**Lab 2 Report**

1. **Project Overview**

This project is a web-based chat interface developed using the Gradio framework, designed to support switching and invoking multiple large language models (LLMs). It provides features such as streaming response and multi-turn conversation history management, allowing users to interact with different models in natural language more conveniently.

The system supports calling models like OpenAI and DeepSeek via unified interface encapsulation, enabling modular management and easy expansion. The frontend is built using Gradio, featuring clear components and a clean, intuitive interaction design.

1. **Project Functions**

|  |  |
| --- | --- |
| Function Category | Description |
| Multi-Model Support | Enables successful API calls to LLMs to answer user questions, with a basic input-output interface. |
| Streaming Response | Implements token-level real-time output to enhance the interactive experience. |
| Basic Chat Functionality | Supports LLM API calls to answer user questions through a simple chat interface. |
| Multi-Turn Conversation | Maintains context by recording previous messages within the same session, allowing the model to generate more coherent responses. |
| Chat History Management | Allows users to view past conversations with the LLM and resume chats from previous dialogue records. |

**3.1 Are There Differences in API Calls Across Different Models?**

**If So, What Are They?**

**3.1.1 Differences in Request Structure**

* OpenAI (GPT-3.5/4) uses a messages field that contains a list of role-based dialogue elements (system, user, assistant).
* Anthropic Claude uses prompt or messages but does not strictly separate system/user/assistant roles.
* DeepSeek / Moonshot generally follow OpenAI's format but with minor adjustments in field names (e.g., stream vs. streaming).

These structural differences require the developer to implement a dynamic adaptation mechanism in the backend to ensure prompt compatibility across models.

**3.1.2 Different Authentication Mechanisms**

**Each model provider uses its own authentication scheme:**

* OpenAI and DeepSeek typically use a Bearer Token in the header.
* Claude requires an x-api-key in the request header.
* Moonshot may also require an additional signature-based verification.

Securely managing API keys and injecting the correct headers at runtime is a crucial part of safe and reliable system development.

**3.1.3 Differences in Response Formats**

The returned data structure also varies across models:

* OpenAI: response['choices'][0]['message']['content']
* Claude: response['completion']
* DeepSeek: response['data']['content']

Therefore, the system needs to implement a "unified response parser" to ensure consistent downstream processing regardless of the model source.

**3.2 Design Philosophy Behind the User Interface: Why This Layout Was Chosen and the Purpose of Each Component**

**Interface Preview:**



1. **Model Selection:**



* Purpose:

Allows users to switch between different LLMs in the chat interface, helping them experience and compare the behaviors of various models.

* Design Reasoning:

Adding a model selection component makes it convenient for users to switch models according to different tasks. This enhances usability and emphasizes the distinct characteristics of each LLM.

1. **Conversation Management:**



* Purpose:

Enables users to manage their sessions—such as creating new conversations or deleting existing ones.

* Design Reasoning:

Since users may engage in multiple conversations simultaneously, introducing a session management component allows for easier context switching and organized interactions.

1. **Chat Window Layout:**



* Purpose:

Provides a clear and focused area where users can view the model’s replies.

* Design Reasoning:

Placing the conversation output in a large, right-side window emphasizes the model’s response and improves readability and user focus.

1. **Input Type Selector:**

* Purpose:

Allows users to choose different types of input—text, image, audio, etc.

* Design Reasoning:

In many cases, users may want to upload images or send voice messages. By offering input-type selection, the interface becomes more intuitive and flexible, improving user experience.

**3.3 The Impact of Adjustable Parameters in Large Language Models (e.g., Temperature, Top-p, etc.) on Generated Output**

The sampling temperature controls the randomness of the generated output. When the temperature is low, the model’s responses are more deterministic and focused; when the temperature is higher, the model tends to generate more creative content, but it may also introduce more uncertainty or irrelevant information.

The nucleus sampling threshold (Top-p) limits the random selection of the next word to the smallest set of tokens whose cumulative probability exceeds a certain value *p*. Unlike Top-k sampling, which selects from a fixed number of top candidates, Top-p sampling dynamically selects tokens based on cumulative probability. A lower Top-p value leads to more concentrated and deterministic outputs, while a higher value introduces more diversity and creativity. By adjusting these parameters flexibly, one can balance the accuracy and diversity of responses according to the specific task requirements.

**3.4 Prompt Styles Can Influence LLM Output – What Style Do You Recommend for Daily vs. Professional Use?**

In the application of large language models, the style of prompt design has a significant impact on the quality and tone of the generated results. Specifically, different usage scenarios require differentiated prompting strategies to enhance the relevance, accuracy, and practicality of the generated content.

Firstly, in everyday usage scenarios, users tend to prefer natural and flexible human-machine interactions, where questions are often open-ended and ambiguous. Therefore, it is recommended to use a natural language style for such prompts—expressing needs as if talking to a friend. These prompts are loosely structured and emphasize the model’s language understanding and generation capabilities, without the need to provide excessive background information or specify output format. This style is well-suited for creative writing, emotional companionship, and everyday knowledge Q&A. Its advantages include being user-friendly, easy to get started with, and highly adaptable. However, its disadvantages lie in the relative lack of output stability and format control.

In contrast, for tasks requiring a high degree of professionalism—such as technical document generation, academic report writing, code analysis, and legal interpretation—it is advisable to adopt a more structured and instruction-driven prompting style. Structured prompts can significantly improve the accuracy, logical consistency, and controllability of the model’s output, and also contribute to better reproducibility, meaning more consistent results across multiple generations. The downside of this style is that it requires higher prompt-writing effort, demanding users to have a certain level of task abstraction and command expression skills.