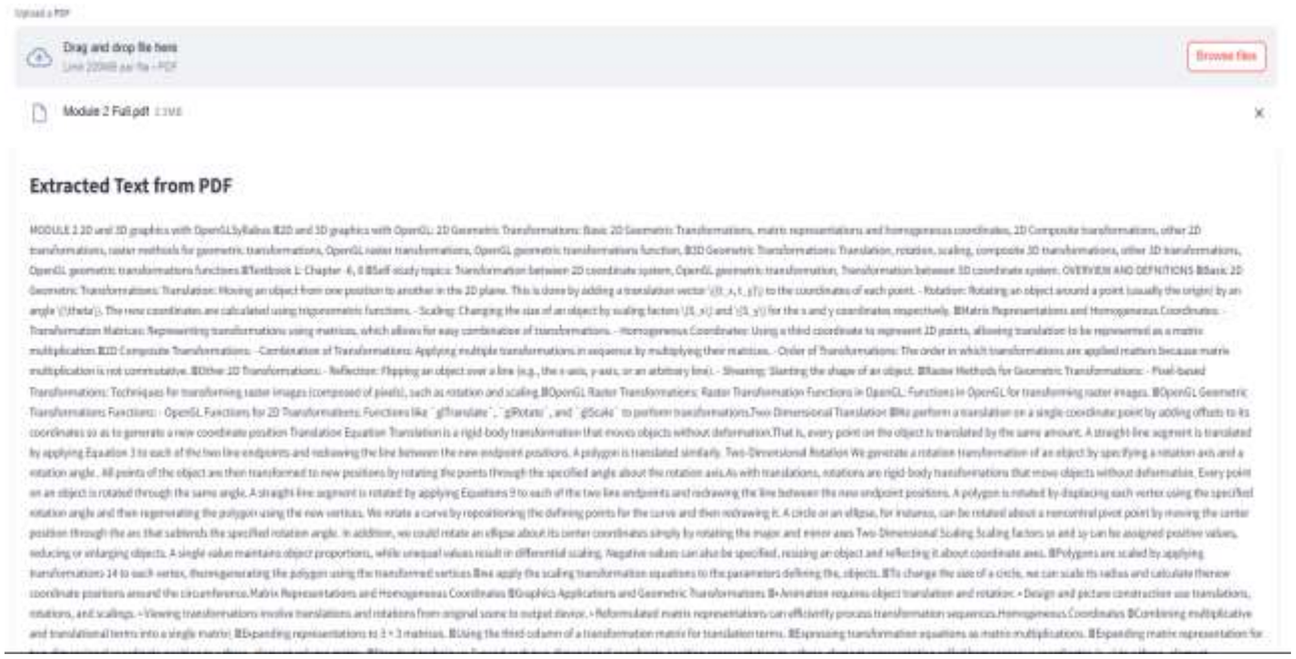
Ask a question about the text:

GIVE SOME FACTS

Answer to Your Question

* The origins of AI can be traced back to ancient times, with myths and legends about artificial beings endowed with intelligence by master craftsmen. * The formal birth of AI as an academic discipline is often dated to 1956, during the Dartmouth Conference. * Early AI efforts focused on symbolic methods and problem-solving, with notable successes like the Logic Theorist and the General Problem Solver. * Expert systems, developed in the 1970s and 1980s, were designed to mimic the decision-making abilities of human experts in fields like medicine and chemistry. * Machine learning, a subfield of AI, focuses on developing algorithms that enable computers to learn from and make predictions based on data. * Deep learning, a subset of machine learning inspired by the structure and function of the human brain, revolutionized the field in the early 21st century. * In 2016, AlphaGo, an AI program, defeated a world champion Go player, showcasing the power of reinforcement learning. * AI is transforming industries like healthcare, finance, transportation, retail, and agriculture. * The rapid advancement of AI raises ethical and societal concerns, such as job displacement, privacy, and biased decision-making.

FIG 5.6: QUESTION AND ANSWER SUMMARY



Upload a PDF

Drag and drop file here

Limit 200MB per file • PDF

Browse files

Module 2 Full.pdf 2.1MB

Extracted Text from PDF

MODULE 2 2D and 3D graphics with OpenGLSyllabus 2D and 3D graphics with OpenGL: 2D Geometric Transformations: Basic 2D Geometric Transformations, matrix representations and homogeneous coordinates, 2D Composite transformations, other 2D transformations, raster methods for geometric transformations, OpenGL raster transformations, OpenGL geometric transformations function, 3D Geometric Transformations: Translation, rotation, scaling, composite 3D transformations, other 3D transformations, OpenGL geometric transformations functions Textbook 1: Chapter -6, 8 Self-study topics: Transformation between 2D coordinate system, OpenGL geometric transformation, Transformation between 3D coordinate system. OVERVIEW AND DEFINITIONS Basic 2D Geometric Transformations: Translation: Moving an object from one position to another in the 2D plane. This is done by adding a translation vector \((t_x, t_y)\) to the coordinates of each point. - Rotation: Rotating an object around a point (usually the origin) by an angle \(\theta\). The new coordinates are calculated using trigonometric functions. - Scaling: Changing the size of an object by scaling factors \(S_x\) and \(S_y\) for the x and y coordinates respectively. Matrix Representations and Homogeneous Coordinates: - Transformation Matrices: Representing transformations using matrices, which allows for easy combination of transformations. - Homogeneous Coordinates: Using a third coordinate to represent 2D points, allowing translation to be represented as a matrix multiplication 2D Composite Transformations: -Combination of Transformations: Applying multiple transformations in sequence by multiplying their matrices. - Order of Transformations: The order in which transformations are applied matters because matrix multiplication is not commutative. Other 2D Transformations: - Reflection: Flipping an object over a line (e.g., the x-axis, y-axis, or an arbitrary line). - Shearing: Slanting the shape of an object. Raster Methods for Geometric Transformations: - Pixel-based Transformations: Techniques for transforming raster images (composed of pixels), such as rotation and scaling. OpenGL Raster Transformations: Raster Transformation Functions in OpenGL: Functions in OpenGL for transforming raster images. OpenGL Geometric Transformations Functions: - OpenGL Functions for 2D Transformations: Functions like `glTranslate`, `glRotate`, and `glScale` to perform transformations.Two Dimensional Translation We perform a translation on a single coordinate point by adding offsets to its coordinates so as to generate a new coordinate position Translation Equation Translation is a rigid-body transformation that moves objects without deformation.That is, every point on the object is translated by the same amount. A straight-line segment is translated by applying Equation 3 to each of the two line endpoints and redrawing the line between the new endpoint positions. A polygon is translated similarly. Two-Dimensional Rotation We generate a rotation transformation of an object by specifying a rotation axis and a rotation angle. All points of the object are then transformed to new positions by rotating the points through the specified angle about the rotation axis.As with translations, rotations are rigid-body transformations that move objects without deformation. Every point on an object is rotated through the same angle. A straight-line segment is rotated by applying Equations 9 to each of the two line endpoints and redrawing the line between the new endpoint positions. A polygon is rotated by displacing each vertex using the specified rotation angle and then regenerating the polygon using the new vertices. We rotate a curve by repositioning the defining points for the curve and then redrawing it. A circle or an ellipse, for instance, can be rotated about a noncentral pivot point by moving the center position through the arc that subtends the specified rotation angle. In addition, we could rotate an ellipse about its center coordinates simply by rotating the major and minor axes Two-Dimensional Scaling Scaling factors sx and sy can be assigned positive values, reducing or enlarging objects. A single value maintains object proportions, while unequal values result in differential scaling. Negative values can also be specified, resizing an object and reflecting it about coordinate axes. Polygons are scaled by applying transformations 14 to each vertex, then regenerating the polygon using the transformed vertices We apply the scaling transformation equations to the parameters defining the objects. To change the size of a circle, we can scale its radius and calculate the new coordinate positions around the circumference.Matrix Representations and Homogeneous Coordinates Graphics Applications and Geometric Transformations • Animation requires object translation and rotation. • Design and picture construction use translations, rotations, and scalings. • Viewing transformations involve translations and rotations from original scene to output device. • Reformulated matrix representations can efficiently process transformation sequences.Homogeneous Coordinates Combining multiplicative and translational terms into a single matrix. Expanding representations to 3 × 3 matrices. Using the third column of a transformation matrix for translation terms. Expressing transformation equations as matrix multiplications. Expanding matrix representation for

FIG 5.7: PDF UPLOAD

CONCLUSION AND FUTURE WORK

The "Text Summarization Using GPT Models" project successfully demonstrates the use of generative AI to automate text summarization. The developed tool offers multiple summarization styles, ensures high-quality output, and handles various text types. The user-friendly interface allows easy interaction, making the tool accessible to a wide range of users. By providing concise and coherent summaries, the project significantly enhances information processing and accessibility. Through its current capabilities, the project enables users to upload documents, generate concise and accurate summaries, and interact with the text in novel ways. The integration of Google Generative AI and the seamless user interface provided by Streamlit demonstrate the potential for AI to transform the way we interact with and comprehend large volumes of text. However, the potential for further enhancement is vast. Future developments can focus on integrating more advanced summarization models, supporting multiple languages, offering customizable summary options, and incorporating interactive and visual elements to improve user engagement. Enhancements in user experience, such as real-time feedback, topic clustering, and multi-document summarization, can further enrich the platform's functionality. Performance optimization, scalability, and robust security measures will ensure the platform remains reliable and secure, accommodating a growing user base and maintaining data privacy compliance. Additionally, expanding accessibility through mobile applications, API integrations, and features for users with disabilities will broaden the project's reach and utility. In conclusion, the Generative AI Text Summarization project stands at the forefront of AI-driven innovation, providing valuable tools for efficient text processing. With continuous improvement and adaptation, this project has the potential to revolutionize how we manage and derive insights from textual data, making information more accessible, comprehensible, and actionable for users across various domains.

FUTURE WORK

The Generative AI Text Summarization project, while currently functional and beneficial, has ample room for future enhancements. These improvements can focus on expanding the capabilities, improving the accuracy and efficiency, and providing a more seamless and user-friendly experience. Here are some potential future enhancements:

1. Enhanced Summarization Models

- **Integration of Advanced Models**: Incorporate the latest state-of-the-art models in natural language processing (NLP) like OpenAI's GPT-4 or future versions, which offer more accurate and nuanced text summarization capabilities.
- **Multilingual Support**: Extend the summarization capabilities to support multiple languages, allowing users to upload and summarize texts in various languages.
- **Customizable Summarization**: Allow users to customize the level of detail in summaries by setting parameters such as length, focus on specific sections, or inclusion of certain keywords.

2. Improved User Interface and Experience

- **Interactive Summaries**: Provide interactive summaries where users can click on summarized points to expand them for more detail, enhancing readability and user engagement.
- **Real-time Feedback**: Implement real-time feedback mechanisms where users can rate the quality of the summaries, which can be used to continuously improve the model.
- **Visual Summaries**: Incorporate visual elements such as infographics or mind maps to represent summarized content, making it easier for users to grasp key points quickly

3. Advanced Features and Functionalities

- **Contextual Question Answering**: Enhance the question-answering feature by enabling the model to understand and answer more complex and context-specific queries.

- **Topic Modeling and Clustering**: Integrate topic modeling to automatically identify and cluster related content, providing users with organized and thematically grouped summaries.
- **Summarization of Multiple Documents**: Enable the summarization of multiple documents simultaneously, providing an aggregated summary that highlights common themes and insights across different texts.

4. Performance and Scalability

- **Optimized Performance**: Optimize the backend processes to reduce the latency and improve the speed of text extraction and summarization, ensuring a smooth user experience even with large documents.
- **Scalable Infrastructure**: Develop a scalable architecture that can handle a high volume of requests and support concurrent users without compromising performance.

5. Security and Privacy

- **Data Privacy Compliance**: Ensure that the project complies with data privacy regulations such as GDPR or CCPA, particularly when handling sensitive or personal information in uploaded documents.
- **Secure Data Handling**: Implement robust security measures to protect user data, including secure file uploads, encrypted storage, and secure API communications.

6. Integration and Accessibility

- **API Integration**: Develop a robust API that allows third-party applications to leverage the text summarization capabilities, broadening the usage scenarios.
- **Mobile App Development**: Create mobile applications for iOS and Android platforms to make the summarization services more accessible to users on the go.
- **Accessibility Features**: Incorporate features such as text-to-speech and support for screen readers to make the platform accessible to users with disabilities.

7. Continuous Learning and Adaptation

- **Adaptive Learning Models**: Implement models that adapt and improve over time based on user interactions and feedback, continuously enhancing the summarization quality.
- **Domain-Specific Models**: Develop specialized models for different domains (e.g., legal, medical, academic) to provide more accurate and contextually relevant summaries for domain-specific texts.

By focusing on these future enhancements, the Generative AI Text Summarization project can evolve into a more powerful, user-friendly, and versatile tool that meets the diverse needs of users and stays at the forefront of technological advancements in AI-driven text processing

REFERENCES

[1] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training. OpenAI.

[2] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems (NeurIPS 2020).

[3] Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. Proceedings of the ACL-04 Workshop, 74–81.

[4] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020). Transformers: State-of-the-Art Natural Language Processing. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations.

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is All You Need. Advances in Neural Information Processing Systems, 5998-6008.

[6] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding by Devlin et al. (2018).

[7] Streamlit for Teams Tutorial https://youtu.be/R2nr1uZ8ffc

[8] A Survey on Text Summarization: Approaches, Datasets, Evaluation Metrics, and Recent Advances by Guo et al. (2022).

[9] Abstractive Text Summarization using Sequence-to-Sequence RNNs and Beyond by Chopra et al. (2016).

[10] Neural Abstractive Text Summarization with Sequence-to-Sequence Models by Rush et al. (2015).