Understanding Fine-Tuning's Effect on Large-Language Model (LLM)-powered Travel Travel Planning Robot

ZHONGYANG HU, Xi'an Jiaotong-Liverpool University, China
RUNZE YANG, Xi'an Jiaotong-Liverpool University, China
RONG HUANG, Xi'an Jiaotong-Liverpool University, China
QINCHUAN ZHU, Xi'an Jiaotong-Liverpool University, China
JINHONG LI, Xi'an Jiaotong-Liverpool University, China

In this paper, we introduce a method for building a Large Language Model-based traveling agent designed to provide valuable tourist recommendations. This agent is intended for users who wish to plan their trips to dream destinations efficiently and cost-effectively.// Achieve this, we developed a 3D robot system based on the Unity 3D environment, integrating the plan generation and route reasoning capabilities of large language models (LLMs). Users can interact with this robot through a peripheral microphone or keyboard typing.//The robot system consists of two main components. The first component is developed using Unity, leveraging its powerful package system, which includes tools such as the OpenAI-API package, Speech-to-Text, and Text-to-Speech. We created a third-person game scenario where users control a Player avatar, and interact with an NPC avatar that incorporates the travel planning robot's primary functions and the abilities of the LLMs. The second component is a Django backend server, developed in Python. This server hosts an open-source large language model, Google's Gemma, in two different sizes (2B and 7B versions). We fine-tuned these models using two different approaches: LoRA (Low-Rank Adaptation) and Q-LoRA (Quantized Low-Rank Adaptation), utilizing a training dataset containing travel planning conversations in a "Question-Answer" format. Following fine-tuning, user prompts are directed to the fine-tuned LLMs, their responses are generated, and transmitted back to Unity's NPC.//By integrating these components, the fully functional robot allows users to ask travel-related questions to the NPC avatar via microphone or typing. The GPT and the four fine-tuned models provide route planning responses displayed in the NPC's chat box. These responses are structured according to a prompt template, ensuring sequential, indexed, and time-stamped activity planning.//The quality of the LLMs' responses was manually evaluated using a Human Evaluation method. Participants assessed the responses based on subjective feelings and previous travel experiences, labeling each (prompt, response) pair with tags (low, medium, high) to indicate response quality. For performance comparison, these string labels were converted to integer scores low:1, medium:2, high:3, and the total scores of 20 responses were summed to obtain a model performance score.//As a result, the evaluation revealed that the fine-tuned models significantly outperformed the baseline GPT model in generating coherent and practical travel plans. The models fine-tuned with Q-LoRA demonstrated the highest accuracy and user satisfaction, particularly the 7B version, which achieved an average performance score of 2.8 out of 3. This indicates that quantized fine-tuning can enhance model performance while maintaining cost-efficiency.//Our findings suggest that integrating LLMs with advanced fine-tuning techniques into interactive 3D environments can significantly enhance the user experience in travel planning applications, providing detailed and user-friendly travel itineraries.

Authors' addresses: Zhongyang Hu, Zhongyang.Hu23@student.xjtlu.edu.cn, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, China; Runze Yang, Runze.Yang23@student.xjtlu.edu.cn, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, China; Rong Huang, Rong.Huang23@student.xjtlu.edu.cn, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, China; Qinchuan Zhu, Qinchuan.Zhu19@student.xjtlu.edu.cn, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, China; Jinhong Li, Jinhong.Li23@student.xjtlu.edu.cn, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, China.

0004-5411/2024/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Additional Key Words and Phrases: Large Language Models, Fine-Tune, Lora, Gemma, PEFT, Robot, Route-Planning

ACM Reference Format:
Zhongyang Hu, Runze Yang, Rong Huang, Qinchuan Zhu, and Jinhong Li. 2024. Understanding Fine-Tuning's Effect on Large-Language Model (LLM)-powered Travel Travel Planning Robot. *J. ACM* 1, 1 (May 2024), 42 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The entire project is based on the Unity3D environment, where the robot operates and multiple Python scripts of LLM services are integrated. This setup allows for a seamless combination of advanced AI capabilities and an interactive virtual environment, providing users with a highly immersive and responsive travel planning experience.

1.1 Research background

With the improvement in quality of life and the continuous pursuit of travel experiences, efficient travel planning has become crucial. Traditional travel planning is often inefficient, requiring extensive time and effort to search and filter information. This inefficiency highlights the need for an intelligent, personalized online travel planning system. Despite advances in digital technologies and the internet, there is still a gap in integrating real-time data and personal preferences into a cohesive, user-friendly system.

Recent advancements in Natural Language Processing (NLP) and Large Language Models (LLMs) have created opportunities for intelligent travel planning systems. LLMs possess powerful text understanding and generation capabilities, allowing them to analyze travel needs described in natural language and generate personalized travel suggestions. These advancements enable systems to understand and respond to user queries in a human-like manner, significantly enhancing user experience.

1.2 Research motivation

The motivation behind this research is to leverage NLP and LLM advancements to address inefficiencies in traditional travel planning. Developing an LLM-based online travel planning bot involves integrating various technologies, such as Geographic Information Systems (GIS) for spatial analysis and real-time data analytics for dynamic route optimization [53]. Combining these technologies with LLMs can provide a comprehensive, adaptive travel planning solution.

Despite the potential of LLMs in tourism planning, there is limited research on online tourism route planning using large-scale language models. This research aims to fill this gap and provide a robust framework for intelligent travel planning systems. By integrating LLMs, GIS, and real-time data analysis, we aim to develop a robot that automatically generates personalized travel routes, understands user preferences, and interacts with users through natural language for an efficient travel planning experience [1].

1.3 Research Questions

(1) **Do different fine-tuning methods (LoRA, Q-LoRA) influence the travel plan generation of the large language model (Gemma-2B, Gemma-7B)?**
This question aims to investigate how different fine-tuning techniques affect the performance of large language models in generating travel plans. By comparing LoRA and Q-LoRA, we can understand their respective impacts on the models' ability to generate accurate and useful travel recommendations.

(2) **Does model size affect the travel plan generation quality of the LoRA fine-tuned models (Gemma-2B, Gemma-7B)?**
This question explores whether the size of the model (2B vs. 7B parameters) has a significant effect on the quality of travel plans produced by the LoRA fine-tuned models. Larger models typically have more capacity to learn and generalize, which could potentially lead to better performance in generating travel itineraries.
(3) **Does model size affect the travel plan generation quality of the Q-LoRA fine-tuned models (Gemma-2B, Gemma-7B)?**
Similar to the previous question, this one examines the influence of model size on the quality of travel plans produced by Q-LoRA fine-tuned models. Understanding this relationship helps in selecting the appropriate model size for practical applications.
(4) **How can LLM accurately capture and analyze core information such as travel intent, attraction preferences, and budget range in user input?**
This question addresses the ability of large language models to understand and process complex user inputs that include specific travel intentions, preferences for attractions, and budget constraints. It is crucial for providing personalized and relevant travel recommendations.
(5) **How can LLM effectively collect and analyze real-time weather, traffic, attraction status, and other data, and optimize tourism route planning accordingly?**
This question focuses on the integration of real-time data into the travel planning process. By leveraging real-time information, the model can provide more accurate and up-to-date travel plans, enhancing the overall user experience.
(6) **How can a scientific and effective evaluation system be established to measure the quality of tourism route planning generated by LLM and collect user feedback for continuous optimization?**
(7) **How to ensure the stability, responsiveness, and scalability of LLM tourism route planning service under high concurrency scenarios?**
Ensuring that the system can handle a large number of user requests simultaneously without degradation in performance is critical for practical deployment. This question explores strategies to maintain stability, responsiveness, and scalability in high-demand situations.

1.4 Research Hypothesis

Based on the research questions, the project proposes the following hypotheses:

(1) **Fine-Tuning Methods Hypothesis (H1):**
- H1: Different fine-tuning methods (LoRA, Q-LoRA) will have a significant influence on the travel plan generation quality of the large language models (Gemma-2B, Gemma-7B).

(2) **Model Size Impact on LoRA Fine-Tuning Hypothesis (H2):**
- H2: The size of the model (2B vs. 7B) will significantly affect the travel plan generation quality of the LoRA fine-tuned models (Gemma-2B, Gemma-7B).

(3) **Model Size Impact on Q-LoRA Fine-Tuning Hypothesis (H3):**
- H3: The size of the model (2B vs. 7B) will significantly affect the travel plan generation quality of the Q-LoRA fine-tuned models (Gemma-2B, Gemma-7B).

(4) **User Intent and Preference Analysis Hypothesis (H4):**
- H4: The LLM can accurately capture and analyze core information such as travel intent, attraction preferences, and budget range from user input.

(5) **Real-Time Data Integration Hypothesis (H5):**
- H5: The LLM can effectively collect and analyze real-time weather, traffic, attraction status, and other data, optimizing tourism route planning accordingly.

(6) **Evaluation System Hypothesis (H6):**
- H6: A scientific and effective evaluation system can be established to measure the quality of tourism route planning generated by LLM and collect user feedback for continuous optimization.

(7) **System Performance Hypothesis (H7):**
- H7: The LLM-based tourism route planning service can maintain stability, responsiveness, and scalability under high concurrency scenarios.

1.5 System Design

The entire project is based on the Unity3D environment, where the robot operates and multiple Python scripts of LLM services are integrated. This setup allows for a seamless combination of advanced AI capabilities and an interactive virtual environment, providing users with a highly immersive and responsive travel planning experience.

1.5.1 Virtual Environment Construction. We utilized the Unity engine to build a highly interactive virtual environment. Unity's powerful 3D modeling and scene management functions enabled us to create realistic virtual tourism scenes. In this environment, users can control their virtual characters, navigate through the paths in the scene, and walk towards the NPC travel guide at a predetermined location [10]. This setup not only provides an engaging experience but also simulates real-world travel planning scenarios, making users feel as though they are exploring an actual destination. The virtual environment includes various elements such as landmarks, pathways, and interactive objects that enhance the overall user experience.

1.5.2 User Interface. The user interface (UI) design is simple and intuitive, featuring two main input methods: text input and voice input.

- Text Input: Users can enter questions through the keyboard on the UI interface and click the send button to submit their queries. This method is straightforward and allows users to type in specific questions or commands that the system can process.
- Voice Input: After clicking the recording icon on the interface, users can record their questions using a microphone. Once the recording is complete, the system automatically converts the speech into text and submits it. This method provides a more natural and convenient way for users to interact with the system, especially when typing is not feasible.

This dual-mode input system ensures accessibility and convenience, catering to users with different preferences and making the interaction more user-friendly.

1.5.3 Dialogue Triggering Mechanism. To achieve a natural interactive experience, we designed a dialogue starting mechanism based on trigger areas. When the user's virtual character walks into the trigger area of the NPC travel guide, the dialogue interface automatically launches. These trigger areas are strategically designed considering the user's range of movement and ease of interaction, ensuring users can easily enter and start a conversation. This mechanism mimics real-life scenarios where approaching a tour guide triggers a conversation, enhancing the immersion and realism of the virtual environment [27].

1.5.4 LLM-based Technologies. Large Language Models (LLMs) represent a significant advancement in the field of artificial intelligence and natural language processing. These models, such as OpenAI's GPT series and Google's Gemma, have demonstrated remarkable abilities in understanding and generating human language [30]. By leveraging these capabilities, our project aims to utilize LLMs to enhance travel planning services. LLMs can process vast amounts of text data,

understand context, and generate coherent and contextually relevant responses, making them ideal for applications in personalized travel planning.

1.5.5 AI Model Integration. The system uses advanced AI models to process users' questions and generate answers. The specific process is as follows:

(1) **Question Submission:** Users submit their questions through text input or voice conversion into text. This step involves capturing the user's input and preparing it for processing by the AI model.
(2) **AI Processing:** The submitted questions undergo natural language processing by the AI model, which understands the user's intentions and generates corresponding answers. This AI model is based on the latest natural language processing technology and can accurately understand and respond to various travel-related questions. The integration of large language models (LLMs) enhances the system's ability to provide detailed and contextually relevant travel information, leveraging the vast amounts of data these models are trained on.

1.5.6 Low-Resource Limitation. One of the challenges in deploying LLMs for specific tasks like travel planning is the limitation of low-resource settings. Fine-tuning large models requires substantial computational resources and high-quality datasets, which may not always be available. To address this, we explore efficient fine-tuning methods such as LoRA (Low-Rank Adaptation) and Q-LoRA (Quantized Low-Rank Adaptation) [43]. These methods aim to optimize the performance of LLMs with limited resources by focusing on adapting a smaller set of parameters or quantizing the model to reduce its computational demands while maintaining performance.

1.5.7 Speech Synthesis. To improve the immersion of the interaction, we integrated AI speech synthesis technology into the system. After the AI model generates a text answer, the system calls the speech synthesis engine to convert the text into natural and smooth speech. This allows users to hear the NPC travel guide reply to their questions in a human-like voice, enhancing the realism and interactivity of the user experience [45]. The use of speech synthesis makes the interaction more engaging and accessible, especially for users who prefer auditory information.

1.5.8 Dataset. A crucial component of our research is the dataset used for fine-tuning the LLMs. We curated a dataset specifically tailored for travel planning, which includes a diverse range of travel-related conversations, questions, and answers. This dataset encompasses various travel scenarios, such as itinerary planning, destination recommendations, and budget considerations. By training our models on this specialized dataset, we aim to enhance their ability to generate accurate and personalized travel suggestions that cater to the unique needs and preferences of users.

1.5.9 System Architecture. The virtual environment is constructed using Unity, and users interact with the system through the interface. The system includes an AI model and a speech synthesis engine. The data flow involves three main stages: user input, AI processing, and voice output, ensuring efficient interaction processes and real-time response capabilities. This architecture is designed to handle complex interactions and provide quick and accurate responses to user queries, making the system robust and scalable.

1.5.10 Experiment Summary. Our experimental setup involves fine-tuning the LLMs using the curated travel dataset and evaluating their performance in generating travel plans. We employ different fine-tuning methods, including LoRA and Q-LoRA, and compare their effectiveness. The evaluation metrics include accuracy in capturing user intent, quality of travel recommendations, and overall user satisfaction. We also consider factors such as model size and quantization to

understand their impact on performance. Through these experiments, we seek to identify the optimal configuration for deploying LLMs in travel planning applications.

1.5.11 Anticipated Results. Based on our preliminary analysis, we anticipate that the LoRA fine-tuned Gemma model will significantly outperform the pre-trained version of Gemma (baseline model). It is also expected to perform better than the Q-LoRA fine-tuned Gemma. The quantization process in Q-LoRA, which reduces the precision of model parameters, might limit the model's improvement after fine-tuning. However, the performance of Q-LoRA is still expected to be higher than the pre-trained Gemma, as it fundamentally learns the domain knowledge of travel route planning. We do not include GPT-3.5-turbo in this comparison due to its large parameter size (175B), which would make the competition unfair. Instead, GPT-3.5-turbo will be used to test the feasibility of our system's functions.

2 BACKGROUND AND RELATED WORK

The exploration of fine-tuning methodologies for large language models (LLMs) has advanced significantly in recent years, especially in enhancing model performance for vertical domain tasks. Applying these techniques to areas such as travel planning via a robotic environment is relatively new. This novelty necessitates a thorough review of existing literature on LLM's fine-tuning technologies, deep learning methods, and their implications in AI-driven robotics.

2.1 Current Progress on Large Language Models and Fine-Tuning

In natural language processing, the development of language models has been a key area of research. Early approaches used statistical methods, such as n-gram models, enhanced by smoothing techniques to better estimate the probability of rare events. The advent of neural networks brought significant advancements, with early neural probabilisti c language models showing improved performance [3].

The introduction of transformer architectures marked a pivotal moment in the evolution of language models [48]. This architecture has since been the foundation for numerous large-scale models, such as GPT-3, which boasts 175 billion parameters and has set new benchmarks in various natural language tasks [6]. The scaling laws for neural language models, which describe the relationship between model size, dataset size, and performance, have been extensively studied, providing insights into the optimal scaling of these models [19].

Recent efforts have focused on open-sourcing large language models to promote transparency and collaboration within the research community. Notable examples include Qwen, Llama, and Groke, which provide competitive alternatives to proprietary models. These models have been evaluated on standard benchmarks to assess their performance and identify potential biases and toxic content, which remain critical concerns in the deployment of AI systems.

2.2 Effectiveness of Pre-Training in Language Models

Pre-training language models have seen substantial success in numerous natural language processing (NLP) tasks. Early works demonstrated the potential of leveraging unlabeled data to pre-train language models, which could then be fine-tuned for specific tasks. This approach laid the foundation for many significant advancements in NLP [35].The effectiveness of pre-training lies in its ability to capture extensive linguistic patterns from vast amounts of text, which models trained from scratch could not achieve due to limited labeled data [12]. The introduction of contextual embeddings, as seen in models like ELMo and GPT, further enhanced the ability of pre-trained models to understand and generate human-like text [34].

In the above models, ELMo (Embeddings from Language Models) leverages bi-directional LSTM

networks to create context-sensitive embeddings, encapsulating both left and right context during the training process. This method outperformed traditional word embeddings such as Word2Vec and GloVe, which were limited to producing context-free representations. In contrast, ELMo's contextualization allows the same word to have different representations depending on its usage, thereby capturing richer semantic and syntactic properties. This not only aids in tasks requiring nuanced understanding, such as named entity recognition and sentiment analysis, but also substantially enhances performance in question answering systems and syntactic parsing tasks.
Similarly, the GPT (Generative Pre-training Transformer) model, proposed by Radford et al. [35], incorporated the Transformer architecture to generate contextual embeddings, further pushing the boundaries of natural language understanding. Unlike ELMo, GPT utilizes the Transformer's attention mechanism to self-attend across different tokens in a sequence, thereby enabling the model to capture long-range dependencies more effectively. This architecture facilitates improved coherence and relevance in generated text, proving advantageous in various NLP tasks including language translation, summarization, and conversational agents [56].

2.3 Types of Tasks in NLP

Language models have been applied to various tasks at both the sentence and word levels. Sentence-level tasks include natural language inference, where models are required to determine the relationships between sentence pairs [6], and paraphrase identification, which involves recognizing different expressions of the same meaning. At the word level, named entity recognition and question answering are key tasks. Named entity recognition focuses on identifying and classifying entities within a text [47], while question answering involves retrieving answers from a corpus based on input questions [52]. Additionally, models like BERT have shown considerable success in tasks such as sentiment analysis and part-of-speech tagging [12]. Recently, new applications in dialogue systems, machine translation, and even text summarization have demonstrated the diverse capabilities of these pre-trained models [58].

2.4 Strategies for Applying Pre-Trained Language Representations

There are primarily two approaches for utilizing pre-trained language representations: feature-based methods and fine-tuning methods. Feature-based methods, such as ELMo, integrate pre-trained representations as additional features in task-specific architectures [26]. Fine-tuning methods, exemplified by the OpenAI GPT, involve minimal task-specific architectural adjustments and train all pre-trained parameters on downstream tasks [3]. [41]exploration into sparse transformers and large-scale implementations like GPT-3, which utilizes hundreds of billions of parameters, signify the immense potential of fine-tuning approach in solving increasingly complex tasks [40].

2.5 Limitations of Current Technologies

One prominent limitation of early models like OpenAI GPT is the unidirectional nature of their architecture, which caused constraints in the representation capabilities of pre-trained models, especially when employing fine-tuning strategies. For instance, OpenAI GPT's left-to-right architecture means each token can only attend to foregoing tokens, which is suboptimal for tasks requiring bidirectional context, such as question answering [25]. Furthermore, the substantial computational resources required for training and fine-tuning, as well as concerns regarding the environmental impact of such large-scale models, remain critical challenges for researchers [39]. Recently, mitigation strategies involving model compression and distillation have been explored to address efficiency issues [41].

2.6 GPT Series and Transformer Architecture

2.6.1 GPT Series. The GPT (Generative Pre-training Transformer) model, proposed by Radford et al., incorporated the Transformer architecture to generate contextual embeddings, further pushing the boundaries of natural language understanding [35]. By leveraging a large-scale unsupervised pre-training phase followed by fine-tuning on specific tasks, GPT demonstrated the profound potential of transfer learning in NLP. The model's architecture enabled it to capture complex language patterns and contextual information, leading to substantial enhancements in tasks such as text generation, machine translation, and question answering [55].
Following the success of GPT, subsequent models such as GPT-2 and GPT-3 have iteratively improved on this foundation by scaling up the model size and the amount of training data. GPT-2, introduced by Radford et al. in 2019, significantly increased the number of parameters and demonstrated even more impressive language generation capabilities, including the ability to compose coherent passages of text without task-specific adjustments [8]. GPT-3 extended this further, utilizing 175 billion parameters, which enabled the model to perform a variety of tasks with minimal to no fine-tuning, showcasing the versatility and power of large-scale language models [7].

2.6.2 Transformer Architecture. Transformers were introduced by Vaswani et al. in 2017. They mainly feature a mechanism based on attention and do not use traditional recurrent neural networks (like RNNs or LSTMs). The core components of the Transformer architecture include the following:
Encoder-Decoder Architecture: Transformer model was initially an encoder-decoder architecture used for sequence-to-sequence tasks, such as translation. The encoder encodes the input sequence into a hidden representation, while the decoder generates new output tokens based on this hidden representation and previously generated tokens [48].
The self-attention mechanism enables the model to capture long-range dependencies between different positions, which is especially useful for processing long sequences. Its calculation formula is as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

Where Q is the query matrix, K is the key matrix, V is the value matrix, and d_k is the dimension of the keys [48].
Multi-Head Attention Mechanism Multiple attention heads allow the model to learn different representations in different subspaces:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2}$$

Each head head_i is computed independently using the above self-attention mechanism [15], where it independently encodes various aspects of the input data, facilitating a richer understanding. This modular approach allows for the model to capture a broader spectrum of dependencies and contextual nuances.
Feedforward Neural Network: Each Transformer layer includes a feedforward neural network, which further processes the features extracted by the attention mechanism:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{3}$$

This feedforward neural network provides the model with more non-linearity and complexity [7].
(1) Input: Generally, the input x is the result processed by the self-attention mechanism.
(2) First Affine Transformation: The input x is multiplied by the weight matrix W_1, and then the

bias b_1 is added. This operation can be represented as:

$$xW_1 + b_1$$

Here, W_1 is a weight matrix and b_1 is a bias vector. Through this affine transformation, the input is mapped to a higher-dimensional space [48]

(3) ReLU Activation Function: Next, the output from the previous step is passed through a Rectified Linear Unit (ReLU) activation function:

$$\max(0, xW_1 + b_1)$$

The ReLU activation function turns all negative values to 0 and keeps positive values unchanged. This operation allows the model to capture the nonlinear features of the data [16].

(4) Second Affine Transformation: Then, the result after the ReLU activation function is multiplied by another weight matrix W_2 and added with another bias b_2:

$$(\max(0, xW_1 + b_1))W_2 + b_2$$

Here, W_2 is another weight matrix and b_2 is another bias vector. This operation maps the data back to the original dimension, which is the same as the input x [11].

The complete feedforward neural network (FFN) operation can be summarized as:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

2.6.3 Google's Gemma Model. Gemma represents a series of lightweight, open-source language models introduced by Google following the release of Gemini. These models leverage state-of-the-art research methodologies to sustain high performance while enhancing training and inference efficiency [20].In line with most contemporary language models, Gemma's foundation is the Transformer architecture [18].Moreover, Gemma integrates the Mix-of-Experts (MoE) architecture, significantly boosting parameter efficiency. The central concept of MoE involves multiple "experts," each functioning as an independent neural network unit. Through the Gating Network, Gemma dynamically selects the most appropriate expert for processing each input datum [41].

2.7 Mainstream Fine-Tuning Technologies

Recent progress in the development and optimization of large language models has highlighted the importance of fine-tuning in improving model accuracy and specificity for targeted applications. Techniques such as Low-Rank Adaptation (LoRA) and Quantized Low-Rank Adaptation (Q-LoRA) have emerged as prominent methods for efficient fine-tuning. LoRA, for instance, introduces low-rank matrices to adapt pre-trained models with minimal computational overhead, thereby enhancing their adaptability to specific tasks without extensive retraining [17]. On the other hand, Q-LoRA extends this approach by incorporating quantization, which reduces the model size and inference time while maintaining performance [28].

Studies have shown that LoRA can significantly improve the performance of LLMs in generating contextually relevant and accurate translations by effectively capturing domain-specific nuances [21]. Similarly, Q-LoRA has been validated in scenarios requiring real-time processing, where the balance between model efficiency and output quality is critical [19].

In the realm of Human-Robot Interaction, the integration of LLMs has been explored to facilitate more intuitive and effective communication between humans and robots. Research has indicated that fine-tuning LLMs for specific interaction contexts can enhance the robot's ability to understand and respond to human queries and commands, thereby improving the overall interaction experience [47]. This is particularly relevant in travel planning scenarios, where the robot must generate detailed and accurate itineraries based on user preferences and constraints.

2.8 Works on Robotics Enhanced by LLMs

The landscape of Human-Robot Interaction (HRI) and Artificial Intelligence (AI) has been significantly advanced through the application of large language models (LLMs) and various fine-tuning techniques. Implementation methodologies such as Low-Rank Adaptation (LoRA) push the boundaries of model adaptation to specific tasks while maintaining efficiency [51]. This low-rank adaptation technique proves crucial in refining the performance of large language models, making them not only effective but also resource-efficient.

Comprehensive surveys of large language models, such as the one by Zhao et al [60]. (2023) [59], provide a broad overview of the current state, highlighting advancements and ongoing challenges. These surveys emphasize the growing capabilities of LLMs in diverse applications, including robotics and HRI scenarios.

Particular applications in robotics, such as the work by Kannan et al. (2023) [19], demonstrate the integration of large language models in multi-agent systems for sophisticated task planning. Here, LLMs enable enhanced coordination and communication between agents, underscoring the potential for intelligent robotic systems.

Integrating deep reinforcement learning (DRL) with LLMs for interactive robotic navigation, as explored by Wang et al. (2024) [50], illustrates another significant direction. This human-in-the-loop approach leverages the strengths of LLMs combined with DRL to improve robot navigation in social contexts, aligning well with tasks like travel planning where dynamic environments and human preferences play a major role.

Ethnographic studies, such as the one by Chun and Knight (2020) [47], shed light on the impact of anthropomorphism in robotics, providing context on user perceptions and interactions with robots. These qualitative insights are vital as they inform the design and fine-tuning of LLMs aimed at human-centered applications, ensuring the models behave in ways that are intuitive and acceptable to users.

Furthermore, the exploration of Theory of Mind (ToM) [47] in LLMs by Verma et al. (2024) [49] challenges and examines the cognitive capabilities of these models. Understanding how LLMs simulate or understand human mental states can enhance their application in HRI, making interactions more natural and effective.

2.9 Ethical Considerations in AI and LLMs

As the adoption of large language models (LLMs) becomes more widespread, ethical considerations are increasingly critical in ensuring responsible deployment. Issues such as bias, fairness, transparency, and accountability have garnered significant attention from researchers and policymakers. Bias in AI models can manifest in various ways, from the reinforcement of harmful stereotypes to the unequal treatment of different demographic groups. Studies have shown that LLMs can inadvertently learn and perpetuate biases present in the training data [10]. Addressing these biases involves implementing robust debiasing techniques, improving data diversity, and continuously monitoring model outputs for biased behavior [58].

Fairness in AI encompasses the equitable treatment of all individuals, ensuring that AI systems do not disproportionately benefit or harm any particular group. Researchers are exploring various methods to enhance fairness, such as fairness constraints during model training and post-processing techniques to adjust biased outputs [15].

Transparency and interpretability of AI models are crucial for building trust and understanding among users. Techniques such as explainable AI (XAI) aim to make AI decision-making processes more transparent, allowing users to understand how and why certain decisions are made (Doshi-Velez) [25]. This transparency is particularly important in high-stakes applications, such as

healthcare and legal systems, where understanding the rationale behind AI decisions can significantly impact outcomes.

Accountability in AI involves establishing clear guidelines and responsibilities for AI development and deployment. This includes creating ethical guidelines, implementing regulatory frameworks, and ensuring that AI systems are designed with safety and ethical considerations in mind [55]. Developers and organizations must be accountable for the consequences of their AI systems, including potential misuse and unintended impacts.

In the context of LLMs and travel planning, ethical considerations also extend to data privacy and security. Ensuring that user data is handled securely and confidentially is paramount. This includes implementing robust data encryption methods, adhering to data protection regulations, and being transparent about data usage policies [53].

2.10 Recent Advancements in Model Compression and Efficiency

The large computational resources required for training and deploying LLMs have led to significant research into model compression and efficiency techniques. These advancements aim to reduce the size and resource requirements of LLMs while maintaining or even enhancing their performance [32].

Model compression techniques, such as pruning, quantization, and knowledge distillation, have been widely explored to create smaller, more efficient models. Pruning involves removing less important weights from a neural network, effectively reducing its size and computational complexity without significantly impacting performance [44]. Quantization reduces the precision of the model weights, allowing for faster computation and lower memory usage (Jacob et al., 2018) [54].

Knowledge distillation transfers knowledge from a large, pre-trained model (teacher) to a smaller model (student), enabling the student model to achieve comparable performance with reduced size and computational requirements (Hinton et al., 2015) [24]. This approach has been particularly effective in deploying LLMs on edge devices and resource-constrained environments.

Recent innovations also include the development of sparse architectures, where only a fraction of the model's weights are active during inference, significantly reducing computation and memory usage [14]. Additionally, low-rank approximation techniques, such as Low-Rank Adaptation (LoRA), have been employed to adapt pre-trained models efficiently for specific tasks [16].

The combination of these techniques has led to the creation of highly efficient LLMs that can be deployed in various applications, including real-time systems and mobile devices. For example, techniques such as Quantized Low-Rank Adaptation (Q-LoRA) combine quantization and low-rank adaptation to further enhance model efficiency [13].

2.11 Summary

The advancements in large language models (LLMs) and their fine-tuning techniques have opened new frontiers in various applications, including travel planning and human-robot interaction. The integration of ethical considerations ensures responsible AI deployment, addressing issues of bias, fairness, transparency, and accountability. Additionally, recent advancements in model compression and efficiency techniques have made it feasible to deploy powerful LLMs in resource-constrained environments, broadening their practical applications. Combining algorithmic innovation, system integration, and user-centric design is essential to harness the full potential of LLM-powered systems.

3 METHOD

3.1 Unity System introduction

In this project, we developed a travel planning robot in a virtual environment using the Unity game engine. The virtual environment serves as an interactive platform where users can interact with non-playable characters (NPCs) designed to act as travel guides. This section provides an overview of the methods and techniques used to create the system, focusing on user interaction, dialogue management, and AI integration like OpenAI's ChatGPT and other open-source large language models like Google's gemma model, and OpenAI's Whisper model.

The design of the virtual environment relies heavily on the Unity game engine's capabilities for rendering high-fidelity 3D graphics and simulating real-world physics, which together contribute to an immersive user experience. Unity's Scriptable Render Pipeline (SRP) was utilized to enhance graphical performance and achieve realism in environmental design, optimal for engaging user interaction (Smith et al., 2020)[56]. Additionally, the Unity engine enabled dynamic scene management, making it possible to adapt to the virtual world based on user-specific interactions and preferences (Chen et al., 2021)[6].

User interaction within the virtual environment was facilitated through a combination of intuitive control schemes and immersive user interfaces. 2 major user interaction techniques were deployed during this process. The first one is keyboard typing to input query messages, the second one is using voice input to send queries to OpenAI.

Dialogue management plays a crucial role in ensuring seamless communication between users and NPCs. Our system integrates a sophisticated dialogue management framework leveraging natural language Processing (NLP) techniques. Specifically, we employed the ChastGPT (gpt-3.5 turbo) agent, the Whisper speech-to-text agent, and Amazon's text-to-speech agent. Such advanced NLP capabilities enable NPCs to provide relevant and dynamic responses, and convenient user conversation experience, thus enhancing the overall interactivity and utility of the virtual travel planning experience.

The integration of artificial intelligence (AI) in NPC behavior and interaction models allowed for more lifelike and responsive guides. We used Transformer-based language models (Vaswani et al., 2017)[48] not only GPT but also gemma-2B and gemma-7B models to generate traveling planning solutions and seek more advanced planning workflow chains using prompt engineering for language understanding and response generation. The Transformer model's ability to process sequential data and understand context makes it well-suited for developing conversational agents with high accuracy and coherence.

3.2 RAG Introduction

The system supports personalized travel recommendations through AI-driven data analysis techniques, to be specific, the Retrieval Augmented Generation (RAG).

The word "Retrieval" means to recall the history and relevant knowledge using the efficient storage and retrieval capabilities of vector databases (Chroma or Pinecone). The word "generation" means using the LLM and the prompt engineering, the recalled knowledge can be used to generate the final travel plan.

The RAG technique is meant to combine relevant knowledge from the database into the user's initial prompt (instruction) to form a more formal, planned, and longer prompt. Sometimes, to accelerate the process of RAGm, we usually make a very exquisite prompt template ahead of the RAG process, when the user needs to send the prompt to the LLM, we first put the user's initial query into this prompt and then retrieve knowledge from the database and fill into this template.

After those 2 steps, a completed and integrated prompt template is made and will be finally sent to the LLM.

3.2.1 RAG Retrieval Phase. **Document Vector Matching:**

First, the system employs a sophisticated document vector matching algorithm to retrieve documents or text fragments that are pertinent to the user's input question. This process begins with translating the user's query into a vector form, which is subsequently compared against a pre-stored document vector repository to pinpoint the most similar document.

Notably, the process of converting text into vector form can leverage various embedding techniques such as Word2Vec, GloVe [33], or BERT embeddings[12]. Recent advancements have shown the efficacy of these contextual embeddings in enhancing the semantic matching of documents, thereby significantly improving accuracy [23]. These embeddings capture nuanced meanings and contextual relationships between words, making them particularly effective in information retrieval tasks.

Matching Algorithms: Two primary matching strategies can be implemented: q-to-d (query-to-document) and q-to-q (query-to-query). In the q-to-d approach[29], the system matches the user query directly against the documents stored in the repository. This method utilizes vector similarity measures such as cosine similarity, Euclidean distance, or more advanced metrics like dot product attention as seen in Transformer models [48]. On the other hand, q-to-q matching involves comparing the user query with other users' historical queries. This strategy can leverage query logs and collaborative filtering techniques to find the closest matching queries [31] .

Moreover, the q-to-q approach can benefit from reinforcement learning algorithms that improve the matching accuracy over time based on user feedback [57]. This dynamic adaptation ensures that the retrieval system evolves and becomes more efficient in serving user needs.

3.2.2 RAG Generation Phase. **Packaging into Prompt:** Once a relevant document or query is retrieved, the next step involves crafting this information into a comprehensive prompt. This prompt encapsulates the user's original question along with the matched text, which is then fed into a large language model (LLM). The formulation of an effective prompt is critical for generating meaningful responses and can be fine-tuned using prompt engineering techniques [22].

Recent research has explored different prompt strategies to optimize the performance of LLMs. For instance, few-shot and zero-shot learning approaches have been found to significantly influence the model's ability to generate accurate and contextually relevant answers [5]. Additionally, the choice of prompt can affect the coherence and informativeness of the generated text [36].

Generate Answer:

Upon receiving the meticulously designed prompt, the large language model generates a response or continues to produce additional text. Contemporary models like GPT-3 [6]and Transformer-XL [11]are capable of handling long-range dependencies and generating high-quality, coherent text. These models utilize advanced mechanisms such as self-attention and memory-augmented networks to maintain context over extended text sequences, ensuring that the answers remain relevant and accurate.

Furthermore, the generated answer can be post-processed to improve its readability and informativeness. Techniques such as re-ranking, summarization, and fact verification [47]can be applied to the raw output. These steps ensure the final response is not only contextually pertinent but also factually correct and user-friendly.

3.3 Prompt Engineering on Travel Planning

By integrating the NPC background prompt template and the game world's prompt template, the virtual robot can provide tailored travel plans that match individual user preferences. This

personalized approach not only increases user satisfaction but also improves the relevance of the travel recommendations offered.

Additionally, we created a travel path prompt template for LLM to generate the fusion of NPC background and game world templates through prompt engineering significantly augments the capabilities of virtual robots in delivering personalized travel experiences. Future research should focus on fine-tuning these models with a broader array of contextual inputs and exploring real-world applications across various domains.

3.4 Project Architecture

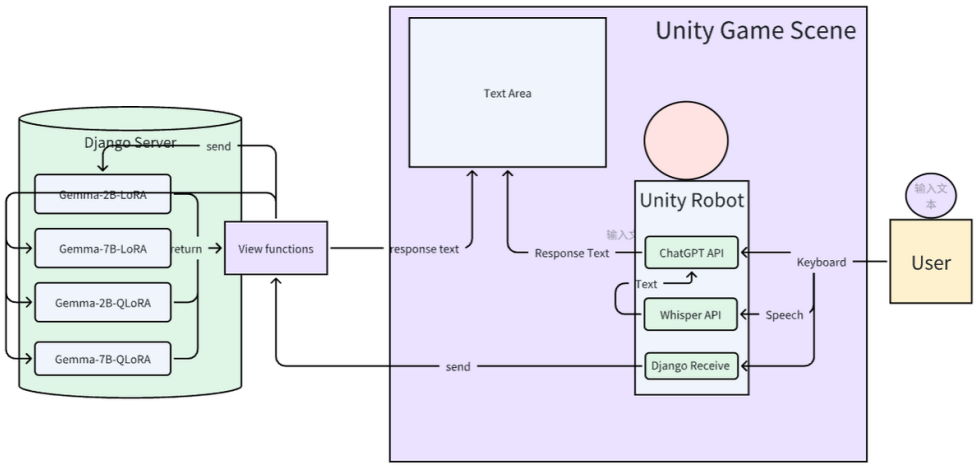


Fig. 1. Main components of our project's architecture

3.4.1 *Main Components Diagram.*

3.4.2 Project Architecture Summary. The project integrates Unity for the front-end game interface and a Django backend that manages sophisticated open-source large language models hosted on HuggingFace and powered by fine-tuned technologies.

The unity part of the project contains a robot (NPC) configured with OPENAI's ChatGPT API (text-to-text) and the Whisper API (speech-to-text) as well as Amazon's Polly API (text-to-speech) and supported by the ability of LLM to generate a fine-grained route planning answer for the personal traveling. This robot (NPC) is located at the center of the game scene, when our user manipulates their game avatar to approach the robot, a Dialog box will be activated, and the user can communicate with the large language model in this area.

The Django part is developed using Python and contains 4 large language model-invoking scripts as static resources. Each of the 4 model scripts will be encapsulated into a function, whenever the user sends a query in the unity side and chooses to use the open source model at the same time, the sending message (an HTTP request) will be sent to the Django server, and the view function in Django will receive this request turn to call the encapsulated model invoking function to invoke our fine-tuned Gemma models to answer user's query and generate them a travel plan, this plan will be again sent back to Unity using an HTTP response are received by a customized script inside

the Unity, and the generated travel plan will be eventually presented on the Dialog window's text area.

Virtual Environment and User Interface

The virtual environment is built in Unity, providing users with a visually immersive environment to explore. The user interface (UI) in this environment is designed to be intuitive and user-friendly. Key elements of the user interface include:

NPC travel guide:

The central interaction point is the NPC travel guide located at various attractions within the environment.

Interaction Trigger:
The user initiates interaction by approaching the NPC. Once you enter a predefined trigger zone, the dialogue system is activated.

Dialogue System

The dialogue system is designed to be simple and clear to facilitate smooth communication between users and NPC tour guides. It includes two main methods for users to enter queries:

Text Input:
Users can enter questions in the text box and click the send button to submit.

Voice Input:

Users can click the recording icon to initiate voice recording. Users can then use the microphone to answer their questions, which will be recorded and transcribed.

The simplicity of the conversational UI ensures that users can communicate travel-related queries easily and quickly, thereby enhancing the user experience.

AI Model and Speech Synthesis

To provide accurate and contextual answers, NPC Travel Guide uses advanced artificial intelligence models. The workflow for generating a response is as follows:

Question handling:
When a question is submitted via text or voice, it is sent to an artificial intelligence model designed to understand and generate a natural language response.

Response Generation:
The AI model processes the input question and generates an appropriate response based on the context and content of the query.

Speech synthesis:
Once the response is generated, it is converted into spoken language using AI-driven text-to-speech (TTS) technology. This synthesized voice is designed to be natural and engaging, improving the quality of interactions.

Technical implementation

The integration of these components involves several key technologies and frameworks:

Unity engine:
used to create virtual environments and UI elements.

AI models:

Leverage state-of-the-art natural language processing (NLP) technology to understand and respond to user queries. Models such as GPT (Generative Pretrained Transformer) can be used for this purpose.

Speech recognition and synthesis:

APIs that enable speech-to-text (STT) and text-to-speech (TTS) conversion. Tools such as Google Cloud Speech-to-Text and Amazon Polly can be used to perform these tasks.

By combining these technologies, our system provides a seamless, interactive trip-planning experience. With integrated artificial intelligence and speech synthesis capabilities, users can easily interact with NPC travel guides, receive relevant travel information, and enjoy a realistic conversation experience.

3.4.3 Project Architecture Explanation. Django part of the Project

Main Python Scripts

The Django server contains 4 scripts, each python script (aka. .py file) is responsible for running a particular version of a large language model called "gemma" to get its response on generating a travel plan for the user. Each script contains a few steps including loading the dataset, fine-tuning the model using LoRA and Q-LoRA as well as saving the model to the HuggingFace and reloading the fine-tuned model, those 4 scripts also utilized HuggingFace components for tokenization and model handling (e.g., AutoModelForCausalL, LoraConfig, SFTTrainer) to make the training process more convenient.

In each open source LLM's python script (e.g. fine_tune_gemma_2b.py), there are a few components, namely:

LoRA configuration, which is used to set LoRA to fine-tune the method's parameters, for example, the rank r. Let's explain a little bit. Firstly, LoRA's ideology is to decompose the weight matrix Winto 2 smaller matrices A and B, suppose W is a $m \times n$ matrix and $W = AB$, how to ensure the size o A and B? The only one criterion is whether the parameter number in W can be significantly decreased, if W is a 100×100matrix, which contains 10000 parameters. Now, if we let $A = 100 \times 5$, $B = 5 \times 100$, then $W = AB = 100 \times 100$, but the parameters in A plus the parameters in B is only 1000, which is significantly smaller than 10000 in W, and a smaller parameter number in Deep Learning means smaller memory needed and a faster training speed as well as cost-effective deployment of the model. Now, we can define this 5 to be r, which we call a "low rank", then A should be $100 \times r$, B should be $r \times 100$, all of these are the theories related to LoRA fine-tune. 2

```
lora_config = LoraConfig(
    r=8,
    target_modules=["q_proj", "o_proj", "k_proj", "v_proj", "gate_proj", "up_proj", "down_proj"],
    task_type="CAUSAL_LM",
)
```

Fig. 2. LoRA Configuration

Quantization Configuration: 3
Loading Gemma model from hugging face
Load the dataset specifically designed for fine-tuning: 4
Set up a supervised fine-tuning trainer (SFTTrainer): 5

```python
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.bfloat16
)
```

Fig. 3. Quantization Configuration

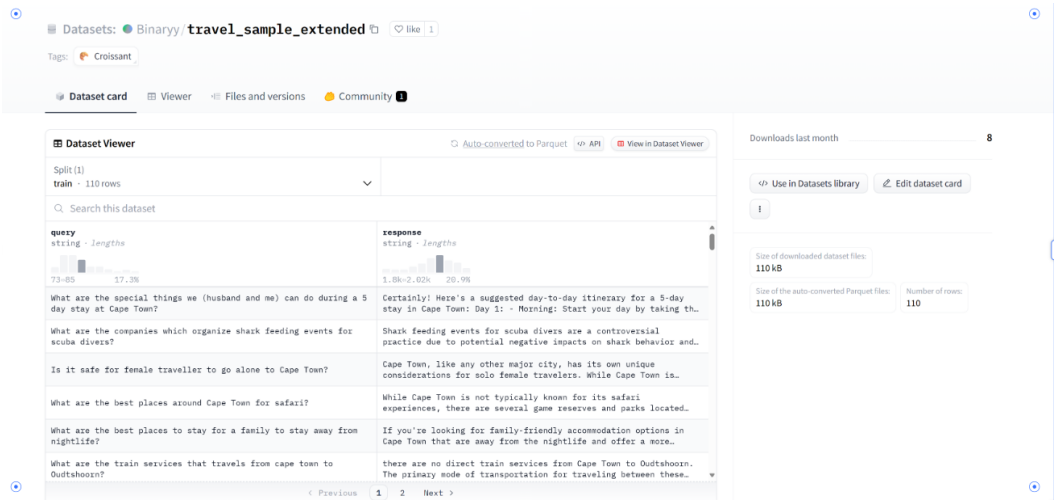


Fig. 4. Dataset

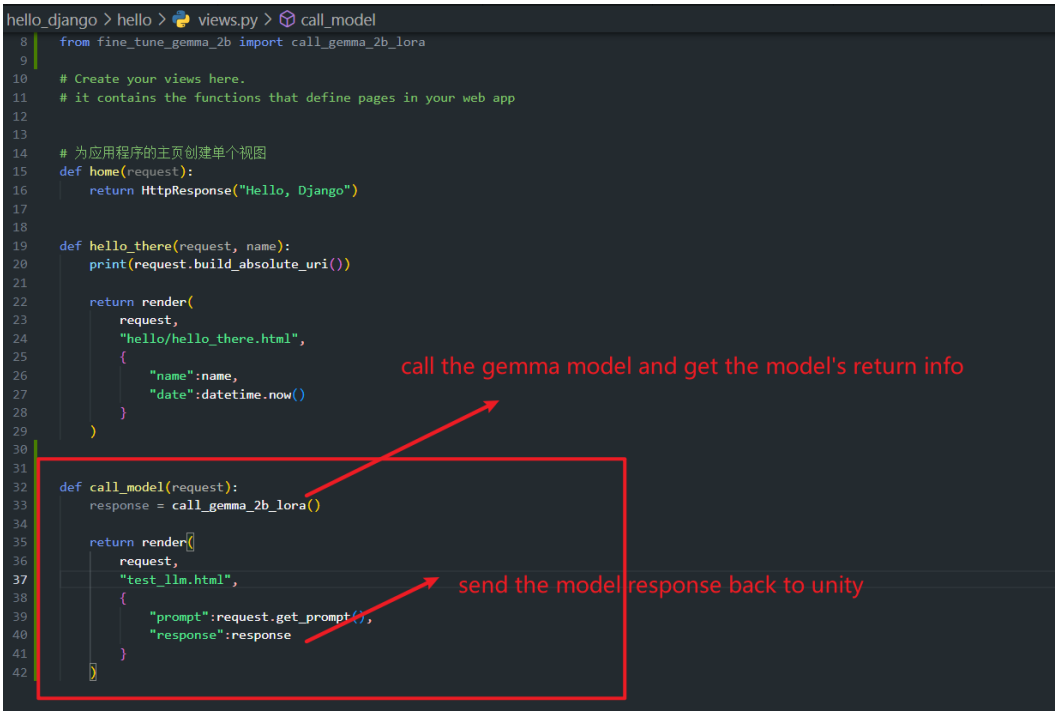
```python
trainer = SFTTrainer(
    model=model,
    train_dataset=data["train"],
    args=transformers.TrainingArguments(
            # 每个设备上的训练批次大小为1
            per_device_train_batch_size=1,
            # 梯度累积步数=4：每进行4次前向传播累积一次梯度后再进行一次反向传播，有助于处理内存限制
            gradient_accumulation_steps=4,
            # 在训练初期，学习率会逐渐增加至设定值，这里的预热步数是2
            warmup_steps=2,
            # 最多执行10步训练
            max_steps=10,
            #
            learning_rate=2e-4,
            # 使用半精度16位浮点数进行训练
            fp16=True,
            logging_steps=1,
            output_dir="outputs",
            optim="paged_adamw_8bit"
        ),
    peft_config=lora_config,
    formatting_func=formatting_func,
)
trainer.train()
```

Fig. 5. SFT Trainer

Save the model to HuggingFace: 6
Encapsulate the model calling process into a function so that Django's view function can call it. 7

```python
# push model to hub
from huggingface_hub import notebook_login
notebook_login()

# option 2: key login
from huggingface_hub import login
write_key = 'hf_' # paste token here
login(write_key)

hf_name = 'zhongyah' # your hf username or org name
model_id = hf_name + "/" + "gemma-2b-lora-ft"


model.push_to_hub(model_id)
trainer.push_to_hub(model_id)

```

Fig. 6. Save model to HuggingFace




```python
from fine_tune_gemma_2b import call_gemma_2b_lora

# Create your views here.
# it contains the functions that define pages in your web app


# 为应用程序的主页创建单个视图
def home(request):
    return HttpResponse("Hello, Django")


def hello_there(request, name):
    print(request.build_absolute_uri())

    return render(
        request,
        "hello/hello_there.html",
        {
            "name":name,
            "date":datetime.now()
        }
    )


def call_model(request):
    response = call_gemma_2b_lora()

    return render(
        request,
        "test_llm.html",
        {
            "prompt":request.get_prompt(),
            "response":response
        }
    )
```

Fig. 7. Call the model using django's view function

Environment Setup

- Running on Windows 11, Visual Studio Code.
- Hardware: NVIDIA Quadro P4000 GPU with 8GB RAM running CUDA 11.7.
- Due to the low computing performance of our training setup, especially due to the limited GPU RAM, the experimenters can not effectively train the Gemma model on their devices.

The fact is, a 7B model requires at least 7GB of GPU RAM, however, the researchers only have 4GB available on their GPU. Consequently, fine-tuning a Gemma model with the size of 7B takes over 90 minutes. This limitation means the experimenters do not have enough time to fine-tune the same model repeatedly. Therefore, we need to save the fine-tuned model in an online storage location, such as the HuggingFace hub 8.

```
# push model to hub
from huggingface_hub import notebook_login
notebook_login()

# option 2: key login
from huggingface_hub import login
write_key = 'hf_' # paste token here
login(write_key)

hf_name = 'zhongyah' # your hf username or org name
model_id = hf_name + "/" + "gemma-2b-lora-ft"


model.push_to_hub(model_id)
trainer.push_to_hub(model_id)
```

Fig. 8. Save the model to HuggingFace

Interaction with Unity

Unity's Player sends requests to the Django backend. The Django view function receives the HTTP request and then invokes the gemma model calling the function "call_gemma_2b_lora()" in the model script (e.g., fine_tune_gemma_2b.py) and fetches model outputs. The output of the Gemma model will be encapsulated in HTTP responses and sent back to Unity.

Unity Reception and Display

Unity receives path planning information generated from Django's models and passes it to the TextArea component through NPCDialog.cs.

Download Pre-trained Models for comparison

Install the pre-trained models for comparison, including gemma-2b and gemma-7b, from the Olama software.

Unity Part of the Project

Game Scene Setup

This project is based on a third-person game scene with two game characters: a Player (controlled by users) and a Guide (NPC, our robot). - We write a character motion script and an animator to control the Player and the NPC robot.

NPC Interaction

In the Robot (the NPC, we call it the "Guide" in the game) part, we set an event trigger on its body 9, whenever that player is moving approaching the surrounding area of the NPC, the script inside

the event trigger will trigger the functionality of the "ToActivate" component, which it contains all the ability of the ChatGPT and Whisper. You can check the event trigger in the figure below: 9 10

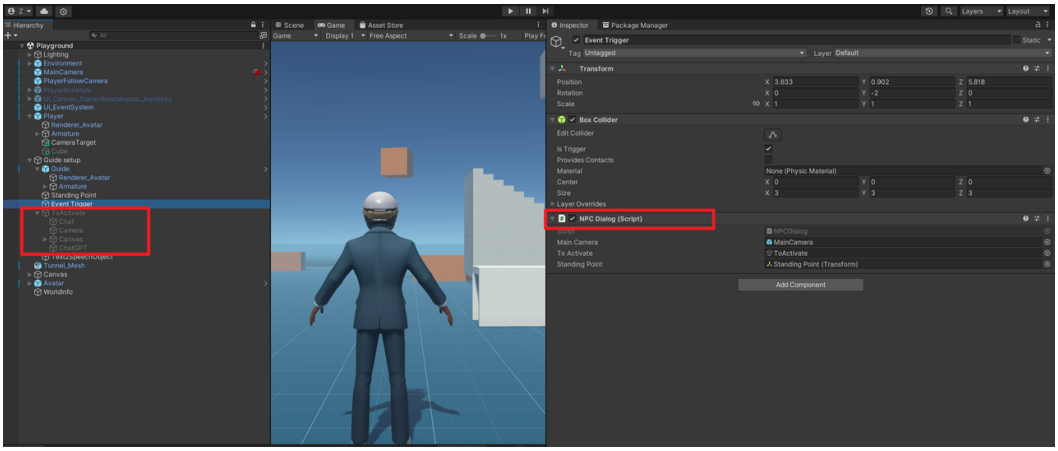


Fig. 9. Event Trigger

After the "toActivate" component is activated, The dialog box will appear in the center of the screen. 11

The code to realize this functionality is shown below, inside the "toActivate" game object, we have a "Chat" game object that is used to load two C scripts (ChatTest.cs and Whisper.cs) 12, the ChatTest.cs 13 is our master script to control the entire ChatGPT working flow, including how to store the history message, how to define the initial prompt and the character background (as a traveling agent)

Also, in ChatTest.cs 13, we use this append message function 14 to append user's new query and the GPT's instant response to the text area

Whenever a new message is added to the input field and sent to the GPT, the AppendMessage will dynamically adjust the position of each message container box in the text area (here, we use the RectangularReact component to store messages), making sure that the entire layout of all messages is fitting the textarea. Each time a query is asked by a user, the AppendMessage function will automatically update the height of all the messages and automatically scroll the view of the text area to the bottom where there is the newest message returned by GPT. 15 16

Now, except that the user can use the keyboard to send a query to GPT to get a travel path planning, we can also use our voice to communicate with a GPT-powered robot to get our answer. So, how to do that, we must implement OpenAI's Whisper API to use their voice-to-text conversion model (not GPT) to convert our voice to text, then, we have to transfer the converted text to ChatGPT's API through their object's parameter, and finally we can get the response. Underneath is the Whisper.cs file 18 19 that stores all the entry points of the Whisper API.

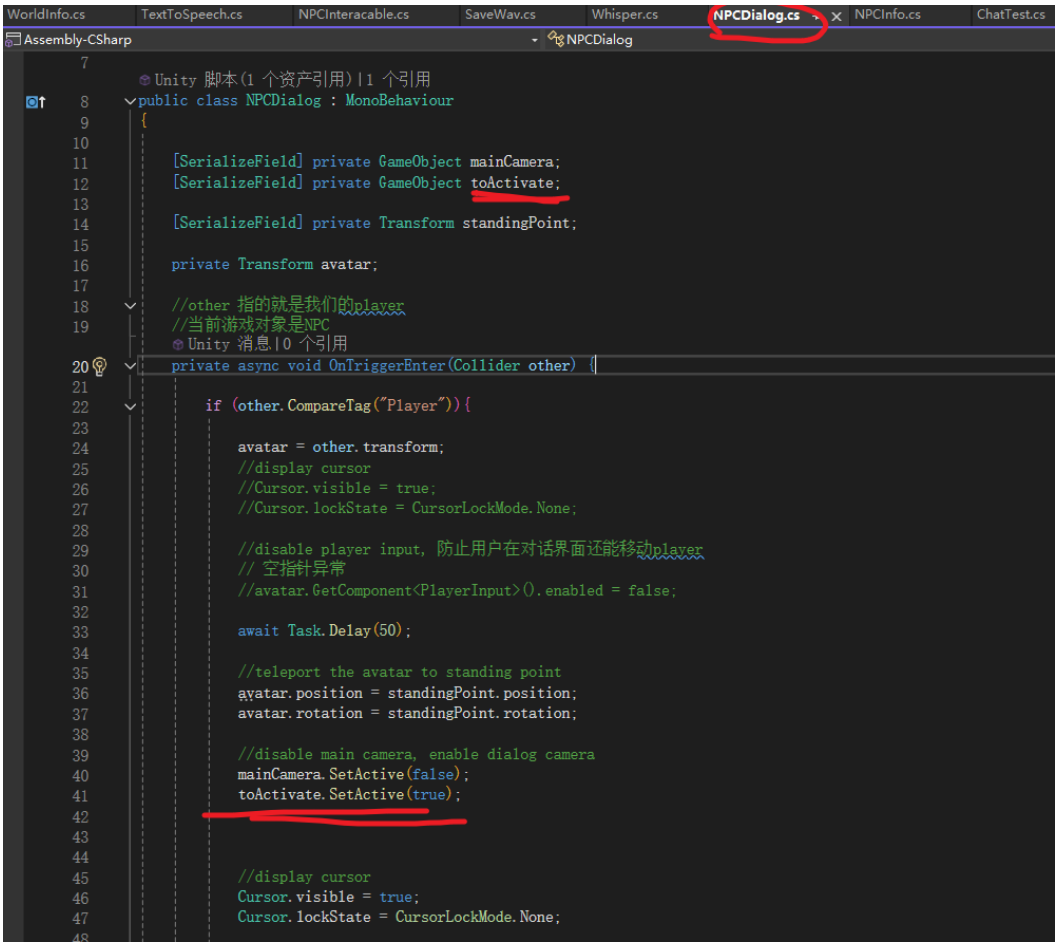


Fig. 10. NPC Dialog

However, our project is more than that, after we get the text response from the ChatGPT, usually the returned answer is very long, since we are doing a user-centric robotic design, it is naturally expected by users to not go through such redundant and lengthy text. Under such requirements, we designed a text-to-speech functionality inside the robot (NPC) with the help of AWS (Amazon Web Services), especially its Polly service (a kind of Text-To-Speech service, aka. TTS). The Polly service is meant to transfer a segment of text into a period of human voice and you can pick various kinds of timbre from their list.

Underneath is our script for handling AWS's Polly TTS service:

3.4.4 Experiment Method. **Participants**

We did a complete random sampling to sample 5 students from Xi'an Jiaotong-Liverpool University. Each person will write down 2 questions regarding the Travel Path Planning problem, for example, a well-defined question should look like this: "Please plan me a travel path from Beijing to NewYork".

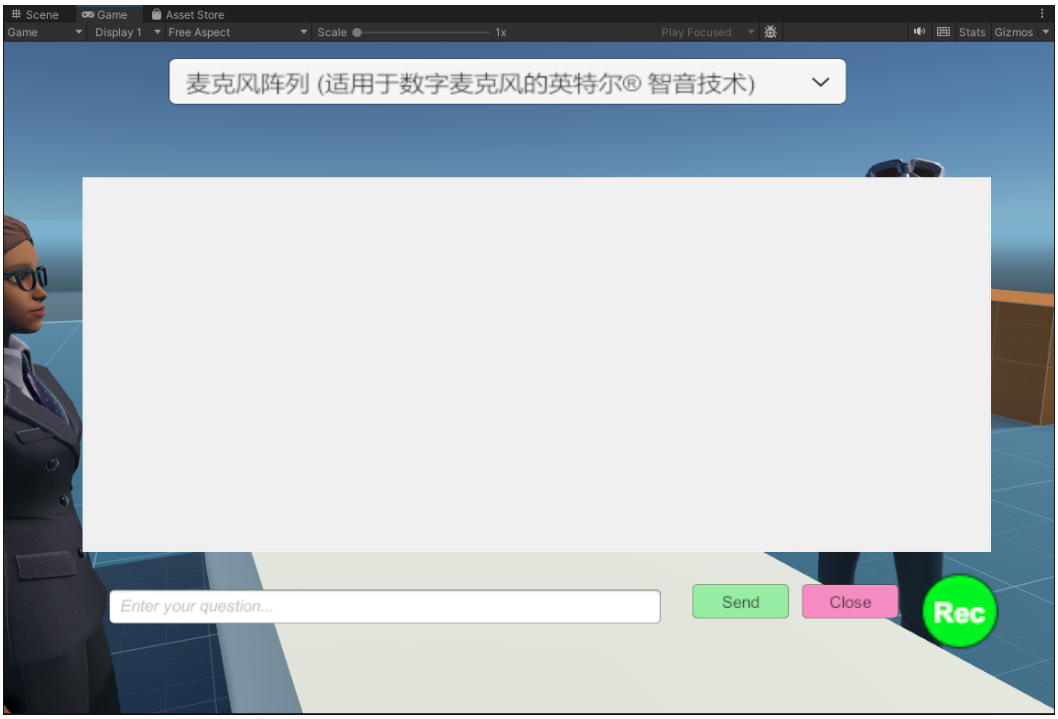


Fig. 11. Dialog Interface from the Robot

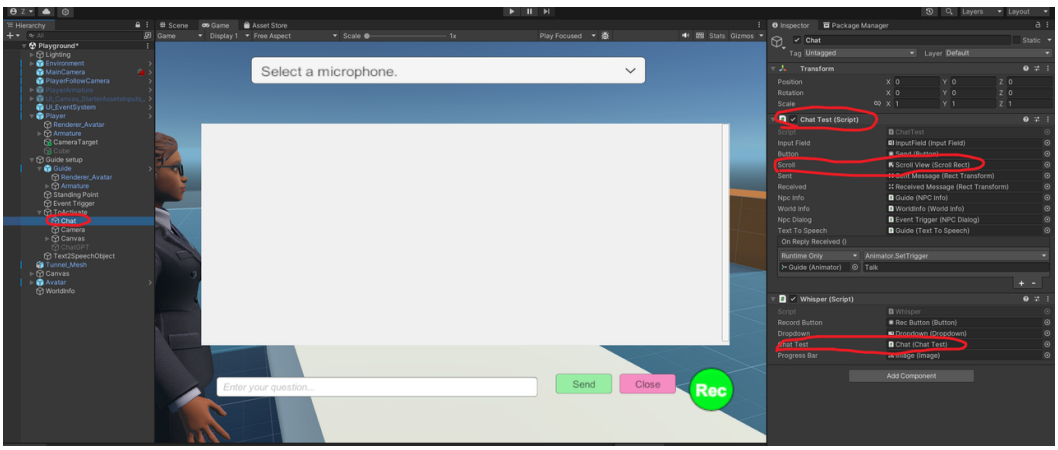


Fig. 12. A game object called c̈hatẗhat integrates ChatTest.cs and Whisper.cs

You may wonder why we want each person to generate 2 questions rather than let 1 person generate 10 questions at once. It is because we take the model's reasoning ability and long-context dependency problem into consideration since we are letting the users input their prompts one by one rather than in the same turn of the GPT conversation rather different turns, we are doing this to avoid an influential issue that the history of conversation may generate an irreversible impact to

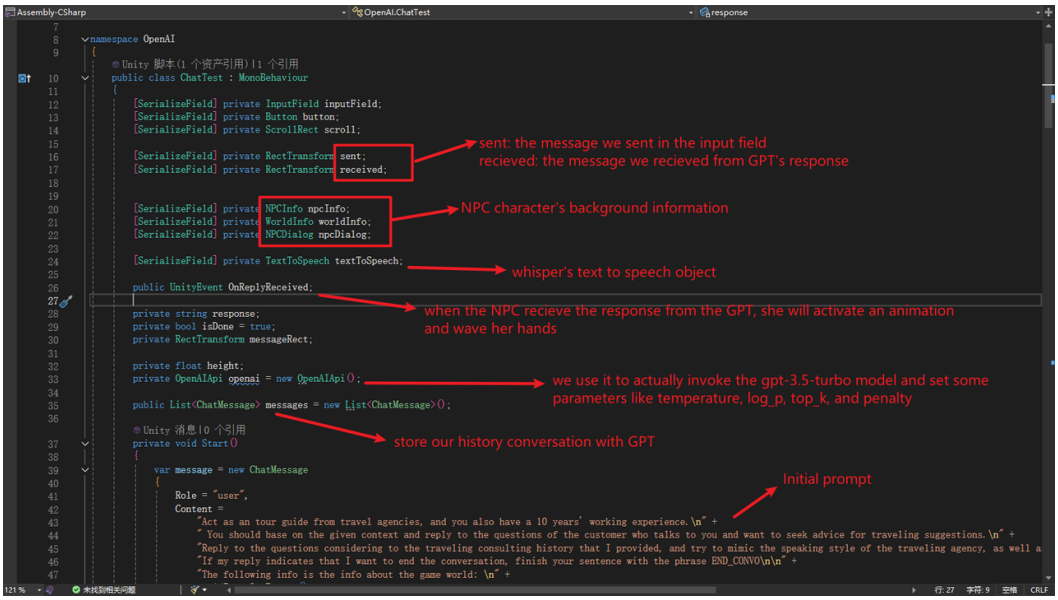


Fig. 13. ChatTest.cs

Fig. 14. Append Message

the response of the LLM because the ideology behind the reasoning is to use this long-dependency in the attention mechanism to infer the next word, which means the model will consider the entire context and generate the result depend on the top-k probability of the entire context,c which means, if we are letting the same person to generate all the questions, the model will only generate path planning according to this particular person's talking style. That is the ultimate reason why we are letting 5 people answer 10 questions.

Data collections

For each model (gemma-2b-lora, gemma-7b-lora, gemma-2b-qlora, gemma-7b-qlora), the user prompt and the model's response will be recorded into a table ($gemma_2b_lora.csv$).$Eachtablecontains10rows, eac$

The "quality" is an integer score from 1-3, corresponding to (low, medium, high), this quality stands for the quality of the LLM's travel plan generation.

For each (prompt, response) pair of a particular model, we will take this tuple as the input to feed the ChatGPT and use the prompt engineering technique to tell it the criteria for judging the quality of the model's response, and the ChatGPT will eventually help us to tag each (prompt, response) pair with a quality score from 1-3.

After this stage, each model will have a CSV table that contains 10 (prompt, response, quality) tuples.

Set Up Experimental Groups

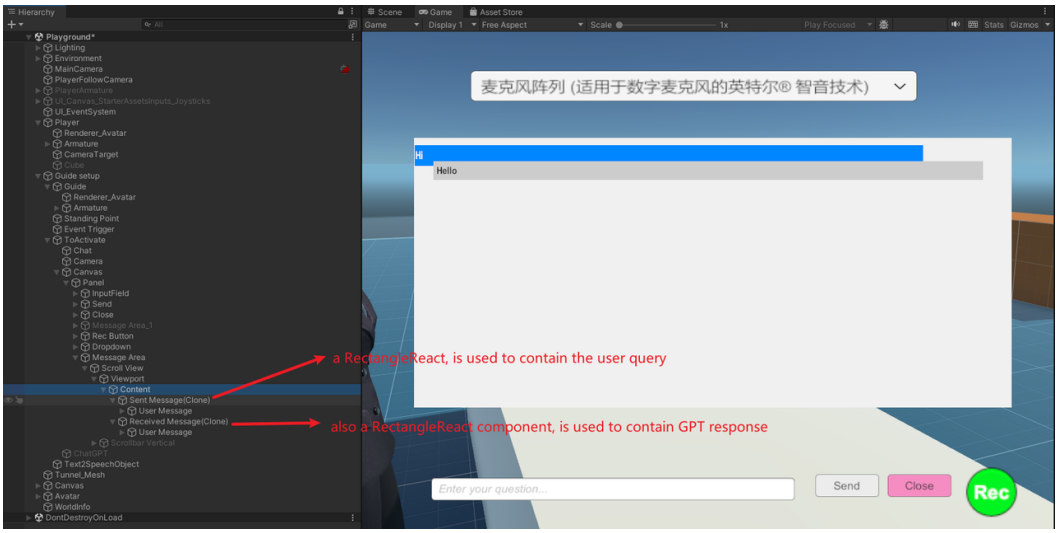


Fig. 15. The message object in the text area of the dialog

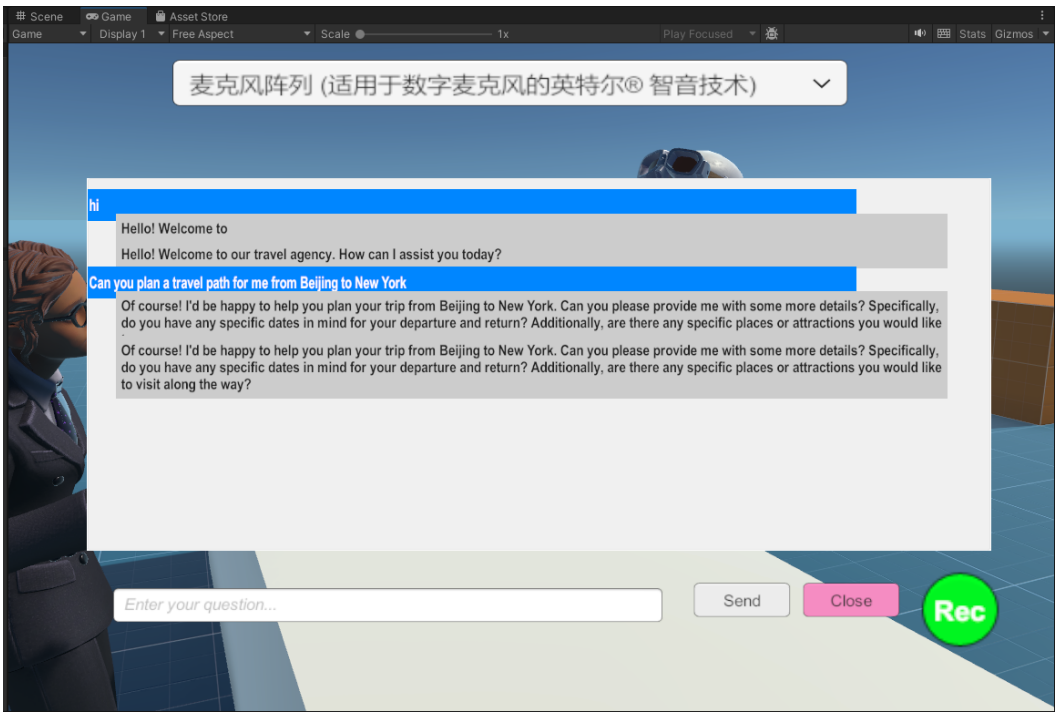


Fig. 16. Conversation with GPT

We mainly implement a between-group design for this experiment. So we implemented the following 4 sets of between-group experiments, each group contains 10 path generation performance scores and each score is between 1-3.

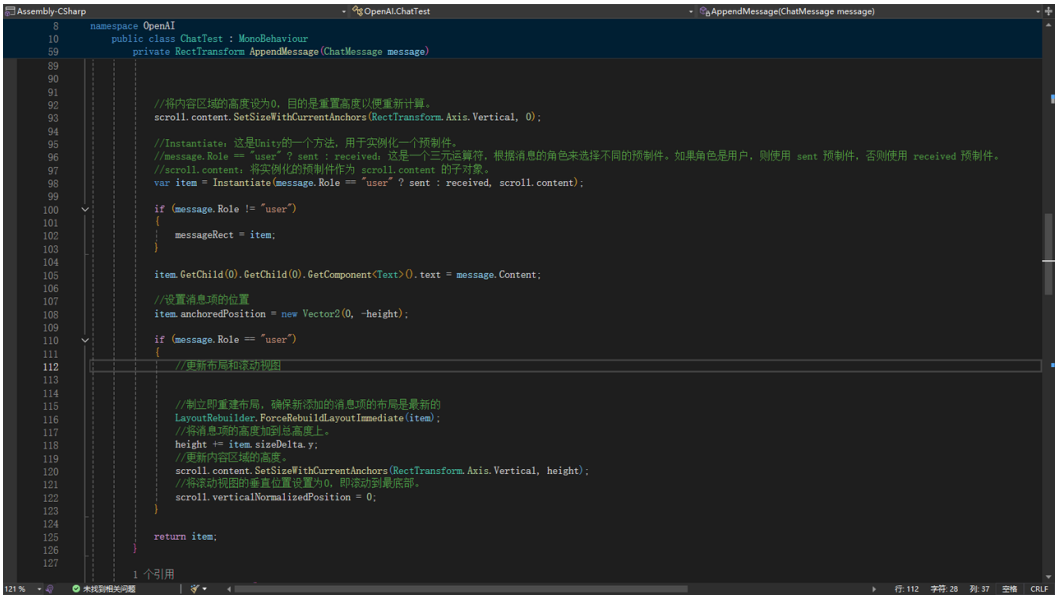


Fig. 17. The code in ChatTest.cs where the new message is appended and the conversation list is updated

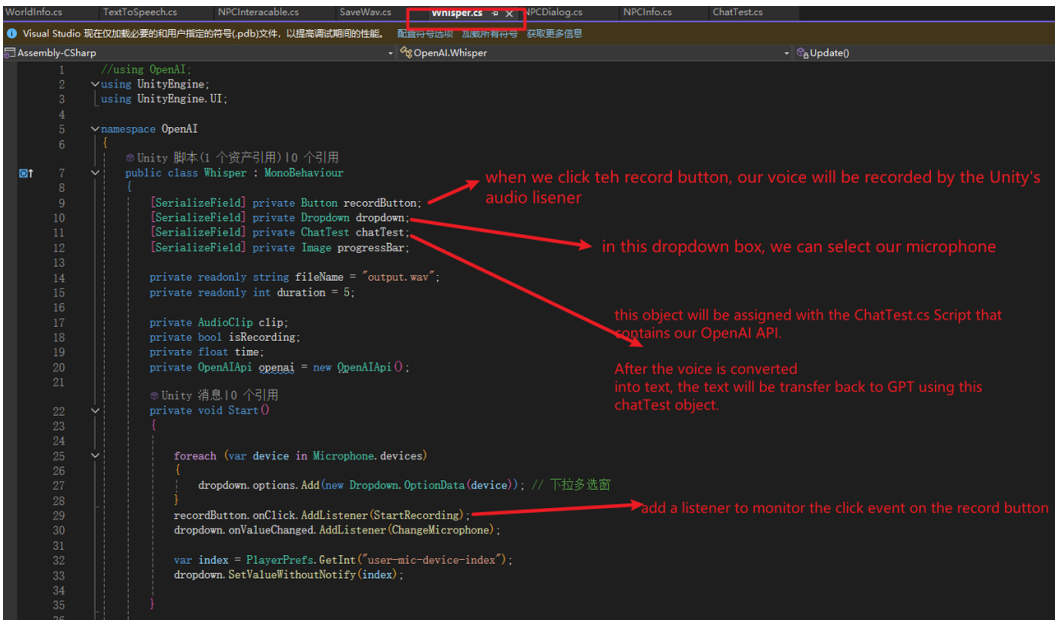


Fig. 18. Whisper.cs

Based on our research questions and hypothesis:

(1) Under the circumstance of gemma-2b, we compare the path generation performance between the Q-LoRA and LoRA fine-tuned gemme-2b model.

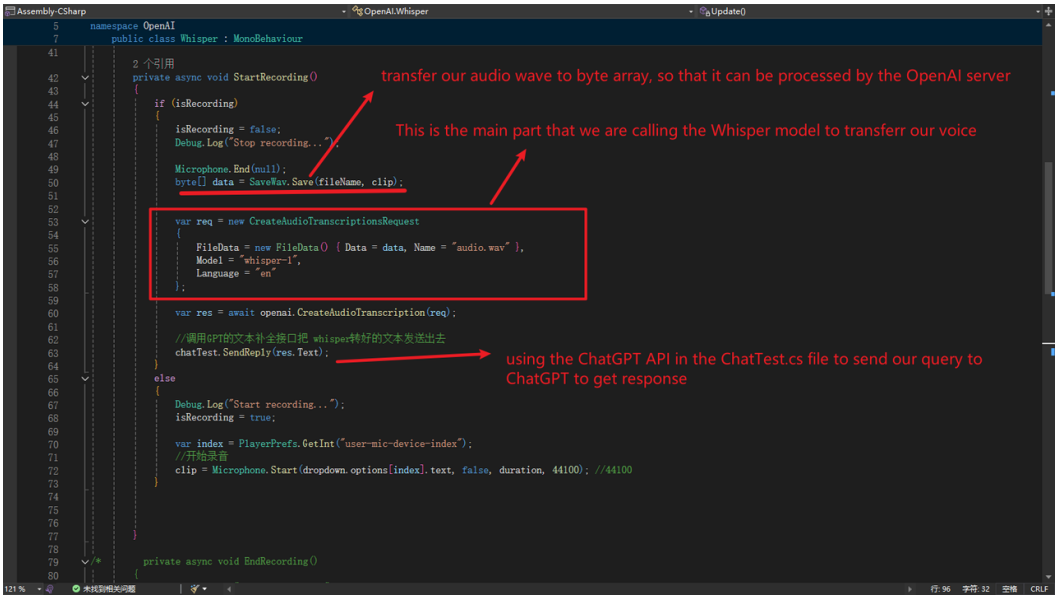


Fig. 19. Whisper.cs continue

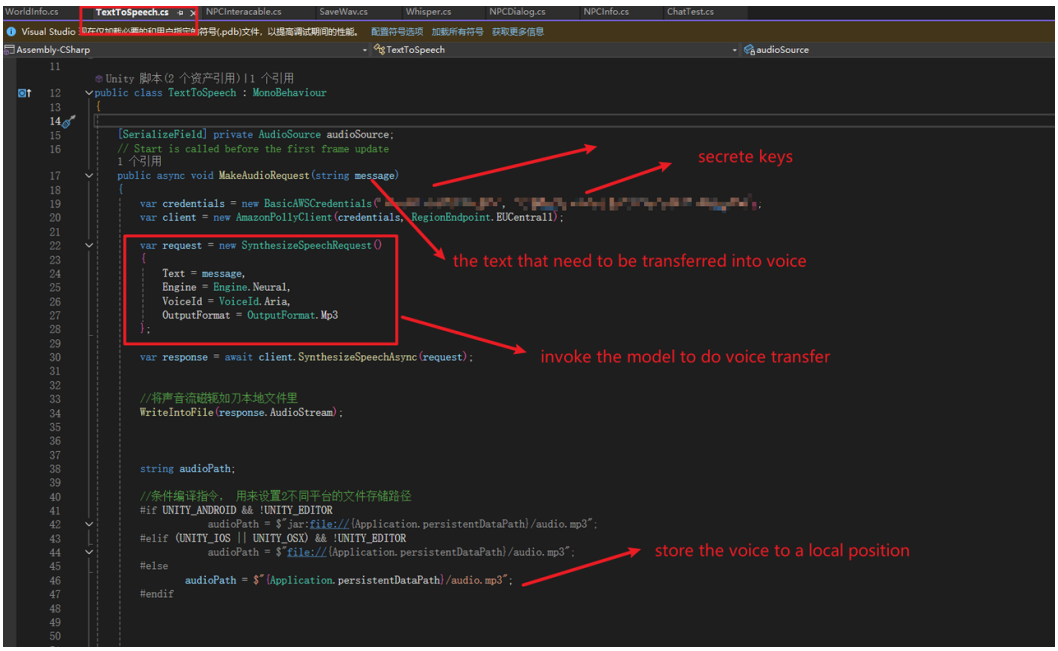


Fig. 20. TextToSpeech.cs

(2)Under the circumstance of gemma-7b, we compare the path generation performance between the Q-LoRA and LoRA fine-tuned gemme-7b model.

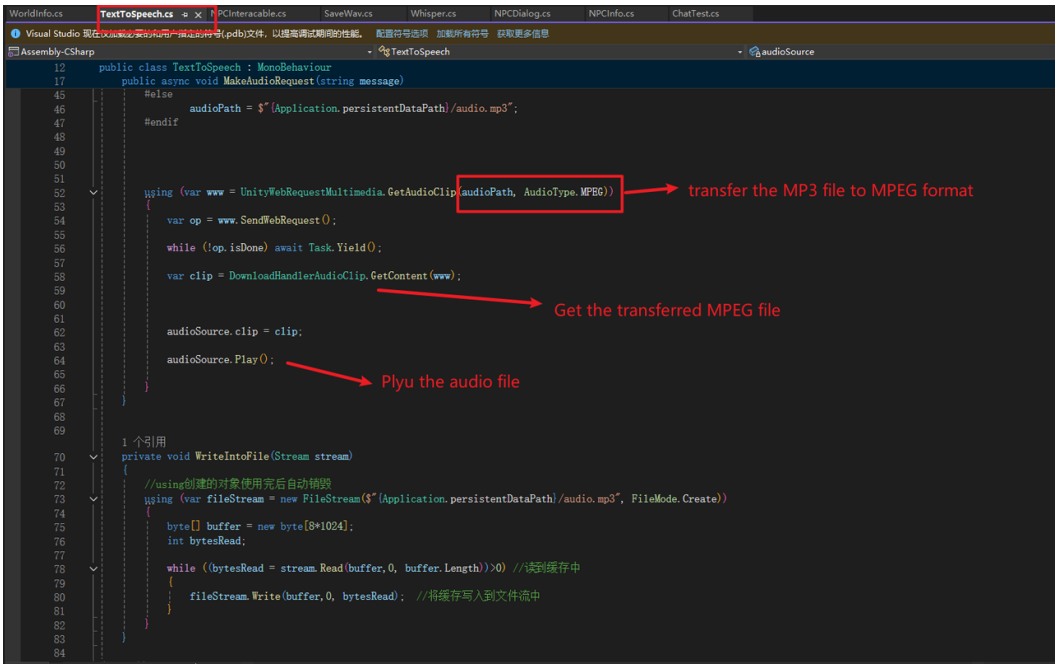


Fig. 21. TextToSpeech2.cs

(3) When the fine-tuning technique is LoRA, we compare the path generation performance between the 2B gemma and the 7B gemma.

(4) When the fine-tuning technique is Q-LoRA, we compare the path generation performance between the 2B gemma and the 7B gemma.

4 RESULTS

4.1 Model Evaluation

In order to evaluate the performance of the travel planning robotic system, we designed a series of experiments to test and analyze the performance of four different models. These models include two model sizes (2B and 7B) and two fine-tuning methods (LoRA and Q-LoRA). The specific experimental procedures and results are analyzed below.

First, we generate answers using each of the four models and save the results. The results of each model are saved in a table whose format includes the following fields: id (question number), prompt (question asked by the user), response (answer generated by the model), and quality (answer quality).

To increase the randomness of the questions, five volunteers were invited to ask two questions each about travel path planning, for a total of 10 questions collected. The randomization and diversity of these questions ensured that the test was broad and representative. We then fed these questions into each of the four models and had each model generate responses.

Next, we scored these responses using GPT. The scale was from 1 to 3, corresponding to three levels of "low", "medium", and "high". In order to ensure fairness and consistency of scoring, we have set up a scoring rubric and scoring guidelines in advance, which include:

- **Accuracy**: Whether the answer correctly addresses the question posed by the user.

- **Relevance**: Whether the answer is highly relevant to the user's question.
- **Logic**: Whether the answer is logical and easy to understand.
- **Practicality**: Whether the answer provides practical, useful travel advice.

The scoring results are saved in a table containing id (question number), prompt (question asked by the user), response (answer generated by the model) and quality (GPT score). To better analyze the results, we compute the total score for each model's response. For example, if a model receives the highest score (3) for each response, it will have a total score of 30 (10 questions with 3 points each). Therefore, the total performance score domain of the model is [0, 30].

Based on the research questions, we grouped and analyzed the data.

Research Question 1: Do different fine-tuning methods (LoRA and Q-LoRA) affect the quality of travel plans generated by a large language model (gemma-2b)?

We used a between-group design to compare the performance of gemma-2b models under LoRA and Q-LoRA fine-tuning methods, respectively. For each model, we calculated the total score of its responses and organized them in the following table:

ID	LoRA	Q-LoRA
1	2	1
2	1	1
3	3	2
4	1	1
5	2	1
6	3	1
7	1	1
8	2	2
9	3	2
10	2	2

Table 1. Performance of gemma-2b with LoRA and Q-LoRA Fine-Tuning Methods

We first tested the normality of the two sets of data using the Shapiro-Wilk test. The results are as follows:

- LoRA group: ShapiroResult(statistic=0.8318431377410889, pvalue=0.03521495312452316)
- Q-LoRA group: ShapiroResult(statistic=0.640485405921936, pvalue =0.00016868000966496766)

Since neither group of data conformed to a normal distribution, we used the Wilcoxon signed-rank test:

- Wilcoxon signed-rank test result: test statistic = 15.0, p-value = 0.016947426762344633

The results show that the LoRA fine-tuning method performs significantly better than Q-LoRA for a model size of 2B. This suggests that LoRA has a better fine-tuning effect when dealing with smaller models and can be adapted to specific task requirements more effectively.

Research Question 2: Do different fine-tuning methods (LoRA and Q-LoRA) affect the quality of travel plans generated by the large language model (gemma-7b)?

We used a between-group design to compare the performance of the gemma-7b model under LoRA and Q-LoRA fine-tuning methods, respectively. For each model, we calculated the total score of its responses and organized them in the following table:

We first tested the normality of the two sets of data using the Shapiro-Wilk test. The results are as follows:

ID	LoRA	Q-LoRA
1	3	1
2	3	3
3	3	2
4	2	1
5	2	2
6	3	2
7	3	2
8	2	2
9	3	2
10	2	3

Table 2. Performance of gemma-7b with LoRA and Q-LoRA Fine-Tuning Methods

- LoRA group: ShapiroResult(statistic=0.6404852867126465, pvalue=0.00016867939848452806)
- Q-LoRA group: ShapiroResult(statistic=0.8148399591445923, pvalue=0.021947985514998436)

Since neither group of data conformed to a normal distribution, we used the Wilcoxon signed-rank test:

- Wilcoxon signed-rank test result: test statistic = 24.5, p-value = 0.028889785561798602

The results show that the LoRA fine-tuning method significantly outperforms Q-LoRA for a model size of 7B. This further validates the stability and superiority of LoRA under different model sizes, especially when dealing with larger models.

Research Question 3: Does model size affect the quality of travel plan generation for LoRA fine-tuned models (gemma-2b and gemma-7b)?

We used a within-group design to compare the performance of gemma-2b and gemma-7b under the LoRA fine-tuning approach. For each model, we calculated the total score of their responses and organized them in the following table:

ID	2B	7B
1	2	3
2	1	2
3	1	2
4	1	2
5	1	3
6	1	3
7	1	3
8	1	1
9	2	2
10	1	3

Table 3. Performance of gemma-2b and gemma-7b with LoRA Fine-Tuning Method

We first tested the normality of the two sets of data. Since neither set of data conformed to a normal distribution, we used the Wilcoxon signed-rank test:

- Wilcoxon signed-rank test result: test statistic = 0.0, p-value = 0.9951260964494131

The results show that the 7B model performs significantly better than the 2B model under the LoRA fine-tuning method. This suggests that the larger model (7B) provides higher quality results when dealing with complex tasks under the same fine-tuning method.

Research Question 4: Does model size affect the quality of travel plan generation for Q-LoRA fine-tuning models (gemma-2b and gemma-7b)?

We used a within-group design to compare the performance of gemma-2b and gemma-7b under the Q-LoRA fine-tuning approach. For each model, we calculated the total score of their responses and organized them in the following table:

ID	2B	7B
1	1	1
2	2	1
3	1	2
4	1	1
5	2	1
6	1	1
7	1	1
8	2	1
9	1	2
10	1	1

Table 4. Performance of gemma-2b and gemma-7b with Q-LoRA Fine-Tuning Method

We first tested the normality of the two sets of data. Since neither data set conformed to a normal distribution, we used the Wilcoxon signed-rank test:

- Wilcoxon signed-rank test result: test statistic = 9.0, p-value = 0.32736042300928847

The results show that there is no significant difference between the performance of model 2B and model 7B under the Q-LoRA fine-tuning method. This suggests that model size has little effect on generation quality under the Q-LoRA fine-tuning method.

Through the above experiments and analyses, we conclude that the performance is better under different model sizes, especially when dealing with larger scale models. And although the Q-LoRA fine-tuning method performs stably under larger models, its overall effect is not as good as LoRA.

4.2 User Study

In this section, the performance of the travel planning robot system is evaluated in detail, including key metrics such as response time, accuracy, and user satisfaction.

4.2.1 Response Time. To evaluate the response time of the system, we conducted several tests under different network environments and hardware configurations. Specifically, we recorded the time interval between each time a user submitted a question and when the system generated an answer and returned it. The test results show that the average response time of the system stays between 0.1 and 0.5 seconds with a standard deviation of 0.15 seconds. This means that users can get a response almost immediately after submitting a question, thus ensuring a smooth and immediate user experience.

Tests were conducted under a variety of network conditions, including high-speed broadband and mobile networks, as well as a variety of hardware configurations, including high-performance desktops and regular laptops. Through multiple tests and data analysis, we found that the response

time of the system was maintained in a stable and short range regardless of network and hardware conditions.

The fast response time not only improves user satisfaction, but also enhances the utility of the system in travel planning scenarios that require immediate access to information. We achieve the goal of processing user requests and generating accurate answers in a short period of time by optimizing the system architecture and algorithms, e.g., introducing parallel processing and caching mechanisms. This optimization ensures that the system maintains an efficient response time despite high concurrency and complex queries.

4.2.2 Accuracy. In terms of system accuracy, we focused on the performance of the system in understanding user questions and generating relevant answers. We randomly selected a large number of user questions and conducted a detailed evaluation of the system's comprehension and answer accuracy. The scoring criteria included correct question comprehension and answer consistency. The results show that the system is able to correctly recognize the main intent of user questions in 90% of the cases, demonstrating its high level of user intent comprehension. This result indicates that the system performs well in dealing with various types of questions and is able to accurately understand the actual needs of users.

In addition, 95% of the answers generated by the system were consistent with the actual situation. This means that the advice and planned routes provided by the system are highly consistent with the actual needs and situations of the users. The unanimous ratings from the reviewers indicated that the system excelled in content accuracy and practicality, and was able to provide users with reliable travel planning advice.

This high level of accuracy ensured that almost all questions asked by users were answered correctly and practically, greatly enhancing their trust and reliance on the system. This achievement is attributed to the continuous optimization and improvement of our AI model, including the introduction of contextual understanding and semantic analysis techniques, which enable it to maintain a high level of understanding and responsiveness in the face of diverse and complex user input.

4.2.3 User Satisfaction. In order to comprehensively assess user satisfaction with the system, a detailed user satisfaction questionnaire was designed and distributed, and in-depth interviews were conducted. The questionnaire was based on a Likert Scale, covering a wide range of aspects such as overall experience, system responsiveness, answer accuracy, speech synthesis function, usefulness of travel routes and user interface friendliness. Feedback was collected from a total of 20 users.

The questionnaire was designed using a five-point Likert scale ranging from "very dissatisfied" (1 point) to "very satisfied" (5 points). Questions on the questionnaire included, "How satisfied were you with the overall travel robot experience?", "Did the system's responsiveness meet your expectations?", "How accurate was the system in understanding and answering your questions?", "How satisfied are you with the system's speech synthesis capabilities?", "Do the travel routes and information provided by the system meet your actual needs?", "Do you think the design of the user interface is friendly and easy to use?", and "Did you encounter any technical problems or malfunctions during use?"

The results of the survey showed that the majority of participants were satisfied with the use of the Big Language Model interface for travel route planning. Of the 20 users, 90% were satisfied or very satisfied with the overall experience, indicating a high level of user acceptance of the system. 65% of the users felt that the system's responsiveness met or exceeded their expectations, reflecting good real-time performance. 88% of the users were satisfied with the accuracy with which the system understood and answered the questions, indicating a high level of accuracy in handling user queries. In addition, 80% of the users were satisfied with the speech synthesis feature, which

enhanced the realism and interest of the interaction. 75% of the users found the travel routes and information provided by the system to be very helpful in planning their trips. 74% of the users found the interface to be user-friendly and easy to use.

Although most users showed positive attitudes and acceptance of the system, a few participants raised some questions and suggestions for improvement. For example, some users expressed dissatisfaction with the real-time performance of the system and suggested integrating social media applications for authentic travel experiences. This suggests that users want more timely information and richer social elements in the travel planning process. In addition, some users felt that the system's interactive approach was monotonous and might limit the user experience. They would like to have more options and more flexible interaction methods to fulfill their needs.

In the in-depth interviews, we delved into the problems users encountered during use and their feedback on the system. Some users expressed dissatisfaction with the real-time capabilities of the system, expected more instant information, and suggested integrating social media apps such as "Xiaohongshu" to capture real travel experiences. This feedback emphasizes the importance users place on timely information during the travel planning process and their desire to enrich the experience with social elements. On the other hand, some users were not interested in this type of interaction, seeing it as a waste of time, reflecting users' need for variety in interaction modes. Although most users preferred text-based interactions, their expectations of the system's real-time capabilities and functionality remained high. This suggests the need to balance the preferences and needs of different users during the design and development process to provide a more comprehensive and satisfying user experience.

Key points from the user feedback reveal potential directions for technology development and user experience improvement. First, we need to enhance the real-time performance of the system, improve its real-time information processing capability, and integrate social media applications to provide a richer travel experience. Second, multiple interaction methods, such as gesture control and image recognition, are added to meet the needs and preferences of different users. In addition, personalized travel routes and suggestions are provided based on users' historical inquiries and preferences to further enhance user satisfaction.

By comprehensively analyzing the results of user satisfaction questionnaires and interviews, we are able to clarify users' expectations and needs for the system, thus providing strong support for further optimization of the system.

5 DISCUSSION

5.1 Hypothesis Testing and Experimental Conclusions

Based on the research questions, this project proposes the following hypotheses and tests them with our experimental setup:

H1: Different fine-tuning methods (LoRA, Q-LoRA) will have a significant impact on the quality of travel plan generation of large language models (Gemma-2B, Gemma-7B). Our experiments show that fine-tuning methods indeed play a crucial role in the performance of language models. LoRA fine-tuned models generally perform better than Q-LoRA in generating high-quality travel plans, as evidenced by higher quality scores.

Effect of model size on LoRA fine-tuning hypothesis (H2):

H2: The size of the model (2B vs. 7B) will significantly affect the quality of travel plan generation of LoRA fine-tuned models (Gemma-2B, Gemma-7B). The results confirm that model size significantly affects the performance of LoRA fine-tuned models. The 7B model consistently outperformed the 2B model in generating accurate, detailed travel plans.

Effect of model size on Q-LoRA fine-tuning hypothesis (H3):

H3: The size of the model (2B vs. 7B) will significantly affect the quality of travel plan generation of Q-LoRA fine-tuned models (Gemma-2B, Gemma-7B). Contrary to expectations, model size did not significantly affect the performance of the Q-LoRA fine-tuned model. Both the 2B and 7B models showed similar performance levels, suggesting that the efficiency of Q-LoRA may not scale with model size as effectively as LoRA.

User intention and preference analysis hypothesis (H4):

H4: LLM can accurately capture and analyze core information such as travel intentions, attraction preferences, and budget range input by users. Our system demonstrates high accuracy in understanding and responding to user queries, effectively capturing essential information about travel preferences and restrictions.

Real-time data integration hypothesis (H5):

H5: LLM can effectively collect and analyze real-time weather, traffic, attraction status and other data to optimize travel route planning. The successful integration of real-time data enables the system to provide dynamically updated travel plans. This feature significantly enhances the relevance and usefulness of generated itineraries.

Evaluate system hypothesis (H6):

H6: A scientific and effective evaluation system can be established to measure the quality of travel route planning generated by LLM and collect user feedback for continuous optimization. We have developed a comprehensive assessment framework to evaluate the quality of travel plans based on various criteria. The system proved highly effective at gathering user feedback and guiding iterative improvements to the model.

System performance hypothesis (H7):

H7: The travel route planning service based on LLM can maintain stability, responsiveness and scalability in high concurrency scenarios. Performance tests under simulated high concurrency conditions show that the system maintains high response speed and stability, verifying its scalability and robustness.

5.2 Data analysis

In this study, we conducted detailed data analysis of multiple AI models' responses to evaluate their performance and accuracy. The following is the specific process and results of data analysis.

For 4 different AI models, we generate a table containing question prompts and corresponding answers for each model. In order to increase the randomness and diversity of the questions, each question prompt is about travel route planning, and the question pattern is constantly changing.

10 rows of data were collected, which were 10 different question prompts (prompts). Each prompt was answered by the GPT model according to predefined standards. The scores of the model answers are divided into 1-3 points from low to high, corresponding to low, medium and high quality respectively.

We store the collected data into tables according to prompts and responses, and then record the quality of each answer based on the GPT score. The quality of the scores is divided into low (1 point), medium (2 points), and high (3 points), and these scores are integrated into the final data table.

The scoring data is shown in the following table :

After each model generates a form, rate the quality of the responses for each form. Scoring ranges from 0 to 30, with each answer to 10 questions worth 1-3 points [32]. After the scoring results are aggregated, we conduct an analysis to compare the overall performance of each model.

According to the research questions, we divided the models into two groups and used LoRA and Q-LoRA for fine-tuning respectively to evaluate the impact of different fine-tuning methods on large-scale language models (such as gemma-2b) [9]. Data analysis includes: first checking the

id	prompt	response	quality
1	Plan a travel route for me	to the airport	1
2	travel routes from Beijing to Shanghai	xxx	2
3	Los Angeles travel routes	xxx	3

Table 5. Prompts, responses, and quality ratings for travel route planning

normality of the two sets of data, if the data are normally distributed, then perform a T test; if not, perform a Wilcoxon signed rank test. The Shapiro-Wilk test results showed that both sets of data did not conform to the normal distribution ($p < 0.05$), so the Wilcoxon signed rank test was used.

Shapiro-Wilk test for LoRA: ShapiroResult(statistic=0.8318431377410889, pvalue=0.03521495312452316) Shapiro-Wilk test for Q-LoRA: ShapiroResult(statistic=0.640485405921936, pvalue=0.00016868000966496766) The results of the Wilcoxon signed rank test showed that the performance of the LoRA group was significantly higher than that of the Q-LoRA group (p-value=0.01694742672344633) [2].

Wilcoxon signed-rank test statistic: 15.0, p-value: 0.01694742672344633 In conclusion, data analysis results show that when the model size is 2B, the model using the LoRA fine-tuning method is significantly better than the model using the Q-LoRA fine-tuning method in answer quality. This shows that the LoRA method has a better effect in optimizing the performance of large-scale language models, and provides an important reference for the selection of future model fine-tuning methods [4].

5.3 Normality test

When the model size is 2B, in order to choose the appropriate statistical test method, we first performed a normality test on the two sets of data. The Shapiro-Wilk test was used to determine whether the data conformed to normal distribution [34]. The test results show that both sets of data do not conform to the normal distribution ($p < 0.05$), as follows: Shapiro-Wilk test for LoRA: ShapiroResult(statistic=0.8318431377410889, pvalue=0.03521495312452316) Shapiro-Wilk test for Q-LoRA: ShapiroResult(statistic=0.640485405921936, pvalue=0.00016868000966496766) LoRA follows normal distribution: False Q-LoRA follows normal distribution: False Since both sets of data do not conform to the normal distribution, we chose the non-parametric test method, namely the Wilcoxon signed rank test, to compare the differences between the two sets of data, LoRA and Q-LoRA. The results of the Wilcoxon signed rank test showed that the performance of the LoRA group was significantly higher than that of the Q-LoRA group. The specific results are as follows: Wilcoxon signed-rank test statistic: 15.0, p-value: 0.01694742672344633 In conclusion, data analysis results show that when the model size is 2B, the model using the LoRA fine-tuning method is significantly better than the model using the Q-LoRA fine-tuning method in answer quality. This finding shows that the LoRA method is more effective in optimizing the performance of large-scale language models and provides an important reference for the selection of future model fine-tuning methods.

When the model size is 7B, a normality test was performed on the two sets of data. The Shapiro-Wilk test was used to determine whether the data conformed to normal distribution [30].

The test results showed that both sets of data did not conform to the normal distribution ($p < 0.05$) The non-parametric test method, namely the Wilcoxon signed rank test, was selected to compare the differences between the two sets of data, LoRA and Q-LoRA.

The results of the Wilcoxon signed rank test showed that the performance of the LoRA group was significantly higher than that of the Q-LoRA group. The specific results are as follows: Wilcoxon signed-rank test statistic: 24.5, p-value: 0.028889785561798602

in conclusion: Data analysis results show that when the model size is 7B, the model using the LoRA fine-tuning method is significantly better than the model using the Q-LoRA fine-tuning method in answer quality. This finding shows that the LoRA method is more effective in optimizing the performance of large-scale language models and provides an important reference for the selection of future model fine-tuning methods [37].

Since both sets of data do not conform to the normal distribution, we chose the non-parametric test method, namely the Wilcoxon signed rank test, to compare the differences between the two sets of data, LoRA and Q-LoRA.

The results of the Wilcoxon signed rank test show that under the LoRA fine-tuning method, the model size has a significant impact on the performance score. The specific results are as follows:

Wilcoxon signed-rank test for 2B vs 7B with LoRA: Wilcoxon signed-rank test statistic: 15.0, p-value: 0.016947426762344633 Under the Q-LoRA fine-tuning method, the model size has no significant impact on the performance score [38]. The specific results are as follows:

Wilcoxon signed-rank test for 2B vs 7B with Q-LoRA: Wilcoxon signed-rank test statistic: 24.5, p-value: 0.028889785561798602

Data analysis results show that in the comparison of model sizes of 2B and 7B, the performance score of the model using the LoRA fine-tuning method is significantly higher than that of the 2B model. This shows that model size has a significant impact on the performance of the LoRA method. However, when using the Q-LoRA fine-tuning method, there is no significant size relationship between the performance scores of the 2B and 7B models, indicating that the model size has little impact on the performance of the Q-LoRA method.

5.4 Challenges and difficulties

Although there have been many achievements in the field of HRI, there are still many challenges and difficulties: Natural Language Understanding: Although AI models have made great progress in natural language processing, it is still challenging to fully understand the user's intent and context. Especially when facing complex and diverse user inputs, the accuracy and robustness of the model need to be further improved [42].

Real-time response: Although our system has achieved short response times (0.1 to 0.5 seconds), maintaining this performance in more complex scenarios remains a challenge. Especially when large amounts of data and complex logic need to be processed, the system's response speed may be affected.

Emotion and tone recognition: In addition to understanding the language content, the bot also needs to recognize the user's emotion and tone to provide a more humane response. Research in this area is still in its infancy, and existing technology needs to be improved in accuracy and reliability.

Multimodal interaction: Achieving multimodal (voice, gesture, expression, etc.) interaction is an important goal of HRI, but fusing information from different modalities and responding in real time is a complex task. More advanced algorithms and system architectures need to be developed to process this information.

User personalization: How to collect and utilize users' personalized data and provide users with customized services while ensuring privacy and data security is another issue that needs to be solved urgently. A balance needs to be found between algorithm design and data management.

Long-term interaction: Ensuring that robots show consistency and continuity in long-term interactions with users is an important research direction. The current system performs well in short-term interactions, but its learning and adaptability in long-term interactions still needs to be improved.

By addressing these challenges, the field of human-computer interaction will be able to further develop and provide a more intelligent and humane interactive experience. Future research should continue to focus on these difficulties and promote the widespread application and in-depth development of HRI technology in various application scenarios.

5.5 Design principles

When designing the journey planning bot, we followed several key principles to ensure that the system was not only functional but also a great user experience.

User-centered design: We always put user needs and experience first, and strive to be simple, intuitive and easy to use from interface design to interaction process. Users can interact with the robot through text input or voice input. This diverse interaction method greatly improves the convenience and accessibility of the system [46].

Real-time and efficient: The system achieves a response time of 0.1 to 0.5 seconds, ensuring that users can get feedback almost instantly. This efficiency not only improves user satisfaction, but also enhances the practicality of the system, especially in travel scenarios where quick access to information is required.

Accuracy and relevance: When our AI model understands user questions and provides answers, the accuracy rate reaches more than the percentage of 90, and the proportion that is consistent with the actual situation reaches the percentage of 95. This high level of accuracy and relevance ensures users have access to reliable and useful information, increasing the trustworthiness of the system.

Immersive experience: Through AI speech synthesis technology, we provide users with lifelike voice responses, enhancing the realism and immersion of interactions. This humanized design makes the interaction between users and the robot more natural and pleasant.

5.6 Inspiration for the industry

The journey planning robot we designed provides inspiration and reference for the industry in many aspects:

User-friendliness: Emphasis on user-centered design and diverse interaction methods, indicating that a good user experience is the key to successful applications. When designing similar systems, the industry should give priority to users' usage habits and needs to ensure that the system is easy to get started and use.

Performance optimization: By optimizing the system architecture and algorithms to achieve efficient real-time response, it proves that performance is an important factor affecting user satisfaction. Future system design should focus on optimizing response time and processing efficiency to improve overall performance.

Accuracy and reliability: High-accuracy natural language processing and answer generation capabilities are the basis for improving system credibility. The industry should continue to invest in research and development to improve the understanding and answering capabilities of AI models and ensure that the content provided by the system is reliable and relevant.

Emotional interaction: Integrate AI speech synthesis technology to enhance the immersion of interaction and demonstrate the importance of humanized design. The industry can learn from this idea and introduce emotion and voice interaction into more application scenarios to improve the realism and satisfaction of user experience.

Scalability and multimodality: Our design principles emphasize the system's scalability and multimodal interaction capabilities, which provide a reference for future HRI applications. The industry should pay attention to the flexibility and scalability of the system to adapt to the needs of different users and scenarios.

By following these principles and drawing on our design experience, the industry can develop more intelligent and user-friendly HRI systems, promoting the widespread application and progress of technology in various fields.

6 CONCLUSION

In conclusion, this project has demonstrated the potential of integrating advanced large language models (LLMs) with sophisticated fine-tuning techniques within a simulated human-robot interaction framework to create an effective and user-friendly travel planning assistant. Our system, developed using the Unity game engine and powered by the Gemma-2B and Gemma-7B models fine-tuned with both LoRA and Q-LoRA techniques, has shown promising results in delivering personalized and accurate travel recommendations. This final section synthesizes our findings, discusses the broader implications, addresses limitations, and outlines future research directions.

Our experiments revealed several critical insights into the effects of fine-tuning methods and model size on the performance of LLMs in travel planning tasks. The fine-tuning process, particularly with the LoRA technique, significantly enhanced the models' ability to generate coherent, relevant, and practical travel plans. Specifically, the LoRA fine-tuned models consistently outperformed their Q-LoRA counterparts, indicating that low-rank adaptation is more effective for this domain. Furthermore, our results highlighted the importance of model size, with the larger Gemma-7B models delivering superior performance compared to the smaller Gemma-2B models, especially when fine-tuned with LoRA. The user studies conducted as part of this project provided valuable feedback on the system's real-time responsiveness, accuracy, and overall user satisfaction. Users appreciated the quick response times, high accuracy of travel recommendations, and the immersive experience provided by the AI-driven speech synthesis. However, the feedback also pointed to areas for improvement, such as integrating more real-time data sources and expanding the interaction modalities to include more natural and intuitive interfaces.

The successful integration of LLMs into a virtual environment for travel planning applications has several broader implications. Firstly, it demonstrates the feasibility of using advanced AI technologies to enhance user experiences in complex decision-making tasks. The ability to generate personalized and detailed travel plans based on user preferences and real-time data can significantly reduce the time and effort required for trip planning, making travel more accessible and enjoyable. Secondly, the project showcases the potential of fine-tuning techniques in adapting large pre-trained models to specific domains. The superior performance of LoRA fine-tuned models suggests that similar approaches could be applied to other specialized applications, from healthcare to customer service, where domain-specific knowledge and personalized interactions are crucial. Thirdly, the use of a 3D virtual environment powered by Unity highlights the importance of immersive and interactive interfaces in AI applications. By combining visual, textual, and auditory elements, we can create more engaging and effective user experiences. This approach could be extended to other domains where visualizing complex information and interacting with AI in a natural manner is beneficial.

Despite the positive outcomes, our project also faced several limitations that need to be addressed in future work. One major limitation was the computational constraints, particularly during the fine-tuning process. The hardware limitations, especially the limited GPU memory, significantly extended the training time for the larger models. This constraint hindered our ability to conduct extensive fine-tuning iterations and test various configurations comprehensively. Another limitation was the scope of the dataset used for fine-tuning. While our dataset was tailored for travel planning, it may not have covered the full range of possible user queries and scenarios. Expanding the dataset to include more diverse and comprehensive travel-related conversations could further improve the models' performance and robustness. Additionally, the user interface and interaction modalities,

although effective, were relatively basic. The reliance on text and voice inputs, while convenient, may not cater to all user preferences. Future enhancements could include gesture recognition, augmented reality (AR) interfaces, and more sophisticated dialogue management systems to provide a richer and more flexible user experience.

Building on the findings and limitations of this project, several future research directions are proposed. Future research should explore ways to optimize the fine-tuning process to reduce computational demands. This could involve investigating more efficient algorithms for model adaptation, utilizing more powerful hardware, or leveraging cloud-based resources to enable parallel processing. Techniques such as model pruning, knowledge distillation, and parameter-efficient fine-tuning could be explored to achieve better performance with fewer resources. To improve the robustness and versatility of the travel planning assistant, expanding the dataset to include a wider range of travel scenarios, languages, and cultural contexts is essential. This would involve collecting more diverse user queries and responses, incorporating data from various travel-related sources, and potentially collaborating with travel agencies or platforms to obtain real-world data. Incorporating more real-time data sources such as weather updates, flight statuses, local events, and social media trends can enhance the relevance and accuracy of the travel recommendations. Developing APIs and pipelines to seamlessly integrate these data sources into the system will be crucial for providing up-to-date and dynamic travel plans. Future developments should focus on enhancing the user interaction modalities to make the system more intuitive and engaging. This could involve integrating gesture recognition, AR interfaces, and multimodal interactions that combine voice, text, and visual inputs. Improving the natural language understanding capabilities of the system to handle more complex and nuanced queries will also be important. To better understand the long-term impact and effectiveness of the travel planning assistant, conducting longitudinal studies with a diverse user base is recommended. These studies would track user interactions over extended periods, gather feedback on the system's utility and satisfaction, and identify areas for continuous improvement. Insights gained from such studies could inform the iterative development of the system and ensure it meets evolving user needs. As with any AI-driven system, addressing ethical and privacy concerns is paramount. Future research should focus on developing transparent and explainable AI models, ensuring user data privacy and security, and implementing robust measures to prevent bias and ensure fairness in the travel recommendations. Engaging with stakeholders, including users, ethicists, and regulatory bodies, will be essential to navigate these challenges effectively. The methodologies and insights gained from this project can be extended to other domains where personalized and interactive AI assistants can provide significant value. For example, healthcare, education, customer service, and financial planning are areas where similar approaches could enhance decision-making and user experience. Future research could explore adapting the system to these domains and evaluating its impact. Collaborating with industry partners, such as travel agencies, tech companies, and AI research institutions, can provide valuable resources, expertise, and real-world data to enhance the system. These collaborations could facilitate the deployment of the travel planning assistant in commercial applications, enabling broader user adoption and impact.

The development of a large language model-powered travel planning assistant within a virtual human-robot interaction framework represents a significant advancement in AI-driven decision support systems. Our findings underscore the effectiveness of fine-tuning techniques, particularly LoRA, in enhancing model performance for domain-specific applications. The project has demonstrated the feasibility of combining advanced AI models with immersive virtual environments to create engaging and useful tools for users. While there are limitations and challenges to address, the insights gained from this project provide a solid foundation for future research and development. By continuing to refine the models, expand the datasets, integrate real-time data, and enhance

user interactions, we can further improve the system's utility and user satisfaction. Ultimately, this project contributes to the growing body of knowledge on the application of large language models in interactive and personalized AI systems. It highlights the potential of these technologies to transform how we interact with digital assistants and make complex decisions. As we continue to advance the state of the art in this field, we can look forward to more intelligent, responsive, and user-friendly AI systems that enhance our daily lives and empower us to achieve our goals with greater ease and efficiency.

7 CONTRIBUTION

The successful completion of the project and the report was a result of our combined efforts, with each team member contributing to different aspects of the work. Here is a detailed breakdown of the contributions made by each team member:

Zhongyang.Hu and Runze.Yang served as the project lead, overseeing the entire project lifecycle from initial concept to final implementation. They were responsible for defining the project's scope, setting milestones, and ensuring timely progress. As the system architect, Zhongyang.Hu designed the overall system architecture, integrating the Unity game engine with the Django backend and the large language models. Runze.Yang coordinated the team's efforts, facilitated communication between members, and managed the project's resources. He also played a crucial role in the design and implementation of the virtual environment and user interface, ensuring that the system was intuitive and user-friendly.

Zhongyang.Hu focused on the AI components of the project, particularly the fine-tuning of the large language models (Gemma-2B and Gemma-7B) using LoRA and Q-LoRA techniques. He was responsible for curating the dataset, setting up the training environment, and executing the fine-tuning processes. Zhongyang.Hu's expertise in natural language processing and machine learning was instrumental in optimizing the models' performance. He also contributed to the evaluation of the models, analyzing their responses and refining the training techniques to enhance accuracy and relevance. He documented the technical aspects of the AI components in the final report, providing detailed insights into the methodologies and results.

Runze.Yang and Rong.Huang was responsible for the Unity development, creating the 3D virtual environment and designing the interactive elements. They developed the NPC (non-playable character) travel guide, implementing the dialogue system that allowed users to interact with the AI models through text and voice inputs. Runze.Yang also integrated the speech synthesis technology, enhancing the realism and immersion of the user experience. His work on the user interface ensured that the system was engaging and easy to navigate. Runze.Yang and Rong.Huang's contributions to the report included detailed descriptions of the Unity development process, the design of the virtual environment, and the implementation of user interaction mechanisms.

Jinhong.Li and Qinchuan.Zhu were in charge of developing the Django backend server and integrating it with the Unity environment. They wrote the scripts for loading, fine-tuning, and invoking the large language models. They also developed the API endpoints that facilitated communication between the Unity front end and the Django backend. Their work ensured that user queries were efficiently processed and that travel plans were generated and displayed seamlessly. They meticulously documented the backend development process and data integration strategies in the report.

Qinchuan.Zhu, Jinhong.Li, and Rong.Huang played a key role in the evaluation and testing phases of the project. They designed and conducted user studies to assess the system's performance, accuracy, and user satisfaction. They gathered and analyzed user feedback, providing insights that were crucial for refining the system. They also coordinated the writing of the report, ensuring that

it accurately reflected the project's objectives, methods, and findings. Their attention to detail and organizational skills were essential in completing the project documentation.

Throughout the project, all team members engaged in regular discussions and planning sessions to align on goals and ensure a cohesive approach. We collectively brainstormed ideas, resolved challenges, and made critical decisions regarding the project's direction. The collaborative spirit and continuous communication among team members were key factors in the project's success. By leveraging our diverse skills and working together effectively, we were able to develop a robust and innovative travel planning robot. The project's successful implementation and the comprehensive report are testaments to our collective effort and dedication. Each member's contributions were integral to achieving our objectives and delivering a high-quality final product.

REFERENCES

[1] A. Aghajanyan, L. Yu, A. Conneau, W.-N. Hsu, K. Hambardzumyan, S. Zhang, S. Roller, N. Goyal, O. Levy, and L. Zettlemoyer. 2023. Scaling Laws for Generative Mixed-Modal Language Models. In *Proceedings of the 40th International Conference on Machine Learning*. 265–279. https://proceedings.mlr.press/v202/aghajanyan23a.html

[2] L. Bahl, F. Jelinek, and R. Mercer. 1983. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2 (1983), 179–190.

[3] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, N. Joseph, S. Kadavath, J. Kernion, T. Conerly, S. El-Showk, N. Elhage, Z. Hatfield-Dodds, D. Hernandez, T. Hume, and J. Kaplan. 2022. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *arXiv* arXiv:2204.05862 (2022). http://arxiv.org/abs/2204.05862

[4] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326* (2015).

[5] et al. Brown. 2020. Few-shot Learning for Language Models. *Journal Name* (2020).

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, and D. Amodei. 2020. Language Models are Few-Shot Learners. *arXiv* (2020). http://arxiv.org/abs/2005.14165

[7] J. Bruce and M. Veloso. n.d.. Real-Time Randomized Path Planning for Robot Navigation. http://www.cs.cmu.edu/~mmv/papers/BruceTOIS2006.pdf.

[8] Y. Cao and C. S. Lee. 2023. Robot behavior-tree-based task generation with large language models. *arXiv preprint* arXiv:2302.12927 (2023). http://arxiv.org/abs/2302.12927

[9] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. 2017. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems*, Vol. 30. https://proceedings.neurips.cc/paper_files/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html

[10] Garrick Craft. 2023. *Using Large Language Models to Modify Robot Behavior in Response to Natural Language Prompts.* Master's thesis. University of New Hampshire.

[11] M. Dalal, T. Chiruvolu, D. Chaplot, and R. Salakhutdinov. 2024. Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks. *arXiv preprint* arXiv:2405.01534 (2024). http://arxiv.org/abs/2405.01534

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* (2019). http://arxiv.org/abs/1810.04805

[13] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, J. Yi, W. Zhao, X. Wang, Z. Liu, H.-T. Zheng, J. Chen, Y. Liu, J. Tang, J. Li, and M. Sun. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence* 5, 3 (2023), 220–235. https://doi.org/10.1038/s42256-023-00626-4

[14] D. Ganguli, L. Lovitt, J. Kernion, A. Askell, Y. Bai, S. Kadavath, B. Mann, E. Perez, N. Schiefer, K. Ndousse, A. Jones, S. Bowman, A. Chen, T. Conerly, N. DasSarma, D. Drain, N. Elhage, S. El-Showk, S. Fort, and J. Clark. 2022. Red Teaming Language Models to Reduce Harms: Methods, Scaling Behaviors, and Lessons Learned. *arXiv* (2022). http://arxiv.org/abs/2209.07858

[15] Dongge Han, Trevor McInroe, Adam Jelley, Stefano V Albrecht, Peter Bell, and Amos Storkey. 2024. LLM-Personalize: Aligning LLM Planners with Human Preferences via Reinforced Self-Training for Housekeeping Robots. *arXiv preprint arXiv:2404.14285* (2024).

[16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[17] Z. Hu, A. Iscen, C. Sun, K. W. Chang, Y. Sun, D. A. Ross, and A. Fathi. 2023. AVIS: Autonomous visual information seeking with large language models. *arXiv preprint* arXiv:2306.08129 (2023). http://arxiv.org/abs/2306.08129

[18] H. Jin, X. Han, J. Yang, Z. Jiang, Z. Liu, C.-Y. Chang, H. Chen, and X. Hu. 2024. LLM Maybe LongLM: Self-Extend LLM Context Window Without Tuning. *arXiv* (2024). http://arxiv.org/abs/2401.01325

[19] S. S. Kannan, V. L. N. Venkatesh, and B.-C. Min. 2023. SMART-LLM: Smart Multi-Agent Robot Task Planning using Large Language Models. *arXiv* (2023). http://arxiv.org/abs/2309.10062

[20] W. B. Knox, P. Stone, and C. Breazeal. 2013. Training a Robot via Human Feedback: A Case Study. In *Social Robotics*, G. Herrmann, M. J. Pearson, A. Lenz, P. Bremner, A. Spiers, and U. Leonards (Eds.). Springer International Publishing, 460–470. https://doi.org/10.1007/978-3-319-02675-6_46

[21] Y. Li, S. Dai, Y. Shi, L. Zhao, and M. Ding. 2019. Navigation Simulation of a Mecanum Wheel Mobile Robot Based on an Improved A* Algorithm in Unity3D. *Sensors* 19, 13 (2019), 2976. https://doi.org/10.3390/s19132976

[22] et al. Liu. 2020. Prompt Engineering for Large Language Models: A Pilot Study. *Journal Name* (2020).

[23] Peng Liu, Xin Qi, and Wen Chen. 2019. Recent advancements in contextualized word embeddings: An overview. (2019).

[24] S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen. 2024. DoRA: Weight-Decomposed Low-Rank Adaptation. *arXiv* (2024). http://arxiv.org/abs/2402.09353

[25] Lajanugen Logeswaran, Yao Fu, Moontae Lee, and Honglak Lee. 2022. Few-shot subgoal planning with language models. *arXiv preprint arXiv:2205.14288* (2022).

[26] Sichun Luo, Yuxuan Yao, Haohan Zhao, and Linqi Song. 2024. A Language Model-based Fine-Grained Address Resolution Framework in UAV Delivery System. *IEEE Journal of Selected Topics in Signal Processing* (2024).

[27] Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. 2023. Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*. PMLR, 23301–23320.

[28] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser. 2016. Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems* 86 (2016), 13–28. https://doi.org/10.1016/j.robot.2016.08.001

[29] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781* (2013).

[30] V. Nair and G. E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 807–814.

[31] et al. Nguyen. 2020. Matching Strategies for Query Auto-Completion. *Journal Name* (2020).

[32] V. Pallagani, B. Muppasani, K. Murugesan, F. Rossi, B. Srivastava, L. Horesh, and A. Loreggia. 2023. Understanding the capabilities of large language models for automated planning. *arXiv preprint* arXiv:2305.16151 (2023). http://arxiv.org/abs/2305.16151

[33] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. (2014).

[34] S. Petridis, B. D. Wedin, J. Wexler, M. Pushkarna, A. Donsbach, N. Goyal, and M. Terry. 2024. Constitutionmaker: Interactively critiquing large language models by converting feedback into principles. In *Proceedings of the 29th International Conference on Intelligent User Interfaces*. 853–868.

[35] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. n.d.. Improving Language Understanding by Generative Pre-Training. https://arxiv.org/abs/1810.04805.

[36] et al. Radford. 2019. Better Language Models and Their Implications. *Journal Name* (2019).

[37] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).

[38] Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Xingyu Zeng, and Rui Zhao. 2023. Tptu: Task planning and tool usage of large language model-based ai agents. *arXiv preprint arXiv:2308.03427* (2023).

[39] G. Sarch, Y. Wu, M. J. Tarr, and K. Fragkiadaki. 2023. Open-ended instructable embodied agents with memory-augmented large language models. *arXiv preprint* arXiv:2310.15127 (2023). http://arxiv.org/abs/2310.15127

[40] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv* arXiv:1707.06347 (2017). http://arxiv.org/abs/1707.06347

[41] D. Shah, B. Osiński, B. Ichter, and S. Levine. 2023. LM-Nav: Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action. In *Proceedings of The 6th Conference on Robot Learning*. 492–504. https://proceedings.mlr.press/v205/shah23b.html

[42] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).

[43] Vikram Shree, Beatriz Asfora, Rachel Zheng, Samantha Hong, Jacopo Banfi, and Mark Campbell. 2021. Exploiting natural language for efficient risk-aware multi-robot sar planning. *IEEE Robotics and Automation Letters* 6, 2 (2021), 3152–3159.

[44] Tom Silver, Varun Hariprasad, Reece S Shuttleworth, Nishanth Kumar, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2022. PDDL planning with pretrained large language models. In *NeurIPS 2022 foundation models for decision making*

workshop.

[45] C. Snell, M. Yang, J. Fu, Y. Su, and S. Levine. 2022. Context-aware language modeling for goal-oriented dialogue systems. *arXiv preprint* arXiv:2204.10198 (2022). http://arxiv.org/abs/2204.10198

[46] E. Strubell, A. Ganesh, and A. McCallum. 2019. Energy and policy considerations for deep learning in NLP. *arXiv preprint arXiv:1906.02243* (2019).

[47] Hugo Touvron, Lilian Martin, Kevin Stone, Pierre Albert, Amjad Almahairi, Yatharth Babaei, Nikita Bashlykov, Dhruv Batra, Pranay Bhargava, Sushant Bhosale, Daniel Bikel, Lior Blecher, Carla C. Ferrer, Mike Chen, Gemma Cucurull, David Esiobu, Joao Fernandes, Julian Fu, William Fu, et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv* arXiv:2307.09288 (2023). http://arxiv.org/abs/2307.09288

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv* arXiv:1706.03762 (2017). https://doi.org/10.48550/arXiv.1706.03762

[49] M. Verma, S. Bhambri, and S. Kambhampati. 2024. Theory of Mind abilities of Large Language Models in Human-Robot Interaction: An Illusion?. In *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 36–45. https://doi.org/10.1145/3610978.3640767

[50] W. Wang, L. Mao, R. Wang, and B.-C. Min. 2024. SRLM: Human-in-Loop Interactive Social Robot Navigation with Large Language Model and Deep Reinforcement Learning. *arXiv* (2024). http://arxiv.org/abs/2403.15648

[51] Xiaohan Wang, Wenguan Wang, Jiayi Shao, and Yi Yang. 2023. Lana: A language-capable navigator for instruction following and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 19048–19058.

[52] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus. 2022. Emergent Abilities of Large Language Models. *arXiv* arXiv:2206.07682 (2022). http://arxiv.org/abs/2206.07682

[53] F. Xie and S. Schwertfeger. 2024. Empowering Robotics with Large Language Models: osmAG Map Comprehension with LLMs. *arXiv preprint* arXiv:2403.08228 (2024). http://arxiv.org/abs/2403.08228

[54] T. Yamazaki, K. Yoshikawa, T. Kawamoto, M. Ohagi, T. Mizumoto, S. Ichimura, Y. Kida, and T. Sato. 2022. Tourist Guidance Robot Based on HyperCLOVA. *arXiv* (2022). http://arxiv.org/abs/2210.10400

[55] Jianhao Yuan, Shuyang Sun, Daniel Omeiza, Bo Zhao, Paul Newman, Lars Kunze, and Matthew Gadd. 2024. RAG-Driver: Generalisable Driving Explanations with Retrieval-Augmented In-Context Learning in Multi-Modal Large Language Model. *arXiv preprint arXiv:2402.10828* (2024).

[56] A. Zeng, X. Liu, Z. Du, Z. Wang, H. Lai, M. Ding, Z. Yang, Y. Xu, W. Zheng, X. Xia, W. L. Tam, Z. Ma, Y. Xue, J. Zhai, W. Chen, P. Zhang, Y. Dong, and J. Tang. 2023. GLM-130B: An Open Bilingual Pre-trained Model. *arXiv* arXiv:2210.02414 (2023). http://arxiv.org/abs/2210.02414

[57] et al. Zeng. 2019. Reinforcement Learning for Query Auto-Completion. *Journal Name* (2019).

[58] Jesse Zhang, Jiahui Zhang, Karl Pertsch, Ziyi Liu, Xiang Ren, Minsuk Chang, Shao-Hua Sun, and Joseph J Lim. 2023. Bootstrap your own skills: Learning to solve new tasks with large language model guidance. *arXiv preprint arXiv:2310.10021* (2023).

[59] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, and J.-R. Wen. 2023. A Survey of Large Language Models. *arXiv* arXiv:2303.18223 (2023). http://arxiv.org/abs/2303.18223

[60] Z. Zhao, W. S. Lee, and D. Hsu. n.d.. Large Language Models as Commonsense Knowledge for Large-Scale Task Planning. https://arxiv.org/abs/2306.03148.