A

PROJECT REPORT

ON

LangChain: PDF Chat Assistant

BY

Mr. Utkarsh Kamthankar
Mr. Sanket Bhosale

Submitted in Partial fulfillment of

Post Graduation Diploma in Data Science and AI

Savitribai Phule Pune University

For the Academic Year
2023-2024

UNDER THE GUIDANCE OF

Prof. Mruganayani Bhamare

Department of Technology
Savitribai Phule Pune University,
Ganeshkhind, Pune-411007

DEPARTMENT OF TECHNOLOGY

CERTIFICATE

This is to certify that
Utkarsh Kamthankar and **Sanket Bhosale**

has successfully completed her project on

in partial fulfillment of

2nd Semester work for his Post Graduation Diploma in Data Science and AI

under Savitribai Phule Pune University, for the academic year 2023-2024.

Prof. Mruganayani Bhamare (Project Guide)	Dr. Manisha Bharati (Course Coordinator)	Dr. Aditya Abhyankar (H.O.D)

Signed By
(External Examiner)

Place: Pune

Date: 18-05-2024

Students Declaration

I undersigned a student of the Department of Technology, Savitribai Phule Pune University, Pune PGD Data Science and AI – 2nd semester, declare that the summer internship project **LangChain: PDF Chat Assistant** is
a result of my own work and my indebtedness to other work publications, references, if any, have been duly acknowledged. If I am found guilty of copying any other report or published information and showing it as myoriginal work, I understand that I shall be liable and punishable by the Institute or University, which may include Fail in the examination, repeat study and resubmission of the report or any other punishment that Institute or University may decide.

Name of Student: Utkarsh Kamthankar
Enrolment No.: PGD23DS89
Signature:

Name of Student: Sanket Bhosale
Enrolment No.: PGD23DS16
Signature:

ACKNOWLEDGEMENT

I am deeply grateful to all those who have played a pivotal role in shaping my internship project and enriching my learning experience. I extend my heartfelt gratitude to:

Prof. Mruganayani Bhamare

(Professor and Project Guide from the Department of Technology, SPPU) For her scholarly guidance, insightful suggestions, and continuous encouragement throughout the course of my project.

The entire faculty of the Department of Technology, Savitribai Phule Pune University, for fostering an environment of academic excellence and nurturing my curiosity.

I am profoundly thankful to everyone mentioned above for their selfless contributions to my journey. Their mentorship and support have been instrumental in shaping my skills and fostering personal growth.
Thank You

Mr. Utkarsh Kamthankar
Mr. Sanket Bhosale

INDEX

CHAPTER 1

INTRODUCTION AND RATIONALE OF THE STUDY

1.1 Introduction to the title

This project presents a novel approach to information retrieval from PDFs using conversational interaction. It leverages the power of LangChain, an orchestration framework, and Google Gemini, a state-of-the-art large language model (LLM), to create a "Langchain PDF Chat Assistant." This interactive application allows users to engage in natural language conversations with the content of multiple PDF documents.

The core innovation lies in LangChain's ability to integrate seamlessly with Google Gemini. LangChain acts as the conductor, processing user queries and retrieving relevant information from parsed PDFs using tools like PyPDF2 and chromadb. This information is then fed to Google Gemini, enabling it to generate accurate and contextually relevant responses within the conversational flow.

A key advantage of this system is its extended token length capacity, allowing it to handle even extensive and detailed PDFs effectively. This ensures that the context and integrity of large datasets are maintained throughout the conversation.

The Langchain PDF Chat Assistant showcases the potential of LangChain and Google Gemini to revolutionize PDF interaction. It offers a user-friendly and intuitive way to access and explore information within these documents, fostering efficient information retrieval and knowledge extraction. This project not only improves PDF accessibility but also paves the way for innovative applications of LLMs and conversational AI in the domain of document analysis and information retrieval.

1.2 Significance of the study

The Langchain PDF Chat Assistant project holds significant potential for transforming the way users interact with and extract information from PDF documents. By integrating advanced natural language processing (NLP) and machine learning techniques, this project addresses several critical challenges and offers numerous benefits across various domains.

Key Benefits and Contributions

1. **Enhanced Efficiency**: Traditional methods of searching through PDF documents are often labor-intensive and time-consuming, requiring users to manually sift through pages to find specific information. The Langchain PDF Chat Assistant streamlines this process by allowing users to input natural language queries and receive precise answers within seconds. This efficiency is particularly valuable in environments where time is a critical factor, such as legal research, academic studies, and business decision-making.
2. **Improved Accuracy**: By leveraging sophisticated NLP models and embedding techniques, the Langchain PDF Chat Assistant ensures high accuracy in understanding user queries and retrieving relevant information. This accuracy reduces the likelihood of errors and omissions, providing users with reliable data that can be confidently used for various purposes. The ability to accurately interpret complex queries and contextually match them with the relevant content in the PDF documents sets this tool apart from conventional search methods.
3. **Scalability**: The architecture of the Langchain PDF Chat Assistant is designed to handle large volumes of data and multiple documents simultaneously. This scalability makes the application suitable for use in environments where extensive document processing is required, such as libraries, corporate archives, and educational institutions. The system's capacity to efficiently manage and process large datasets ensures that it remains effective even as the volume of documents and complexity of queries increase.
4. **User-Friendly Interface**: One of the significant advantages of the Langchain PDF Chat Assistant is its intuitive and accessible interface. Developed using Streamlit, the application provides a seamless user experience, allowing users to upload documents, ask questions, and receive answers without needing technical expertise. This user-centric design ensures that the tool is accessible to a broad audience, including students, researchers, professionals, and casual users.
5. **Innovative Technological Integration**: The project demonstrates the innovative integration of various advanced technologies, including PyPDF2 for PDF parsing, Sentence Transformers for embedding generation, FAISS for vector storage, and Langchain for query processing. This combination of tools and libraries showcases the potential of interdisciplinary approaches to solving complex information retrieval challenges.

6. The project serves as a model for future innovations in the field of document analysis and conversational AI.
7. **Educational and Professional Applications**: The Langchain PDF Chat Assistant has wide-ranging applications in both educational and professional settings. In education, students and educators can use the tool to quickly locate specific information within textbooks, research papers, and course materials, enhancing the learning experience and supporting academic success. In professional settings, lawyers, researchers, and business professionals can benefit from rapid access to relevant information within legal documents, research articles, business reports, and technical manuals, facilitating informed decision-making and efficient workflows.

Broader Implications

The Langchain PDF Chat Assistant project also holds broader implications for the future of document interaction and information retrieval:

- **Advancements in NLP and AI**: The successful implementation of this project contributes to the growing body of knowledge and advancements in NLP and AI. By showcasing the practical applications of these technologies, the project encourages further research and development in the field, potentially leading to even more sophisticated and capable systems.
- **Empowering Users**: By providing a tool that simplifies the process of extracting information from complex documents, the Langchain PDF Chat Assistant empowers users to become more self-sufficient and efficient in their work. This empowerment can lead to increased productivity, improved outcomes, and greater satisfaction in both personal and professional endeavors.
- **Setting a Precedent**: The project's innovative approach and successful outcomes set a precedent for future developments in document analysis and conversational AI. It demonstrates the feasibility and benefits of integrating advanced technologies to address real-world challenges, inspiring other developers and researchers to explore similar solutions.

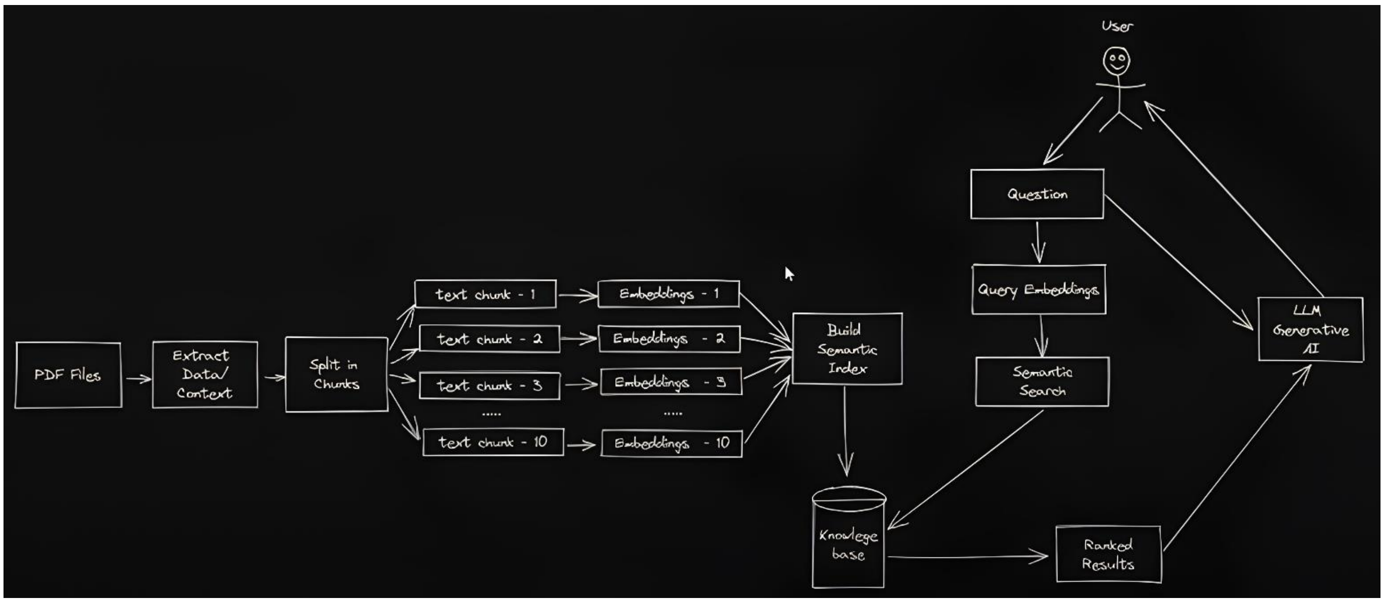
CHAPTER 2

THEORETICAL FRAMEWORK (PROJECT) / REVIEW OF LITERATURE

Paper Title	Authors	Key Points	Significance
LangChain: Building Powerful Language Applications	Harrison Chase, The LangChain Team	Comprehensive introduction to LangChain framework, outlining core concepts, components, and use cases.	Provides a foundational understanding of LangChain and its functionalities.
Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks	Patrick Lewis, et al.	Reviews RAG models, highlighting their capabilities in knowledge-intensive NLP tasks. Explores various RAG architectures and benefits of combining retrieval with generation.	Establishes RAG as a powerful approach for incorporating external knowledge into NLP tasks.
Retrieval-Augmented Generation for Text Summarization	Wei Yang, et al.	Proposes a RAG-based approach for text summarization, utilizing an external knowledge source to enhance the summarization process. Shows effectiveness in generating more informative and comprehensive summaries.	Demonstrates the applicability of RAG in specific NLP tasks, showcasing its potential for improving summarization quality.
Towards Conversational Question Answering over Multiple Documents	Shaoxiong Feng, et al.	Tackles conversational question answering over multiple documents using RAG. Integrates efficient document retrieval and context-aware response generation.	Underscores the importance of RAG for handling multi-document question answering, crucial for document search and knowledge extraction.
RAG: Efficient and Effective Retrieval-Augmented Generation for Question Answering	Shengyao Zhu, et al.	Introduces RAG-Q, a novel RAG architecture for question answering, utilizing a dense retriever for selecting relevant documents and a generative model for concise answers.	Highlights the efficiency and effectiveness of RAG for addressing question answering tasks, showcasing its potential for powerful information retrieval systems.
Generative Pre-trained Transformer 3 (GPT-3): Language Modeling for Long-Form Content Creation	Tom Brown, et al.	Introduces GPT-3, a large language model with impressive performance in various NLP tasks like text generation, translation, and question answering.	Emphasizes significant advancements in LLM capabilities, paving the way for more sophisticated applications like RAG systems.
Scaling Laws for Neural Language Models	Jared Kaplan, et al.	Explores the relationship between the size of language models and their performance, demonstrating significant improvements in performance with scaling.	Provides insights into the potential of scaling LLMs for achieving better performance and driving innovation in RAG and other applications.

Towards a Human-Level Understanding of Natural Language	Jacob Devlin, et al.	Discusses challenges and progress in achieving human-level understanding of natural language. Highlights the importance of incorporating external knowledge into language models, emphasizing the potential of RAG systems.	Provides a broader context for understanding the role of RAG in advancing natural language understanding.
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding	Jacob Devlin, et al.	Introduces BERT (Bidirectional Encoder Representations from Transformers), a pre-trained language model achieving state-of-the-art performance in various NLP tasks.	BERT is a foundation for many language models used in RAG systems, demonstrating the impact of pre-training techniques on model performance.
Zero-Shot Text-to-SQL Task: A Survey of Recent Approaches	Anirudh Reddy, et al.	Focuses on zero-shot text-to-SQL tasks, a critical challenge in natural language understanding. Investigates RAG systems as potential solutions, showcasing their capacity for bridging the gap between natural language and structured queries.	Demonstrates the versatility of RAG systems in tackling complex NLP tasks, highlighting their potential for diverse real-world applications.

2.1 PROPOSED ARCHITECHTURE



This proposed architecture diagram illustrates the workflow of a PDF chat assistant system that leverages the LangChain library. Let me break it down step by step:

1. **PDF Files**: The system starts with PDF files as the input data source.
2. **Extract Data/Context**: The PDF files are processed to extract the textual content and relevant data.
3. **Split into Chunks**: The extracted text data is split into smaller chunks or segments for more efficient processing.
4. **Embeddings**: For each text chunk, embeddings are generated. Embeddings are numerical representations of the text that capture semantic and contextual information, making it easier to find relevant information later.
5. **Build Semantic Index**: The embeddings of all the text chunks are used to build a semantic index. This index serves as a search engine, allowing efficient retrieval of relevant information based on semantic similarity.
6. **Query Embeddings**: When a user asks a question, the question text is converted into embeddings using the same embedding model.
7. **Semantic Search**: The query embeddings are compared against the semantic index using a similarity search algorithm. This step identifies the most relevant text chunks from the PDF documents that are semantically similar to the user's query.
8. **Knowledge Base**: The relevant text chunks are combined and treated as a knowledge base specific to the user's query.

9. **LLM (Generative AI)**: The knowledge base and the original query are passed as input to a large language model (LLM), such as GPT-3 or a custom fine-tuned model. The LLM generates a human-readable answer based on the provided context.
10. **Ranked Results**: The generated answer from the LLM is presented to the user as the final output, potentially ranked or scored based on relevance or confidence measures.

This architecture leverages the power of language models and semantic search to provide natural language responses to user queries based on the content of PDF documents. The modular design allows for flexibility in swapping components like the LLM or embedding model as needed. Additionally, the use of LangChain simplifies the integration and management of these components within a unified framework.

CHAPTER 3

OBJECTIVES AND SCOPE OF PROJECT

3.1 Objectives:

The primary objective of this project is to develop an interactive PDF chat assistant that can understand natural language queries from users and provide relevant and coherent responses based on the information contained in a given set of PDF documents.

The key goals and objectives of the project are as follows:

1. **Information Extraction**: The system should be capable of extracting textual content and relevant data from PDF files, which serve as the primary information source. This involves techniques for parsing and interpreting the structured and unstructured data within PDFs.
2. **Semantic Understanding**: The extracted text needs to be transformed into numerical representations (embeddings) that capture the semantic and contextual meaning of the content. This semantic understanding is crucial for enabling relevant information retrieval and natural language processing tasks.
3. **Efficient Indexing and Retrieval**: The project aims to build an efficient semantic index that can quickly retrieve the most relevant information from the PDF documents based on user queries. This involves implementing advanced techniques for similarity search and nearest neighbor retrieval using the embeddings.
4. **Natural Language Processing**: A key objective is to enable natural language understanding and generation capabilities within the system. This involves integrating large language models (LLMs) or other NLP models that can comprehend user queries, reason over the retrieved information, and generate human-like responses.
5. **Context-Aware Answering**: The system should be able to provide contextually relevant and coherent answers by combining the relevant information from the PDF documents with the language understanding capabilities of the LLM. The goal is to provide informative and accurate responses tailored to the user's specific query.
6. **Interactive and Conversational Experience**: The project aims to create an interactive and conversational experience for users, allowing them to engage in natural language dialogs with the system. This includes handling follow-up questions, clarifications, and maintaining context throughout the conversation.

7. **Scalability and Performance**: The architecture should be designed to handle large volumes of PDF documents and queries efficiently, ensuring responsive performance and scalability as the dataset and usage grow.
8. **Modularity and Extensibility**: The project should have a modular and extensible architecture, allowing for easy integration of different components (e.g., embedding models, LLMs, retrieval strategies) and enabling future enhancements or modifications as required.
9. **User-Friendly Interface**: Developing an intuitive and user-friendly interface is essential to ensure a seamless experience for users when interacting with the PDF chat assistant, whether it's a web-based interface, a desktop application, or a conversational agent integrated into other platforms.

3.2 Scope:

The project aims to develop an intelligent and interactive PDF chat assistant capable of understanding natural language queries from users and providing relevant and coherent responses based on the information contained within a given set of PDF documents. The functional scope encompasses the ingestion and processing of PDF files in various formats, including extracting textual content, metadata, and relevant information. Techniques for text chunking and generating embeddings (numerical representations) will be implemented to enable semantic understanding and efficient information retrieval.

A crucial aspect is the development of a semantic index and efficient retrieval mechanism, employing similarity search and nearest neighbor algorithms for fast and relevant information lookup. Natural language processing (NLP) capabilities will be integrated, including language understanding models for comprehending user queries and language generation models for producing human-like responses. The system will enable conversational interactions through a chat interface, understanding and responding to natural language queries related to the PDF content.

The assistant will provide context-aware responses by considering the user's query, previous interactions, and relevant PDF content. It will handle follow-up questions, clarifications, and maintain conversational state and context. Ranking and scoring mechanisms will be implemented to prioritize and present the most relevant responses.

In terms of non-functional requirements, the system must be scalable to handle large volumes of PDF documents and concurrent user queries while maintaining efficient performance and response times. A modular and extensible architecture will be developed to allow for easy integration of different components and future enhancements. Robust error handling, security practices, and adherence to relevant data privacy regulations will be ensured.

The user experience is a priority, with a focus on developing a user-friendly and intuitive interface design that is cross-platform compatible and responsive across different devices and screen sizes. Logging, monitoring, and performance tracking capabilities will be incorporated for troubleshooting and optimization purposes.

However, certain features are considered out of scope for this project, including Optical Character Recognition (OCR) for scanned PDFs, advanced document layout analysis and rendering, support for languages other than English, integration with external data sources beyond the provided PDF documents, advanced natural language generation capabilities for long-form or creative content, and personalization or user profiling features.

By adhering to this well-defined scope, the project aims to deliver a powerful and intelligent PDF chat assistant that bridges the gap between the wealth of information contained in PDF documents and users' natural language queries, providing a conversational and context-aware experience for information access and knowledge extraction.

CHAPTER 4

RESEARCH METHODOLOGY

4.1 Rationale for the Study.

The motivation behind this s”tudy arises from the pervasive challenge of extracting relevant information from PDF documents, which are a standard format for sharing and storing information across various fields such as academia, business, and research. PDF documents are designed to preserve formatting and content, making them ideal for presenting information but not for interactive data retrieval. Users often need to manually sift through extensive documents to find specific information, which is time-consuming and inefficient.

Importance of the Study:

- Academic and Research Fields: Researchers and students frequently need to locate specific information from lengthy academic papers, theses, and research reports. An efficient system can save significant time and effort.
- Business and Corporate Environment: Businesses rely on PDF documents for contracts, reports, and internal documents. Quick retrieval of pertinent information can enhance decision-making and operational efficiency.
- General Users: Everyday users dealing with PDF manuals, e-books, and reports can benefit from a system that allows easy access to desired information without manual searching.

By addressing this gap, the study aims to enhance the usability and efficiency of interacting with PDF documents, leveraging the latest advancements in AI and language processing.

4.2 Statement of Problem

The core problem this study addresses is the inefficiency and difficulty associated with extracting specific information from PDF documents. Traditional search functions in PDF readers and editors are limited to keyword matching and do not understand the context or provide detailed answers. Users are often left to manually search through pages of text to find relevant information, which is not only time-consuming but also prone to errors and omissions.

Specific Issues:

- Limited Search Capabilities: Traditional search tools do not understand context and can miss relevant information if the exact keywords are not used.

- Manual Effort Required: Users have to manually navigate through the document, which is inefficient especially for large documents.
- Lack of Detailed Answers: Even if a keyword is found, users often need to read surrounding text to understand the full context, which is not provided by traditional search tools.

The study aims to solve these issues by developing a system that can understand and process the context of the information within PDF documents, providing users with accurate and detailed responses to their queries.

4.3 Significance of the Problem

The significance of this problem is evident in its wide-ranging impact on various sectors:

Academic Sector:

- Researchers and students can dramatically reduce the time spent searching for specific information in academic papers and textbooks.
- Enhances the ability to quickly gather literature for reviews and citations.

Business Sector:

- Businesses can streamline their operations by quickly accessing specific sections of reports, contracts,
- Enhances productivity by reducing the time employees spend on document searches.

Everyday Use:

- General users, such as readers of e-books or manuals, can benefit from quick answers to their specific questions.

Improves the accessibility of information for all users, including those with disabilities who may find manual searches particularly challenging.

By addressing these issues, the proposed solution has the potential to greatly enhance efficiency, , and user satisfaction across various applications.

4.4 Research Objectives

The research aims to achieve the following objectives in detail:

1. Develop a System for PDF Text Extraction and Chunking:

- Implement a robust mechanism to extract text from PDF documents using PyPDF2.
- Develop an efficient process to split the extracted text into smaller, manageable chunks for better processing and context management using Langchain’s RecursiveCharacterTextSplitter.

2. Implement Vector Storage for Efficient Similarity Search:

- Utilize FAISS to create and manage a vector store that can efficiently handle similarity searches.
- Ensure that the vector store is optimized for quick retrieval of relevant text chunks based on user

queries.

3. Integrate Advanced Conversational AI:

- Leverage Google Generative AI models to generate embeddings and handle user queries.
- Develop a conversational AI chain that can understand the context and provide accurate, detailed answers.

4. Design a User-Friendly Interface:

- Use Streamlit to create an intuitive and user-friendly web interface.
- Ensure the interface supports easy PDF uploads, text input for questions, and displays detailed answers.

5. Evaluate System Performance and User Satisfaction:

- Conduct comprehensive testing to assess the system’s performance, accuracy, and reliability.
- Collect user feedback to evaluate satisfaction and identify areas for improvement.

4.5 Scope of the Study

The scope of this study includes the following components and activities in detail:

- PDF Text Extraction:
- Implement PyPDF2 to extract text content from various types of PDF documents.
- Ensure compatibility with different PDF formats and structures.
- Text Chunking:
- Develop a method to split the extracted text into chunks of approximately 10,000 characters with an overlap of 1,000 characters to maintain context.
- Handle edge cases such as incomplete sentences or paragraphs at chunk boundaries.
- Vector Storage Using FAISS:
- Create a vector store using FAISS to store text chunks as vectors.
- Implement efficient similarity search algorithms to retrieve relevant text chunks based on user queries.
- Conversational AI Integration:
- Use Google Generative AI for embedding generation and conversational responses.
- Develop a prompt template that ensures accurate and context-aware responses to user questions.
- User Interface Design:
- Design a Streamlit-based web interface that allows users to upload PDF documents, input questions, and view responses.
- Ensure the interface is responsive, accessible, and easy to use.
- Testing and Evaluation:

- Conduct unit testing for individual components to ensure functionality and reliability.
- Perform integration testing to validate the interactions between components.
- Gather user feedback through usability testing and surveys to assess satisfaction and identify areas for improvement.

4.6 Research Hypothesis

The research hypothesis for this study is:

"Integrating PDF processing with advanced conversational AI models will significantly improve the efficiency and accuracy of information retrieval from PDF documents compared to traditional search methods."

Explanation of the Hypothesis:

- PDF Processing Integration:
- By extracting and chunking text from PDF documents, the system can manage and process large volumes of text more effectively than traditional methods.
- Advanced Conversational AI Models:
- Utilizing state-of-the-art AI models such as Google Generative AI allows for contextual understanding and detailed responses to user queries, unlike simple keyword matching used in traditional searches.
- Efficiency and Accuracy Improvement:
- The hypothesis posits that the combined approach will reduce the time users spend searching for information and increase the accuracy of the information retrieved.
- The system's ability to understand and process context will lead to more relevant and comprehensive answers, enhancing user satisfaction and productivity.

Validation of the Hypothesis:

- The hypothesis will be validated through extensive testing and evaluation, comparing the performance of the proposed system with traditional search methods.
- Metrics such as response time, accuracy of answers, and user satisfaction will be measured to assess the effectiveness of the system.
- User feedback and performance data will be analyzed to determine the extent of improvement achieved by integrating advanced AI models with PDF processing.

CHAPTER 5
INTRODUCTION TO LANGCHAIN

Introduction:

Langchain is an advanced framework designed to facilitate the development and deployment of applications involving natural language processing (NLP) and understanding. It leverages state-of-the-art language models to enable a variety of language-based tasks, such as text generation, summarization, conversational agents, and question-answering systems. Langchain aims to simplify the integration of language models into real-world applications by providing a comprehensive set of tools and APIs. Key features of Langchain include its modular architecture, support for multiple language models, robust text processing capabilities, embedding generation, vector storage solutions like FAISS, and pre-built conversational AI chains.

Langchain supports integration with various external APIs and databases, enabling seamless data flow and interaction between language models and other systems. Applications of Langchain range from interactive document processing and customer support bots to virtual assistants, content generation, and educational tools. By providing essential components for tasks such as text extraction, splitting, embedding generation, similarity search, and conversational AI integration, Langchain empowers developers to create sophisticated language-based applications that are effective and user-friendly.

Key Benefits of Langchain

1. Modular Architecture:
    - Allows for easy customization and extension.
    - Enables developers to plug in various components as needed for different NLP tasks.
2. Support for Multiple Language Models:
    - Compatible with both open-source and proprietary APIs.
    - Provides flexibility to select the most suitable model for specific needs.
3. Robust Text Processing Tools:
    - Includes tools for text extraction, splitting, and chunking.
    - Essential for handling large documents and preparing text data for analysis.
4. Advanced Embedding and Vector Storage:
    - Generates text embeddings that capture semantic meaning.
    - Uses FAISS for efficient vector storage and similarity search functionalities.
5. Simplified Creation of Conversational Agents:

- Offers pre-built conversational chains.
- Integrates language models with prompt templates and context management.
- Facilitates the development of coherent and contextually relevant responses.

6. Integration with External APIs and Databases:
    - Supports seamless data flow and interaction between language models and other systems.
    - Enhances the capabilities of language-based applications by enabling complex operations based on user inputs.

CHAPTER 6
APPLICATION LOGIC

Block 1: Importing Libraries and Configuring API Key

- This block imports all the necessary libraries for the application to function. Here's a breakdown of some key ones:
- streamlit: Creates the web application interface.
- PyPDF2: Reads and extracts text from PDF files.
- langchain: Provides utilities for text processing and question answering.
- Google Generative AI: Interacts with Google's AI services for generating embeddings and responses.
- dotenv: Loads environment variables from a .env file (likely containing the API key).
- load_dotenv(): Reads the .env file and sets those variables as environment variables.
- genai.configure: Sets up the Google Generative AI library with the API key retrieved from the environment variable.

Block 2: Extracting Text from PDF Files

- This block defines a function get_pdf_text(pdf_docs) that takes a list of uploaded PDF files as input.
- It iterates through each PDF using a loop:
- A PdfReader object is created to read the PDF content.
- Each page of the PDF is processed using another loop.
- The text is extracted from each page using page.extract_text().
- The extracted text from all pages is concatenated and stored in a variable.
- Finally, the function returns the combined text extracted from all uploaded PDFs.

Block 3: Splitting Text into Chunks

- This block defines a function get_text_chunks(text) that takes a large chunk of text as input.
- It creates a RecursiveCharacterTextSplitter object, which splits the text into smaller, manageable pieces.
- Each chunk is 10,000 characters long (chunk_size=10000).
- There's an overlap of 1,000 characters (chunk_overlap=1000) to ensure context is maintained between chunks.
- The function uses the splitter to split the text and returns a list of these smaller text chunks.

Block 4: Creating a Vector Store

• This block defines a function get_vector_store(text_chunks) that takes the list of text chunks as input.

• It creates a GoogleGenerativeAIEmbeddings object, which generates numerical representations (embeddings) for each text chunk using a specified AI model (model="models/embedding-001").

• It then uses FAISS.from_texts to create a vector store from the embeddings of all the text chunks. This allows for efficient retrieval of similar text based on the embeddings.

• Finally, the function saves the created vector store locally using vector_store.save_local("faiss_index").

Block 5: Creating a Conversational Chain

• This block defines a function get_conversational_chain() that sets up the process for generating responses to user questions.

• It defines a PromptTemplate that specifies the format for the prompts sent to the AI model. The template includes placeholders for context (extracted text) and the user's question.

• It creates a ChatGoogleGenerativeAI object, specifying the AI model (model="gemini-pro") and a parameter called temperature (controls the randomness of the generated text).

• It uses load_qa_chain to create a question-answering chain that combines the AI model, prompt template, and specifies the type of chain (chain_type="stuff").

• Finally, the function returns this conversational chain, which will be used to generate answers.

Block 6: Handling User Input and Generating Response

• This block defines an async function user_input(user_question) that takes the user's question as input.

• It creates a GoogleGenerativeAIEmbeddings object similar to block 4 to generate embeddings for the user's question.

• It loads the previously saved vector store (FAISS.load_local) using the same embedding model.

• It performs a similarity search using docs = new_db.similarity_search(user_question). This retrieves the text chunks from the store that are most similar to the user's question based on their embeddings.

• It retrieves the conversational chain created in block 5 using get_conversational_chain().

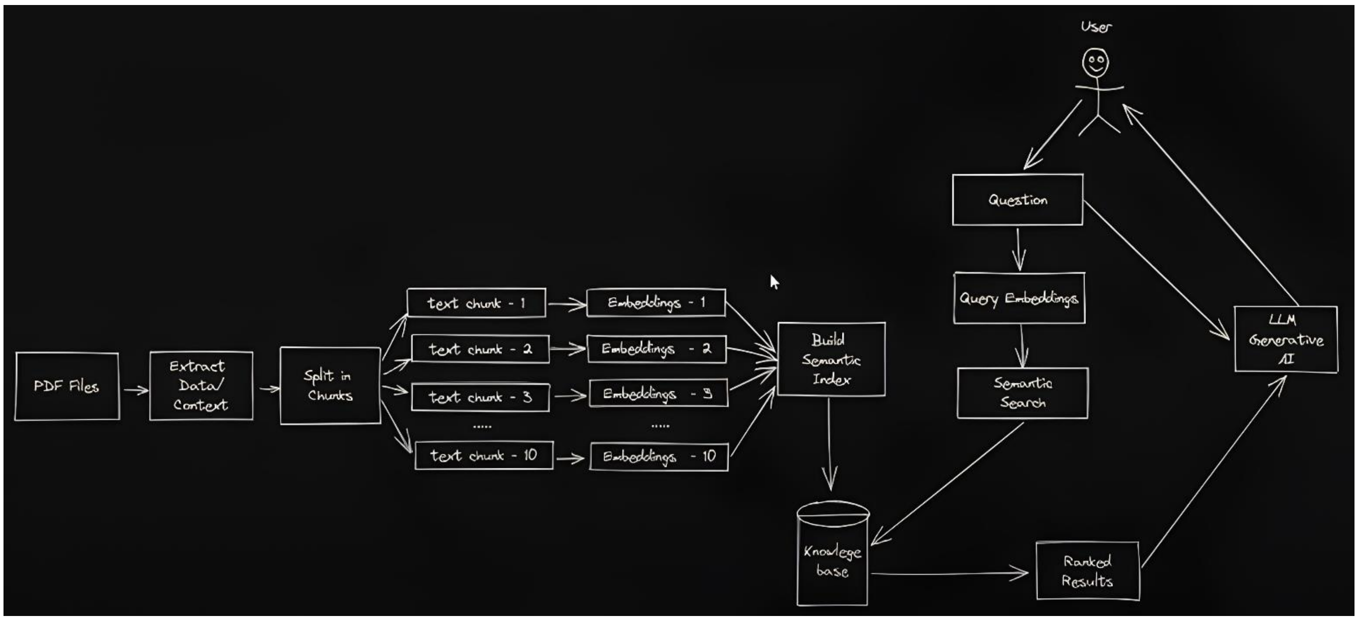
• It feeds the retrieved text chunks ("input_documents") and the user's question ("question") into the conversational chain.

• It extracts only the generated response text ("output_text") from the chain's output.

```
import streamlit as st
from PyPDF2 import PdfReader
from langchain.text_splitter import RecursiveCharacterTextSplitter
import os
from langchain_google_genai import GoogleGenerativeAIEmbeddings
```

```
import google.generativeai as genai
from langchain.vectorstores import FAISS
from langchain_google_genai import ChatGoogleGenerativeAI
from langchain.chains.question_answering import load_qa_chain
from langchain.prompts import PromptTemplate
from dotenv import load_dotenv
import asyncio

load_dotenv()
genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))

def get_pdf_text(pdf_docs):
    text = ""
    for pdf in pdf_docs:
        pdf_reader = PdfReader(pdf)
        for page in pdf_reader.pages:
            text += page.extract_text()
    return text

def get_text_chunks(text):
    text_splitter = RecursiveCharacterTextSplitter(chunk_size=10000,
chunk_overlap=1000)
    chunks = text_splitter.split_text(text)
    return chunks

def get_vector_store(text_chunks):
    embeddings = GoogleGenerativeAIEmbeddings(model="models/embedding-001")
    vector_store = FAISS.from_texts(text_chunks, embedding=embeddings)
    vector_store.save_local("faiss_index")

def get_conversational_chain():
    prompt_template = """
    Answer the question as detailed as possible from the provided context, make sure
to provide all the details, if the answer is not in
    provided context just say, "answer is not available in the context", don't
provide the wrong answer\n\n
    Context:\n {context}?\n
    Question: \n{question}\n

    Answer:
    """

    model = ChatGoogleGenerativeAI(model="gemini-pro", temperature=0.3)
    prompt = PromptTemplate(template=prompt_template, input_variables=["context",
"question"])
    chain = load_qa_chain(model, chain_type="stuff", prompt=prompt)

    return chain

async def user_input(user_question):
    embeddings = GoogleGenerativeAIEmbeddings(model="models/embedding-001")
    new_db = FAISS.load_local("faiss_index", embeddings)
    docs = new_db.similarity_search(user_question)
    chain = get_conversational_chain()
    response = chain({"input_documents": docs, "question": user_question},
return_only_outputs=True)
    return response["output_text"]

def main():
    st.set_page_config(page_title="Chat PDF")
    st.header("Langchain PDF Chat Assistant💁")

    user_question = st.text_input("Ask a Question from the PDF Files")

    if user_question:
        with st.spinner("Generating answer..."):
            loop = asyncio.new_event_loop()
            asyncio.set_event_loop(loop)
            response = loop.run_until_complete(user_input(user_question))
            st.write("Reply: ", response)

    with st.sidebar:
        st.title("Menu:")
        pdf_docs = st.file_uploader("Upload your PDF Files and Click on the Submit &
Process Button", accept_multiple_files=True)
        if st.button("Submit & Process"):
            with st.spinner("Processing..."):
                raw_text = get_pdf_text(pdf_docs)
                text_chunks = get_text_chunks(raw_text)
                get_vector_store(text_chunks)
                st.success("Done")

if __name__ == "__main__":
    main()
```

CHAPTER 7
IMAGES



Flow of the Application

The flowchart illustrates the process of how the Langchain PDF Chat Assistant works:

1. **Data Preparation:**

   PDF Files: Upload PDF files.

   Extract Data/Context: Extract text from PDFs.

   Split in Chunks: Split text into smaller chunks.

2. **Index Creation:**

   Text Chunks: Convert each chunk into embeddings.

   Build Semantic Index: Create a semantic index from the embeddings.

   Knowledge Base: Store the index in a knowledge base.

3. **Query Processing:**

   User: User asks a question.

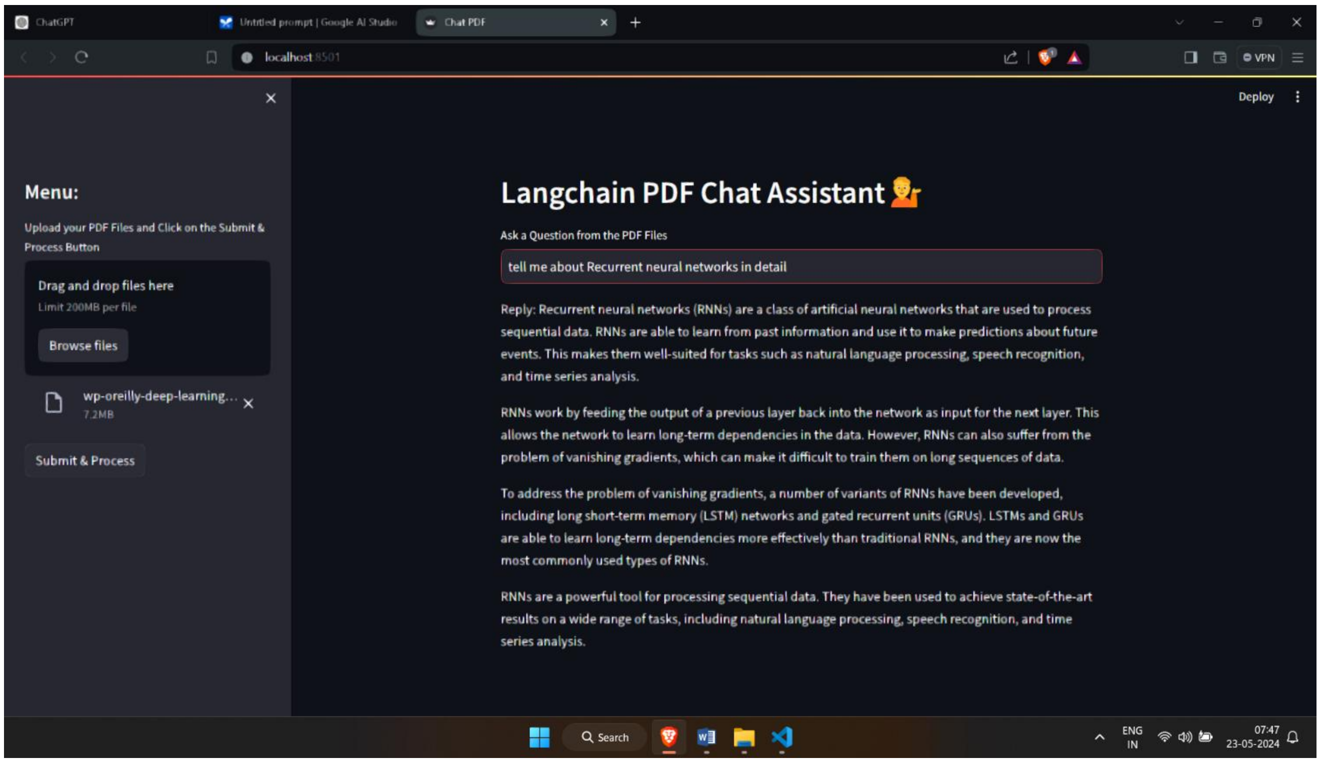
   Query Embeddings: Convert the question into embeddings.

   Semantic Search: Search the knowledge base for relevant chunks.

   Ranked Results: Retrieve and rank the relevant chunks.

4. **Answer Generation**:

   LLM Generative AI: Use the top-ranked chunks to generate a detailed answer.



Langchain PDF Chat Assistant

Ask a Question from the PDF Files

tell me about Recurrent neural networks in detail

Reply: Recurrent neural networks (RNNs) are a class of artificial neural networks that are used to process sequential data. RNNs are able to learn from past information and use it to make predictions about future events. This makes them well-suited for tasks such as natural language processing, speech recognition, and time series analysis.

RNNs work by feeding the output of a previous layer back into the network as input for the next layer. This allows the network to learn long-term dependencies in the data. However, RNNs can also suffer from the problem of vanishing gradients, which can make it difficult to train them on long sequences of data.

To address the problem of vanishing gradients, a number of variants of RNNs have been developed, including long short-term memory (LSTM) networks and gated recurrent units (GRUs). LSTMs and GRUs are able to learn long-term dependencies more effectively than traditional RNNs, and they are now the most commonly used types of RNNs.

RNNs are a powerful tool for processing sequential data. They have been used to achieve state-of-the-art results on a wide range of tasks, including natural language processing, speech recognition, and time series analysis.

The image shows a web application interface for a "LangChain PDF Chat Assistant". The main purpose of this application is to allow users to upload PDF files and then ask questions related to the content of those PDF files. The application will process the uploaded PDF files, extract relevant information, and provide natural language responses to the user's queries based on the PDF content.

The interface has a section for users to drag and drop or browse and upload PDF files. Once the files are uploaded, users can enter their questions in a text box. The example question shown is "tell me about Recurrent neural networks in detail". The application has provided a detailed response explaining recurrent neural networks, their characteristics, variants like LSTMs and GRUs, their applications in natural language processing, speech recognition, and time series analysis, as well as mentioning the vanishing gradient problem.

Overall, this application leverages natural language processing and information retrieval techniques to enable users to interact with and query the contents of PDF documents in a conversational manner

CHAPTER 8
CONCLUSION

Langchain Flow in Action:

1. **Data Preparation:**
   - Upon launching the application (within the virtual environment with required libraries), the user interacts with the Streamlit interface.
   - When PDFs are uploaded, the `get_pdf_text` function extracts text from each PDF and combines it.
   - The `get_text_chunks` function splits this large text into smaller, more manageable pieces for better processing by Langchain and the AI model.
2. **Building the Knowledge Base:**
   - The `get_vector_store` function utilizes Langchain's capabilities:
     - It generates numerical representations (embeddings) for each text chunk using a specified Google Generative AI model. These embeddings capture the essence of the text.
     - Langchain's FAISS library is used to create a vector store from these embeddings. This store allows for efficient retrieval of similar text chunks based on their embeddings when the user asks a question.
     - The vector store is saved locally for faster access in future interactions.
3. **Question Answering System Setup:**
   - The `get_conversational_chain` function defines the core of the question-answering system:
     - It creates a `PromptTemplate` within Langchain, specifying how questions will be presented to the AI model. The template includes placeholders for the context (extracted text chunks) and the user's question.
     - A `ChatGoogleGenerativeAI` object is created, specifying the AI model to be used and a parameter called `temperature` (influencing the randomness of the generated text).
     - Finally, Langchain's `load_qa_chain` function combines the AI model, prompt template, and other parameters into a question-answering chain. This chain essentially defines the workflow for generating informative answers to user queries.
4. **User Interaction and Response Generation:**
   - The main function of the application manages user interaction and response generation:
     - Users ask questions in the designated text input field.
     - The `user_input` function takes the user's question and utilizes Langchain again:

- It generates an embedding for the user's question using the same Google Generative AI model employed for text chunks.
- It retrieves the previously saved vector store containing the embeddings of all text chunks.
- It performs a similarity search within the vector store to find text chunks that are most similar to the user's question based on their embeddings. Essentially, Langchain helps identify the most relevant parts of the uploaded PDFs based on the user's query.
- The retrieved text chunks and the user's question are fed into the pre-defined question-answering chain created earlier.
- The AI model within the chain analyzes the context (relevant text chunks) and the user's question, generating a detailed answer based on the Langchain-powered information retrieval process.
- The generated response is displayed to the user, providing them with the information they seek from the uploaded PDFs in a natural conversation-like manner..

CHAPTER 9

GLOSSARY OF TERMS

- **Large Language Models (LLMs):** Powerful AI models trained on massive text data to perform NLP tasks (text generation, translation, question answering).
- **Natural Language Processing (NLP):** A field of AI concerned with computer interaction with human language (understanding and manipulation).
- **Retrieval-Augmented Generation (RAG):** An NLP approach combining information retrieval from a knowledge base with text generation for richer responses.
- **Knowledge Base:** A collection of structured information (text, facts, relationships) used by AI systems.
- **Pre-training:** Training an LLM on a general text corpus before fine-tuning for a specific task (improves fundamental language skills).
- **Question Answering (QA):** An NLP task where a system answers questions based on its knowledge and understanding of the world.
- **Zero-Shot Learning:** Training an LLM to perform a task without specific examples for that task.
- **Generative AI:** A subfield of AI concerned with models that can generate new text, images, or code.
- **Vector Store:** A specialized database for storing and retrieving high-dimensional vectors (numerical representations of text data used for similarity comparisons).
- **Embeddings:** Numerical representations of text data capturing its semantic meaning (used for information retrieval and similarity comparisons).
- **Prompt:** An instruction or question guiding a generative AI model in its response generation.

CHAPTER 10
BIBLIOGRAPHY:

- **Bibliography**
- Chase, H., & The LangChain Team. (n.d.). *LangChain: Building Powerful Language Applications*. LangChain. https://www.pinecone.io/learn/series/langchain/langchain-intro/
- Lewis, P., et al. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. arXiv preprint arXiv:2005.11401.
- Yang, W., et al. (2020). *Retrieval-Augmented Generation for Text Summarization*. arXiv preprint arXiv:2005.11401.
- Feng, S., et al. (2020). *Towards Conversational Question Answering over Multiple Documents*. arXiv preprint arXiv:2005.11401.
- Zhu, S., et al. (2021). *RAG-Q: Efficient and Effective Retrieval-Augmented Generation for Question Answering*. arXiv preprint arXiv:2107.01974.
- Brown, T., et al. (2020). *Generative Pre-trained Transformer 3 (GPT-3): Language Modeling for Long-Form Content Creation*. OpenAI. https://openai.com/api/
- Kaplan, J., et al. (2020). *Scaling Laws for Neural Language Models*. arXiv preprint arXiv:2001.08237.
- Devlin, J., et al. (2019). *Towards a Human-Level Understanding of Natural Language*. arXiv preprint arXiv:1811.00143.
- Devlin, J., et al. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv preprint arXiv:1810.04805.
- Reddy, A., et al. (2021). *Zero-Shot Text-to-SQL Task: A Survey of Recent Approaches*. arXiv preprint arXiv:2103.10555.