Chatbot Using LLM with Streamlit and Hugging Face

Objective

The purpose of this practical exercise was to develop an interactive chatbot interface using a Large Language Model (LLM) from Hugging Face. The chatbot is deployed as a web application using Streamlit, and it is capable of maintaining a short memory of previous interactions using LangChain's conversational framework.

Tools

- **Libraries Used**:
  - streamlit for UI
  - transformers from Hugging Face
  - langchain for chaining LLM calls and memory management
- **Model**: microsoft/DialoGPT-medium (also tested: distilgpt2)
- **Development Platform**: macOS Terminal + Browser

Dataset

There was no dataset required. The chatbot works in a live chat setting, generating its responses dynamically based on the user's input.

Methodology

1. **Model Selection**
   We chose microsoft/DialoGPT-medium because it is trained specifically for dialog-style responses. We also tested distilgpt2 for lightweight deployments.
2. **Pipeline Setup**
   The model was loaded using Hugging Face's pipeline() method for text generation, and then wrapped with HuggingFacePipeline from LangChain.
3. **Prompt Engineering**
   A simple input-output pattern was used:

User: *Your message here*
Bot:

The model was left to generate a short continuation that served as the chatbot's reply.

4. **Memory Management**
   To simulate ongoing conversation, LangChain's ConversationBufferMemory was used, preserving recent user inputs and generated replies.
5. **Streamlit Interface**
   - A chat interface with st.chat_input() and st.chat_message() displayed messages.
   - Sidebar settings allowed dynamic change of model (DialoGPT, distilgpt2) and token limits.

Results & Observations

- **Response Relevance**: DialoGPT performed reasonably well on casual conversations and simple explanations. However, it sometimes generated vague or cut-off responses, expected behavior for medium-sized un-finetuned LLMs.
- **Latency**: Responses generated in < 2 seconds on local machine (MacBook M1) with distilgpt2, and ~4 seconds with DialoGPT.
- **Memory Performance**: LangChain memory kept recent context, but due to model limitations, it could not always leverage it effectively. Adding few-shot examples in prompts would improve continuity.
- **Usability**: The Streamlit interface allowed seamless interaction and fast prototyping. Its chat_input() and chat_message() components mimicked a modern chat UI.

Conclusion

This exercise demonstrated how an LLM can be quickly turned into an interactive chatbot using Streamlit and LangChain. While base models like DialoGPT can handle

general conversation well, more advanced models or fine-tuning would be required for professional applications. Overall, this project provided a strong foundation for building intelligent, interactive NLP assistants.