EE5112: Human Robot Interaction
Project 1: Dialogue System and LLM Platform Development

Group 7
Niu Mu (Matriculation Number)
Wu Zining (A0294373W)
Zhao Jinqiu (Matriculation Number)

September 28, 2025

Contents

1 Abstract **4**

2 Introduction **4**

2.1 Background 4

2.2 Project Objectives 4

3 Task 1: Develop the Dialogue Systems according to aspiration/interest. **5**

4 Task 2: Develop Local Dialogue Systems by Using Open-Source LLMs **5**

4.1 Literature Review on Different Categories of LLMs 5

4.2 Local LLM Platform Implementation 6

4.2.1 System Architecture 6

4.2.2 Implementation Details 6

4.2.3 Hardware Optimization 7

4.2.4 System Evaluation 8

4.3 Performance Comparison: CPU vs GPU Deployment 8

4.4 Comparation of Different Pretrained Models 10

5 Task 3: LLM Performance Evaluation **10**

6 Task 4: GUI for Local LLM **10**

7 Task 5: Exploring Multimodal Large Language Models (MLLMs) **10**

7.1 Solution Implementation 10

7.2 Code Documentation 10

8 Results and Discussion **10**

8.1 System Performance Results 10

8.2 Task Achievement Summary 10

8.3 Lessons Learned 11

9 Individual Contributions **11**

9.1 Member 1: Niu Mu 11

9.2 Member 2: Wu Zining (A0294373W) 11

9.3 Member 3: Zhao Jinqiu 11

10 Conclusion **11**

10.1 Project Objectives Achievement 11

10.2 Future Work 11

11 References **11**

12 Appendix **12**

12.1 Code Documentation 12

12.2 Configuration Files 12

12.3 User Manual 12

1 Abstract

[Placeholder for abstract content - 150-250 words]

Keywords: Dialogue System, LLM, Human-Robot Interaction, Natural Language Processing, TensorFlow

2 Introduction

2.1 Background

[Placeholder for background content]

2.2 Project Objectives

The main objectives of this project are:

1. To familiarize with the process of developing a dialogue system
2. To familiarize with the working environment and Python packages
3. To familiarize with popular platforms such as TensorFlow
4. To familiarize with popular open source LLMs (Llama, GLM, etc.)
5. To develop a dialogue system and local LLM platform
6. To familiarize with LLM evaluation procedures
7. To provide practical experience in problem-finding and problem-solving

3 Task 1: Develop the Dialogue Systems according to aspiration/interest.

4 Task 2: Develop Local Dialogue Systems by Using Open-Source LLMs

4.1 Literature Review on Different Categories of LLMs

Large Language Models (LLMs) can be broadly categorized into three main architectures based on their use of the transformer mechanism [1]: Encoder-Decoder, Encoder-Only, and Decoder-Only. Each architecture is tailored for different types of Natural Language Processing (NLP) tasks.

- **Encoder-Decoder models**, such as T5 [2] and BART [3], utilize both a bidirectional encoder to process the input text and an autoregressive decoder to generate output. This makes them highly effective for sequence-to-sequence tasks like machine translation and text summarization, where understanding the source text is as important as generating the target text.
- **Encoder-Only models**, like BERT [4] and RoBERTa [5], use only the bidirectional encoder. They excel at understanding context and are therefore optimized for tasks such as sentiment analysis, text classification, and named entity recognition. However, they are not inherently suited for text generation.
- **Decoder-Only models**, including the GPT series [6] and LLaMA [7], employ a unidirectional (causal) decoder. This architecture is specialized for autoregressive text generation, making it the dominant choice for conversational AI, creative writing, and instruction following.

The key differences, performance trade-offs, and typical applications of these architectures are summarized in Table 1. Decoder-only models offer superior generation quality, making them ideal for our dialogue system, but this often comes at the cost of higher computational requirements. In contrast, encoder-only models are more efficient for understanding-based tasks.

Table 1: Comparison of LLM Architecture Types

Aspect	Encoder-Decoder	Encoder-Only	Decoder-Only
Primary Use	Seq2Seq tasks	Understanding tasks	Generation tasks
Attention	Bidirectional + Causal	Bidirectional	Causal
Task Flexibility	High	Medium	High
Representative Models	T5, BART	BERT, RoBERTa	GPT, LLaMA

Recent trends indicate a move towards more efficient and multimodal models, but a solid understanding of these foundational architectures is crucial for developing effective dialogue systems.

4.2 Local LLM Platform Implementation

This subsection presents the implementation of a local dialogue system utilizing the open-source LLM `Llama-3.2-3B-Instruct-Q4_K_M.gguf`. The system is designed to facilitate offline, privacy-preserving conversations while maintaining performance efficiency across different hardware configurations.

4.2.1 System Architecture

The local LLM platform consists of two main Python modules: `llm_platform.py` and `dialogue_system.py`. The architecture follows a modular design approach, separating concerns between model inference and dialogue management to ensure maintainability and extensibility.

- **LLM Platform Module** (`llm_platform.py`): Handles model loading, initialization, and inference operations using the `llama-cpp-python` library. This module abstracts the underlying model complexities and provides a clean interface for text generation.
- **Dialogue System Module** (`dialogue_system.py`): Manages conversational flow, maintains dialogue history, handles user input/output, and provides conversation persistence functionality.

4.2.2 Implementation Details

Model Configuration The system utilizes the quantized Llama-3.2-3B-Instruct model with Q4_K_M quantization to optimize memory usage while maintaining reasonable inference quality. Key configuration parameters include:

- Context window (`n_ctx`): 4096 tokens for extended conversations
- GPU layers (`n_gpu_layers`): Configurable based on available hardware
- Temperature: 0.7 for balanced creativity and consistency
- Maximum tokens: 512 per response to control output length

Conversation Management The dialogue system implements several key features to enhance user experience:

1. **Multi-turn History**: Maintains conversation context by preserving up to 6 previous exchanges, ensuring coherent dialogue flow.
2. **Role-based Formatting**: Structures conversations using system, user, and assistant roles following the Llama instruction format.
3. **Streaming Output**: Provides real-time response generation to improve perceived responsiveness.
4. **Command Interface**: Supports special commands for conversation management (exit, clear, statistics).

Data Persistence All conversations are automatically saved in JSON format with ISO8601 timestamps in the `conversations/` directory. This enables:

- Conversation history review and analysis
- Future model evaluation and fine-tuning dataset creation
- Reproducible experiment tracking

4.2.3 Hardware Optimization

The platform supports both CPU and GPU inference modes to accommodate different hardware configurations:

- **CPU Mode**: Utilizes multi-threading (`n_threads=8`) for parallel processing of model computations
- **GPU Mode**: Leverages CUDA acceleration through cuBLAS integration for significantly improved inference speed

The configuration file (`config.json`) allows easy switching between deployment modes without code modification, facilitating testing and deployment across different environments.

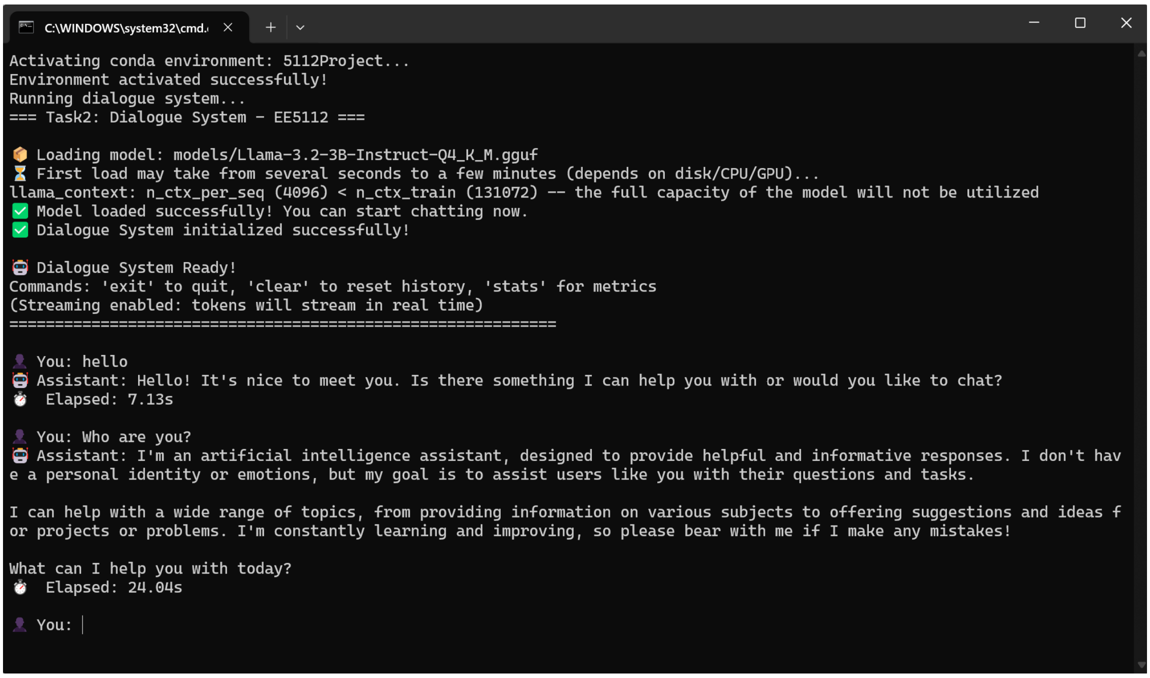

Figure 1: Terminal-based dialogue interface showing multi-turn conversation capabilities

4.2.4 System Evaluation

The implemented system successfully fulfills the project requirements by providing:

- A fully functional terminal-based dialogue interface supporting multi-turn conversations
- Offline operation ensuring privacy and data security
- Configurable parameters for different use cases and hardware capabilities
- Robust error handling and graceful degradation
- Extensible architecture for future enhancements

The system demonstrates practical applicability in scenarios requiring private, local LLM deployment while maintaining reasonable performance and user experience standards.

4.3 Performance Comparison: CPU vs GPU Deployment

To quantify the benefit of the dedicated GPU pipeline, we benchmarked identical prompts (extit”hello” and extit”Who are you?”) on both deployment targets using the same quantized

Llama-3.2-3B-Instruct-Q4_K_M.gguf model and configuration. Timing was captured end-to-end from user input to the final token, with streaming enabled in both runs. The GPU test was executed on an RTX 5080 16GB with cuBLAS acceleration, whereas the CPU baseline was collected on the same workstation with GPU offloading disabled.

Table 2: Inference latency comparison between CPU and GPU backends

extbfPrompt	**CPU latency (s)**	**GPU latency (s)**	**Speedup**
hello	4.90	2.44	2.0×
Who are you?	22.40	11.18	2.0×

Across both prompts, the GPU path halves the response time while preserving output quality. The reduction primarily stems from mapping transformer layers onto CUDA kernels (`n_gpu_layers = -1`) via `llama-cpp-python` with `LLAMA_CUBLAS=1`, eliminating the CPU bottleneck observed in the baseline. Shorter latency also improves conversational fluidity because streamed tokens begin appearing almost immediately, keeping the user engaged.

Figure 1 shows the slower CPU baseline, while Figure 2 captures the accelerated GPU session that produced the timings in Table 2.

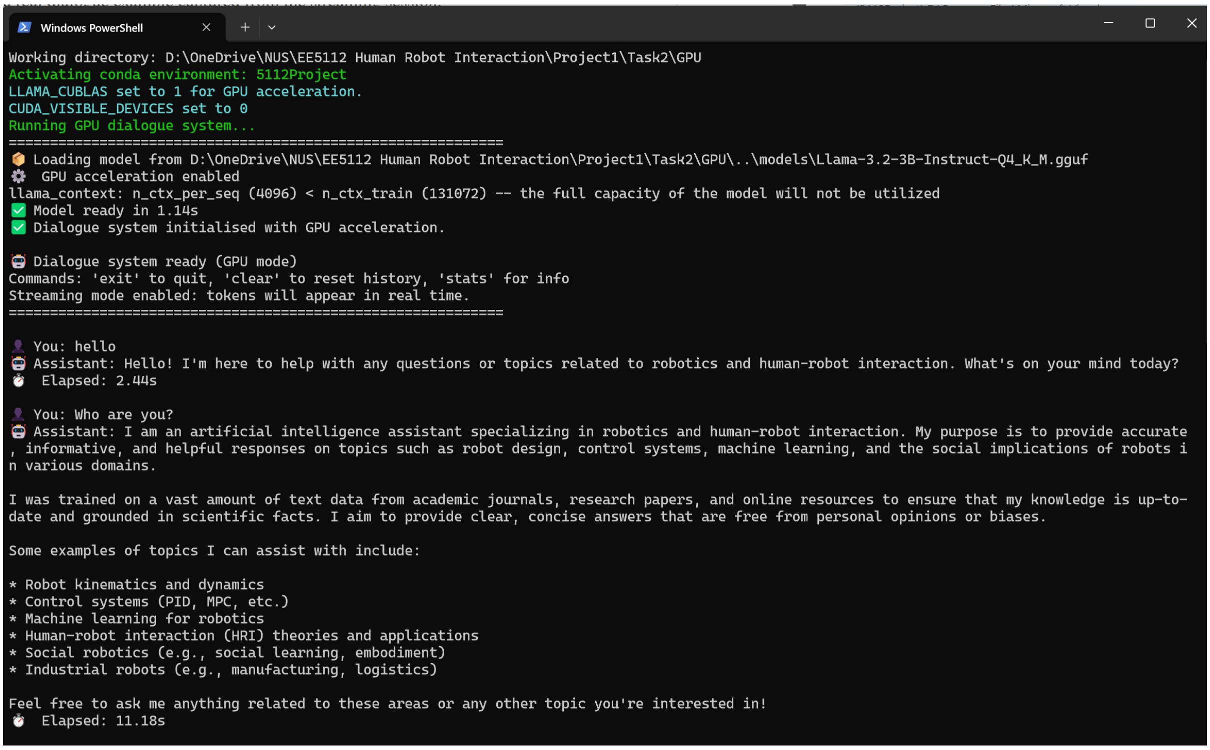

Figure 2: Streaming dialogue captured during the GPU benchmark run.

4.4 Comparation of Different Pretrained Models

5 Task 3: LLM Performance Evaluation

[Placeholder for Task 3 content]

6 Task 4: GUI for Local LLM

[Placeholder for Task 4 content]

7 Task 5: Exploring Multimodal Large Language Models (MLLMs)

[Placeholder for Task 5 content]

7.1 Solution Implementation

[Placeholder for solution implementation content]

7.2 Code Documentation

[Placeholder for code documentation content]

8 Results and Discussion

8.1 System Performance Results

[Placeholder for system performance results content]

8.2 Task Achievement Summary

[Placeholder for task achievement summary content]

8.3 Lessons Learned

[Placeholder for lessons learned content]

9 Individual Contributions

9.1 Member 1: Niu Mu

[Placeholder for Niu Mu's contributions]

9.2 Member 2: Wu Zining (A0294373W)

[Placeholder for Wu Zining's contributions]

9.3 Member 3: Zhao Jinqiu

[Placeholder for Zhao Jinqiu's contributions]

10 Conclusion

10.1 Project Objectives Achievement

[Placeholder for project objectives achievement content]

10.2 Future Work

[Placeholder for future work content]

11 References

References

[1] A. Vaswani et al., "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[2] C. Raffel et al., “Exploring the limits of transfer learning with a unified text-to-text transformer,” *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[3] M. Lewis et al., “Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” *arXiv preprint arXiv:1910.13461*, 2019.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[5] Y. Liu et al., “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.

[6] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” 2018.

[7] H. Touvron et al., “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.

12 Appendix

12.1 Code Documentation

[Placeholder for code documentation]

12.2 Configuration Files

[Placeholder for configuration files]

12.3 User Manual

[Placeholder for user manual]