Zhejiang University / University of Illinois at Urbana-Champaign Institute

Senior Design Individual Report

CAMPUS TOUR GUIDE BY AI-POWERED AUTONOMOUS SYSTEM

By

Author Name: Hao Ren
Student ID: 3200110807

Individual Report for Senior Design, Spring 2024
Major: Electrical Engineering
Grade: 2020
Supervisor: Simon Hu
TA: Xinlong Huang

Project No. 21

Senior Design Individual Report Checklist

Notes: Please filled out by the supervisor after the defense.

Author Name		Student ID		Supervisor	
Major & Grade		Tel.		Professional title of Supervisor	
Report title				Grade assessment	

Inspection Content	Evaluation (√)	
	A	**B**
1. Title Page (using a unified title page, blue recommended)		
2. Commitment Statement		
3. Acknowledgement		
4. Abstract (150 words or less)		
5. Contents (with corresponding page numbers)		
6. Main body (page numbers start from this point)		
7. Reference		
8. Appendix (as needed)		
9. Author Resume		
10. Senior Design Report Task Assignment		
11. Senior Design Report Assessment Form		

Inspector: Inspection Date:

Senior Design Individual Report
Commitment Statement

1. I solemnly promise that the Senior Design Individual Report submitted was completed in strict accordance with the relevant regulations of the institute and university under the guidance of the supervisor.

2. In my Senior Design Individual Report, except where specifically and explicitly indicated, the report does not contain results that have been published or written by others.

3. Any contribution to this report made by the teammates with whom I worked has been clearly stated in the report and acknowledged.

4. I promise that I have not falsified data or committed other similar acts in the completion of the Senior Design Individual Report.

5. If there is any infringement of intellectual property rights or breach of academic integrity in this Senior Design Individual Report, I shall bear the corresponding legal responsibility.

6. I fully understand that Zhejiang University has the right to retain and send copies and electronic records of this report to relevant departments or institutions. I authorize Zhejiang University to compile all or part of the contents of this report into a relevant database for retrieval and dissemination, and can use photocopying, microprinting or scanning and other means of reproduction to preserve and archive the report.

Author: Date:

Supervisor: Date:

Acknowledgement

I'm deeply thankful to my parents, friends and colleagues who have worked with me along the way.

I appreciate the effort my parents paid to raise me and support me through college. College is impossible for me without them.

I also appreciate friends that were and are by my side along the way, as well as the help I gained from this project's supevisor Dr. Simon Hu, and the TA Xinlong Huang. I'm deeply thankful to work with the team for the project, including Weiang Wang, Yuntong Gu and Xuanbo Jin.

Special thanks to the 16 other members in my high school OI team. It's impossible to be where I am without their support. We are there for each other over 8 years which is about one half of my life. We've been through suffering and losses, yet we made it so far. My sincerest wish is that we can continue being there for each other today, tomorrow, and in the future.

Special thanks to my colleagues and my advisor Dr. Kang at Microsoft, who teaches me so much in senior year. This simple AI agent system is impossible without my research experience in his group.

Also, huge thanks to all the people I knew at ZJU-UIUC campus and at UIUC. Many of you played critical roles in my college years.

I won't mention any specific names here, as I think that will be a very unfair thing to do for those who might not be mentioned. And I hope I can make you feel that you are important to me in real life instead of writing it here. Words are pale and paper means far less than the effort people pay in real life to defend their believes and love their loved ones.

Abstract

This report presents a multi-agent AI system designed to provide real-time, interactive guidance in complex environments like universities. Traditional approaches, including human guides and early automated systems, face issues of cost, scalability, and adaptability. Our solution leverages AI agents to efficiently handle data collection, processing, command parsing, and response generation. This system shifts from one guide serving many users to many AI agents serving individual users, optimizing performance and user experience. Key components include robust data management, effective task distribution among agents, and stringent ethical and safety measures. Our findings demonstrate the system's potential to deliver accurate, timely information and actions, ensuring a scalable, versatile, and cost-effective solution for real-time guidance. Future research will focus on enhancing interaction models, scalability, and ethical frameworks to further improve AI-driven interactive systems.

Contents

1 Introduction

1.1 Problem and Solution Overview

Anyone entering a place for the first time, like an university, can be quite challenging. Knowing where you are, how to get to your destination, how to optimize your routes, knowing factors that will influence your routes can be complicated. Having a real-time interactive system that guides people through this process is needed. It has been possible yet not able to scale because it's not open-sourced, and its hardware isn't standardized, and is expensive. The interaction isn't versatile enough to adapt well under the ever-changing applications. A cheap and versatile solution is needed.

1.2 Visual Aid

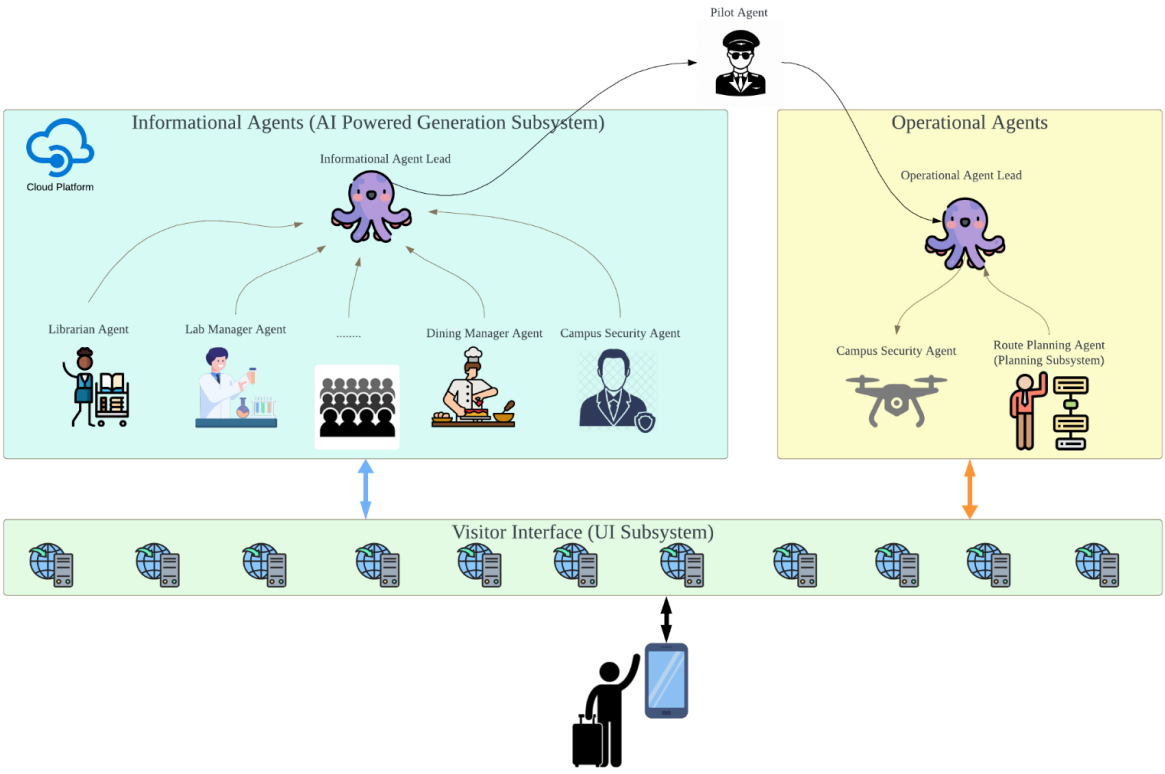

Figure 1: Multi-agent AI operation Visual Illustration

1.3 Motivation

The most traditional paradigm is have human tour guide guiding a group of visitors. Then when the era of electrical and electronics engineering came, engineers started automating this process. They designed specific pipelines to emulate this process. However,

this automation has intrinsic problem with cost, scalability, as well as generality. Currently, when artificial intelligence prevails, inserting AI as a component in the pipeline has significant downsides. Apart from previous problems, most of the components in the pipeline is not AI-powered, wasting computational resources and efficiency. In light of these problems, we completely shift the pipeline design to a Multi-agent AI operation network design, with each of its component AI powered. It can scale easily as all agents have simple and general interfaces. The average cost is also amortised as the number of agents increase. Most importantly, the paradigm of all the down stream applications, including campus tour guide, is shifted from 1 guide leading 100 visitors to 100 guides leading 1 visitors, letting users to these applications fully exploit the power of AI.

1.4 High-level requirements list

- The AI agents network system should be **responsive**. It should respond to user's request appropriately. It should give appropriate guidance to user in both **informational and operational** ways.
- The **interface** must be clean and useful. It should gives user easy access to the service.
- The system must distribute and **map** the request to the correct agent who is most useful in a certain service and is able to merge and **reduce** the response from many agents into one organized response. The choice of agent must be optimized to ensure the best holistic accuracy.

1.5 Single Agent Archictecture

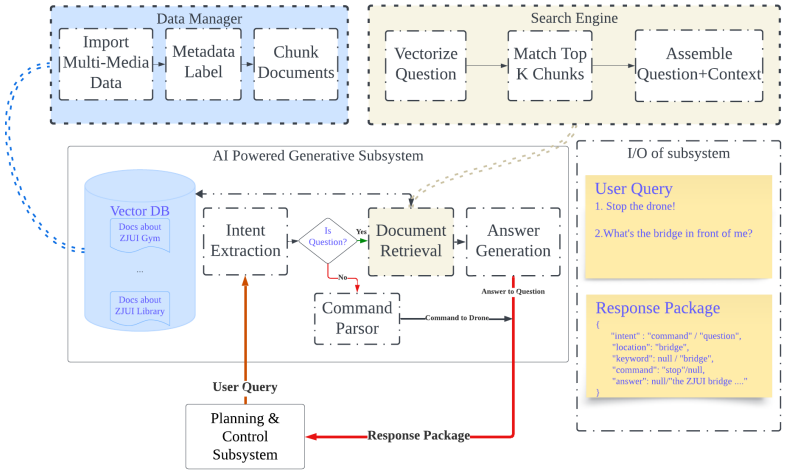

Figure 2: Architect of each agent

2 Literature Review

AI agents have been with us for many years, and with the rapid development of Large Language Models (LLMs) such as ChatGPT, AI agents have become a hot topic for engineering and research [1]. However, since the single-agent framework has many limitations such as self-inconsistency, hallucinations, and limited knowledge and skills, multi-agent frameworks have become increasingly popular [2].

Originally, there was a chat paradigm where an LLM handled the user's request and provided an answer [3]. The user's request is called a prompt, and the prompts along with their answers form the context of the conversation. The context has a length called context length. Due to the limited attention span of LLMs, this chat paradigm is intrinsically bounded by the context length, typically 16K tokens for GPT-3.5 and up to 128K for the newest GPT-4 model [1], [3]. A common engineering and research practice to increase the context length involves a fusion of retrieval and generation, meaning the context is composed by a search engine that fetches only the relevant pieces of information from a knowledge base to generate new information. This introduces modular designs built on top of the chat paradigm. As these modules become more versatile and their numbers increase over time, people have begun to assemble compositions of modules as human agents. This forms another paradigm for AI generation tasks backed by LLMs, known as AI agents.

The paradigm of AI agents is a framework backed by chat APIs to conduct various tasks. Initially, these agents were informational agents, meaning their abilities were limited to Question-Answer (QA) type tasks [3], [4]. As the accuracy and ability of backend LLMs increased, people began to ground the results of informational agents to operate on different objects. These objects could include commands to computers, such as reboot commands, or commands to different machines, like unmanned aerial vehicles (UAVs) [3].

The introduction of operational agents created new opportunities but also had limitations. Because the design of an agent is targeted toward a specific type of workload, its generality is limited [5]. Additionally, due to the hallucination issues of LLMs, the single-agent approach might require heavy restrictions on the results it generates. To address these problems, multi-agent systems have been developed to increase the diversity of answers and enhance the coverage and versatility of the system [2], [6].

In recent studies, multi-agent systems (MAS) have been identified as a powerful solution to model and solve problems in complex and dynamic environments [2], [5]. They offer an alternative way to design distributed control systems by enabling multiple agents to work together to achieve common goals. The landscape of MAS applications spans various domains, including healthcare, systems engineering, and complex networks [4], [6].

For example, in healthcare, agents can assist in managing patient information and providing decision support [6]. In systems engineering, MAS and complex network theories are combined to address optimization and control challenges [4]. Additionally, design patterns for MAS have been systematically reviewed to identify best practices and trends

in the field [7].

3 Methdology

This section delves into three major aspects of the work carried out:

- **Data-side Tasks:** These tasks encompass all activities related to the collection, processing, and formulation of data. This includes gathering data from various sources, cleaning and organizing the data, and preparing it in a format suitable for use by AI models and agents.
- **Agent-side Tasks:** These tasks involve the generation of insights from the data and the execution of actions based on those insights. This includes the distribution of tasks among different AI agents, the parsing of commands, and the retrieval of relevant information to answer queries.
- **Ethics and Safety:** This involves ensuring that the AI systems operate within ethical guidelines and do not pose safety risks. This includes grounding AI outputs to prevent hallucinations and ensuring the safety of AI actions.

Design Motivation and Design Alternatives

Information is unavailable and unorganized

Data are ubiquitous and exist in various formats, including multimedia. Efficiently collecting data for an entire campus is labor-intensive. However, the greater challenge lies in effectively utilizing this multimedia data and retrieving the most relevant information from an enormous dataset.

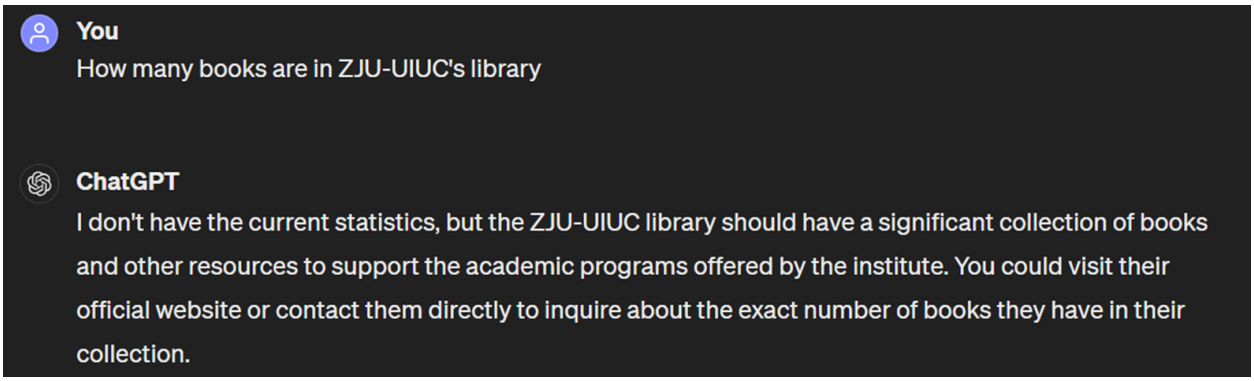

Figure 3: Problems emerged when queries are associated campus related data

Figure 3 illustrates the problem that may arise when a user queries campus-related information. Due to the relatively small amount of relevant data compared to the total dataset size, the agent struggles to retrieve the corresponding data based on the user's intent.

Data path is very long for each query

Unlike applications hosted on a single device, our system is distributed across multiple devices, including two personal computers (PCs) acting as host servers, one web server, a mobile phone, a UAV, and several cloud services hosted by OpenAI. This system spans different countries and continents.

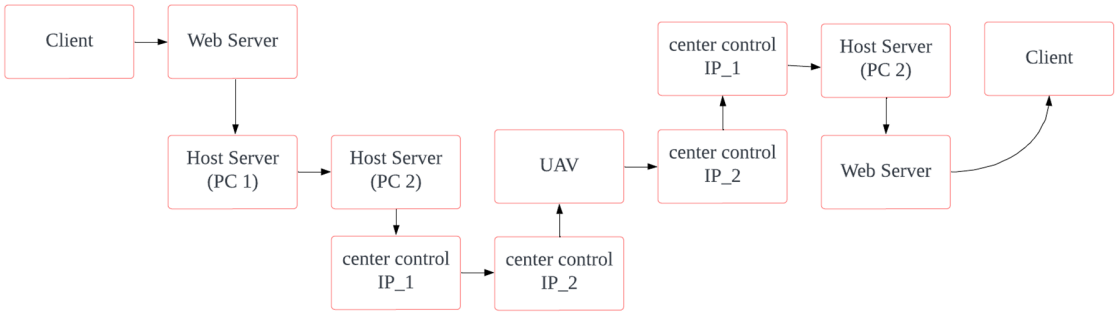

Figure 4: Long datapath for each user's query

The long datapath shown in Figure 4 suggests that it is beneficial to group the devices and abstract the services behind them as subsystems. Based on this idea, we grouped the service managing the web server and mobile phone as the user interface subsystem. We grouped the service managing the UAV and the host server that communicates with it as the planning and control unit. We also grouped the service hosted on OpenAI and the host server that communicates with it as the AI-powered response generation subsystem.

Huge gap between informational and operational

Even with proper grounding, where the OpenAI agent is provided with relevant information and its intended domain of expertise, the generative responses produced by the LLM can be highly unstable and unpredictable.

Another significant observation from our initial testing is that there is a 2 out of 25 chance that GPT could generate a malicious response potentially harmful to the UAV. This requires careful grounding, meaning we must establish a set of restrictions on communication between AI agents and the UAV.

Complete paradigm change needed

Our group proposes a multi-agent network, focusing on generalization while supporting automation and visualization. Imagine a system built for a single-CPU game: with more resources, a person can add another CPU, but the system must be entirely rebuilt because it cannot distribute work across multiple CPUs. The same logic applies to our

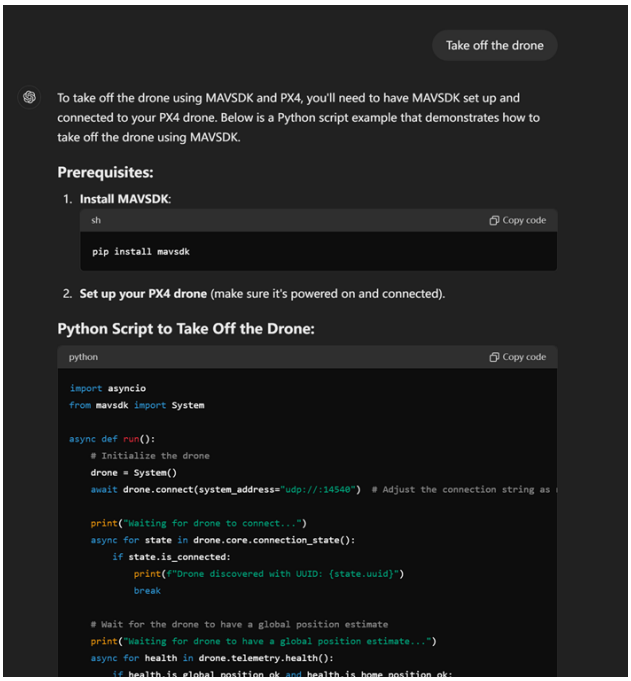

Figure 5: Problems when user types in the command

system. Imagine a system using a UAV to guide users and answer questions about 10 locations. This system would need a complete overhaul if expanded to 10 UAVs or required to answer questions about 1000 locations, as it is designed for 1 UAV and 10 locations. It cannot manage large datasets or multiple UAVs. Our system, however, can handle arbitrarily large amounts of knowledge with a sufficiently large database and can expand to support any number of operational agents, given the resources to build them.

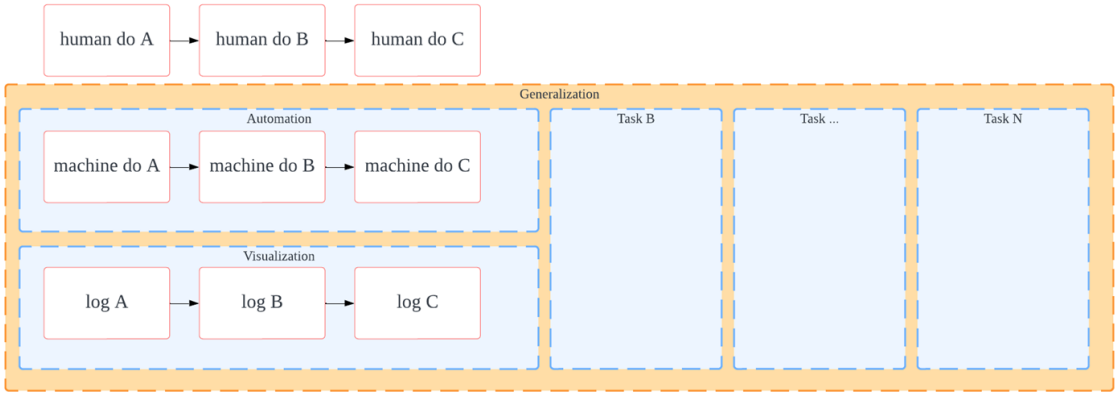

Figure 6: Paradigm change

Data-side Tasks

The first aspect focuses on data-related tasks. While there are many impressive advancements in large language models, they often face a trade-off between generality and accuracy. Specifically, commercial large language models and their frameworks, such as various AI agents, may not perform optimally for specific tasks like providing a campus tour of ZJU-UIUC if the data is not meticulously processed. To tailor these frameworks and agents to this specific downstream task, we have undertaken several key activities:

1. **Mining Insights from Data:** Extracting valuable information from raw data to inform decision-making processes.
2. **Handling Different Types of Data with Appropriate Methods:** Employing various techniques to process different data formats efficiently.
3. **Multi-media Data Gathering, Cleaning, and Tagging:** Collecting data from multiple media sources, cleaning it to remove noise and inconsistencies, and tagging it to facilitate easy retrieval and analysis.
4. **Data Evaluation and Visualization:** Assessing the quality and relevance of the data and visualizing it to derive insights and support data-driven decisions.

Agent-side Tasks

The second aspect concentrates on tasks related to AI agents. To establish a well-structured multi-agent AI operational network, it is essential to perform the following tasks:

1. **Agent Distribution:** Allocating tasks and responsibilities among different AI agents to ensure efficient operation.
2. **Command Parsing:** Interpreting and processing commands issued to the AI system to ensure accurate execution.
3. **Retrieval by Location:** Locating relevant information based on geographical or positional data.
4. **Retrieval by Embedding:** Using embeddings to find and retrieve information that is semantically similar to the input query.
5. **Answer Generation:** Formulating accurate and contextually appropriate responses to user queries based on the retrieved information.

Ethics and Safety

AI systems, particularly those designed for informational and operational tasks, are prone to the issue of AI hallucination, where the system generates responses that are plausible but incorrect or unfounded. Even when an AI generates a well-formulated response, transforming this response into actions for a UAV interface can be hazardous if not rigor-

ously verified. To mitigate these risks and ensure the safety and reliability of AI-generated outputs, efforts have been dedicated to the following grounding activities:

Efforts in AI safety are crucial, as highlighted by various studies and initiatives. OpenAI's approach to AI safety emphasizes iterative deployment and stakeholder engagement to ensure safe and beneficial AI development [8]. The AI safety landscape includes significant contributions from academic institutions like Stanford University, which focuses on building trustworthy AI systems [9]. The SafeAI workshops, organized by AAAI, are platforms for discussing AI safety engineering, ethical design, and regulatory standards [10]. Furthermore, MLCommons' AI Safety Benchmark aims to evaluate and improve the safety of AI systems, addressing challenges like hallucination and robustness [11].

1. **Informational Grounding:** Verifying the accuracy and reliability of the information generated by the AI before it is used or presented.
2. **Operational Grounding:** Ensuring that the actions proposed by the AI are safe, feasible, and aligned with operational guidelines and safety protocols.

3.1 Data-side work

3.1.1 Datpath and IO

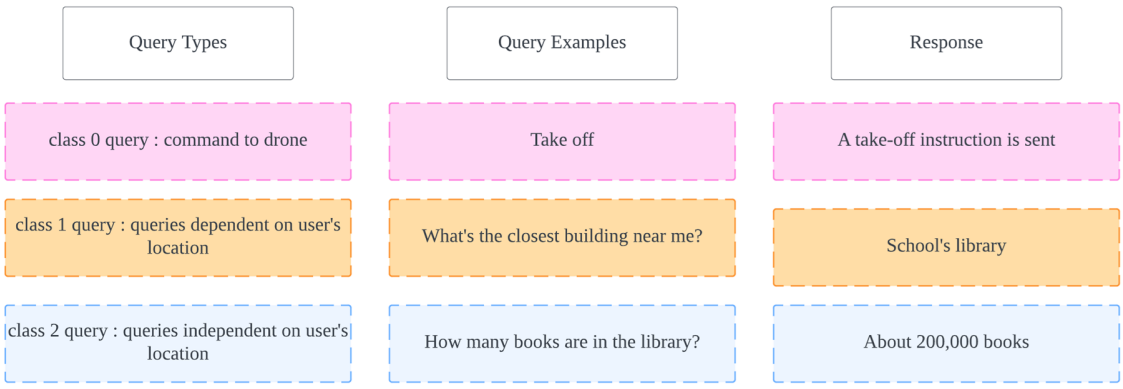

Figure 7: Datapath and IO of the AI subsystem

To illustrate the works done on the data side, it's important to understand the datapath and input/output (IO) of the multi-agent AI system. The diagram provides a clear overview of the query types, examples, and responses within this AI subsystem.

The diagram categorizes queries into three distinct classes based on their nature and dependencies:

Class 0 Query: Command to Drone

Description: These are direct commands issued to the drone, instructing it to perform specific actions.
Example: "Take off"
Response: The system processes this command and sends a take-off instruction to the drone.

Class 1 Query: Queries Dependent on User's Location

Description: These queries require knowledge of the user's current location to provide relevant information.
Example: "What's the closest building near me?"
Response: The system uses the user's location to determine and provide the nearest building, such as the "School's library."

Class 2 Query: Queries Independent of User's Location

Description: These queries do not depend on the user's location and seek general information.
Example: "How many books are in the library?"
Response: The system provides the required information, for instance, "About 200,000 books."

3.1.2 Mining Insights for Different Data

For different data, we have different insights:

- Most data sources are associated with one location, with each location having approximately 10,000 words of data relevant to the ZJU-UIUC campus.
- Uncategorized data (data sources that aren't associated with any locations or are associated with too many locations) tend to be on average ten times longer than categorizable data.
- Certain types of data, such as map-related data and instruction sets, are consistently used for specific types of queries.

The diagram above illustrates the data retrieval process backed by a search engine, highlighting the handling of different query types. For class 1 and 2 queries, the system retrieves location-specific data, general information, and maps. For class 0 queries, it retrieves and sends instruction sets to the drone.

3.1.3 Handle Different Types of Data with Different Methods

Handling different types of data efficiently requires tailored methods. Here are some strategies we employ:

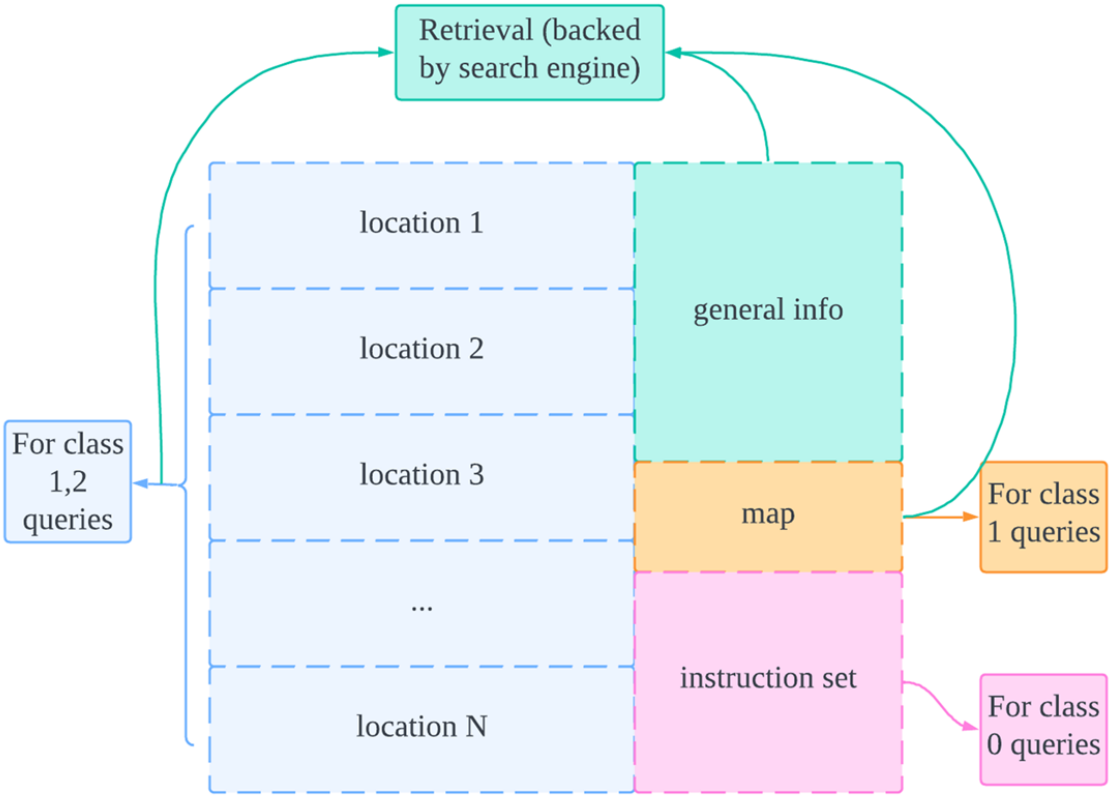

Figure 8: Handle different data differently

- **Categorization and Tagging:** Data is categorized based on its relevance to specific locations or topics. This ensures that queries can be matched with the most pertinent information quickly.
- **Data Cleaning and Preprocessing:** We implement robust data cleaning techniques to remove noise and inconsistencies from the raw data. This preprocessing step is crucial for maintaining the accuracy and reliability of the AI system.
- **Multi-source Integration:** Information from various sources is integrated to form a comprehensive dataset. This involves merging data from text files, maps, instruction sets, and other multimedia sources.
- **Query-based Filtering:** Depending on the type of query (class 0, 1, or 2), different filters are applied to retrieve the most relevant data. For instance, map data is prioritized for location-based queries, while instruction sets are prioritized for drone commands.

Each method ensures that the data is not only comprehensive but also easily accessible and relevant to the user's query. The integration of these methods into the AI system en-

hances its ability to provide accurate and contextually appropriate responses, improving the overall user experience and operational efficiency.

3.1.4 Multi-media data gathering, cleaning and tagging

The GPT model does not have the ability to answer questions related to ZJU-UIUC. It failed to answer the 100 testing questions completely. To supply pertinent information regarding users' queries, we need a set of data tailored for our application. The data we will use are in multiple forms as shown in the figure below. These forms include digital PDFs and paper-based materials placed at various locations within the campus. We utilize several methods to handle these different forms of information.

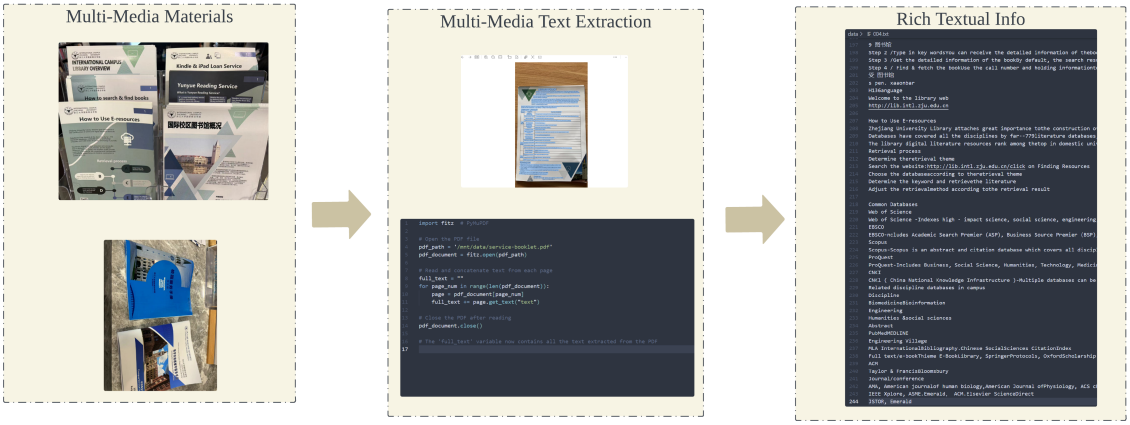

Figure 9: Workflow of Data Collection

1. **OCR Tool Backed by WeChat:** For paper-based materials, we use an Optical Character Recognition (OCR) tool integrated within WeChat. This tool allows us to scan physical documents and convert them into digital text.
2. **Python Tools to Parse PDFs:** For digital PDFs, we employ Python libraries, such as PyMuPDF, to extract text from PDF files. This method ensures that we can handle a large volume of documents efficiently.

The code for parsing multi-media material can be found in this directory.

The process of converting these diverse data sources into uniform textual information involves several steps. Initially, multi-media materials are gathered, which include brochures, posters, and other paper-based information. These materials are then subjected to text extraction processes using the mentioned tools. Finally, the extracted text is cleaned and tagged appropriately to ensure its relevance and accuracy.

This transformation allows our AI agent to access and utilize a comprehensive database of information, enabling it to answer user queries with precise and varied responses. The

goal is to enrich the knowledge base of the AI system, ensuring that it can handle a wide range of questions related to the ZJU-UIUC campus with ease and accuracy.

3.2 Agent-side work

The design of an agent enables it to complete two types of commands: Question or Command. Each agent is responsible for a specific type of tasks, which can be either informational and operational. Also, we have a leading agent in both informational and operational sub-network. We will first introduce the design of the single agent as well as related works. And then we will go through the part where we link different agents to form a functioning multi-agent network.

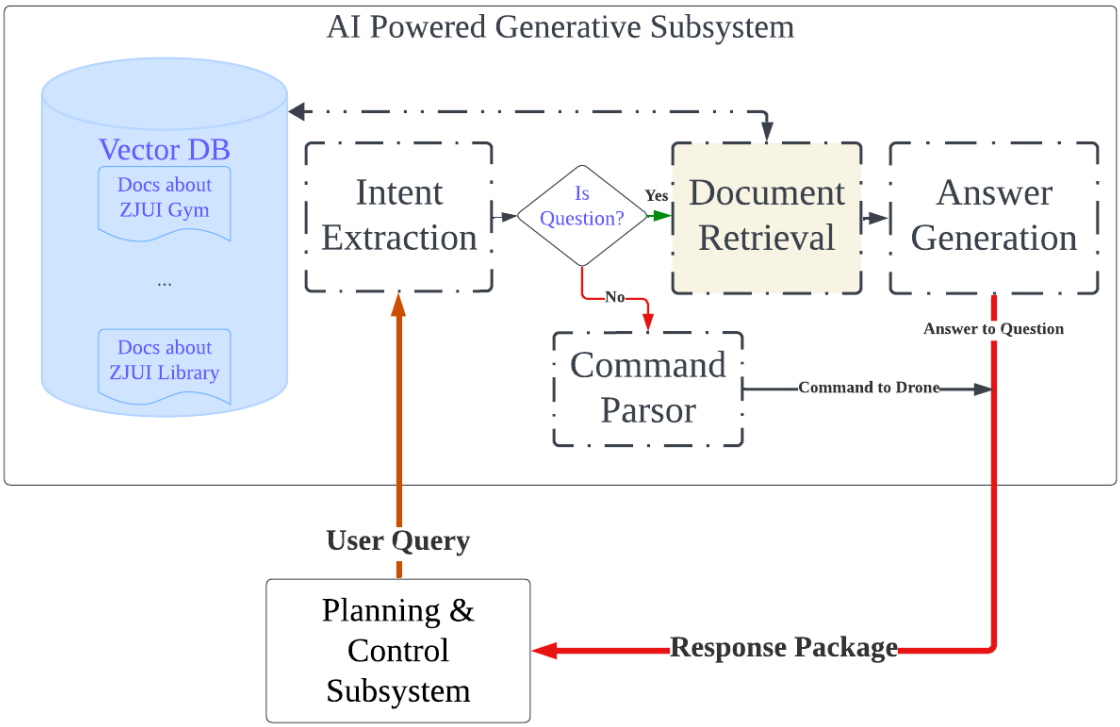

Figure 10: High level design of the agent

3.2.1 Notation and Data Structure

3.2.2 Single-agent Design

The system can be broken into the following parts:

- Vector DB storing related campus material

Table 1: Input and Output of the AI-Powered Response Generation Subsystem

Field Name	Type	Meaning
User Query	Input	Campus related query or a command to drone
User Location	Input	Current GPS location of user
Answer	Output	Answer to user's question
Command	Output	Parsed output Command to the drone

Table 2: Vector DB Data Store Table

Field Name	Type	Meaning
d_id	INTEGER	Indexing of the entries
location	STRING	Location of the described place
outlook	STRING	Appearance of the location
keyword	STRING	keyword of the description
description	STRING	a text describing related info about a location in ZJUI campus

- Command Parsing module which converts the user's intent to formulated command.
- Intent Extraction module extracting user's intent.
- Search Engine module search for related entries in Vector DB
- Answer Generation

The network contains an intent identification module, to parse the intent. It relays the request to different portion of network. For example, the class 0 query (queries related to command) are routed to the operational agents network. Because GPT knows not so much about ZJU-UIUC campus, we need a retrieval module, to retrieve related data processed by various components on data-side. And to finally compose an answer, we will use the previously obtained context as well as intent, and feed them into the answer generation unit.

Unlike usual linguistic tasks that be completed by a simple call to openAI APIs[12], this tasks requires offloading logics to retrieval models and huge effort on data collection as well as processing. The retrieval model requires a clear evaluation scheme to make the system **sustainable and scalable**.

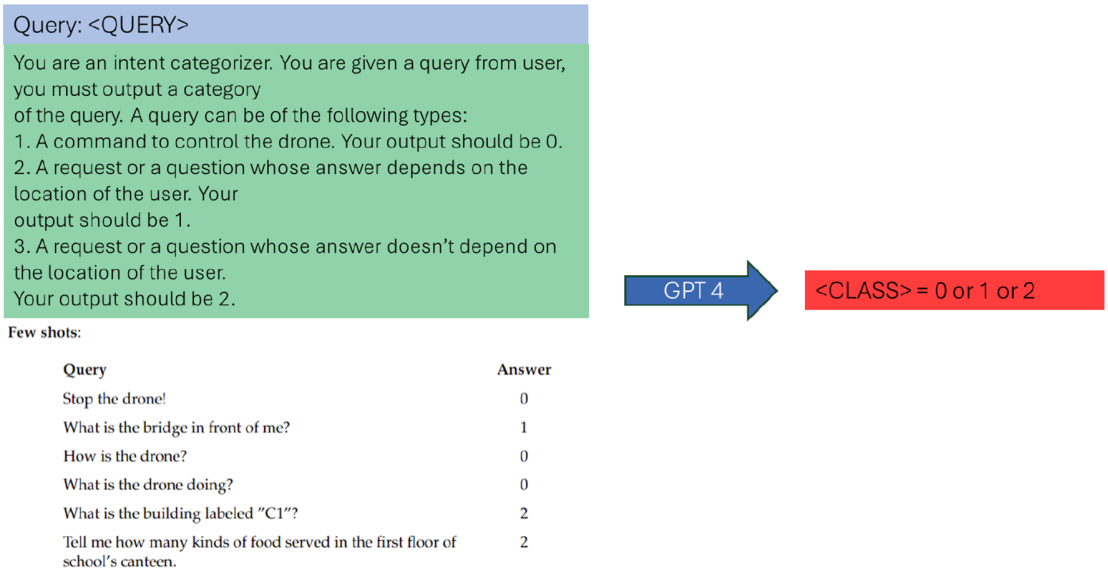

Figure 11: Intent Identification Workflow

3.2.3 Intent Identification Module

The intent identification module is used to differentiates 3 types of users' input listed in the instruction part of the prompt. This module is implemented by a simple prompt to GPT 3.5. Because of the limitation of GPT models, which can only reach a $\tilde{7}0\%$ accuracy [12] on a general testing multiple choice dataset, the accuracy for handling a specific task tailored to our application needs to carefully evaluated.

I did preliminary verification of this module by testing 100 questions with labeled answer. Our module reaches an accuracy of 61%, giving us some space to improve this module. The 100 questions are visualized in Figure 4 and Figure 5. Figure 4 is a word cloud for the questions, and Figure 5 is the distribution of the questions of 3 categories mentioned above in the prompt.

Prompt:

You are an intent categorizer. You are given a query from user, you must output a category of the query. A query can be of the following types:

1. A command to control the drone. Your output should be 0.
2. A request or a question whose answer depends on the location of the user. Your output should be 1.
3. A request or a question whose answer doesn't depend on the location of the user. Your output should be 2.

Few shots:

Query	Answer
Stop the drone!	0
What is the bridge in front of me?	1
How is the drone?	0
What is the drone doing?	0
What is the building labeled ”C1”?	2
Tell me how many kinds of food served in the first floor of school’s canteen.	2

Table 3: Queries and their corresponding answer classifications.

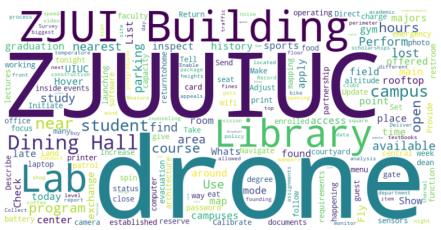

Figure 12: Word Cloud for Query Keywords

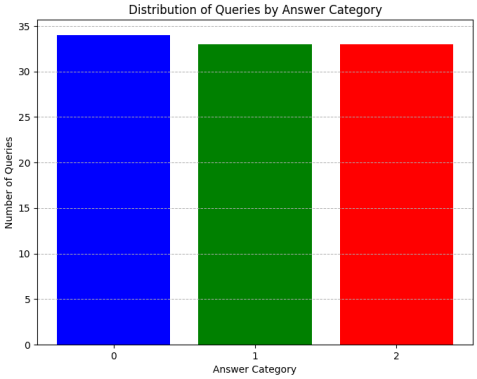

Figure 13: Distribution of Queries by Answer Category

Testing result by GPT

```
Passed for query: Fly the drone to the ZJUI Building.
Passed for query: Where can I find the nearest restroom?
Failed for query: What time does the Dining Hall open?
Expected: 2
Got: 1
Failed for query: Take a photo of the Library.
Expected: 0
Got: 1
.....
.....

Passed for query: Where can I get first aid in the ZJUI Building?
```

```
Passed for query: Broadcast a live feed from the drone
during the campus festival.
Passed for query: Are there any quiet study areas in the Library?
Failed for query: What is the average class size for
undergraduate courses at ZJU-UIUC?
Expected: 2
Got: 1
Passed for query: Is there a place to recharge electric
vehicles near the Dining Hall?
Failed for query: What sustainability initiatives are in
place at ZJU-UIUC?
Expected: 2
Got: 1
```

Score: 61/100

The alternative to this solution is

1. Upgrading the backing model to GPT4.
2. Conduct further prompt engineering listed in this article [13] to further improve the accuracy.

3.2.4 Retrieval Module: Search Engine

The information retrieval module utilizes Retreival-Augment-Generation. After carefully studying this method [14], I choose the most advanced modular design for the module. The whole design of this agent consists of a multi-media input processing unit, a search engine unit played as a retriever, as well as a generation unit that synthesize the retrieved information and generate corresponding output.

The document retriever associates the query with the corresponding location and fetches the data we gathered for the location. However, this will not be sufficient when the data we collect for a single location grows in size, as it will exceed the token limit imposed by GPT. A token limit is the maximum number of tokens you can feed to an LLM at once. To address this problem, the retriever chunks the external material into ”chunks,” each with a summary. It then vectorizes the chunks into vectors. The vectorization tool is provided by OpenAI as well. The distance between the vectors represents the similarity of the semantics for the text that is vectorized. We utilize this trait to find the top-K similar chunks to the question by matching the vector of the question to the vectors of the chunks.

Using this chunking method, we can efficiently address the token limit problem with the retriever. Preliminary testing and verification were done by manually checking all tested questions' logs to see if the retriever works correctly. This process is shown in Figure 14. I checked the number of questions for which the retriever fetches the correct external material. This checking was done manually but can further be automated by aligning a GPT evaluator with human evaluation. This means we will write a script to let

Algorithm 1 RAGAgent Initialization and Vector Store Handling

```
procedure INITIALIZE(vector_store_id, assistant_id)
    Initialize environment
    Create OpenAI client
    if assistant_id is provided then
        Retrieve existing assistant
    else
        Create new assistant with name and instructions
    end if
    if vector_store_id is provided then
        Update assistant with vector store
    end if
end procedure
procedure CREATE_VECTOR_STORE(file_paths)
    Create new vector store
    Open and read files (PDF/TXT) into streams
    Upload and poll file batch to vector store
    Print status and file counts
    Update assistant with new vector store
    return vector store ID
end procedure
procedure UPDATE_VECTOR_STORE_TO_ASSISTANT(vector_store_id)
    Update assistant with vector store ID
    return vector store ID
end procedure
```

GPT verify the RAG process and align its evaluation with human evaluation. If they are aligned, it can be shown that the GPT evaluation result is reliable and hence can automate the evaluation process.

3.2.5 Answer Generation Module

The Answer Generation Module is a crucial component of the Retrieval-Augment-Generation (RAG) framework. This module synthesizes the retrieved information to generate accurate and contextually relevant responses to user queries. The diagram below illustrates the workflow of the Answer Generation Module.

Workflow Description

1. User Query: - The process begins with a user submitting a query. This query is directed to the QA assistant, which is designed to assist in providing answers about specific topics—in this case, related to the ZJU-UIUC campus.

2. Contextual Information and Reference Nodes: - The QA assistant receives the query along with a list of reference nodes. These nodes contain various pieces of external mate-

Algorithm 2 Query Handling

```
procedure QUERY(query)
    Create thread with user query
    Create and poll run with thread ID and assistant ID
    List messages from thread
    Extract and process message content and annotations
    Print message content and citations
end procedure
Main Function
Initialize RAGAgent with vector_store_id and assistant_id
Perform query on the initialized agent
```

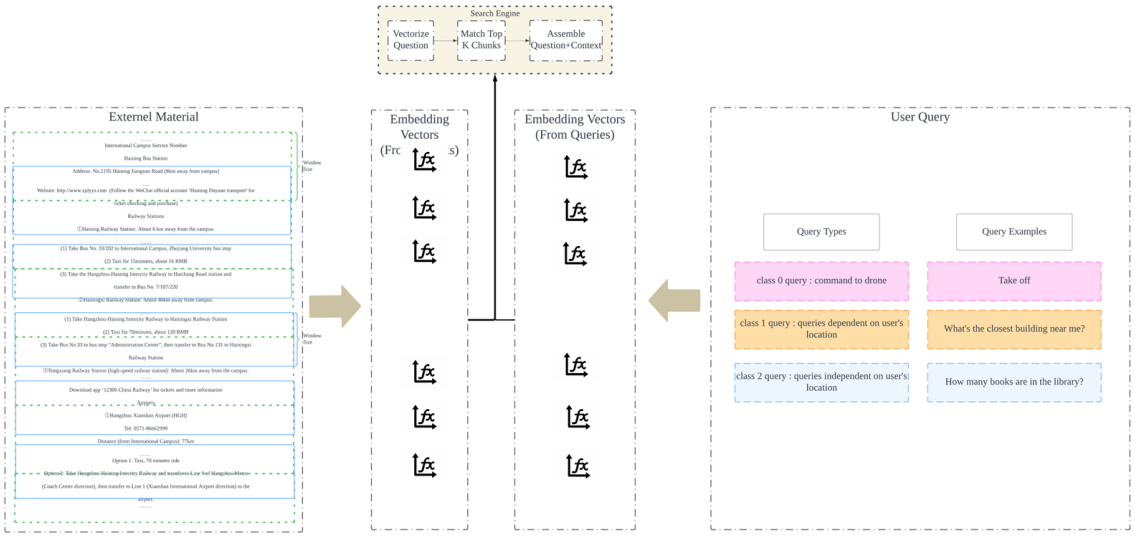

Figure 14: Workflow of Retrieval Module

rial that might be relevant to answering the query. Each node includes:

- `d_id`: A unique identifier for the node.
- `location`: The location context of the material.
- `keyword`: Key terms associated with the material.
- `outlook`: A brief summary or outlook of the content.
- `text`: The actual external material that might help in answering the query.

3. Processing by GPT-3.5: - The assembled context, which includes the user query and the relevant reference nodes, is then processed by the GPT-3.5 model. The model uses its advanced natural language understanding capabilities to interpret the query within the provided context. - GPT-3.5 analyzes the query and the associated reference nodes to

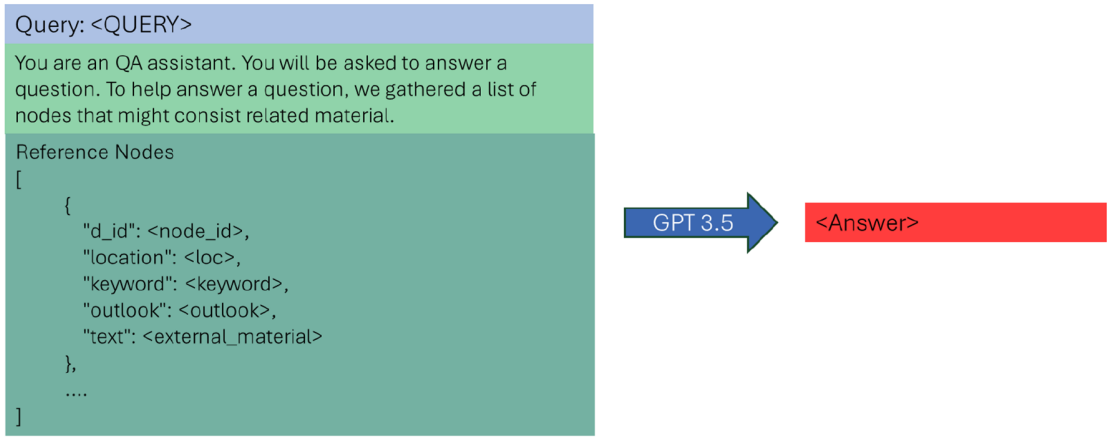

Figure 15: Answer generation workflow

generate a comprehensive and accurate answer. This involves understanding the nuances of the query, extracting pertinent information from the reference nodes, and formulating a coherent response.

4. Answer Generation: - Finally, the GPT-3.5 model produces an answer based on the synthesized information. This answer is then provided back to the user, ensuring it is both relevant and contextually appropriate.

Advantages of the Answer Generation Module

The Answer Generation Module leverages the power of advanced language models like GPT-3.5 to provide detailed and accurate responses. By integrating contextual information from reference nodes, the module ensures that the generated answers are not only based on a comprehensive understanding of the query but also grounded in relevant external materials. This enhances the reliability and relevance of the responses provided to users.

3.2.6 Connecting agents to a network

Each agent has its own ability, connecting agents together involves code that distributes tasks to different agents. These pieces of code are modeled as agent leads, as shown in the diagram. Operational agents and informational agents are both backed by OpenAI's API to generate responses. The informational network and operational network are not independent of each other. GPS signals and commands are passed between the two subnetworks on demand. The code responsible for this communication is modeled as the pilot agent shown in the diagram. The safety and grounding units are omitted here and in the diagram and will be discussed in later chapters.

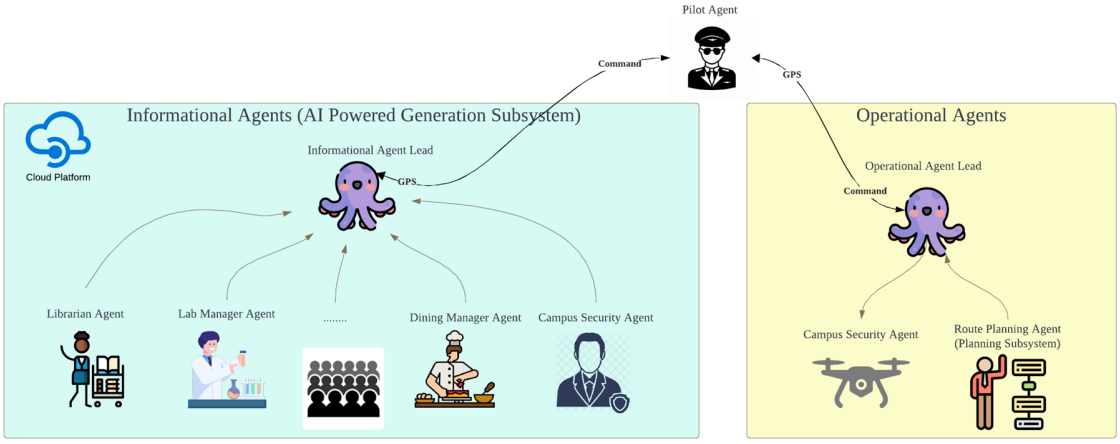

Figure 16: Architect of agent network

Informational Agents

Informational agents form the AI-powered generation subsystem. Each informational agent has a specific role and accesses a cloud platform for information retrieval and processing. For example:

- **Librarian Agent**: Manages information about library resources and services.
- **Lab Manager Agent**: Provides details about laboratory facilities and schedules.
- **Dining Manager Agent**: Offers information on dining options, menus, and services.
- **Campus Security Agent**: Supplies security-related information and protocols.

The informational agent lead coordinates these agents, ensuring that the relevant information is provided based on user queries.

Operational Agents

Operational agents constitute the execution subsystem, where commands are translated into actions. Each operational agent performs specific tasks based on the commands received:

- **Campus Security Agent**: Manages security operations, potentially including surveillance and emergency responses.
- **Route Planning Agent**: Handles planning and optimization of routes for campus navigation.

The operational agent lead ensures that the commands are executed correctly, coordinating between different operational agents.

Pilot Agent

The pilot agent serves as the communication bridge between the informational and operational agents. It processes GPS signals and commands, ensuring that both networks can function cohesively. When a user query involves both information retrieval and execution, the pilot agent coordinates the task, ensuring seamless operation.

Integration and Coordination

The integration of these agents into a cohesive network allows for efficient task distribution and execution. Informational agents gather and process data, while operational agents act on the processed information. The pilot agent ensures that both networks communicate effectively, using GPS signals and command relays to maintain synchronization.

This architecture provides a robust framework for handling a wide range of tasks, from answering queries to executing complex operations on campus. By leveraging OpenAI's API and a well-coordinated agent network, we can deliver precise and timely responses to user needs.

3.3 Safety and Ethics Work

3.3.1 Addressing AI Hallucination and Grounding

Algorithm 3 UAVOPAgent Class Initialization and Methods

Input: OpenAIITF object
procedure INITIALIZE($OpenAI\_itf$)
    $self.OpenAI\_itf \leftarrow OpenAI\_itf$
end procedure
procedure QUERY($query, prompt \leftarrow' uav\_op\_agent.prompt'$)
    $prompt\_content \leftarrow$ ReadFile($prompt\_dir, prompt$)
    $config\_data \leftarrow$ LoadYAMLFile($agents\_config\_dir,' uav\_op\_agent.yaml'$)
    $locations \leftarrow config\_data['location']$
    $message\_list \leftarrow$ CreateMessageList($prompt\_content, locations, query$)
    $result \leftarrow self.OpenAI\_itf$.GetChatCompletionContent($message\_list, 0$)
    return GroundResult($result$)
end procedure
procedure GROUNDRESULT($result$)
    $content \leftarrow$ ExtractContent($result$)
    $command, location \leftarrow$ ExtractCommandAndLocation($content$)
    $config\_data \leftarrow$ LoadYAMLFile($agents\_config\_dir,' uav\_op\_agent.yaml'$)
    $command\_dict \leftarrow config\_data['command']$
    $location\_dict \leftarrow config\_data['location']$
    if IsValidCommand($command, command\_dict$) **and** (**not** $location$ **or** IsValidLocation($location, location\_dict$)) **then**
        $command \leftarrow \{command : command\_dict[command]\}$
        if $location$ **is None then**
            return FormatAsJSON($command, None$)
        else
            return FormatAsJSON($command, \{location : location\_dict[location]\}$)
        end if
    else
        return None
    end if
end procedure

AI agents suffer greatly from hallucination, which means the generation of plausible but incorrect or nonsensical information. This can be particularly problematic in scenarios requiring precise and accurate responses, such as UAV operations, where incorrect information can lead to dangerous or illegal actions [15], [16].

To mitigate these risks, AI agents must be carefully grounded, especially on the operational side of the AI agent network. Grounding involves validating and verifying the generated outputs against predefined rules and contexts to ensure their accuracy and safety [17].

Algorithm 4 Main Function to Run UAVOPAgent

1: $itf \leftarrow$ OpenAIITF()
2: $agent \leftarrow$ UAVOPAgent(itf)
3: Print(agent.Query("Take off the drone"))
4: Print(agent.Query("Land the drone"))
5: Print(agent.Query("Do a barrel roll"))
6: Print(agent.Query("Fly the drone to the library"))
7: Print(agent.Query("Fly the drone to the center of the lake and land it"))

Hence, the following sample algorithm showcases the effort we put into grounding the UAV operational agent. It ensures that no illegal command will be output by the UAV operational agent under any circumstances [18].

3.3.2 Algorithm Description

The algorithm for the `UAVOPAgent` class ensures the grounding of commands issued to UAVs. Here's a detailed breakdown:

3.3.3 Initialization

The `Initialize` procedure sets up the OpenAI interface required for the agent to function. It prepares the agent to handle queries by establishing the necessary environment.

3.3.4 Query Handling

The `Query` procedure handles user queries. It reads a predefined prompt, loads configuration data, and prepares a list of messages to be processed. The query is sent to the OpenAI model, and the response is obtained.

3.3.5 Grounding Results

The `GroundResult` procedure ensures that the generated command and location are valid. It extracts the relevant content, validates the command against predefined rules, and formats the result appropriately. If the command or location is invalid, it returns `None`, ensuring no illegal command is output.

3.3.6 Main Function

The main function demonstrates the execution of the `UAVOPAgent`. It initializes the agent and processes several sample queries, showcasing the grounding process in action. This ensures that the UAV operational agent only executes safe and valid commands.

By incorporating stringent validation and grounding procedures, the `UAVOPAgent` ensures the safe and reliable operation of UAVs, mitigating the risks associated with AI hallucinations.

3.4 Evaluation

3.4.1 End2End Answer Generation: Testing and Verification

Combining data collection, intent identification, document retriever, as well as answer generator, we have an end to end solution which accepts a user's query and output a corresponding answer. I have evaluated the intent identification part as well as the RAG part. I then evaluated the end-to-end process by using human evaluation as well as comments. This process both let the designer case-study an appropriate amount of questions, and provide few-shots examples so that we can also conduct automation of evaluation and align it with human evaluation.

The verification is conducted with the following methodology. I first design 15 different questions across 4 testing locations with abundant external materials. Next, I use these 15 questions I designed as few-shots examples to let GPT generate 15 another examples randomly across the 4 testing locations. The results shows that among 30 testing questions, human identified 13.3% incorrect answers to human asked questions, and 0% incorrect answers to GPT asked questions. The score graded by human is also displayed in Figure 6 accross different locations.

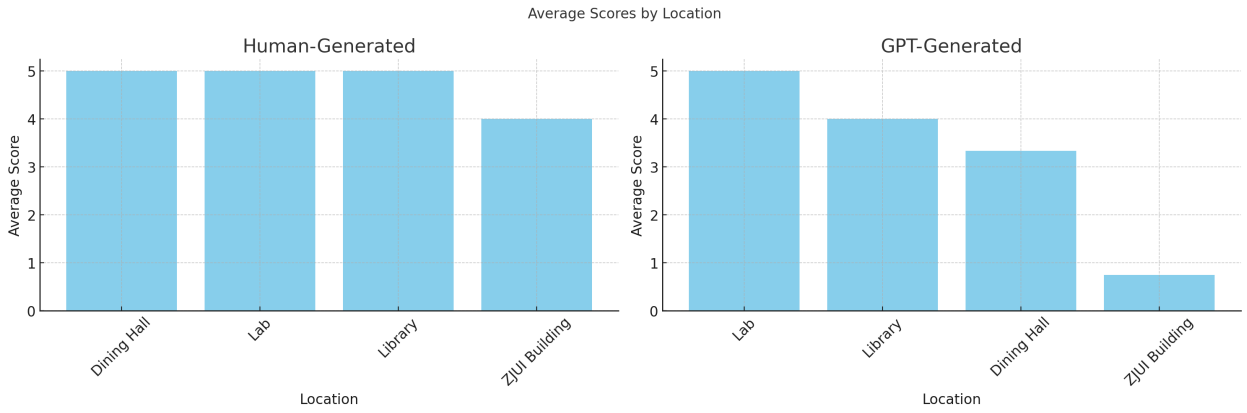

Figure 17: Evaluation of answer distributed by testing locations

As shown in figures above, our end-2-end evaluations prove to be accurate and valid against questions to ZJU-UIUC campus, and can provide sufficient guidance to visitors. We have our code for the agent open-sourced, along with testing dataset, testing scripts in the following project repository.

To further investigate the problmes of the current design and conduct efficient version iterations, we visualize the comments to the experiment results in the word clouds shown in Figure 8. We find that a source of problem is still token limit of LLM module being exceeded. This is because we haven't implemented chunking part as discussed above. This is considered a next step accorinding to the pace of other subsystems of this project.

We have demonstrate the end-2-end workflow in Figure 9. On the left is input to our design, and on the right is the output. As one can see, we support batch input mode and

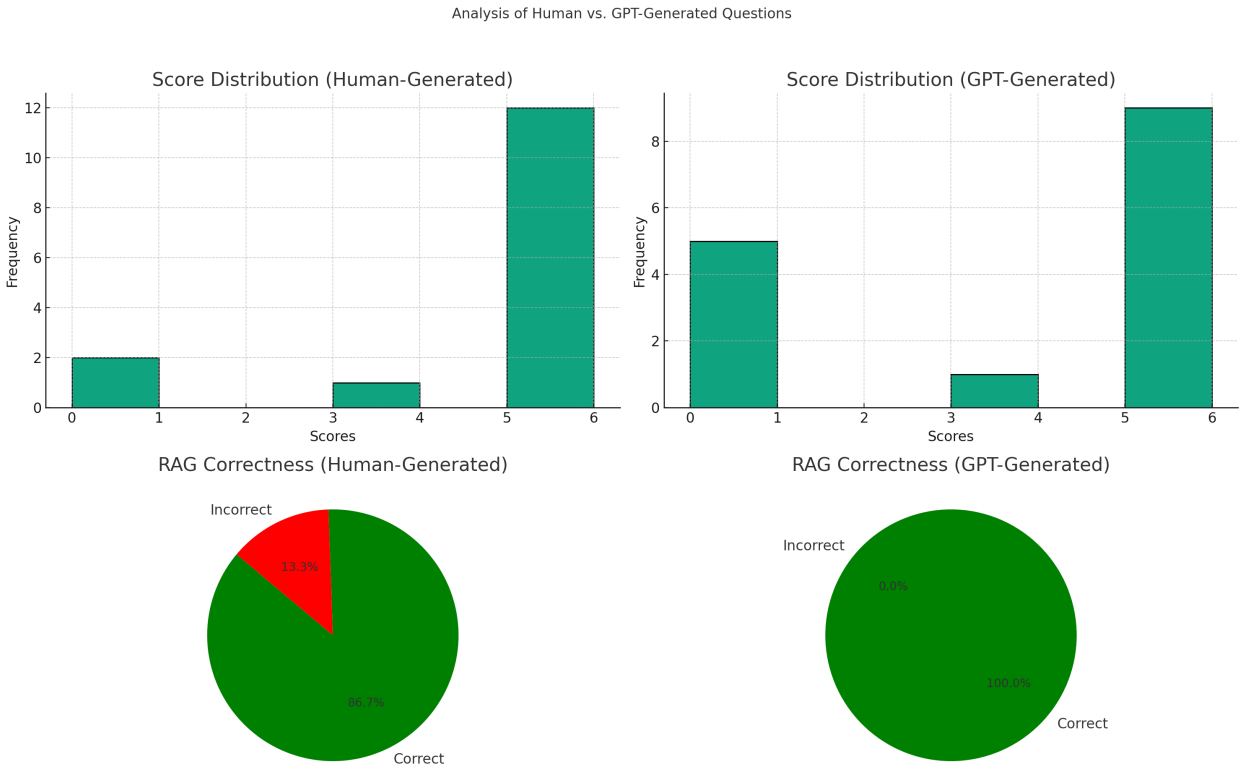

Figure 18: Analysis of End to End Accuracy as well as RAG accuracy

output-mode. And the process is properly logged.

3.4.2 AI-powered Response Generation Subsystem Interface

The code is available at project_repos

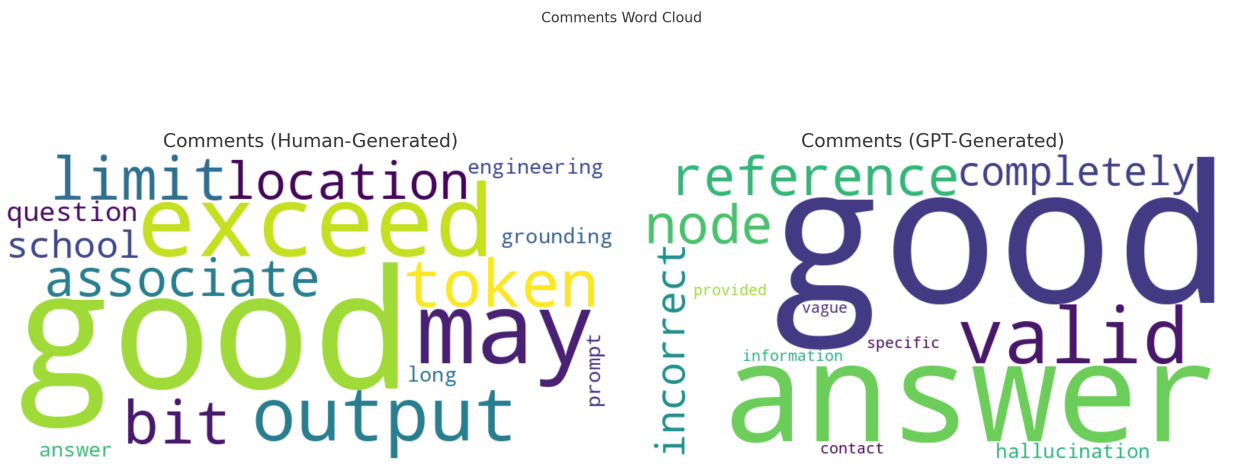

Figure 19: Major problems with current module for human and gpt-generated questions

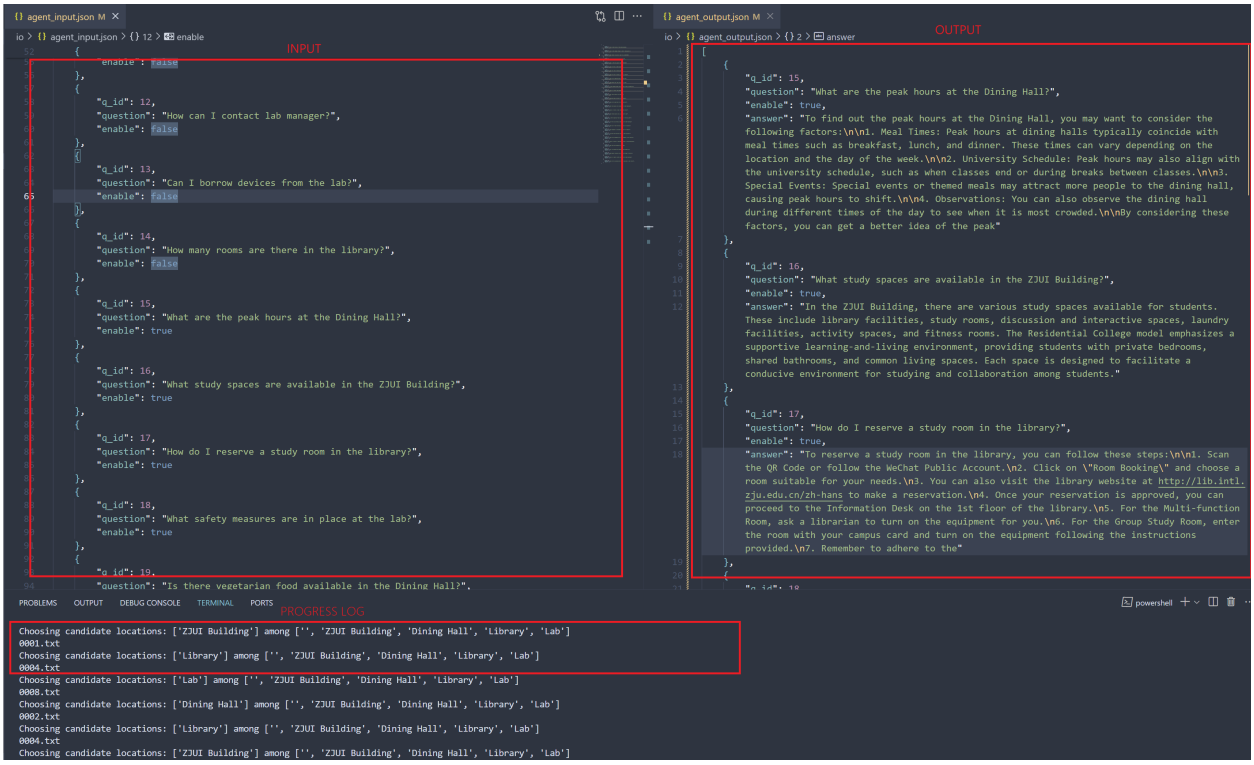

Figure 20: Interface Demo

4 Cost Analysis

4.1 Labor

The labor cost is calculated based on the working hours and wage pricing of each team member. We set the hourly wage at 100 RMB based on market research and the skill levels of team members. Considering the total project duration of 8 weeks with 40 hours of work per week, the total working hours per team member are: 320 hours. Therefore, the labor cost per team member is:

$$100 \text{ RMB/hour} \times 320 \text{ hours} = 32000 \text{ RMB}$$

We chose an hourly wage of 100 RMB, which is based on market wage levels and the skill and experience levels of team members. According to survey data from the Institute of Electrical and Electronics Engineers (IEEE) [19], the average salary for graduates in Electrical and Computer Engineering (ECE) is around 200,000 RMB per year. Calculated on a full-time basis, the average hourly wage is approximately 100 RMB.

4.2 Parts

The table below provides a breakdown of the parts and their estimated costs:

Description	Manufacturer	Part #	Quantity	Cost (RMB)
Drone	PixHawk	MFP450	1	5174
Mavlink Module	Amovlab	-	1	680
Temperature Sensor	Aosong	DHT11	2	10
Acceleration Sensor	MiraMEMS	DA213B	2	15
LCD Screen	Touglesy	LCD1602	1	20
PCB Board	Custom	-	1	50
Simple Application Server	Alibaba Cloud	-	1	49 per month
ChatGPT4 API	OpenAI	-	1	240 per month

Table 4: Parts List and Estimated Costs

4.3 Grand Total

The grand total cost of the project can be calculated by summing up the labor cost and the cost of parts:

- Labor: 32,000 RMB
- Parts: 6,819 RMB

Grand Total: 32000 + 6819 = 38,819 RMB

5 Conclusion

In this report, we explored the comprehensive architecture and functionality of an AI-powered multi-agent system designed for the ZJU-UIUC campus. This system integrates various specialized agents, each with unique roles, to deliver precise and contextually relevant information and services.

Informational Agents

We delved into the roles of informational agents, which form the AI-powered generation subsystem. These agents, such as the Librarian Agent, Lab Manager Agent, Dining Manager Agent, and Campus Security Agent, are crucial for gathering, processing, and providing data pertinent to their specific domains. Coordinated by the informational agent lead, these agents ensure that user queries are answered with the most relevant information retrieved from a cloud platform.

Operational Agents

The operational agents constitute the execution subsystem, where the commands generated by the informational agents are executed. This includes agents like the Campus Security Agent and the Route Planning Agent, which handle security operations and navigation planning, respectively. The operational agent lead ensures that these commands are carried out accurately and efficiently.

Pilot Agent

The pilot agent serves as the communication bridge between the informational and operational agents. It manages the transfer of GPS signals and commands, ensuring seamless interaction between the two subsystems. This coordination is crucial for tasks that require both information retrieval and action execution.

Addressing AI Hallucination and Grounding

A significant focus of our system is on mitigating AI hallucinations, which can result in incorrect or nonsensical information. Grounding, especially in the operational side of the AI agent network, ensures that all commands are validated and safe. The UAVOPAgent class exemplifies this effort by strictly validating and grounding UAV commands to prevent any illegal or unsafe actions.

Retrieval-Augment-Generation Framework

The system utilizes a Retrieval-Augment-Generation (RAG) framework to enhance the accuracy and relevance of responses. This involves chunking large data sets to address token limits, vectorizing these chunks, and using similarity matching to retrieve the most relevant information. This approach ensures that the system can handle extensive data efficiently while providing accurate responses to user queries.

Integration and Coordination

The integration of informational and operational agents into a cohesive network, facilitated by the pilot agent, allows for efficient task distribution and execution. This architecture ensures that the system can handle a wide range of tasks, from simple information retrieval to complex operational commands, providing a robust and reliable service to the users.

In conclusion, the multi-agent system for the ZJU-UIUC campus represents a sophisticated and well-coordinated approach to leveraging AI for campus management. By addressing key challenges such as AI hallucination and ensuring seamless integration between information retrieval and operational execution, this system sets a strong foundation for advanced AI applications in campus environments and beyond.

Safety and Ethics

To follow the IEEE ethics manual [20], I did some steps to ground the output of our AI-agent so it will be reliable and not propagate misinformation as well as unethical information.

References

[1] MDPI, "Chatgpt and open-ai models: A preliminary review," *Future Internet*, 2022.

[2] J. P. Müller and K. Fischer, *Application Impact of Multi-Agent Systems and Technologies: A Survey*. Springer, 2014, pp. 27–53.

[3] G. et al., "Gpt (generative pre-trained transformer) – a comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions," *arXiv preprint arXiv:2311.10242*, 2022.

[4] M. Herrera, M. Pérez-Hernández, A. K. Parlikad, and J. Izquierdo, "Multi-agent systems and complex networks: Review and applications in systems engineering," *Processes*, vol. 8, no. 3, p. 312, 2020.

[5] A. Sturm and O. Shehory, *The landscape of agent-oriented methodologies*. Springer Berlin Heidelberg, 2014, pp. 137–154.

[6] D. Isern and A. Moreno, "A systematic literature review of agents applied in healthcare," *Journal of Medical Systems*, vol. 40, no. 2, p. 43, 2016.

[7] M. Falco and G. Robiolo, "A systematic literature review in multi-agent systems: Patterns and trends," in *XLV Conferencia Latinoamericana de Informática, Centro Latinoamericano de Estudios de Informática (CLEI)*, IEEE, 2019.

[8] OpenAI, "Our approach to ai safety," *OpenAI*, 2023. [Online]. Available: https://openai.com/our-approach-to-ai-safety.

[9] S. University, *Stanford ai safety*, 2023. [Online]. Available: https://aisafety.stanford.edu/.

[10] S. Workshop, *Safeai 2023 – aaai's workshop on artificial intelligence safety*, 2023. [Online]. Available: https://safeai.webs.upv.es/.

[11] M. A. S. W. Group, "Introducing v0.5 of the ai safety benchmark from mlcommons," 2024. [Online]. Available: https://ar5iv.org/abs/2404.12241.

[12] OpenAI, J. Achiam, S. Adler, *et al.*, *Gpt-4 technical report*, 2024. arXiv: 2303.08774 [cs.CL].

[13] L. Weng, "Prompt engineering," *lilianweng.github.io*, Mar. 2023. [Online]. Available: https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/.

[14] P. Zhao, H. Zhang, Q. Yu, *et al.*, *Retrieval-augmented generation for ai-generated content: A survey*, 2024. arXiv: 2402.19473 [cs.CV].

[15] IBM, "What are ai hallucinations?," 2023. [Online]. Available: https://www.ibm.com.

[16] M. Sloan, "When ai gets it wrong: Addressing ai hallucinations and bias," 2023. [Online]. Available: https://mitsloanedtech.mit.edu.

[17] SuperAnnotate, "Ai hallucination: Complete guide to detection and prevention," 2023. [Online]. Available: https://www.superannotate.com.

[18] arXiv, "Human-machine teaming for uavs: An experimentation platform," 2023. [Online]. Available: https://ar5iv.org.

[19] "IEEE (Institute of Electrical and Electronics Engineers) Salary Survey," 2022.

[20] IEEE. ""IEEE Code of Ethics"." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 02/08/2020).

Appendix A Requirement & Verification Table

Table 5: Subsytem Index Table

Subsystem Name	Subsystem Index(S-Index)
AI Powered Response Generation Subsystem	1
User Interface Subsystem	2
Planning and Control Subsystem	3
Sensor Unit Subsystem	4

Table 6: Requirement & Verification Table

S-Index	Requirement	Verification	Points
1	Detect user intent with at least 25% accuracy	Test with a diverse set of input queries and verify the accuracy of intent detection.	2
1	Detect user intent with at least 50% accuracy	Continue testing and refining to achieve higher accuracy.	2
1	Generate response within 60 seconds	Measure retrieval time with various queries to ensure performance within the initial time limit.	2
1	Generate response within 40 seconds	Measure retrieval time with various queries to ensure performance within the initial time limit.	2
1	Generate response within 30 seconds	Optimize system to improve performance and meet the final time requirement.	1
1	Can fetch correct external material with 20% accuracy	Testing the agent's RAG accuracy by using manually labeled dataset containing 30 questions for 4 testing locations.	1

Continued on next page

Table 6 continued from previous page

S-Index	Requirement	Verification	Points
1	Can fetch correct external material with 50% accuracy	Testing the agent's RAG accuracy by using manually labeled dataset containing 30 questions for 4 testing locations.	2
1	Can fetch correct external material with 70% accuracy	Testing the agent's RAG accuracy by using manually labeled dataset containing 30 questions for 4 testing locations.	1
1	Generated answers must match the user's intent with an accuracy of at least 25% in given dataset	Compare generated answers from the dataset containing 30 questions with human labeled answer to calculate initial accuracy.	3
1	Generated answers must match the user's intent with an accuracy of at least 50% in given dataset	Compare generated answers from the dataset containing 30 questions with human labeled answer to calculate initial accuracy.	3
2	The web server must handle and route messages with less than 2 seconds latency	Test message routing on the server under load and measure latency	2
2	The client interface must provide intuitive access for users to submit queries and control the UAV	Conduct usability testing with participants to assess ease of use and intuitiveness	1
2	UAV command buttons must send correct instructions to the UAV subsystem with 100% accuracy	Test each button and verify that the correct command is sent to the UAV subsystem	1
2	The web server must send instructions and questions separately to different hosts	Test two hosts if they receive correct messages	1
2	Obtain the user's GPS position as the starting position	Test if the two hosts receive the GPS signal	1
2	The host which process the questions can display answers correctly	Check if the UI can display reasonable answers	2

Continued on next page

Table 6 continued from previous page

S-Index	Requirement	Verification	Points
2	The host which process the instructions can send commands to UAV	UAV can take-off, Stop, Continue, Land correctly and in time	2
3	Must accurately process user commands and drone status within 10 second	Perform stress testing with simultaneous user commands and verify response time	3
3	Must accurately process user commands and drone status within 1 second	Perform stress testing with simultaneous user commands and verify response time	3
3	Must optimize the UAV route based on the current status	Test with different scenarios (no-fly zones, different areas) to verify route optimization	3
3	Should maintain a secure and encrypted connection to the remote server	Verify the encryption standards and conduct penetration testing to assess security	1
3	Must integrate seamlessly with the PX4 APIs for flight control	Execute a series of flight tests to ensure proper integration and control	1
4	Power supply successfully power the hardware unit	Power supply LED works correctly	1
4	Sensors can operate properly	Connect the sensors to the test board Use oscilloscope to read the output data.	2
4	LCD screen can display normally	Connect the LCD screen to the test board Provide appropriate power. Send test data to the LCD screen Check if the screen displays correctly.	2
1,2 3,4	End to End works correctly	Can complete a guide for appointed locations while iteracting with visitors	5

Appendix B Resume

Author Name	Hao Ren
Student ID	3200110807
Educational background	2017.9.1 - 2020.7.1, Zhilin High School 2020.9.1 - 2024.7.1, Zhejiang University
Awards received	MCM Final List
Participation in projects	Architect the project and workflow Build multi-agent AI network to power the system Divide the work to different members Understand each part of the design and unblock hard problem Have 1-on-1 meeting with members to understand the blocking issues and give constructive advice

Senior Design Report Task Assignment

1. **Report Title: Smart Power Routing with MPPT-Based Wind Turbine**

2. **Guidelines from supervisors regarding schedule and requirements for project and report:**

Schedule:

Requirements:

Notes: Start date: January 15, 2024; End date: May 24, 2024.

Supervisor: Professional title:

3. **Institute Review Comments:**

Dean: Date:

Senior Design Individual Report Assessment Form

1. Supervisor's comments on the Senior Design Individual Report:

Supervisor: Professional title:

2. Defense Committee's comments on the Senior Design Individual Report:

Grade:

Defense Committee Chairman: Date: