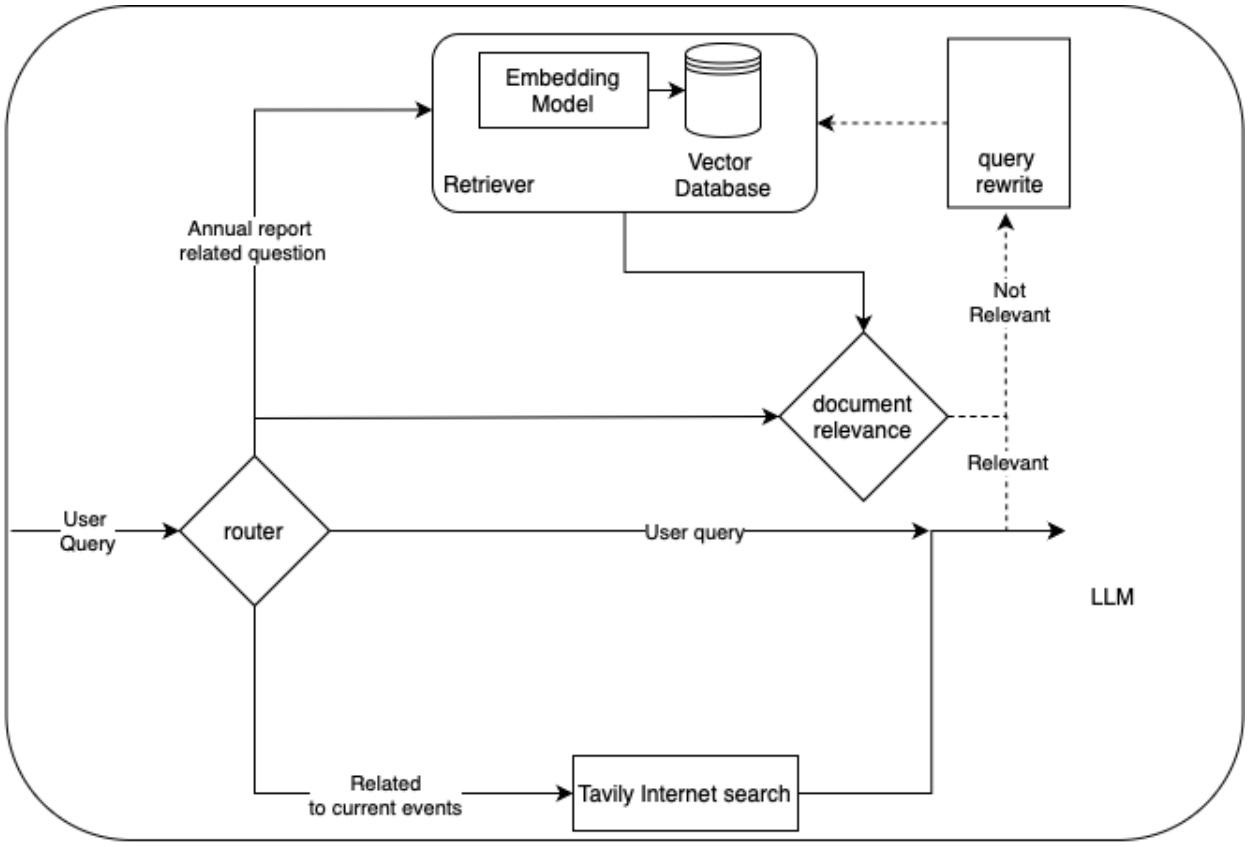
Assignment: Implementing a Retrieval-Augmented Generation (RAG) Chatbot with Internet Search



Architecture overview of the RAG chatbot

Introduction

In this assignment, your task is to develop a Retrieval-Augmented Generation (RAG) chatbot using LangGraph. The chatbot will integrate a robust retrieval mechanism, internet search functionality, and large language models (LLMs). It will intelligently handle queries related to NetSol's financial report, current events, and general questions by routing them to the appropriate workflows.

You are required to process the financial report at the following URL:
NetSol Financial Statement 2024 Part 1

The chatbot must be capable of handling complex tables, extracting detailed information, and providing accurate answers.

You will connect all the components into a Gradio app and deploy it on HuggingFace Spaces, where it will be tested.

Assignment Objectives

1. **Data Ingestion:** Use **unstructured.io** to process the PDF report into a structured format that supports complex table extraction and detailed analysis.
2. **Workflow Design:** Implement a routing mechanism that identifies if a user query is:
    - Related to NetSol's finances.
    - A question about current events requiring live data.
    - A general information request.
3. **RAG Implementation:** Use LangGraph to integrate a retrieval system for queries on the PDF and **Tavily** for internet-based queries.
4. **Evaluation:** Implement RAGAS metrics to evaluate the chatbot's performance for:
    - Answer **Correctness**
    - Answer **Faithfulness**
5. **Deployment:** Build a Gradio-based user interface for the chatbot and deploy it on HuggingFace Spaces.

Workflow Requirements

Refer to the **attached diagram** for structuring your workflow. The workflow must:

1. Route the user query to determine the appropriate module (retriever, internet search, or LLM response).
2. If related to the financial report, retrieve relevant information using vector embeddings.
3. If related to current events, leverage Tavily to fetch live information.

4. Use an LLM to provide contextual answers, rephrase queries, or enhance the user experience.

Tasks

1. **Process the PDF:**
    - Use **unstructured.io** to ingest the financial report and extract structured data, ensuring proper handling of complex tables and data points.
2. **Develop Query Routing:**
    - Implement a query routing system to classify user queries into three categories: finance-related, live-event related, or general.
3. **Build RAG Workflow:**
    - Integrate a retrieval mechanism using an embedding model and vector database for finance-related questions.
    - Use **Tavily** for internet search to address current event queries.
    - Connect these modules with a large language model to generate the final responses.
4. **Implement Evaluation Metrics:**
    - Use **RAGAS** metrics to measure:
        - **Correctness**: Does the response correctly address the query?
        - **Faithfulness**: Is the response grounded in the retrieved data?
5. **Create a Gradio App:**
    - Build an interactive user interface using Gradio.
    - Deploy the app on HuggingFace Spaces, making it publicly accessible for testing.

Deliverables

1. **Source Code:** Submit the complete source code for your implementation.

2. **Processed Data:** Provide the structured data extracted from the financial report.
3. **Evaluation Results:** Include RAGAS evaluation scores and a summary of correctness and faithfulness.
4. **Gradio App:** Share the link to your deployed app on HuggingFace Spaces.

Submission Instructions

- Package your source code and processed data into a shared repository (GitHub or similar).
- Provide a README with setup instructions for running the code and app.
- Include the evaluation results in the repository.
- Submit the HuggingFace Spaces URL for your Gradio app.

Deadline

You have **one week** from the date of assignment distribution to complete and submit your work.

Evaluation Criteria

1. **Technical Accuracy:**
    - Proper implementation of data ingestion, RAG workflow, and query routing.
2. **Performance Metrics:**
    - High scores in RAGAS metrics for correctness and faithfulness.
3. **Deployment Quality:**
    - A functional and user-friendly Gradio app.
4. **Code Quality:**
    - Clean, well-documented, and modular code.

Reference Diagram

Use the **attached diagram** as a visual guide for designing and implementing the RAG workflow. Ensure all components are accurately integrated and follow the outlined steps.

Good luck! This is your opportunity to showcase your understanding of RAG workflows and demonstrate your ability to build end-to-end applications.