ABSTRACT: MULTI-FILE RETRIEVAL-AUGMENTED GENERATION (RAG) CHATBOT WITH GRADIO INTERFACE:

This project implements a modular, production-ready Retrieval-Augmented Generation (RAG) system capable of performing question answering over heterogeneous document formats. The system is designed for local deployment and integrates a dynamic Gradio Blocks-based user interface with a backend pipeline built using LangChain, Hugging Face Transformers, FAISS, and Sentence Transformers.

SYSTEM OVERVIEW

The architecture supports ingestion, semantic indexing, and querying of multiple document types including PDF, CSV, DOC/DOCX, PPT/PPTX, XLS/XLSX, and TXT. The pipeline is structured into discrete stages, each triggered interactively via Gradio components.

BACKEND PIPELINE

1. MODEL INITIALIZATION

Users specify a Hugging Face model name (e.g., ,) through a text input. The model is loaded using the Hugging Face API and wrapped for downstream use in a LangChain chain. Model load status is displayed to the user.

2. DOCUMENT UPLOAD AND PREPROCESSING

Uploaded files are parsed and normalized into text using format-specific loaders. The text is segmented using to ensure consistent chunking across formats. This enables uniform embedding and retrieval performance.

3. VECTOR STORE CREATION

Text chunks are embedded using the model. The resulting vectors are stored in a FAISS index, which supports fast approximate nearest neighbor search. This index serves as the retriever for the QA chain.

4. QA CHAIN CONSTRUCTION

A LangChain chain is instantiated, connecting the FAISS retriever to the loaded LLM. This chain enables contextual question answering with source-aware responses. The chain setup is triggered via a dedicated Gradio button and its status is displayed.

GRADIO INTERFACE DESIGN

The user interface is constructed using and organized into two primary columns:

- Left Column: Workflow Configuration
- Model selection textbox and load button
- Multi-file upload widget supporting various formats
- Button to create the FAISS vector database
- Button to initialize the QA chain
- Status textboxes for each stage to provide feedback
- Right Column: Chatbot Interaction

• A persistent component for real-time dialogue

• Textbox for user queries

• Submit button to send queries to the bot

• Clear button to reset the chat history

CHAT LOGIC AND CALLBACK FLOW

The chatbot interaction is managed by a function, which:

• Validates the input query

• Passes the query and chat history to a custom function

• Appends the response to the chat history and returns the updated state

GRADIO CALLBACKS ARE DEFINED FOR EACH INTERACTIVE COMPONENT:

• triggers model loading

• handles document ingestion

• creates the FAISS index

• initializes the QA chain

• handle user queries

• resets the chat interface

EXAMPLE QUESTIONS

The interface includes a set of predefined example queries to guide users:

• What is the main topic discussed in the documents?

• Can you summarize the key points from the uploaded files?

• What are the important dates or numbers mentioned?

• Explain the methodology described in the documents.

• What conclusions can be drawn from the data?

EXTENSIBILITY AND MODULARITY

The system is designed with modularity in mind, allowing easy substitution of models, retrievers, and document loaders. Each stage is independently triggered and monitored, ensuring transparency and reproducibility. The use of LangChain and Gradio enables rapid prototyping and deployment of advanced document-based QA systems.