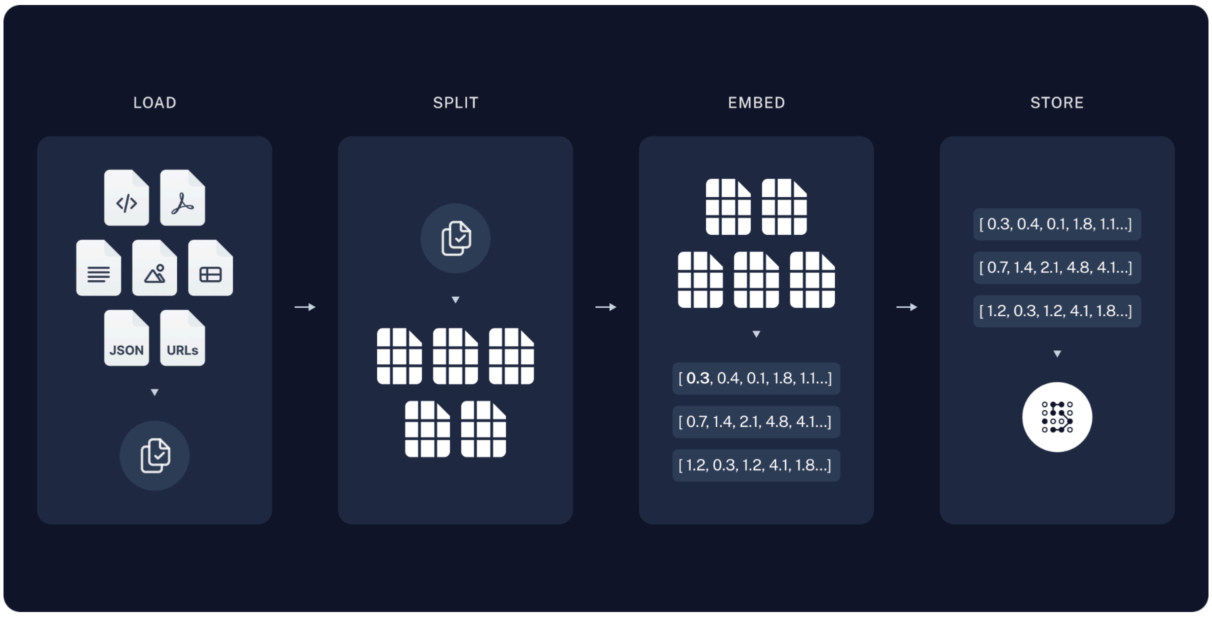
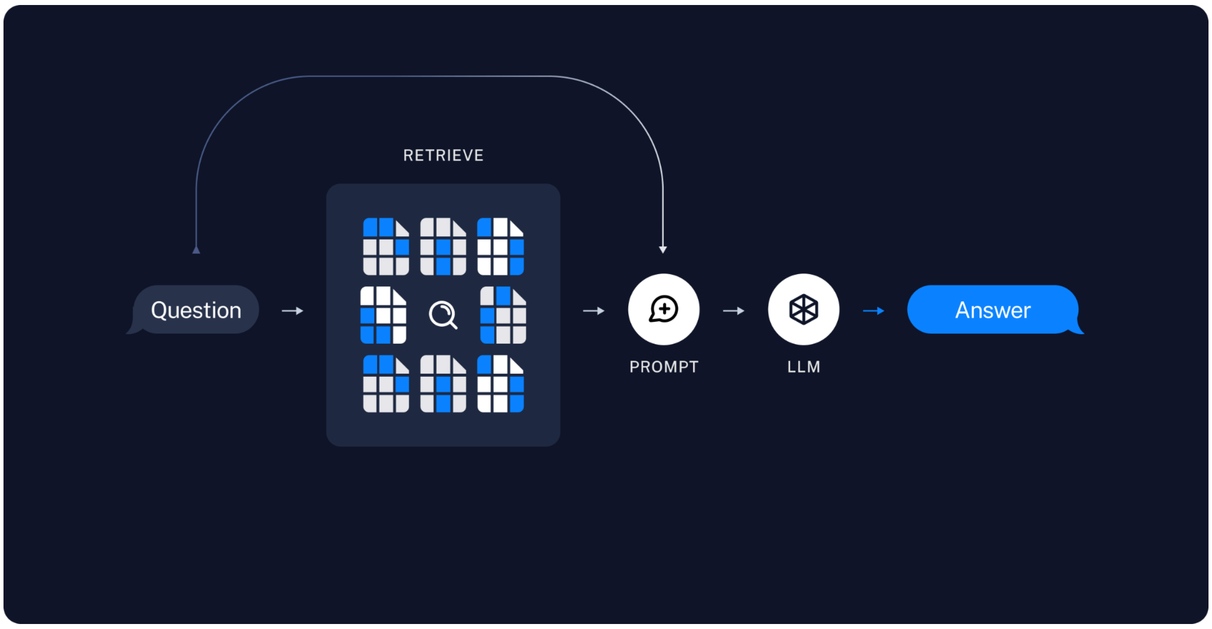
RAG project report

Approach and Methodology

The RAG (Retrieval-Augmented Generation) project aimed to build an answering questions system based on specific PDF documents using LangChain, FastAPI, and Streamlit.





Here's a detailed outline of the approach and methodology used:

1. **Problem Definition**:
   - The goal was to create a question-answering system that leverages pre-defined documents for generating accurate and context-specific responses.

2. **Technology Stack**:
    - **Backend**: FastAPI for handling API requests.
    - **GUI**: Streamlit for a user-friendly interface.
    - **Model**: LangChain integrated with AI21 for embeddings and LLM.

3. **Data Preparation**:
    - Extracted text from the provided PDF documents.
    - Used LangChain's text splitter to manage large text chunks for better processing and embedding.

4. **Model Integration**:
    - Employed AI21's embedding and LLM capabilities through LangChain to convert text into embeddings and generate responses.
    - used FAISS (Facebook AI Similarity Search) for vector storing and efficient similarity searches among the document embeddings.

5. **Application Development**:
    - **API Endpoints**: Created FastAPI endpoints to handle queries and return answers.
    - **User Interface**: Developed a Streamlit application for users to input questions and view responses.

6. **Dockerization**:
    - Created a Dockerfile to containerize the application, ensuring consistent environments across different setups.
    - Utilized Docker Compose for managing multi-container setups if needed.

Docker is an open-source platform that enables developers to build, deploy, run, update, and manage containers.
Containers are standardized, executable components that combine application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.

Key Benefits of Docker:

1. **Consistency:** Ensures the application behaves the same in all environments by encapsulating container dependencies and configurations.

2. **Isolation:** Each container runs independently, avoiding conflicts between different applications or services.
3. **Portability:** Docker containers run on any system that supports Docker, providing a consistent environment across development, testing, and production.
4. **Resource Efficiency:** Uses fewer resources than traditional virtual machines by sharing the same OS kernel, allowing more applications to run on a single machine and saving hardware costs.
5. **Scalability:** Facilitates application scalability with multiple container instances and efficient workload distribution using tools like Kubernetes or Docker Swarm, adapting quickly to demand, supporting vertical scaling, and providing built-in load balancing.
6. **Faster Development and Deployment:** Simplifies development and deployment by providing identical local and production environments, reducing compatibility issues, speeding up the development cycle, and automating the deployment process.

Challenges

1. **Outdated Documentation and Tutorials:**
   - **Evolving Field:** The field of AI and model deployment is rapidly evolving, which means that many available sources are not up-to-date.
   - **Deprecated Methods:** most tutorials and documentation rely on `RetrievalQA`, which has been deprecated by LangChain.
2. **Finding Suitable Models:**
   - **Efficiency vs. Parameter Count:** identifying models with a low number of parameters that remained efficient and effective for the project's requirements.
   - **Model Exploration:** including Ollama, Facebook/BERT, Mestralai, and various models from HuggingFace.
   - **Final Choice:** After extensive testing, I determined that AI21 models, well-supported in LangChain, offered the optimal balance of performance and parameter efficiency.

Suggestions and Improvements:

1. **Chat Bot:** Instead of a simple input-output QA agent, implementing a chatbot would provide a more interactive and user-friendly experience.

2. **Testing and Quality Assurance:** Implement comprehensive testing strategies, including unit tests, integration tests, and end-to-end tests, to ensure the reliability and robustness of the application.

3. **Bigger Database:** Add more related documents and PDFs, which will improve the accuracy and relevance of the responses.