HCLTech | Supercharging Progress™

AI & Cloud Native Labs
Junior AI Engineer

Coding Exercise

[Junior AI Engineer – Coding Exercise v0.1]
[Prepared by – Ashish S]
[Public]

Contents

1 AI Junior Engineer – Coding Exercise

1.1 Exercise

Stateless LLM Wrapper Development with LangChain for Enhanced Reasoning

1.2 Overview

Develop a software wrapper that integrates a base Language Model (LLM) with LangChain to provide enhanced reasoning capabilities, focusing on real-time interactions without maintaining conversation history. This setup should facilitate handling complex reasoning queries akin to conversational AI platforms like ChatGPT.

1.3 Tools Needed:

- Language Model of choice for natural language understanding and generation.
- LangChain for enhanced logical reasoning.
- Python as the primary programming language for backend development.
- A UI Framework (e.g., React or Angular etc) for creating an interactive front-end.
- API Development Tools for integrating the LLM with the web interface and LangChain.

You are allowed to use GitHub CoPilot, ChatGPT, Gemini or any other tool

1.4 Task

- Select and Set Up the LLM:
  - Choose a robust Language Model (like GPT) that can understand and generate human-like text. Set up the model in your development environment.

- Integrate LangChain:
  - Incorporate LangChain into your setup to enhance the LLM's reasoning capabilities, enabling it to handle complex, logic-based queries more effectively.

- Develop the User Interface:
  - Design: Create a clean and intuitive interface using a modern UI framework (e.g., React.js). The design should include a text input for queries and an area to display responses.
  - Implementation: Develop the frontend components, ensuring the interface is responsive and user-friendly.

- API Endpoint:
  - Set up a /query endpoint in the backend using a server framework like Flask to receive user queries, process them with the integrated LLM and LangChain, and return the responses in JSON format.

- Process Queries:
  - Implement logic on the backend side to receive user queries from the frontend.
  - Use the LLM integrated with LangChain to process these queries and generate responses.

- Handle Responses:
  - Develop functionality to send the processed responses back to the frontend for display to the user.

- Test and Validate:
  - Functional Testing: Test the complete system to ensure it works as expected, with the LLM and LangChain correctly processing queries and returning the appropriate responses.

- Usability Testing: Ensure the user interface is easy to navigate and use, and that it meets the needs of the end users.

- Deployment:
  - Prepare the application for deployment.
  - Choose a suitable hosting environment and deploy the application, making sure it is accessible to users

1.5 Optional Task

- **RAG Model Integration:**

  Integrate a Retrieval-Augmented Generation (RAG) model to enhance the LLM's ability to provide more informed and context-rich responses. This integration would allow the wrapper to pull relevant data from a vast knowledge base during interactions, improving the accuracy and relevance of responses to complex queries.

  Business Use Case Example:

  In a customer support scenario for a tech company, the RAG-enhanced LLM can quickly access a vast repository of tech support documents and past customer interaction logs to provide detailed troubleshooting steps, product recommendations, and technical explanations tailored to customer queries, thereby reducing response times and increasing customer satisfaction.

  Sample Data

- **Containerization:**

  Implement containerization using Docker to ensure that the application can be easily deployed, scaled, and managed across different environments. This would help in maintaining consistent performance and simplifying the deployment process on various platforms, including cloud-based services.

1.6 Submission Requirements:

- **Source Code:**

  - Upload the complete source code to a GitHub repository. Include detailed documentation in the README on how to set up, install, and operate the system, emphasizing its integration with LangChain and its stateless operation.

- **Demonstration Video:**

  - Produce a video demonstrating the wrapper's capabilities in handling complex reasoning interactions without maintaining any conversation history. Show how it processes individual reasoning queries independently. Provide access to the video through the GitHub repository.

- **API Documentation:**

  - Document the API endpoints created for interacting with the LLM and LangChain. Include parameters, expected responses, and example requests to guide developers in integrating with the wrapper, highlighting the enhanced reasoning and stateless nature of the interactions.

1.7 Sample Data for Optional RAG

You can create multiple such mock document

Setting Up Your Wireless Printer

Troubleshooting Network Connectivity Issues

HCLTech | Supercharging Progress™

hcltech.com