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 25, 2025

Contents

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

2.3 Project Scope

[Placeholder for project scope content]

3 Task 1: Development Environment Setup

3.1 Python Environment Configuration

[Placeholder for Python environment setup content]

3.2 TensorFlow Platform Familiarization

[Placeholder for TensorFlow familiarization content]

3.3 Development Tools and Libraries

[Placeholder for development tools content]

4 Task 2: LLM Platform Development

4.1 Open Source LLM Exploration

4.1.1 Literature Review on LLM Categories

Large Language Models (LLMs) can be categorized into three main architectural paradigms based on their transformer configurations: encoder-decoder, encoder-only, and decoder-only models. Each architecture has distinct characteristics, strengths, and applications in natural language processing tasks.

Encoder-Decoder Models Encoder-decoder models employ a dual-transformer architecture where the encoder processes input sequences to generate contextual representations, while the decoder generates output sequences based on these representations. This architecture is particularly effective for sequence-to-sequence tasks such as machine translation, text summarization, and question answering.

Key Characteristics:

- Bidirectional attention in the encoder captures context from both directions
- Unidirectional attention in the decoder enables autoregressive generation
- Explicit separation between understanding (encoding) and generation (decoding) phases

Representative Models:

- **T5 (Text-to-Text Transfer Transformer)**: Treats all NLP tasks as text-to-text problems, achieving state-of-the-art performance across diverse benchmarks
- **BART (Bidirectional and Auto-Regressive Transformers)**: Combines bidirectional encoder with autoregressive decoder, excelling in text generation and denoising tasks
- **mT5**: Multilingual extension of T5 supporting over 100 languages

Encoder-Only Models Encoder-only models utilize only the encoder component of the transformer architecture, employing bidirectional attention to process input sequences. These models excel at understanding and representation learning tasks rather than text generation.

Key Characteristics:

- Bidirectional attention mechanism captures full context
- Optimized for understanding tasks rather than generation
- Require task-specific heads for downstream applications
- Typically used for classification, named entity recognition, and feature extraction

Representative Models:

- **BERT (Bidirectional Encoder Representations from Transformers)**: Pioneer in bidirectional language modeling, achieving breakthrough performance in NLU tasks
- **RoBERTa**: Optimized version of BERT with improved training procedures and longer training duration
- **DeBERTa**: Enhanced BERT with disentangled attention mechanism and enhanced mask decoder
- **ELECTRA**: More efficient pre-training using replaced token detection instead of masked language modeling

Decoder-Only Models Decoder-only models rely exclusively on the decoder component with causal (unidirectional) attention, making them highly effective for autoregressive text generation tasks. This architecture has become the dominant paradigm for modern conversational AI systems.

Key Characteristics:

- Unidirectional attention prevents information leakage during training
- Optimized for text generation and completion tasks
- Can be fine-tuned for various downstream tasks through instruction following
- Generally require larger model sizes to achieve competitive performance

Representative Models:

- **GPT (Generative Pre-trained Transformer) Series**: GPT-1, GPT-2, GPT-3, and GPT-4 represent the evolution of decoder-only models with increasing scale and capabilities
- **LLaMA (Large Language Model Meta AI)**: Efficient decoder-only model achieving competitive performance with smaller parameter counts
- **GLM (General Language Model)**: Chinese-developed model combining autoregressive and autoencoding approaches
- **PaLM (Pathways Language Model)**: Google's large-scale decoder-only model with 540B parameters

4.1.2 Comparative Analysis

Table 1: Comparison of LLM Architecture Types

Aspect	**Encoder-Decoder**	**Encoder-Only**	**Decoder-Only**
Primary Use	Seq2Seq tasks	Understanding tasks	Generation tasks
Attention Mechanism	Bidirectional + Causal	Bidirectional	Causal
Training Efficiency	Medium	High	Low (for large models)
Inference Speed	Medium	Fast	Slow (for large models)
Task Flexibility	High	Medium	High
Parameter Efficiency	Medium	High	Low
Representative Models	T5, BART	BERT, RoBERTa	GPT, LLaMA

Performance Trade-offs:

- **Encoder-Decoder Models**: Offer balanced performance for both understanding and generation tasks, but require more computational resources due to dual architecture
- **Encoder-Only Models**: Excel at understanding tasks with high efficiency, but limited generation capabilities
- **Decoder-Only Models**: Superior generation quality and conversational abilities, but require significant computational resources for training and inference

Application Scenarios:

- **Encoder-Decoder**: Machine translation, text summarization, question answering systems
- **Encoder-Only**: Sentiment analysis, named entity recognition, text classification, feature extraction

- **Decoder-Only**: Conversational AI, creative writing, code generation, instruction following

4.1.3 Recent Trends and Future Directions

Recent developments in LLM architectures show several emerging trends:

- **Scale Integration**: Modern models increasingly combine multiple architectural paradigms (e.g., encoder-decoder with decoder-only components)
- **Efficiency Optimization**: Focus on reducing computational requirements while maintaining performance through techniques like knowledge distillation and model compression
- **Multimodal Integration**: Extension of decoder-only models to handle multiple modalities (text, vision, audio)
- **Specialized Architectures**: Development of task-specific architectures optimized for particular domains or applications

This comprehensive understanding of different LLM architectures provides the foundation for selecting appropriate models for specific applications in dialogue systems and local LLM platforms.

4.2 Local LLM Platform Implementation

5 Task 3: Dialogue System Development

5.1 Dialogue System Architecture

[Placeholder for dialogue system architecture content]

5.2 Natural Language Processing Components

[Placeholder for NLP components content]

5.3 Multi-modal Communication

[Placeholder for multi-modal communication content]

5.4 Dialogue Management

[Placeholder for dialogue management content]

6 Task 4: System Integration and Testing

6.1 Component Integration

[Placeholder for component integration content]

6.2 System Testing

[Placeholder for system testing content]

6.3 Performance Optimization

[Placeholder for performance optimization content]

7 Task 5: LLM Evaluation

7.1 Evaluation Metrics

[Placeholder for evaluation metrics content]

7.2 Evaluation Framework Implementation

[Placeholder for evaluation framework implementation content]

7.3 Performance Analysis

[Placeholder for performance analysis content]

7.4 Solution Implementation

[Placeholder for solution implementation content]

7.5 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

[Placeholder for references - Use proper citation format]

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]