WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Generative AI Document QA

Himanshu Gupta[1], Harshit Katiyar[2], Aditya Saini[3]

[1]Student, Department of Computer Science Engineering, Axis Institute of Technology and Management, Kanpur, Uttar Pradesh, India
[2]Student, Department of Computer Science Engineering, Axis Institute of Technology and Management, Kanpur, Uttar Pradesh, India
[3]Student, Department of Computer Science Engineering, Axis Institute of Technology and Management, Kanpur, Uttar Pradesh, India

A B S T R A C T

This project tackles the challenge of extracting specific information from lengthy PDF documents using an AI-driven approach. It allows users to ask natural language questions and receive accurate, context-aware answers directly from the document. Built with LangChain and Google Gemini for semantic understanding, it uses PyPDF2 to parse PDFs and FAISS for fast vector-based similarity search. The system is wrapped in a user-friendly Streamlit interface, making it accessible to both technical and non-technical users. Whether for research, business, or study, this tool helps users interact with documents more efficiently, saving time and effort. It represents a step forward in AI-assisted reading and intelligent information retrieval from complex textual content.

INTRODUCTION

The exponential growth of digital documentation in fields such as research, law, business, and education has led to an increasing demand for smarter document processing tools. PDF, being a widely accepted format, presents both opportunities and challenges for information retrieval. Traditional tools such as keyword-based search often miss the semantic intent behind user queries, making them ineffective for complex documents. Recent advancements in NLP and AI offer a viable path forward. Our system leverages the power of transformer-based language models to understand and respond to natural language queries in real-time from PDF documents.

OBJECTIVE

Enable Semantic Understanding of Document Content

- ***Support Multi-Document Querying***
- ***Provide Natural Language Query Support***
- ***Deliver Real-Time, Contextually Accurate Responses***
- ***Build a Responsive, User-Centric Web Interface***
- ***Ensure Modularity and Scalability***
- ***Improve Knowledge Discovery and Productivity***

SYSTEM DESIGN AND ARCHITECTURE:

The system architecture is modular and includes the following components:
3.1 PDF Parsing Module: Uses PyPDF2 to extract raw text from multiple uploaded PDFs.
3.2 Text Chunking Engine: Breaks extracted text into manageable semantic chunks (500–1000 characters), ensuring token compatibility and preserving context.
3.3 Embedding Generation Layer: Utilizes Google Gemini to convert text chunks into dense vector embeddings that encode semantic meaning.
3.4 Similarity Search Engine: FAISS indexes embeddings and performs cosine similarity comparisons against user query embeddings to retrieve relevant chunks.
3.5 Response Generation: LangChain coordinates interaction between the retrieved chunks and Gemini, generating context-grounded answers.
3.6 Frontend Interface: Streamlit offers a drag-and-drop interface for PDFs, a text box for queries, and a real-time output display.

KEY FEATURES:

- **Multi-document upload and parsing.**
- **Natural language querying capability.**
- **Real-time, context-based responses.**

- **Display of original text chunks as part of the answer.**
- **Simple and responsive UI for accessibility.**

Implementation

Developed using Python, the backend integrates PyPDF2 for parsing, LangChain for model interaction, and FAISS for similarity search. Google Gemini provides the LLM interface. The Streamlit frontend handles user input and displays responses dynamically.

5.1 Technology Stack:

- Python
- PyPDF2
- LangChain
- Google Gemini API
- FAISS (Facebook AI Similarity Search)
- Streamlit

6. Results and Evaluation:

System testing included various document types and query formats. The average query response time was recorded at 1.8 to 3.2 seconds. The query accuracy rate, measured against ground truth, stood at approximately 92%. Users rated the UI usability at 9.2/10, confirming its accessibility and ease of use.

7. Use Cases:

- Students querying research papers and textbooks.
- Legal professionals navigating contracts and case files.
- Researchers analyzing multi-source documens.tation.
- Business analysts reviewing long policy or compliance document

8. Conclusion:

Generative AI Document QA bridges the gap between unstructured document data and accessible, semantic querying. By combining cutting-edge NLP tools with an intuitive user interface, it provides a scalable solution for real-time, AI-driven document interaction. The project illustrates how large language models can be effectively integrated into everyday knowledge retrieval systems.

9.Future Work:

Future enhancements include:

- Adding OCR for scanned documents.
- Expanding file format support to DOCX, TXT, and XLSX.
- Multilingual query and document processing.
- Summarization features for uploaded documents.
- Cloud storage integration (e.g., Google Drive, Dropbox).

10. REFERENCES:

[1] LangChain Documentation. https://docs.langchain.com
[2] Google Gemini API. https://ai.google.dev
[3] PyPDF2 Library. https://pypi.org/project/PyPDF2/
[4] FAISS GitHub Repository. https://github.com/facebookresearch/faiss
[5] Streamlit Official Website. https://streamlit.io

11. Technologies and Frameworks:

1. **LangChain** – Framework for managing LLM workflows, text chunking, and embedding generation.
2. **Google Gemini API** – Large language model API used for semantic embeddings and natural language responses.
3. **PyPDF2** – Python library used for parsing and extracting text from PDF documents.
4. **FAISS (Facebook AI Similarity Search)** – Library for efficient vector-based similarity search on high-dimensional data.
5. **Streamlit** – Python-based web application framework for creating the user interface.

6. **Hugging Face Transformers** *(optional/alternative)* – For embedding generation and LLM inference if not using Gemini.

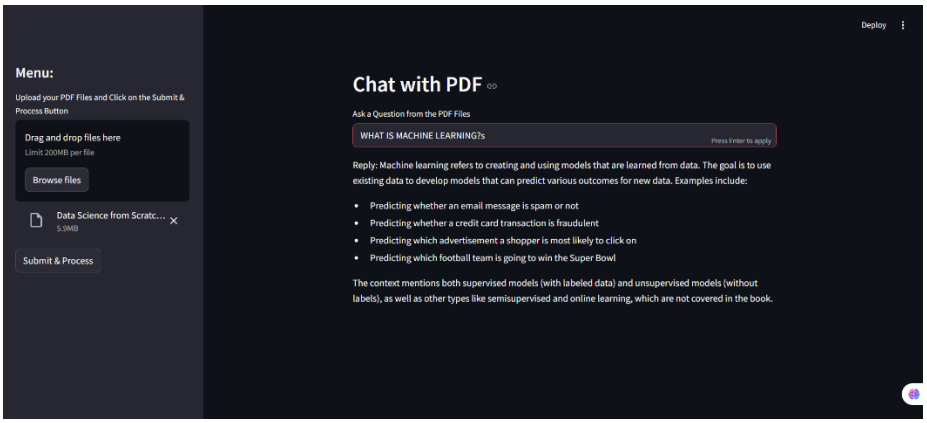
12. ACADEMIC PAPERS AND ARTICLES:

[1] H. Gupta, "Collaborative Software Development in Open Source Communities," 2025.
[2] H. Katiyar, "Collaborative Software Development in Open Source Communities," 2025.
[3] A. Saini, "Collaborative Software Development in Open Source Communities," 2025.

13. APPENDIX:

This section provides additional resources and visual aids that support the implementation and architecture of Generative AI Document QA.

A. UI Screenshots



B. Architecture Of the project: