同济大学

TONGJI UNIVERSITY

Design and Implementation of a Multi-Model Large Language Model Chat System with Gradio Interface

Course Name: Human-Computer Interaction

Author: 2351050 , YANG Ruichen
2350989 , ZHANG Qizheng

April 21, 2025

Abstract

This paper presents the design and implementation of an interactive user interface for **Large Language Model (LLM)** systems using the Gradio framework. The system enables seamless communication with multiple LLM providers through standardized API calls and offers functionalities including single-turn dialogue, multi-turn conversation history, and user-friendly model selection. The architecture allows comparative analysis of LLM behavior under different configurations and highlights the impact of parameters such as temperature and nucleus sampling on model output. The system also introduces practical prompt design strategies for both casual and professional scenarios to optimize the quality and relevance of AI-generated responses.

1 Introduction

In recent years, the rapid development of Large Language Models (LLMs) has significantly improved the capabilities of artificial intelligence in understanding and generating human-like text. This project aims to explore the integration of multiple LLMs into a unified user interface using Gradio, allowing comparative studies of their behaviors and facilitating an enhanced user experience.

2 System Architecture and Implementation

The proposed system is designed to support dialogue interactions with multiple LLMs via API, including models provided by DashScope, OpenRouter, and DeepSeek. The system architecture is composed of two main components:

- **Backend:** The backend leverages the OpenAI-compatible client to streamline interactions with various LLM providers. Despite variations in authentication and endpoint configurations, a uniform request structure is maintained across different models, including a system role for context-setting, user input, and historical conversation records.
- **Frontend:** Gradio [1] is employed to create a responsive, accessible user interface featuring:
  - Model selection dropdown.
  - Multi-turn conversation display.
  - History management for dialogue context retrieval.
  - Real-time streaming of LLM responses.
  - User feedback mechanisms, such as typing indicators and conversation status updates.

3 Discussion on Cross-Model Differences in API Integration

During development, we observed several differences in how APIs for different LLM providers must be integrated. While our system uses an OpenAI-compatible interface for uniformity, the following distinctions are noteworthy:

i. **Endpoint URL Structure:** Each provider has a unique API endpoint structure, requiring individual configuration:
   - DashScope: Uses Alibaba Cloud's dedicated LLM endpoint.
   - OpenRouter: Offers a routing service to multiple public models.

- DeepSeek: Provides a campus-level LLM interface optimized for educational and research use.

These differences necessitate custom endpoint configuration for each provider [2].

ii. **Authentication Mechanism:** Authentication is another area where variations exist. Every provider requires a unique API key, which is typically tied to usage quotas and access rights. To ensure proper authentication, it's crucial that API keys are correctly bound to their respective endpoints.

iii. **Model Identification:** Each provider has its own model naming convention, which must align with their internal registry:

- OpenRouter: Uses fully qualified names, e.g., moonshotai/kimi-vl-a3b-thinking:free.
- DashScope: Employs shorter model names, e.g., qwen-max.

Ensuring the model name matches the provider's naming convention is essential for successful integration.

These observations suggest that even when a unified interface is adopted, developers must account for backend variations when designing multi-provider systems.

4 User Interface Design

Special attention was paid to the UX design of the interface to ensure both technical clarity and user comfort. The UI was iteratively refined through heuristic evaluation and informal user testing to balance visual clarity, usability, and flexibility.

4.1 Design Goals

- Support multiple LLM providers through a unified interface.
- Make conversation history intuitive and easy to navigate.
- Provide real-time feedback during model interactions.

4.2 Component Roles

In the left side of the interface, we have a vertical panel that contains components (1), (2) and (3) as a control center. The right side contains components (4), (5) and (6), dedicated to the conversation. The layout is designed to be responsive, allowing for easy navigation and interaction. The main components are as follows:

(1) **New Chat Button:** Initiates a new conversation, clearing the current chat history.

(2) **Model Selection Dropdown:** Allows users to switch between LLM models easily while showing the provider for transparency.

(3) **Conversation History Dropdown:** Lets users revisit past conversations, aiding continuity and learning, which showing in Figure 2.

(4) **Chatbot Display:** Presents AI responses in a structured markdown-based chat interface, supporting LaTeX for technical answers.

(5) **Text Input Box:** Enables users to input questions or instructions, designed to blend with modern chat app conventions.

(6) **Stop and Send Buttons:** Allow users to manage their conversation sessions and control the response generation dynamically.

(7) **Streaming Response Feature:** Offers real-time response visibility, improving perceived system responsiveness.

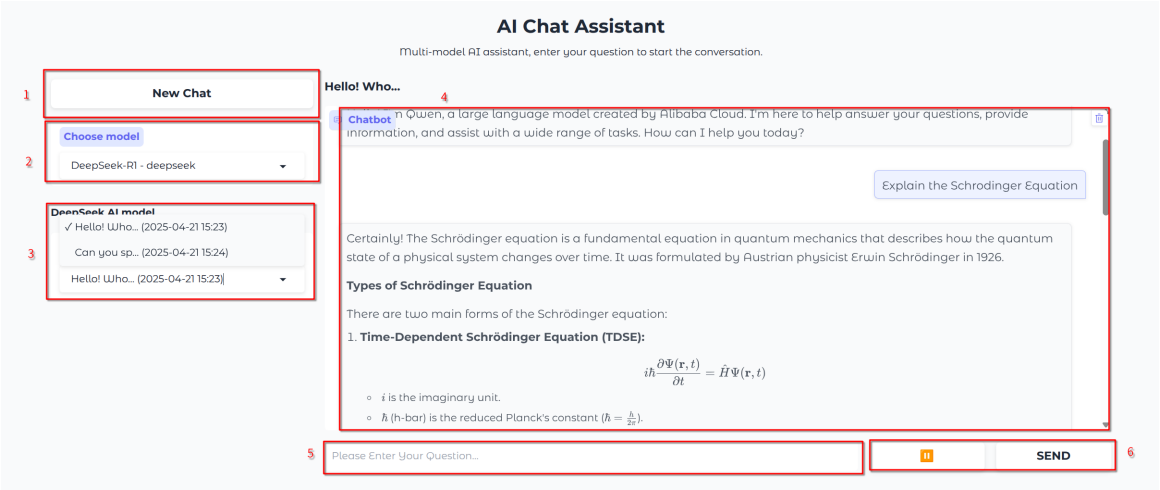

Figure 1: Overall layout of the user interface highlighting the core components

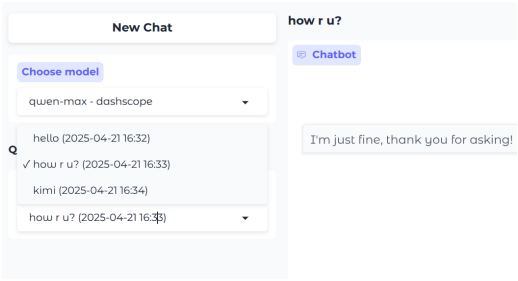

Figure 2: Revisiting and resuming previous dialogues

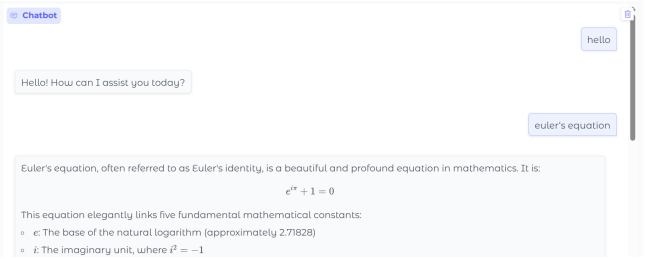

Figure 3: The chat history system maintains context across multiple sessions

4.3 Design Rationale

The UI design choices reflect several key considerations:

- **Minimizing Cognitive Load:** The layout is clean and focused, reducing the mental effort required to use the application.
- **Progressive Disclosure:** Advanced options are available but do not crowd the primary interaction area.
- **Conversation Continuity:** The chat history system allows users to maintain context across multiple sessions, which showing in Figure 3.
- **Visual Feedback:** The ”Thinking...” indicator provides immediate feedback while waiting for model responses.

- **Accessibility:** *Markdown* rendering enhances readability, while LaTeXsupport ensures mathematical expressions display correctly.
- **Real-time streaming feedback:** Enhances transparency and engagement, especially when long responses are being generated.

In summary, the design achieves a thoughtful balance between simplicity and functionality. It caters to both novice users seeking ease of use and advanced users desiring fine-grained control. The well-organized layout and responsive features ensure that users can engage in productive conversations without being burdened by the complexities of underlying AI interactions.

5 Impact of Model Parameters on LLM Output

Modern large language models typically provide several adjustable parameters that allow users to influence the nature and style of the generated responses. The most significant parameters are as follows:

1. **Temperature [3]:** The temperature parameter controls the randomness of the model's output. It determines how likely the model is to select lower-probability tokens over high-probability ones during text generation.

Table 1: Impact of Temperature on LLM Behavior

Temperature	Model Behavior	Recommended Use Cases
0.1 – 0.3	Deterministic, consistent	Technical documentation, fact-based Q&A
0.4 – 0.7	Balanced, creative yet relevant	General conversations, brainstorming
0.8 – 1.0	Highly diverse, sometimes off-topic	Creative writing, idea generation

2. **Top_p (Nucleus Sampling) [4]:** Top_p sampling specifies the cumulative probability mass of the most likely tokens considered during text generation. A lower value constrains the model to select from only the most probable words, while a higher value allows more diverse and unexpected responses. For example, setting top_p=0.9 ensures the model samples only from the smallest set of tokens whose combined probability is at least 90%.
3. **Max Tokens [5]:** This parameter specifies the upper limit for the number of tokens (roughly corresponding to words) in the model's output. Limiting the token count prevents excessively long or verbose responses and controls computational cost, especially for applications that require real-time interaction.
4. **System Messages [6]:** The program currently sets a static system prompt (*You are a helpful assistant.*) for all models, but this field can be customized to significantly influence the assistant's tone, behavior, and response structure. For example, modifying the system prompt to "You are a critical academic reviewer" can encourage more formal and evaluative answers.

Through the careful adjustment of these parameters, developers can align the model's responses with specific application requirements, such as accuracy, creativity, brevity, or fluency.

6 Analysis of Prompt Design and Recommended Strategies

The performance of LLMs is highly sensitive to the way prompts are formulated. Properly structured prompts not only yield more relevant answers but also significantly reduce ambiguity in both casual and technical contexts. Based on practical observations during the system development process, the following strategies are recommended:

6.1 Daily Use Cases [7]

For informal, everyday questions, prompt clarity and conversational framing are crucial:

- **Explicit Intent:** Clearly state the desired outcome to avoid vague or overly broad prompts.
  - Poor: *Tell me something interesting.*
  - Better: *Tell me an interesting fact about deep-sea creatures.*
- **Contextual Framing:** Providing context helps the model generate more targeted responses.
  - Example: *I'm planning to visit Paris in autumn. What clothes should I pack?*
- **Direct Questions and Constraints:** The model performs better when clear constraints are specified.
  - Example: *List three healthy snacks that require no cooking.*
- **Step-by-Step Instructions:** Breaking down complex prompts into logical steps improves output structure and accuracy.
  - Example: *Explain how to bake a cake step-by-step.*

6.2 Professional and Technical Use Cases [8]

For specialized tasks, precise language and role definition are especially important:

- **Technical Jargon:** Use domain-specific terminology to ensure the model understands the context.
  - Example: *Explain the difference between supervised and unsupervised learning.*
- **Role Assignment:** Assigning a specific role to the model can guide its response style and content.
  - Example: *You are a financial analyst. Provide an overview of current stock market trends. You are a software engineer. Review the following Python code for potential optimizations.*
- **Format Expectations:** Specify the desired output format to ensure clarity and usability.
  - Example: *Provide your answer in table format, with three columns: Algorithm, Time Complexity, Space Complexity.*
- **Chain-of-Thought Prompting:** Encourage the model to reason out loud, which can reduce the risk of incorrect answers for complex tasks.
  - Example: *Explain your reasoning step by step before providing the final answer.*

- **Contextual Expertise Specification:** Tailor the model's response to the user's knowledge level to avoid over- or under-explanation.
  - Example: *Explain the theory of relativity as if I were a high school student.*

This approach ensures the generated output matches the user's knowledge level, avoiding over- or under-explanation.

7 Conclusion

This project demonstrates the feasibility and advantages of building an interactive LLM user interface that supports multiple model providers through standardized API communication. Our design approach emphasizes usability, adaptability, and maintainability. Through comparative analysis, we have shown how differences in provider APIs, model configuration parameters, and prompt design strategies can significantly influence the performance and reliability of AI-generated content. These insights are essential for developing user-friendly and application-specific AI tools in both academic and industrial environments.

References

[1] Gradio Team. Gradio documentation. https://www.gradio.app/docs, 2025.

[2] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[3] Paul Windisch, Fabio Dennstädt, Christian Koechli, et al. The impact of temperature on extracting information from clinical trial publications using large language models. *Cureus*, 16(12):e75748, 2024.

[4] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[5] A. Mao. Large language model settings: Temperature, top p and max tokens. https://vectorshift.ai/blog/large-language-model-settings-temperature-top-p-and-max-tokens, January 15 2024.

[6] X. Amatriain. Prompt design and engineering: Introduction and advanced methods. https://arxiv.org/abs/2401.14423, 2024.

[7] Tom's Guide. I write about ai for a living — here's my top 5 chatgpt prompt tips. https://www.tomsguide.com/ai/chatgpt/i-write-about-ai-for-a-living-heres-my-top-5-chatgpt-prompt-tips, April 20 2025.

[8] The Guardian. Ai prompt engineering: learn how not to ask a chatbot a silly question. https://www.theguardian.com/technology/2023/jul/29/ai-prompt-engineering-chatbot-questions-art-writing-dalle-midjourney-chatgpt-bard, July 29 2023.