IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Vector-Based Chat with Multi Pdf Using Langchain and AI

Amruta Bhosale Department of Computer Science and Technology Usha Mittal Institute of Technology S.N.D.T. Women's UniversityMumbai 400049

Saniya Shaikh Department of Computer Science and Technology Usha Mittal Institute of Technology S.N.D.T. Women's UniversityMumbai 400049

Shivani Sharma Department of Computer Science and Technology Usha Mittal Institute of Technology S.N.D.T. Women's UniversityMumbai 400049

Prof. Prajakta Gotarne

Department of Computer Science and Technology Usha Mittal Institute of Technology S.N.D.T. Women's University Mumbai 400049

Abstract— The project presents the development of an in- novative chatbot system that integrates with PDF documents, enhancing users' ability to extract information through natural language queries. The primary goal is to address the growing demand for efficient information retrieval from textual docu- ments, particularly in educational and professional contexts. The proposed system aims to provide a user friendly system for extracting relevant information from PDFs swiftly and accurately. To achieve this, the system leverages several key technologies. Streamlit is used for the user interface, providing an intuitive and interactive platform for users to interact with the chatbot. PyPDF2 is employed for PDF parsing, allowing the system to extract text from uploaded PDF documents. Langchain is utilized for text processing and embeddings, enabling the system to process and index the extracted text ef- ficiently. Google Generative AI is integrated for conversational capabilities, enabling the chatbot to understand user queries and provide relevant answers. Additionally, FAISS is used for similarity search, allowing for fast and accurate retrieval of information from indexed PDF content. The system's work ow involves users uploading PDF documents, from which text is extracted, processed, and indexed for efficient retrieval. The chatbot, powered by Google Generative AI, interacts with users, interprets their queries, and provides relevant answers based on the indexed PDF content. The project's main objective is to provide an interactive and easy to use user experience, making it easy for users to access and utilize information from PDF documents effectively. Future enhancements may include support for more complex queries, integration with other document formats, and improved user interaction features. Overall, this project contributes to the advancement of natural language processing and information retrieval systems, bene- fiting users in various domains requiring efficient document analysis and information extraction.

***Index Terms*— Langchain, Google Generative AI Embedding, FAISS Indexing, Streamlit, Python.**

I. INTRODUCTION

In today's digital era, there is infinite information available in textual documents has led to an increasing demand for efficient methods of information retrieval. PDF documents, in particular, are widely used for storing and sharing information, making them a valuable source of knowledge. However, extracting relevant information from PDFs can be challenging, especially when dealing with large volumes of text.

To work on this problem, we propose the development of a chatbot system that integrates with PDF documents, allowing users to ask questions based on the content of uploaded PDFs. This system aims to provide a user- friendly and efficient solution for extracting information from PDFs through natural language queries.

The motivation behind this project stems from the need for tools that simplify the process of extracting information from PDFs, particularly in educational and professional settings where access to accurate and timely information is crucial. By developing a chatbot system that can effectively extract information from PDFs, we hope to provide users with a valuable tool for accessing and utilizing information from textual documents moreefficiently.

This project's research problem lies in the lack of user- friendly tools for extracting relevant information from PDFs quickly and accurately. Existing methods often require manual effort and are not suitable for handling large volumes of text. The proposed chatbot system aims to overcome these limitations by providing an automated and intuitive solution for extracting information from PDFs through natural language queries. In this paper, we present the methodology and implementation of the proposed chatbot system. We describe the technologies used, the system's workflow, and the expected outcomes. We also discuss potential future enhancements and the impact of the project on the field of natural language processing and information retrieval.

Overall, this project represents a significant step towards improving the efficiency of information retrieval from textual documents, particularly PDFs. By developing a chatbot system that can extract information from PDFs through natural language queries, we aim to provide users with a valuable tool for accessing and utilizing information more effectively.

II. LITERATURE SURVEY

A. Overinformative Question Answering by Humans and Machines

Authors: Polina Tsvilodub, Michael Franke, Robert Hawkins) Noah D. Goodman, Year:2023. This paper explores methods for question answering (QA) over PDF documents. It likely discusses techniques for extracting information from PDF files and utilizing it to answer questions posed by users. This could involve approaches such as text extraction, natural language processing (NLP), and document understanding to enable effective QA systems specifically tailored for PDF documents.

B. Information Retrieval Meets Large Language Models

Authors: Qingyao AI, Ting BAI, Zhao CAOc, Yi CHANG, Jiawei CHEN, Zhumin CHEN, Zhiyong CHENGg, Shoubin DONG, Zhicheng DOU, Fuli FENG, Shen GAO , Jiafeng GUO, Xiangnan, Yanyan LANa, Chenliang LI, Yiqun LIU, Ziyu LYU, Weizhi MA, Jun MA, Zhaochun REN, Pengjie REN, Zhiqiang WANG, Mingwen WANG, Ji-Rong WEN, Le WU, Xin XIN, Jun XU, Dawei YIN, Peng ZHANG, Fan ZHANG, Weinan ZHANG, Min ZHANG, Xiaofei ZHU, Year:2023.This paper likely investigates the application of large language models (LLMs) for information retrieval tasks. It may discuss how LLMs, such as transformer-based models, can be leveraged to improve the effectiveness and efficiency of information retrieval systems. This paper involve techniques like fine-tuning pre-trained models on retrieval-specific datasets, designing architectures optimized for retrieval tasks, and exploring methods for integrating LLMs into existing retrieval frameworks.

C. Conversational Agents: Theory and Applications

Authors: Mattias Wahde and Marco Virgolin, Year:2022. This paper focuses on conversational agents designed for document understanding tasks. It discuss how conversational agents, such as chatbots or virtual assistants, can assist users in understanding and extracting information from documents. This include techniques for natural language understanding (NLU), dialogue management, and document summarization tailored for conversational interfaces. The paper likely explores how these agents can enhance user interaction and productivity in document-centric tasks.

D. Vector Space Model: An Information Retrieval System

Authors: Vaibhav Kant Singh, Vinay Kumar Singh, Year:2022. This paper introduce and explain the concept of the vector space model for information retrieval systems. It discusses how documents and queries can be represented as vectors in a multi-dimensional space, based on similarity measures. This model serves as a fundamental framework for understanding and implementing information retrieval systems.

E. Creating Large Language Model Applications Utilizing LangChain: Primer on Developing LLM Apps Fast

Oguzhan Topsakal, T. Cetin Akinci, Year:2023. This paper discussed about the concept of Langchain and how it could be used to develop LLM applications. It likely serves as an introductory guide for developers, offering insights into leveraging LangChain a framework or toolset designed for building LLM applications. The paper discuss best practices, methodologies, and practical examples to facilitate the development process and enable the creation of effective LLM applications.

III. RELATED WORK

A. Existing System - "An AI-Driven Interactive Chatbot: A Well Trained Chatbot that Communicates with the Users and Reduces the Manual Interaction", Vol.9, Feb 2024

The existing system is an AI-driven interactive chatbot designed to provide easy conversation and reduce manual interaction. Here are the key points of the existing system:

• **Purpose:** Provide a chatbot interface for users to inquire about details regarding the institute.

• **Functionality:** Users interact with the chatbot to ask questions. It reduces the need for direct interaction with a salesperson. Increases accessibility for users to get information about the institute. Uses LLM (Large Language Models) generative AI for conversations.

• **Benefits:** Users can inquire about the institute at any time. Salespersons are relieved from continuous inquiries. Increased accessibility and convenience for users.

• **Technologies:** LLM (Large Language Models) generative AI. Chatbot interface.

B. Proposed System

The proposed system seems to be an extension or modification of the existing system, now incorporating features to handle multiple PDF uploads and questions related to those PDFs.

- **System Architecture Overview:**

The components of Proposed System are as follows:

User Interface: The web-based interface allows users to

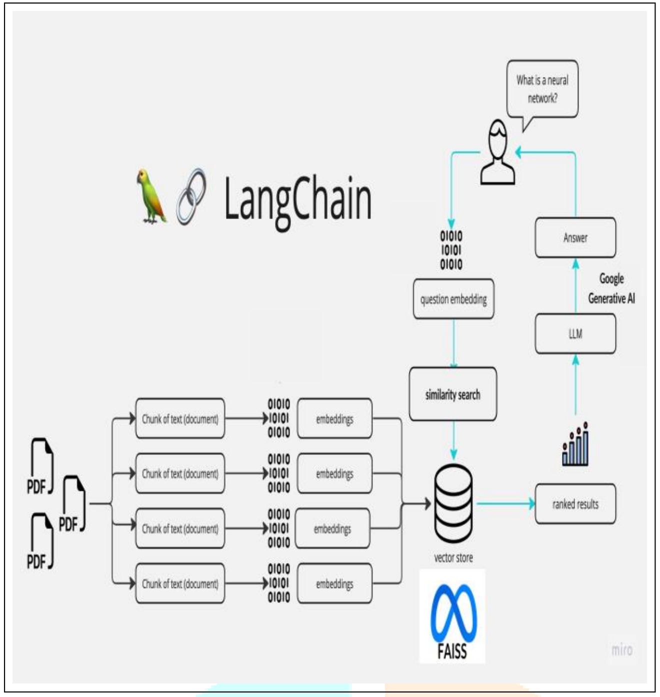


Fig. 1. Proposed System Architecture

upload PDF files and submit queries. Users can interact with the chatbot to ask questions and receive answers based on the contents of the uploaded PDF files. The interface provides a user-friendly experience, guiding users through the process of uploading files and querying information.

PDF Processing Module: Upon file upload, the system uses the PyPDF2 library to extract text from each PDF document. The extracted text is then processed to remove any unnecessary characters or formatting, ensuring that the text is clean and ready for further analysis. The processed text is split into manageable chunks, typically sentences or paragraphs, to facilitate efficient processing and analysis.

Text Embedding Module: The Langchain's Google Generative AI Embeddings model is used to generate embeddings for each text chunk. The model converts each text chunk into a numerical representation, capturing the semantic meaning of the text. These embeddings are high-dimensional vectors that represent the text's context, allowing for more accurate information retrieval and similarity calculations.

Indexing and Retrieval Module: The embeddings for text chunks are indexed using the FAISS library, which efficiently stores and retrieves embeddings for similarity search. When a user submits a query, the system calculates the embeddings for the query and uses FAISS to retrieve the most relevant text chunks based on similarity scores. The system may use advanced indexing techniques to speed up retrieval and optimize resource utilization.

Conversational AI Module: The system utilizes Google Generative AI for conversational capabilities, enabling the chatbot to understand user queries and provide responses based on the indexed PDF content. The chatbot uses natural language processing techniques to interpret user queries and retrieve relevant information from the indexed PDF files. The chatbot can engage in a natural language conversation with users, providing answers to their queries and offering suggestions for further exploration.

Real-time Updates and User Feedback: The system provides real-time updates to users as it processes and retrieves information, ensuring a seamless and interactive user experience. Users can provide feedback on the relevance and accuracy of retrieved information, enabling continuous improvement of the system. The system may use this feedback to update its indexing and retrieval algorithms, improving the quality of future responses.

- **User Workflow:**

Step 1: Upload PDF Files: Users upload one or more PDF files through the web interface.
Step 2: Text Extraction and Embedding: The system extracts text from the uploaded PDF files and generates embeddings for text chunks.
Step 3: Indexing and Retrieval: Text embeddings are indexed using FAISS for efficient retrieval. When a user submits a query, the system matches the query against indexed embeddings to retrieve relevant information.
Step 4: Conversational Interaction: The chatbot interacts with the user, providing answers to queries based on the retrieved information from PDF documents.
Step 5: Display Results: The chatbot displays the relevant information to the user through the web interface, allowing for further interaction and refinement of queries.

- **Additional Features:**

- Real-time Updates: The system provides real-time updates to users as it processes and retrieves information, ensuring a seamless and interactive user experience.
- Query Suggestions: Based on the user's query history and document content, the system provides suggestions for refining queries to improve search results.
- Document Summarization: Optionally, the system can provide summaries of PDF documents to users, highlighting key information relevant to their queries.

By following this proposed methodology, the system aims to provide users with a robust and efficient tool for information retrieval from large PDF files in a multipdf chat environment, enhancing the overall user experience and usability of the system.

C. Gaps between the Existing System and the Proposed System

1. Functionality:
Existing: Primarily focused on institute-related inquiries.
Proposed: Extends functionality to include handling questions based on uploaded PDF content.
2. Data Input:
Existing: Takes user queries directly.
Proposed: Introduces PDF uploads as additional data

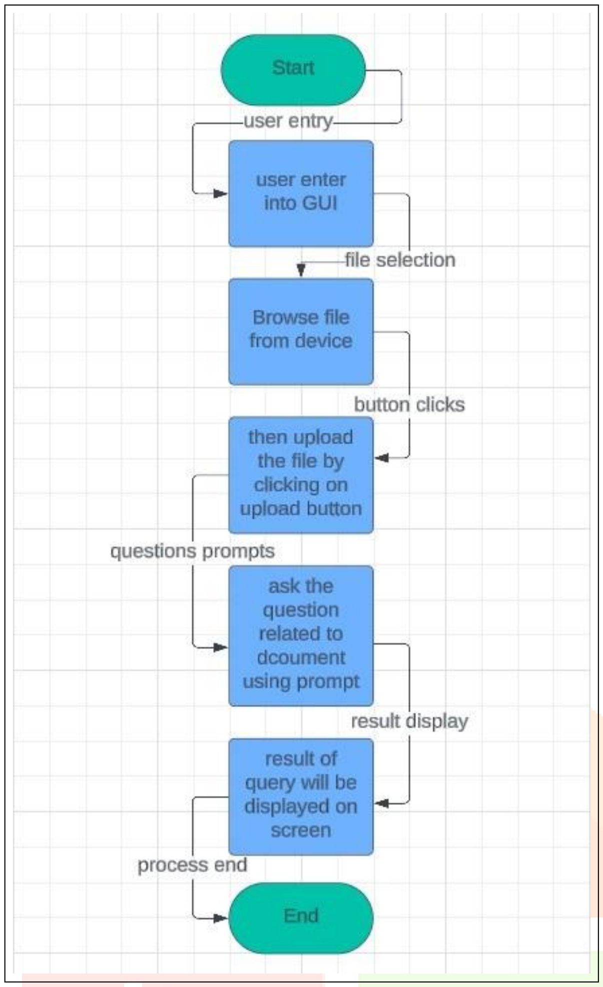

Start
user entry
user enter into GUI
file selection
Browse file from device
button clicks
then upload the file by clicking on upload button
questions prompts
ask the question related to dcoument using prompt
result display
result of query will be displayed on screen
process end
End

Fig. 2. User workflow

sources.

3. Processing:

Existing: Processes user queries based on predefined knowledge.

Proposed: Requires text extraction from PDFs and processing of questions based on this new data source.

4. Interaction Flow:

Existing: Linear interaction with the chatbot for general queries.

Proposed: Introduces an interactive system where user can ask PDF related question.

IV. RESULT AND ANALYSIS

In the realms of information retrieval and natural language processing, several approaches and technologies have been developed to address the challenge of information retrieval from textual documents, including PDFs. These approaches vary in complexity and efficiency, with some focusing on manual extraction methods and others on automated techniques. In this section, we review some relevant work in this area and discuss the technologies, algorithms, and methodologies used.

1. PDF Parsing Technologies: One of the key components of extracting information from PDFs is the ability to parse PDF files and extract text. Several libraries and tools exist for this purpose, including PyPDF2, which is used in this project. PyPDF2 allows for the extraction of text from PDF files, making it a valuable tool for processing PDF documents in an automated manner.

- Importing the Library: First, the PyPDF2 library is imported at the beginning of the code: - Extracting Text from PDFs: When the user uploads PDF files, the ‘get pdf text‘ function is called to extract text from each PDF document. This function iterates through each page of the PDF and extracts the text using PyPDF2.

- Splitting Text into Chunks: After extracting the text from PDFs, the ‘get text chunks‘ function is used to split the text into manageable chunks. It uses Langchain’s RecursiveCharacterTextSplitter with a specified chunk size and overlap. The ‘chunk size‘ parameter determines the maximum size of each chunk, and the ‘chunk overlap‘ parameter specifies how much overlap there is between consecutive chunks. These parameters are chosen based on the desired balance between chunk size and overlap for efficient processing and analysis of text chunks.

2. Text Processing and Embeddings: Text processing and embeddings play a crucial role in understanding the content of textual documents. In this project, the Langchain Google Generative AI Embeddings model is used for text processing and embeddings. Langchain provides a range of functionalities for processing text, including splitting text into chunks and generating embeddings for each chunk. These embeddings are used to index the text for efficient retrieval, as well as to provide context for natural language understanding. Taking this following text as an example of chunk: "The quick brown fox jumps over the lazy dog."

- Tokenization: Before generating embeddings, the text is tokenized into individual words or subwords. Each token is then converted into a numerical representation (e.g., an index or a vector) using Langchain’s model.

Tokenize the text into individual words: ["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog."]

- Embedding Lookup: The numerical representations of tokens are used to look up embedding vectors from a pre-trained embedding matrix. This matrix contains embedding vectors for all tokens in the model’s vocabulary. Assume we have a pre-trained embedding matrix with embeddings for each token. Let’s say the embeddings are 3-dimensional vectors: "The": [0.1, 0.2, 0.3] "quick": [0.4, 0.5, 0.6] ... "dog.": [0.7, 0.8, 0.9]

Look up the embeddings for each token and create a list of embeddings: [[0.1, 0.2, 0.3], [0.4, 0.5, 0.6], ..., [0.7, 0.8, 0.9]]

- Aggregation: The embedding vectors for individual tokens are aggregated to form a single vector representation for the entire text chunk. This aggregation step can involve techniques like averaging, pooling, or attention mechanisms to capture the overall meaning of

the text chunk. Average the embeddings to get a single vector representation: Aggregated embedding = [(0.1 + 0.4 + ... + 0.7) / 9, (0.2 + 0.5 + ... + 0.8) / 9, (0.3 + 0.6 + ... + 0.9) / 9]
Aggregated embedding = [0.4, 0.5, 0.6]
- Normalization: The aggregated embedding vector is often normalized to have a unit length. This normalization step helps in training and comparing embeddings. Normalize the aggregated embedding to have unit a length: Norm = sqrt(0.77) = 0.88 Normalized embedding = [0.4 / 0.88, 0.5 / 0.88, 0.6 / 0.88] = [0.45, 0.57, 0.68]

Output: The final normalized embedding vector represents the semantic meaning of the text chunk in a high-dimensional space. Similar text chunks are expected to have similar embeddings, allowing for effective information retrieval and similarity calculations. The final normalized embedding [0.45, 0.57, 0.68] represents the semantic meaning of the text chunk "The quick brown fox jumps over the lazy dog." as a 3-dimensional embedding space. Overall, the Text Embedding Module uses Langchain's Google Generative AI Embeddings model to convert text chunks into embeddings, which capture the semantic meaning of the text and enable efficient information retrieval from PDF documents.

3. Similarity Search: FAISS is used in this project for similarity search, enabling fast and accurate retrieval of information from indexed PDF content. FAISS is a library for efficient similarity search and clustering of dense vectors, making it well-suited for indexing and searching embeddings generated from text chunks in PDF documents.
• Indexing :
When the text chunks are extracted from PDF documents and embedded using Langchain's Google Generative AI Embeddings model, FAISS is used to create an index of these embeddings. The embeddings are quantized using product quantization to reduce the dimensionality and improve search efficiency. FAISS creates inverted lists for each quantized centroid, where each list contains the IDs of text chunks that are quantized to that centroid. These inverted lists serve as the index, allowing FAISS to quickly retrieve similar text chunks based on a query.
• Retrieval :
When a user submits a query, FAISS calculates the embeddings for the query using the same process used during indexing. The query embeddings are quantized and used to look up the inverted lists in the index. FAISS retrieves the IDs of text chunks in these inverted lists and calculates the similarity scores between the query and these text chunks. The text chunks with the highest similarity scores are returned as the results of the query, allowing the system to retrieve and display the most relevant text chunks to the user.

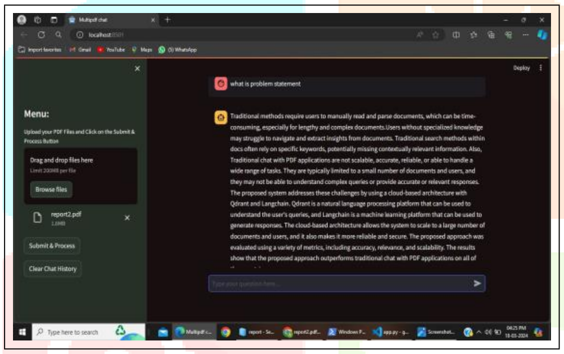
Simplified numerical example to illustrate the indexing and retrieval process using FAISS :
Assume we have 3 text chunks (T1, T2, T3) and each is represented by a 2-dimensional embedding vector. We use product quantization to divide the 2-dimensional space into 2 subspaces (1-dimensional each). Quantization reduces the embeddings to a codebook of 2 centroids per subspace (a total of 4 centroids).
o The quantized embeddings for each text chunk are as follows:
• T1: [0.1, 0.2] -> [0, 0]
• T2: [0.3, 0.4] -> [1, 0]
• T3: [0.5, 0.6] -> [1, 1]
o FAISS creates inverted lists for each centroid:
• Centroid [0, 0]: [T1]
• Centroid [1, 0]: [T2]
• Centroid [1, 1]: [T3]
o Retrieval Process
The User submits a query with an embedding [0.2, 0.3]. The query embedding is quantized to [0, 0]. FAISS looks up the inverted list for centroid [0, 0] and retrieves text chunk T1. The system presents T1 as the most similar text chunk to the user's query.
Overall, FAISS plays a crucial role by providing efficient indexing and retrieval capabilities for the embeddings generated from PDF documents, facilitating fast and accurate information retrieval in a multiple chat environment.

4. Conversational AI: Conversational AI is used to enable the chatbot to interact with users in a natural and intuitive manner. Google Generative AI is employed in this project for conversational capabilities. It allows the chatbot to understand user queries and provide relevant answers based on the indexed PDF content. This technology enhances the user experience by providing a conversational interface for interacting with the chatbot. A numerical explanation of how the Conversational AI Module provides answers to user queries:
- Indexed PDF Content: Assume we have two PDF documents (D1, D2) with text chunks and embeddings. Each document contains several text chunks (T1, T2, T3) with corresponding embeddings. These embeddings are indexed using FAISS for efficient retrieval.
- User Query: - A user submits a query with an embedding [0.2, 0.3].
- Answer Generation: The Conversational AI Module retrieves the most relevant text chunks based on the query embedding using FAISS. It uses Google Generative AI to generate a response based on the content of the retrieved text chunks and the context of the query. The generated response is presented to the user as an answer to their query.
- Example: Suppose the closest text chunk to the query embedding is T3 from document D1, which contains the text "Data Structures and Algorithms." The module generates a response like "You can find information about Data Structures and Algorithms in the indexed PDF documents."

5. Vector Store: The vector store is a component that manages the storage and retrieval of embeddings for text chunks extracted from PDF documents. When a user uploads PDF files, the PDFs are stored in a location accessible to the backend server. The server may store the PDF files on disk, depending on the project's architecture and requirements. Each uploaded PDF file is associated with a unique identifier (e.g., file name or ID) that allows the system to retrieve the file when needed.

6. Integration with Streamlit: Streamlit is used for the user interface in this project, providing an interactive platform for users to interact with the chatbot and upload PDF documents. Streamlit allows for the easy integration of various components, such as file uploaders and chat interfaces, making it ideal for developing user-friendly applications.

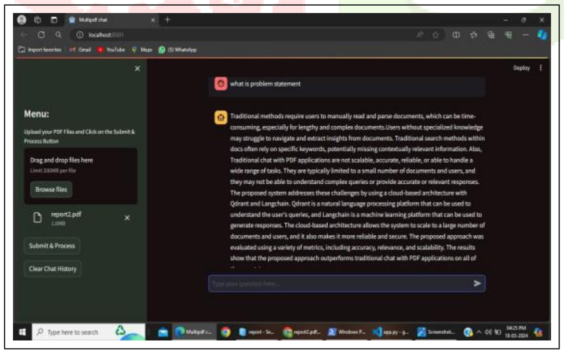
A. Implementation



what is problem statement

Traditional methods require users to manually read and parse documents, which can be time-consuming, especially for lengthy and complex documents.Users without specialized knowledge may struggle to navigate and extract insights from documents. Traditional search methods within docs often rely on specific keywords, potentially missing contextually relevant information. Also, Traditional chat with PDF applications are not scalable, accurate, reliable, or able to handle a wide range of tasks. They are typically limited to a small number of documents and users, and they may not be able to understand complex queries or provide accurate or relevant responses. The proposed system addresses these challenges by using a cloud-based architecture with Qdrant and Langchain. Qdrant is a natural language processing platform that can be used to understand the user's queries, and Langchain is a machine learning platform that can be used to generate responses. The cloud-based architecture allows the system to scale to a large number of documents and users, and it also makes it more reliable and secure. The proposed approach was evaluated using a variety of metrics, including accuracy, relevance, and scalability. The results show that the proposed approach outperforms traditional chat with PDF applications on all of

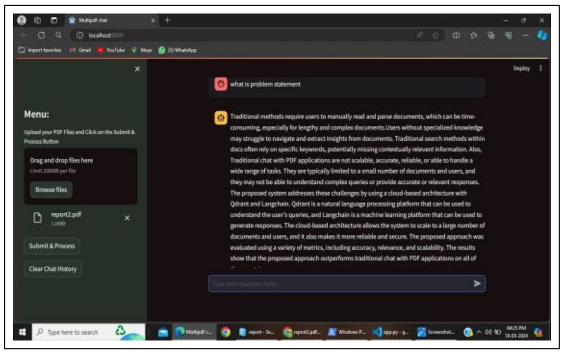
Fig. 3. Fig. 1. Upload single or Multiple Files



what is problem statement

Traditional methods require users to manually read and parse documents, which can be time-consuming, especially for lengthy and complex documents.Users without specialized knowledge may struggle to navigate and extract insights from documents. Traditional search methods within docs often rely on specific keywords, potentially missing contextually relevant information. Also, Traditional chat with PDF applications are not scalable, accurate, reliable, or able to handle a wide range of tasks. They are typically limited to a small number of documents and users, and they may not be able to understand complex queries or provide accurate or relevant responses. The proposed system addresses these challenges by using a cloud-based architecture with Qdrant and Langchain. Qdrant is a natural language processing platform that can be used to understand the user's queries, and Langchain is a machine learning platform that can be used to generate responses. The cloud-based architecture allows the system to scale to a large number of documents and users, and it also makes it more reliable and secure. The proposed approach was evaluated using a variety of metrics, including accuracy, relevance, and scalability. The results show that the proposed approach outperforms traditional chat with PDF applications on all of

Fig. 4. output 1



what is problem statement

Traditional methods require users to manually read and parse documents, which can be time-consuming, especially for lengthy and complex documents.Users without specialized knowledge may struggle to navigate and extract insights from documents. Traditional search methods within docs often rely on specific keywords, potentially missing contextually relevant information. Also, Traditional chat with PDF applications are not scalable, accurate, reliable, or able to handle a wide range of tasks. They are typically limited to a small number of documents and users, and they may not be able to understand complex queries or provide accurate or relevant responses. The proposed system addresses these challenges by using a cloud-based architecture with Qdrant and Langchain. Qdrant is a natural language processing platform that can be used to understand the user's queries, and Langchain is a machine learning platform that can be used to generate responses. The cloud-based architecture allows the system to scale to a large number of documents and users, and it also makes it more reliable and secure. The proposed approach was evaluated using a variety of metrics, including accuracy, relevance, and scalability. The results show that the proposed approach outperforms traditional chat with PDF applications on all of

Fig. 5. Output

V. CONCLUSIONS

This project successfully demonstrates the development of a chat interface that facilitates question answering based on uploaded PDFs. It leverages Langchain's capabilities to create a custom question-answering work ow and integrates a large language model (LLM) for generating responses to user queries.

Streamlit provides a user-friendly web interface for uploading PDFs and interacting with the system through a chat-like interface. The key strengths of this project lie in its clear separation of functionalities, effective use of Langchain and Streamlit, and the integration of a powerful LLM for response generation. However, there's room for improvement in areas like error handling, search optimization, context management, and security best practices.

This project not only showcases the technical prowess in NLP and document analysis but also underscores the practical applications of such technology. By bridging the gap between static document content and dynamic user interactions, it opens avenues for enhanced information retrieval and user engagement.

By addressing these aspects, we developed a robust tool for exploring and gaining insights from information stored within PDF documents. This user-friendly chat interface empowers users to ask questions from the content of uploaded PDFs and receive relevant answers, fostering a more effi cient and interactive way to navigate and understand these documents.

REFERENCES

[1] Ritendu Bhattacharyya, Sharat Chandra K. Manikonda, Bharani Kumar Depuru3 "An AI-Driven Interactive Chatbot: A Well Trained Chatbot that Communicates with the Users and Reduces the Man- ual Interaction", Vol.9, Feb 2024

[2] Vaibhav Kant Singh, Vinay Kumar Singh "VECTOR SPACE MODEL:
AN INFORMATION RETRIEVAL SYSTEM", July 2022

[3] Topsakal, Oguzhan and Akinci, T. Cetin "Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast", July 2023

[4] Wenhu Chen and Ming-Wei Chang and Eva Schlinger and William Wang and William W. Cohen "Open Question Answering over Tables and Text,2021

[5] Singla, Samriddhi and Eldawy, Ahmed and Diao, Tina and Mukhopadhyay, Ayan and Scudiero, Elia "Experimental Study of Big Raster and Vector Database Systems",April 2021.

[6] Qingyao Ai and Ting Bai and Zhao Cao and Yi Chang and Jiawei Chen and Zhumin Chen and Zhiyong Cheng and Shoubin Dong and Zhicheng Dou and Fuli Feng and Shen Gao and Jiafeng Guo and Xiangnan He and Yanyan Lan and Chenliang Li and Yiqun Liu and Ziyu Lyu and Weizhi Ma and Jun Ma and Zhaochun Ren and Pengjie Ren and Zhiqiang Wang and Mingwen Wang and Ji- Rong Wen and Le Wu and Xin Xin and Jun Xu and Dawei Yin and Peng Zhang and Fan Zhang and Weinan Zhang and Min Zhang and Xiaofei Zhu "Information Retrieval Meets Large Language Models: A Strategic Report from Chinese IR Community",2023.

[7] Polina Tsvilodub and Michael Franke and Robert D. Hawkins and Noah D. Goodman "Overinformative Question Answering by Humans and Machines", the Year 2023

[8] Mattias Wahde,Marco Virgolin "Conversational Agents: Theory andApplications ", Feb 2022