Submitted by: Krishna Suthar

Qualifying Assignment: AI agent-based Deep Research

Introduction

The **Deep Research AI Agentic System** is designed to enhance information gathering and answer drafting processes using advanced AI techniques. The system employs a **dual-agent setup** to separate the tasks of research and answer drafting, ensuring a streamlined workflow for research-based problem-solving.

This system utilizes the following tools and frameworks:

- **Tavily** for online information gathering.
- **LangChain** for language model integration.
- **LangGraph** for organizing the workflow using a state graph.

The system is composed of two agents:

1. **Research Agent**: Collects and compiles relevant data based on the user's query.
2. **Answer Drafting Agent**: Generates a detailed response based on the gathered data.

The agents work sequentially within a state graph structure, where the output of the Research Agent serves as input to the Drafting Agent.

Objectives

- Develop an AI-driven research system that can crawl websites using Tavily.
- Implement a dual-agent architecture using LangChain and LangGraph.
- Ensure the system can handle various queries, gather relevant data, and generate coherent, well-researched answers.

Tools and Frameworks

Tavily

Tavily is an API that allows for efficient web scraping and data collection from online sources. This tool is used to gather relevant documents and content related to a research query.

LangChain

LangChain is a framework for building AI-powered applications with language models, facilitating the interaction between language models (like GPT-3.5) and various data sources, making it perfect for handling large volumes of text data.

LangGraph

LangGraph is used to define the workflow of the agents, manage state transitions, and ensure smooth execution from research to answer drafting.

System Architecture

The system is composed of the following key components:

1. **Research Agent**:
   - Collects data from Tavily based on a user's query.
   - Filters and compiles relevant content into a single data structure for use by the Drafting Agent.
2. **Answer Drafting Agent**:
   - Uses LangChain's GPT-3.5 model to process the gathered data.
   - Generates a detailed response by analyzing the compiled content.
3. **LangGraph**:
   - Coordinates the execution of agents by creating a stateful graph where agents are connected and perform actions sequentially.
   - Defines the flow of data between the Research Agent and the Drafting Agent.

Workflow

1. **User Input**: The system prompts the user to input a research query.
2. **Research Agent**:
   - The system uses the Tavily API to search for relevant data based on the user's query.
   - The data is compiled into a document.
3. **Answer Drafting Agent**:
   - The compiled data is passed to the Answer Drafting Agent, which uses a GPT model to generate a detailed answer.
4. **Output**: The system outputs a final, well-structured answer to the user.

Code Explanation

Research Agent

```
def research_agent(state: ResearchState) -> ResearchState:
    results = tavily.search(state["query"], max_results=5)
    docs = [r["content"] for r in results["results"]]
    compiled = "\n\n".join(docs)
    return {"query": state["query"], "data": compiled}
```

The Research Agent queries Tavily for data based on the input query, compiles the results, and returns the gathered content in a structured format.

Answer Drafting Agent

```
def answer_agent(state: ResearchState) -> str:
    prompt = f"Based on the following data, write a detailed, well-researched response:\n\n{state['data']}"
    return llm.invoke(prompt)
```

The Answer Drafting Agent generates an answer by processing the compiled data using the GPT-3.5 model. The agent is designed to handle large chunks of text and produce coherent answers.

LangGraph Workflow

```
def create_graph():
    builder = StateGraph(ResearchState)
    builder.add_node("Research", research_agent)
    builder.add_node("Drafting", answer_agent)
    builder.set_entry_point("Research")
    builder.add_edge("Research", "Drafting")
    builder.add_edge("Drafting", END)
    return builder.compile()
```

LangGraph defines the process flow, specifying the research and drafting steps. It ensures that the workflow progresses sequentially, from data collection to answer generation.

Challenges and Solutions

1. **Handling Large Data Volumes**:
   - Challenge: Research data can be voluminous, making it difficult for the system to process effectively.

    - Solution: The system limits the number of results returned from Tavily and compiles the data into manageable chunks.

2. **Ensuring Coherent Answer Generation**:
    - Challenge: Generating a well-structured and coherent answer from raw data.
    - Solution: The use of GPT-3.5 allows the Drafting Agent to generate detailed and accurate responses based on the structured data.

3. **State Management**:
    - Challenge: Ensuring smooth state transitions between agents.
    - Solution: LangGraph's stateful design ensures that the research and drafting steps are linked and executed in the correct order.

Future Work

- **Advanced Data Filtering**: Implement more sophisticated algorithms for selecting the most relevant data from the search results.
- **User Customization**: Allow users to customize the format and style of the generated answers.
- **Improved Answer Quality**: Integrate additional language models for more accurate and in-depth responses.

Conclusion

The Deep Research AI Agentic System is an innovative approach to automating the research process. By leveraging Tavily, LangChain, and LangGraph, the system effectively gathers relevant data and generates high-quality, detailed answers. The dual-agent architecture enhances the overall workflow, ensuring efficiency and scalability.

GitHub Repository

The full implementation can be found at the GitHub repository: [https://github.com/123krissh/Deep-Research-AI-Agentic-System].