Project Overview Document

Scientific Paper Analyzer: Cloud-Deployed Summarization and Q&A Tool

Team Members:

Ali Talasaz, Evan Parra, Bharath Sai Vallabhaneni, Ashritha Aloori

Abstract:

The proposed project is a cloud-deployed Scientific Paper Analyzer platform designed to make complex academic literature easier to understand. It helps students and researchers quickly grasp the key elements of scientific papers by automatically summarizing findings, methods, and conclusions. Beyond summarization, the system allows users to ask questions in plain language and receive clear, context-aware answers. To achieve this, the platform uses retrieval-augmented generation (RAG), which retrieves the most relevant sections of the paper and combines them with an AI model to generate accurate answers supported by the paper's content. By integrating both summarization and RAG-powered Q&A into a single tool, the project provides a faster, more reliable way to read, explore, and interact with research documents.

High-Level Project Overview:

Our project is a conversational AI assistant that supports students and researchers by simplifying research papers. Users upload a PDF or text file, and the system processes it through three main stages: summarization, retrieval, and Q&A. The application is built on a modular architecture where each component is responsible for a clear function: document parsing, chunking, embedding, retrieval, summarization, and interactive chat. This modularity ensures that each part of the system can be improved or replaced independently, making the design flexible and maintainable. By deploying this system on the cloud, it will scale easily for multiple users, remain accessible anytime, and provide reliable AI-driven academic support. The use of retrieval-augmented generation (RAG) allows the assistant to provide answers that are grounded in the actual content of the paper, reducing the chance of irrelevant or incorrect responses. In addition, conversational memory enables multi-turn interactions, so users can build on previous questions without losing context. Together, these features make the platform a powerful and practical tool for improving the way researchers and students interact with scientific literature.

Project Components (Software Components):

The project is built using a modular architecture, where each component has a distinct role in processing papers and supporting user interaction. Together, these modules work in sequence to provide summarization, retrieval, and conversational Q&A. The key components include:

1. **Document Loader & Preprocessor:** loads PDFs/Arxiv papers and prepares them for analysis.
2. **Summarization Module:** creates a running summary using ChatNVIDIA and custom prompts.
3. **Embedding & Vector Store:** stores paper chunks in FAISS with NVIDIA embeddings.
4. **RAG Retrieval Chain:** fetches relevant chunks based on user queries.
5. **LLM Backend**: powered by NVIDIA NIM (mixtral-8x7b-instruct) or OpenAI API.
6. **Conversational Memory**: stores past Q&A exchanges in a FAISS-based conversation store.
7. **User Interface**: Gradio-based chat interface for real-time interaction.
8. **Cloud Deployment**: Dockerized and deployed on GCP Cloud Run (or AWS/Azure).

Technology Stack:

The implementation of this project relies on a modern technology stack that combines programming tools, AI frameworks, and cloud services. Each element plays a specific role in enabling summarization, retrieval, and interactive Q&A. The main technologies include:

- **Programming Language:** Python
- **Frameworks & Libraries:** LangChain, Pydantic, FAISS, Gradio, PyMuPDF, Rich
- **AI/LLM Tools:** NVIDIA NIM (ChatNVIDIA, nv-embed-v1) or OpenAI API
- **Vector Database:** FAISS (for semantic search)
- **Cloud Platform:** GCP Cloud Run
- **Version Control:** GitHub
- **Other Tools:** ArxivLoader, UnstructuredPDFLoader, RecursiveCharacterTextSplitter

<u>Project Timeline:</u>

- **Requirement Analysis (Week 1)**
  Develop a project requirement document that includes: "User requirements" to identify user roles and specify requirements for each role and "System requirements" to define functional and non-functional requirements, categorized by type.
- **Solution Architecture (Week 2)**
  Design the modular system architecture with document loaders, summarization, RAG pipeline, vector storage, LLM backend, and chat UI.
- **System Modeling (Week 3)**
  Create sequence and class diagrams to model interactions between users, the RAG pipeline, and core components.
- **Prototype Implementation (Weeks 4–5)**
  Develop ingestion, summarization, embeddings with FAISS, retrieval modules, and a simple Gradio interface.
- **Experimental Prototype (Week 6)**
  Test end-to-end functionality on real papers, validate summarization and Q&A accuracy, and measure latency.
- **Deployment (Week 7)**
  Containerize with Docker and deploy to Google Cloud Run for scalability and public access.
- **Evaluation & Closure (Week 8)**
  Collect feedback, refine the workflow, and deliver the final deployed prototype with project report.

<u>Demo Details:</u>

For the final demo, we will showcase:

1. Uploading a scientific paper (PDF).
2. The system automatically summarizing the paper into key points.
3. Asking the system questions in plain English, such as:
   - "What methods does this paper use?"
   - "Summarize the main contributions."
   - "How does this compare to RAG?"
4. The system returning accurate answers grounded in the document.
5. Displaying the running summary and conversational memory across multiple questions.
6. Deployment in the cloud (GCP) with a public URL for access.