LangGraph-AI-Agent

Introduction

This project showcases an advanced LangGraph-based AI Agent capable of answering user queries related to both weather information via OpenWeather API and document-based questions through RAG (Retrieval-Augmented Generation) from a sample PDF. The application is designed to demonstrate multi-step reasoning, API orchestration, and context-aware query routing.

Objective

To build a multi-functional AI agent that can accurately respond to real-time weather queries and provide insightful answers based on document content using LangChain, LangGraph, and LangSmith.

Key Features

- Weather query understanding with keyword recognition and city name extraction
- RAG-based PDF QA using Qdrant vector store
- Intelligent routing between weather and document QA
- LangSmith integration for trace logging and evaluation
- Streamlit-based UI with session persistence and row-aligned history display

Technologies used

- Python
- LangChain, LangGraph, LangSmith
- OpenAI GPT 3.5
- Qdrant Vector DB
- OpenWeather API
- Streamlit
- Hugging Face Spaces for deployment

Architecture

The architecture includes a LangGraph state machine that routes incoming queries either to the weather node or the document QA node based on keyword matching. Each node performs the required action, and the UI displays answers row by row with timing and logs recorded in LangSmith.

Approach

The core approach revolves around a dynamic routing mechanism that intelligently determines whether a user's query is related to weather data or the RAG-based PDF content. This is achieved through a dedicated router node, which scans the query for weather-related keywords like "temperature," "forecast," "humidity," or "rain." If such keywords are detected, the query is routed to the Weather node; otherwise, it is handled by the RAG node using the Qdrant vector database.

The UI is designed to reflect this logic transparently: for every response, the application displays a **Query**, **Answer**, **Source**, and **Reason** in a structured row-wise layout. For example, if the response is derived from the OpenWeather API, the source is displayed as **"OpenWeather"** and the reason is noted as **"Weather-related keywords detected"**. Similarly, if the query pertains to PDF content, the source is **"RAG PDF"** with a corresponding reason. This makes the interaction both interpretable and professional, enhancing user trust in the system.

Challenges Faced with Solutions

1. City name misidentification – Resolved by regex and NLP token filtering
2. Weather keyword misclassification – Fixed with enhanced routing logic
3. UI alignment – Implemented row-by-row display for improved readability
4. Session clear limitations – Removed 'Clear' button to avoid Streamlit reset issues
5. LangSmith integration – Manually embedded environment configs and verified logs

Conclusion

The LangGraph-based AI Agent successfully bridges structured and unstructured data retrieval by combining API-based weather queries with semantic PDF document understanding. Through intelligent routing, precise keyword identification, and source-aware answer justification, the agent delivers accurate and contextually appropriate responses. The clean UI and structured display of queries, answers, sources, and reasons ensure clarity and professionalism. Challenges like city name recognition and weather keyword mapping were resolved through robust NLP enhancements. The project is fully deployable, thoroughly evaluated via LangSmith, and well-documented. This system serves as a solid foundation for intelligent multi-source querying in real-world applications.