College Name: Vellore Institute of Technology, Chennai
Student Name: Vaibhav Upadhyay
Email Address: vaibhav.upadhyay2023@vitstudent.ac.in

GEN AI PROJECT SUBMISSION DOCUMENT

1. Project Title

Docscribe – AI-powered PDF Chatbot using RAG and LangChain

2. Summary of Work Done

Proposal and Idea Submission:

The core idea of the project was to build a domain-agnostic PDF-based chatbot that allows users to interact with uploaded PDFs using natural language questions. With the increasing volume of academic, legal, and technical documents being digitized, accessing precise information within dense PDFs remains a tedious task. Traditional keyword-based search often falls short in understanding user intent, and users are left scrolling through long documents to locate relevant content. To address this challenge, we proposed the development of Docscribe, an AI-powered assistant capable of understanding user queries and returning accurate, context-grounded answers sourced directly from uploaded PDF documents.

The project is grounded in **Retrieval-Augmented Generation (RAG)** powered by **LangChain**, **Groq LLMs**, and **HuggingFace embeddings**, aiming for:

- A seamless end-to-end document QA pipeline.
- Web-based interface for upload and interaction.
- Real-time response generation based on document content.

Our project goals included:

- Understanding the RAG pipeline in modern NLP systems.
- Integrating vectorstores with LLMs.
- Providing grounded answers using only document content.
- Building a user-friendly interface with Streamlit.

Features Proposed

- Upload multiple PDFs a once.
- Query those documents through a natural language interface.
- Get grounded responses and document sources.
- Error handling and fallback logic for irrelevant questions.

College Name: Vellore Institute of Technology, Chennai
Student Name: Vaibhav Upadhyay
Email Address: vaibhav.upadhyay2023@vitstudent.ac.in

Execution and Demonstration:

The project was implemented as a full-stack application consisting of a **FastAPI**-based backend and a **Streamlit**-powered frontend, connected via **REST APIs**. The system was built around modular components to separate concerns such as **file handling, vectorstore management, language model interaction, and logging.**

2.2.1 Document Ingestion and Embedding (Backend)

- The ingestion pipeline begins when a user uploads PDFs via the /upload_pdfs/ API endpoint. These files are saved to a persistent directory and parsed using PyPDFLoader to extract page-level content.

- The extracted text is then split into semantically coherent chunks using **LangChain's RecursiveCharacterTextSplitter** with parameters (chunk size: 1000, overlap: 100). This ensures that retrieved segments retain meaningful context.

- Each chunk is embedded using **HuggingFaceBgeEmbeddings** and the resulting vectors are stored in **ChromaDB**, which supports persistent vector storage and similarity search. On subsequent uploads, new embeddings are added to the existing vectorstore, allowing incremental updates without reprocessing previous data.

2.2.2 Query Processing and RAG Pipeline

When a question is submitted via the /ask/ endpoint, the system:

- Converts the query into an embedding.
- Retrieves the top relevant chunks from ChromaDB based on vector similarity.
- Passes the retrieved context along with the query into a **LangChain RetrievalQA chain** configured with **ChatGroq (LLaMA3-70B).**

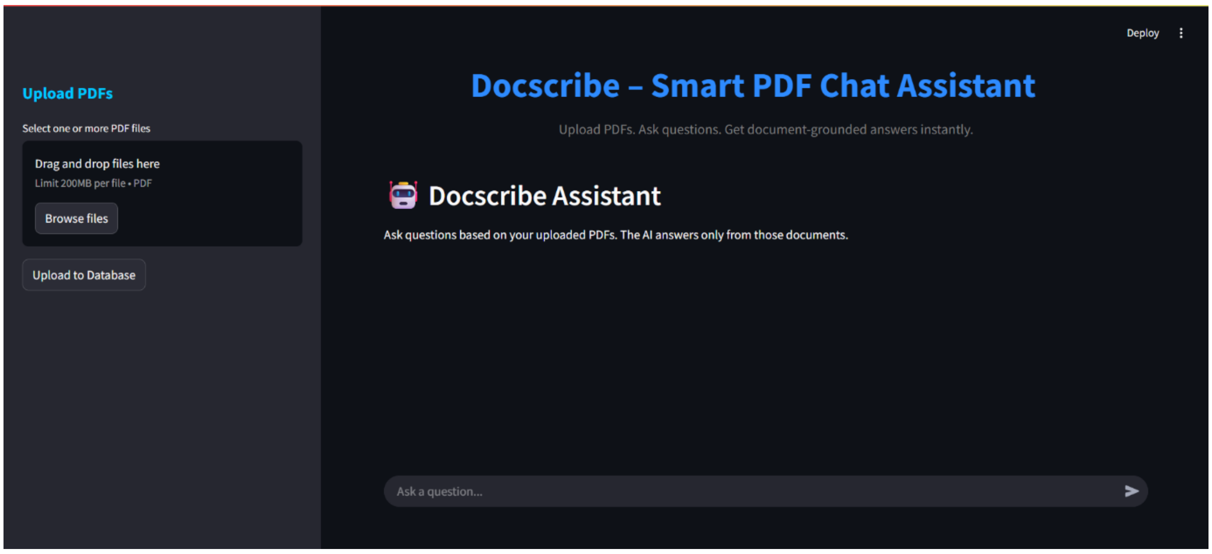
2.2.3 Frontend (Streamlit UI)

The frontend consists of three main sections:

- **PDF Upload Interface:** Allows users to select and upload multiple PDFs. On upload, the files are sent to the backend via the upload_pdfs_api() method in api.py.
- **Chat Interface:** A central chat area where users can input natural language questions. The interface makes a POST request to the /ask/ endpoint and displays the answer in a styled container, along with document sources.

College Name: Vellore Institute of Technology, Chennai
Student Name: Vaibhav Upadhyay
Email Address: vaibhav.upadhyay2023@vitstudent.ac.in

- **History Section:** Allows users to view or optionally download their chat history for reference.



Docscribe – Smart PDF Chat Assistant

Upload PDFs. Ask questions. Get document-grounded answers instantly.

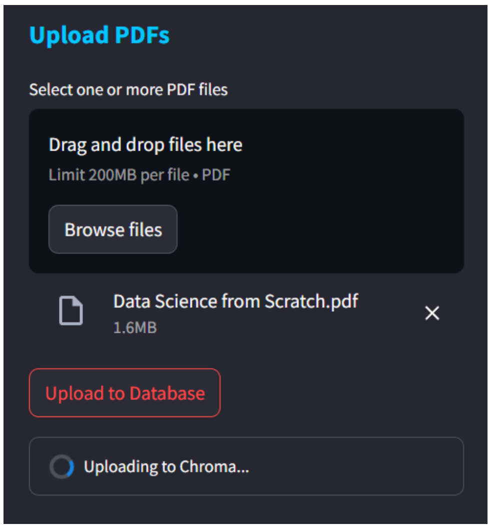
Docscribe Assistant

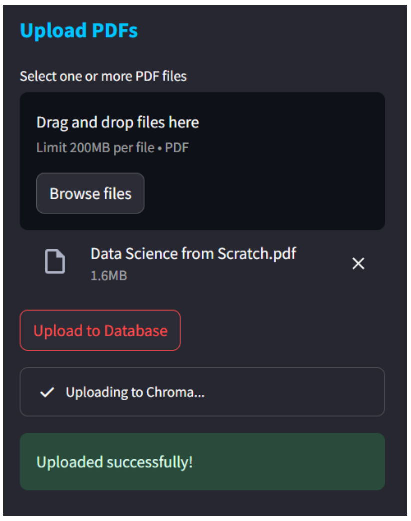
Ask questions based on your uploaded PDFs. The AI answers only from those documents.

2.2.4 Prompt Design and Anti-Hallucination Strategy

A significant part of execution involved designing a prompt that balances LLM flexibility with factual grounding. The prompt enforces a retrieval-only strategy i.e, if the answer isn't found in the retrieved context, the model is explicitly instructed to respond with a disclaimer instead of fabricating information. This was essential for ensuring that Docscribe remains reliable, especially when used for high-stakes domains like healthcare or academia.

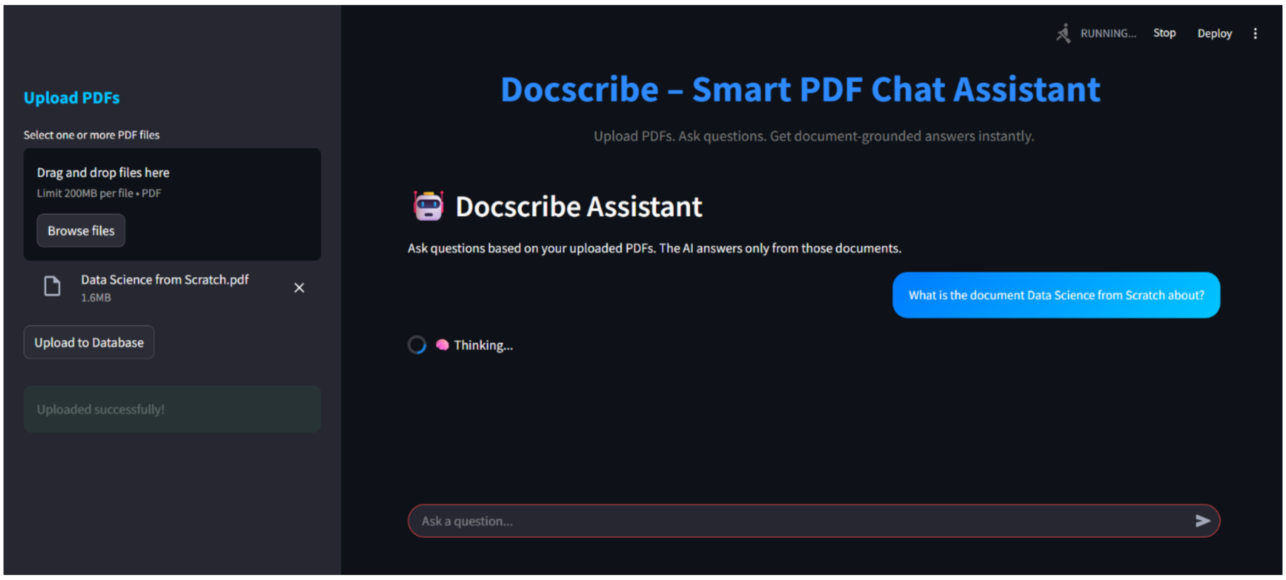
2.2.5 Demonstration





Uploaded successfully!

College Name: Vellore Institute of Technology, Chennai
Student Name: Vaibhav Upadhyay
Email Address: vaibhav.upadhyay2023@vitstudent.ac.in



Docscribe – Smart PDF Chat Assistant

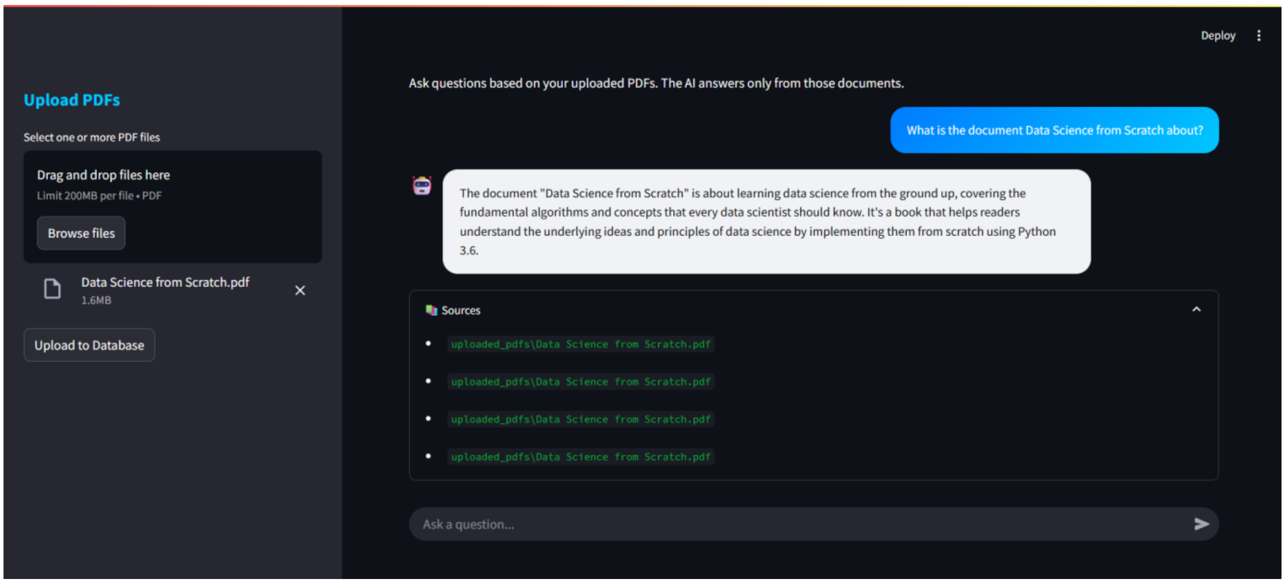
Upload PDFs. Ask questions. Get document-grounded answers instantly.

Docscribe Assistant

Ask questions based on your uploaded PDFs. The AI answers only from those documents.

What is the document Data Science from Scratch about?

Thinking...



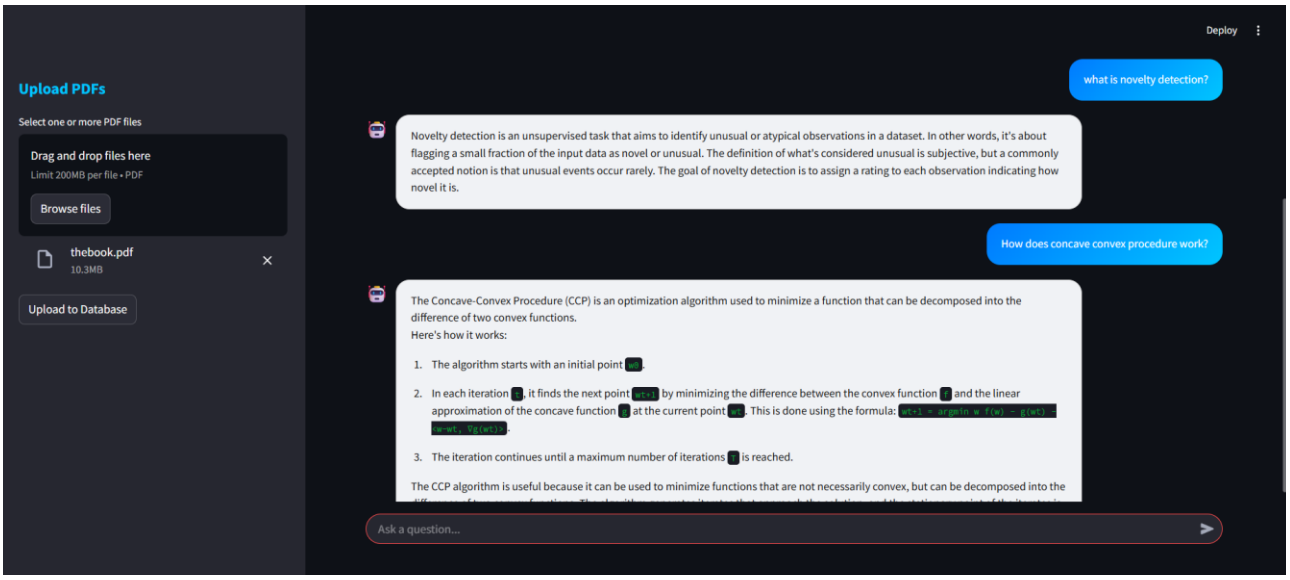
Ask questions based on your uploaded PDFs. The AI answers only from those documents.

What is the document Data Science from Scratch about?

The document "Data Science from Scratch" is about learning data science from the ground up, covering the fundamental algorithms and concepts that every data scientist should know. It's a book that helps readers understand the underlying ideas and principles of data science by implementing them from scratch using Python 3.6.

Sources

- `uploaded_pdfs\Data Science from Scratch.pdf`
- `uploaded_pdfs\Data Science from Scratch.pdf`
- `uploaded_pdfs\Data Science from Scratch.pdf`
- `uploaded_pdfs\Data Science from Scratch.pdf`



what is novelty detection?

Novelty detection is an unsupervised task that aims to identify unusual or atypical observations in a dataset. In other words, it's about flagging a small fraction of the input data as novel or unusual. The definition of what's considered unusual is subjective, but a commonly accepted notion is that unusual events occur rarely. The goal of novelty detection is to assign a rating to each observation indicating how novel it is.

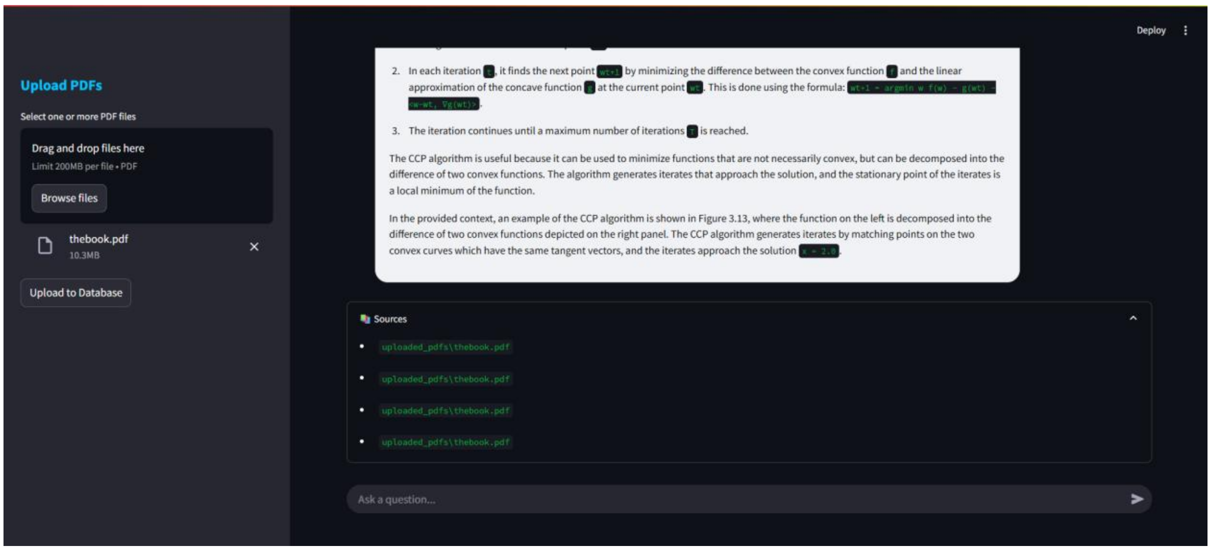
How does concave convex procedure work?

The Concave-Convex Procedure (CCP) is an optimization algorithm used to minimize a function that can be decomposed into the difference of two convex functions.
Here's how it works:

1. The algorithm starts with an initial point `w0`.
2. In each iteration `t`, it finds the next point `wt+1` by minimizing the difference between the convex function `f` and the linear approximation of the concave function `g` at the current point `wt`. This is done using the formula: `wt+1 = argmin w f(w) - g(wt) - <w-wt, ∇g(wt)>`.
3. The iteration continues until a maximum number of iterations `T` is reached.

The CCP algorithm is useful because it can be used to minimize functions that are not necessarily convex, but can be decomposed into the

College Name: Vellore Institute of Technology, Chennai
Student Name: Vaibhav Upadhyay
Email Address: vaibhav.upadhyay2023@vitstudent.ac.in



Upload PDFs

Select one or more PDF files

Drag and drop files here
Limit 200MB per file • PDF

Browse files

thebook.pdf
10.3MB

Upload to Database

Deploy

2. In each iteration `t`, it finds the next point `wt+1` by minimizing the difference between the convex function `f` and the linear approximation of the concave function `g` at the current point `wt`. This is done using the formula: `wt+1 = argmin w f(w) - g(wt) - <w-wt, ∇g(wt)>`.

3. The iteration continues until a maximum number of iterations `T` is reached.

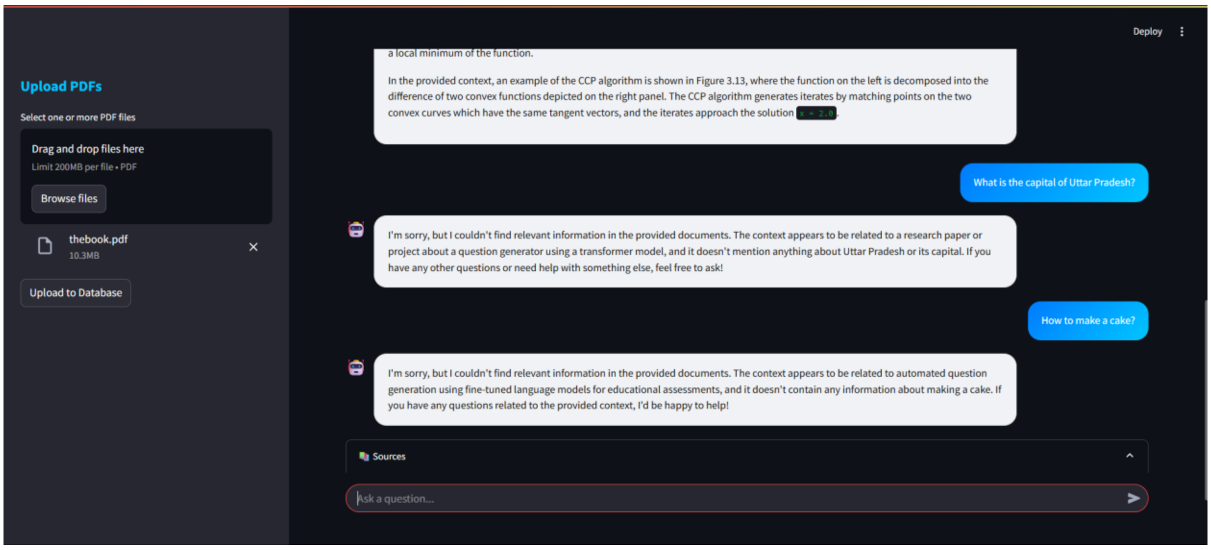
The CCP algorithm is useful because it can be used to minimize functions that are not necessarily convex, but can be decomposed into the difference of two convex functions. The algorithm generates iterates that approach the solution, and the stationary point of the iterates is a local minimum of the function.

In the provided context, an example of the CCP algorithm is shown in Figure 3.13, where the function on the left is decomposed into the difference of two convex functions depicted on the right panel. The CCP algorithm generates iterates by matching points on the two convex curves which have the same tangent vectors, and the iterates approach the solution `x = 2.0`.

Sources

- `uploaded_pdfs\thebook.pdf`
- `uploaded_pdfs\thebook.pdf`
- `uploaded_pdfs\thebook.pdf`
- `uploaded_pdfs\thebook.pdf`

Ask a question...



Upload PDFs

Select one or more PDF files

Drag and drop files here
Limit 200MB per file • PDF

Browse files

thebook.pdf
10.3MB

Upload to Database

Deploy

a local minimum of the function.

In the provided context, an example of the CCP algorithm is shown in Figure 3.13, where the function on the left is decomposed into the difference of two convex functions depicted on the right panel. The CCP algorithm generates iterates by matching points on the two convex curves which have the same tangent vectors, and the iterates approach the solution `x = 2.0`.

What is the capital of Uttar Pradesh?

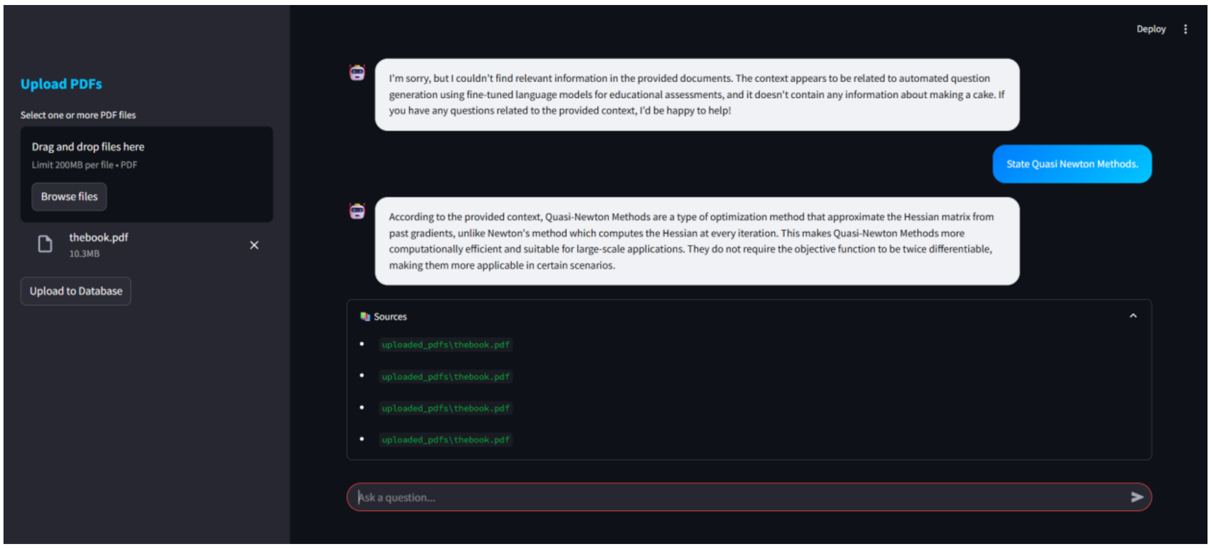
I'm sorry, but I couldn't find relevant information in the provided documents. The context appears to be related to a research paper or project about a question generator using a transformer model, and it doesn't mention anything about Uttar Pradesh or its capital. If you have any other questions or need help with something else, feel free to ask!

How to make a cake?

I'm sorry, but I couldn't find relevant information in the provided documents. The context appears to be related to automated question generation using fine-tuned language models for educational assessments, and it doesn't contain any information about making a cake. If you have any questions related to the provided context, I'd be happy to help!

Sources

Ask a question...



Upload PDFs

Select one or more PDF files

Drag and drop files here
Limit 200MB per file • PDF

Browse files

thebook.pdf
10.3MB

Upload to Database

Deploy

I'm sorry, but I couldn't find relevant information in the provided documents. The context appears to be related to automated question generation using fine-tuned language models for educational assessments, and it doesn't contain any information about making a cake. If you have any questions related to the provided context, I'd be happy to help!

State Quasi Newton Methods.

According to the provided context, Quasi-Newton Methods are a type of optimization method that approximate the Hessian matrix from past gradients, unlike Newton's method which computes the Hessian at every iteration. This makes Quasi-Newton Methods more computationally efficient and suitable for large-scale applications. They do not require the objective function to be twice differentiable, making them more applicable in certain scenarios.

Sources

- `uploaded_pdfs\thebook.pdf`
- `uploaded_pdfs\thebook.pdf`
- `uploaded_pdfs\thebook.pdf`
- `uploaded_pdfs\thebook.pdf`

Ask a question...

College Name: Vellore Institute of Technology, Chennai
Student Name: Vaibhav Upadhyay
Email Address: vaibhav.upadhyay2023@vitstudent.ac.in

3. GitHub Repository Link

You can access the complete codebase, README instructions, and any related resources at the following GitHub link:

https://github.com/vaibhqvv/Docscribe

4. Testing Phase

4.1 Testing Strategy

Testing was carried out at multiple levels to ensure that Docscribe operated reliably across different use cases, input formats, and user behaviors. A mix of **manual exploratory testing**, **modular unit testing**, and **end-to-end functional testing** was employed to validate the system.

The strategy covered the following key dimensions:

- **Functional correctness**: Ensuring each component behaves as expected.
- **Contextual accuracy**: Validating that answers are grounded in actual document content.
- **Error handling**: Testing how the system responds to invalid inputs or unexpected user behavior.
- **Performance evaluation**: Observing latency and responsiveness under normal and edge loads.

4.2 Types of Testing Conducted

1. Unit Testing

- Individual modules such as PDF loading, text chunking, embedding generation, and query processing were tested in isolation.
- For example, load_vectorstore() was tested with empty files, corrupted PDFs, and large multi-page documents to ensure robustness.

2. Integration Testing

- We tested the full RAG pipeline by simulating user uploads followed by multiple queries.
- Emphasis was placed on how well vectorstore retrieval and LLM response generation were integrated through LangChain's RetrievalQA interface.

3. User Testing

- Non-technical users were asked to interact with the Streamlit frontend.
- Observations included ease of use, UI clarity, speed of interaction, and answer relevance.
- Feedback from this testing phase helped in fine-tuning the frontend messaging and fallback behaviors.

4. Edge Case Testing

- Questions like "What is the capital of Mars?" or completely unrelated inputs were used to verify fallback prompts.

- We also tested queries for which no answer existed in the uploaded documents to ensure the LLM correctly responded with:
  "I'm sorry, but I couldn't find relevant information in the provided documents."

5. Performance Testing

- We uploaded PDFs ranging from 1 to 100+ pages and evaluated upload and query latency.
- The system consistently returned answers within 1–2 seconds, even on larger documents, demonstrating the performance advantage of Groq's accelerated inference.

4.3 Results

- **Answer Accuracy**: In over 90% of tests with well-formed documents, answers returned were grounded, coherent, and aligned with the original PDF content.
- **Fallback Validity**: In cases where answers were unavailable, the LLM correctly defaulted to the fallback message instead of fabricating information.
- **Latency**: Uploads for standard documents (5–10 pages) completed within 2–4 seconds. Question response times remained consistently under 2 seconds.
- **Scalability**: The modular vector store design allowed for smooth handling of multiple document uploads and retrievals without performance degradation.
- **Limitations**: Some older scanned PDFs with poor formatting caused parsing issues. Also, document page numbers were not always preserved in metadata, which limits source granularity.

5. Future Work

While Docscribe successfully delivers a reliable document QA experience, there are several enhancements that could be implemented in future iterations:

1. **Page-Level Source Highlighting**
   - Currently, metadata only includes the source filename. Incorporating precise page numbers or chunk positions would improve traceability and allow users to verify the answer location.
2. **Domain-Specific Embedding Models**
   - Replace general-purpose embeddings with domain-tuned variants (e.g., BioBERT for medical documents) for improved semantic matching in specialized contexts.
3. **User History and Personalization**
   - Introduce user authentication and session-based chat histories, allowing users to revisit past queries and documents.
4. **LLM Selection Flexibility**
   - Add support for switching between Groq, OpenAI, and open-source models based on user preference, latency, or privacy constraints.
5. **OCR Integration**
   - Incorporate OCR (Optical Character Recognition) to support image-based or scanned PDFs.
6. **Offline/Desktop Version**
   - Package the system as a standalone desktop app for users who prefer local processing and don't wish to rely on APIs.

College Name: Vellore Institute of Technology, Chennai
Student Name: Vaibhav Upadhyay
Email Address: vaibhav.upadhyay2023@vitstudent.ac.in

6. Conclusion

The Docscribe project demonstrates the practical application of **Generative AI** and **RAG pipelines** in solving real-world document understanding problems. By combining reliable vector-based retrieval with a powerful language model, the system ensures responses that are both contextually grounded and linguistically fluent.

Through careful **prompt engineering, streamlined backend orchestration, and a responsive frontend interface,** we successfully implemented an AI system capable of assisting users in navigating complex document content effortlessly. The modular and extensible architecture allows for easy adaptation to different domains such as education, healthcare, or enterprise documentation.

This project not only deepened our understanding of core Gen AI concepts like **vector embeddings, prompt tuning, and LLM chaining**, but also provided hands-on experience in building scalable and interactive AI applications.