Imperial College London

Department of Earth Science and Engineering

Final Report

A Conversational Multi-Agent System for Smart Campus Energy Management

Author:
Arastun Mammadli[1,2]
am4224@ic.ac.uk
esemsc-am4224

Supervisors:
Prof. Lee Stott[2]
Prof. Gerard Gorman[1]

Submitted in partial fulfillment of the requirements for the MSc degree in Applied Computational Science and Engineering of Imperial College London

August 2025

[1] Imperial College London, [2] Microsoft

Contents

Abstract

Large Language Models (LLMs) are increasingly used in autonomous multi-agent systems. They have seen extensive success in the fields of web navigation [13], software engineering [46], personalised learning and research [7]. However, due to its technical and multi-facet requirements (dynamic control and monitoring, feedback collection, sustainability initiatives), energy management remains a challenging domain. We argue agentic systems can extend traditional energy monitoring only solutions to serve a range of campus stakeholders (students, faculty, and administrators).

This project leverages emerging frameworks in Microsoft's AI Agent Services to demonstrate a proof of concept for a smart energy management system. Building on base agents, it integrates language and search services to enable a robust orchestration. The system includes a user-facing chatbot that can respond to system-related queries, collect student feedback, and assist administrators textual and visual prognostics. The project also designs a synthetic energy data schema with hierarchical infrastructure, usage categories, and environmental context that mirrors real world systems and supports agentic capabilities.

The system is evaluated on modular (individual agents and services) and end-to-end levels. Special focus is given to explore intent routing, access-level based grounding mechanisms, and query complexity. All together the assessments outline practical challenges of composite agentic workflows and their integration with external services. They discuss promising future directions through integrations with real-world energy systems.

1 Introduction

The building sector accounts for 32% of global energy demands [42]. As part of this, universities face significant pressure to reduce their carbon footprint and energy costs [16]. University campuses include a range of spaces (e.g., lecture halls, laboratories, sports facilities, dormitories) with distinct energy profiles and usage patterns. Hence, they make great candidates for complex energy management systems. That is; managing HVAC (heating, ventilation, air-conditioning) activity, lighting levels, and the equipment efficiently [35]. There are range of factors to consider when optimising energy usage. For example, grid prices, environmental conditions (e.g., weather, occupancy [32]), and utilities (e.g., HVAC, lab and teaching equipment [48]). Moreover, student and faculty satisfaction (comfort levels) are also important considerations [15]. Together, this makes centralised control difficult. There is a strong case to adopt smart campus energy management, beyond traditional monitoring only systems.

Past works have mostly explored multi-agent deep reinforcement learning (MADRL) [48, 9, 5] techniques. Multi-agent systems (MAS) are a divide and conquer approach where multiple autonomous agents collaborate (or compete) to solve the problem [6]. They align well with the distributed nature of campus energy (e.g., range of spaces with distinct energy profiles). However, traditional methods have mostly focused on rule-based logic enhanced by traditional AI techniques (e.g., reinforcement learning [48], neural networks [45]). They are energy monitoring only solutions, constrained to technical use with little consideration for user-centric design.

In contrast, Large Language Models (LLMs) as autonomous agents have seen extensive success in content generation and interactive decision making. Particularly in the fields of web navigation [18, 13], software engineering [46], research [7], embodied environments [39], and generalist assistants [10]. Recent works have focused on improving the capabilities of LLM agents through better prompting [43, 47] grounding techniques [14], tool use [4], and multi-agent orchestrations [44]. These developments have led to more factual, grounded, and goal-driven agents. There is an opportunity to explore emerging concepts and frameworks in the context of campus energy management.

1.1 Significance and Originality

Commercially, there is a strong case to adopt smarter energy management systems. They have a strong drive towards sustainability, with many institutions committing to net-zer emission targets [16]. University campuses are also research intensive environments, making them a great testbed. A user-centric energy management system has influence on not just the administrative staff, but also the students and faculty. It can serve as a platform that involves all campus stakeholders, raising awareness in energy and sustainability initiatives.

The project shows originality in combining the technical (structured energy schema, feedback collection, dynamic energy prognostics) and creative (information retrieval and summary, chart generation) domains. Whereas most past works have either focused on technical monitoring systems (e.g., MADRL [48, 9]) or non-technical generalist assistants (e.g., Magentic-one [10]). The work also leverages emerging frameworks (Azure AI Foundry [24], Semantic Kernel [30], Azure AI Language [25], Azure AI Search [26]) to build the system. It showcases a novel integration of specialised agents, NLP services, and a fallback retrieval-augmented generation (RAG) system. Hence, it acts as a experimental testbed for future works involving these frameworks.

1.2 Objectives

First objective is to design a synthetic energy data schema reflective of real-world campus energy data. The schema should also contain heterogeneous data that can be used to comprehensively test the agentic capabilities.

Second objective is to develop a multi-agent orchestration powering a user-facing chatbot. The chatbot should be able to serve a range of campus stakeholders (students, faculty, and administrators). It should include specialised agents (both information retrieval and action based), to accommodate different user needs. The orchestration should also integrate non-LLM NLP services (for intent routing and common question answering), and a fallback RAG system (for robust grounding).

The third objective is to evaluate the system on modular (individual agents and services) and end-to-end levels. To test the system's multi-facet capabilities, the evaluation should cover a range of strategies (e.g., prompt engineering, intent resolution, query complexity). Together, the project should outline practical challenges of complex multi-agent orchestrations and their integration with non-LLM services.

2 Background

2.1 Base Agents

A base agent consists of an LLM model (e.g., GPT-4 [3], Phi4 [1]) and given instructions that guide its behaviour. This is the baseline structure upon which more specialised or task-specific agents can be built. Therefore, the performance of any agentic system fundamentally depends on the underlying base agent [36].

Earlier works [12] explored whether LLMs already contain enough information, not just for linguistic tasks, but to make goal-driven decisions. They were among the first to use LLMs as agents, directy acting on an environment. Later works such as ReAct (Reason and Act) [47] and Reflexion [38] took this further through more complex prompt engineering. ReAct combines reasoning; chain-of-thought (CoT) [43] with acting. Reflexion builds on this by endowing the agent with dynamic memory and self-reflection capabilities. More advanced prompt engineering techniques include; StateAct [36] (leveraging chain-of-states for long-context reasoning) and AdaPlanner [41] (using code-based prompts for reduced ambiguity).

These works have utilised emergent properties in LLMs, leading to a more factual and grounded problem solving capabilities. They lay an important foundation for the development of more complex agentic systems. ReAct is commonly the base agent of modern state-of-the-art (SoTA) approaches [36]. To explore it in the energy management setting, we also use ReAct-style prompting for our specialised agents.

2.2 Grounding and Multi-Agent Systems

In recent years, a range of Information Retrieval (IR) techniques have been incorporated to improve grounding and factuality. Notably, Retrieval-Augmented Generation (RAG) [14] was introduced to combine pre-trained parametric and non-parametric memory for language generation. To address increasing nubmer of real-world applications, the RAG paradigm has evolved significantly to include; standard Naive RAG, Advanced RAG, Modular RAG, Graph

RAG, and more recently Agentic RAG [40]. This work makes use of a Modular RAG approach with hybrid retrieval strategies. More concretely, it leverages Azure AI Search to combine semantic vector search with traditional keyword search.

Another intuitive approach to scale up the performance is to use multiple LLM agents that cooperate and produce a synthesised result [44]. Multi-agent systems can help better incorporate feedback (through agentic conversations or human-in-the-loop), showcase a broad range of capabilities (through individualised settings), and break complex tasks into manageable subtasks. Such systems have made strides in a range of fields including web navigation [13, 18], software engineering [46] and research [7]. We build on this idea by employing 4 specialised agents operated in a group chat setting through a coordinator Triage agent.

3 Methodology

3.1 Synthetic Data

Real-world campus energy datasets are often proprietary, fragmented, or incomplete [31, 19, 20]. This makes it difficult to obtain a comprehensive real-world dataset that is suitable for training and evaluating our system. Synthetic data enables prototyping of novel architecture without waiting for lengthy data access negotiations [17]. Moreover, with heterogeneous data we can better test the agentic capabilities, without being limited to the existing infrastructure. In general, synthetic data generation was used for:

- **Structured Energy Database:** a NoSQL Azure Cosmos [27] database storing; user profiles, campus infrastructure, energy metrics, and room sensor readings.
- **Unstructured Energy Data:** a collection of simulated documents containing manuals, energy initiatives, policies, and FAQs stored in Azure Blob [22].
- **Evaluation Scenarious:** to simulate user queries (based on available energy data or hand-crafted query samples).

The structured data is great at mirroring real-world campus databases (e.g., user profiles, buildings, rooms, energy metrics, sensor readings) [31, 20]. The unstructured data is great at simulating query scenarious (through the Azure Simulator [23]) and providing rich context for the agents (when combined with RAG). It is representative of large unstructured corpora common in institutional energy systems. Together they help test agentic grounding (structured and unstructured) and tool-use capabilities (structured information retrieval). By integrating structured and unstructured data, we question the assumption that only raw sensor and metered data is useful for smart energy systems. Instead we show that emerging LLM agents can leverage unstructured sources, providing stakeholders with a more holistic view of the system (e.g., how campus-wide sustainability initiatives impact energy usage).

3.1.1 Energy Data Schema

We note that synthetic data is not a replacement for real-world data. It is important to mirror real-world scenarios so that our findings are generalisable and applicable to practical settings. To do so, we ground synthetic data generation on existing energy data schemas and monitoring systems. We focus on design (structure) of the data schema and the integrity of data values (accuracy and validity). In total 3 existing energy data schemas were used to guide the synthetic data schema design:

- **Energy Monitoring at Imperial College London:** through the sustainability team we find that they use a custom energy monitoring system. We obtain metered readings from two buildings (MSRH and Faculty) at 15-minute granularity.
- **UNICON dataset [31]:** a comprehensive energy dataset in a university campus setting, containing infrastrucutre data (buildings categorised into; teaching/labs, library, administrative, residential, and sports), metered consumption readings at 15-minute granularity, and environmental data (temperature, humidity, wind speed and direction).
- **Office Building Dataset [19]:** a three-year dataset covering energy consumption (15-minute intervals), occupancy counts and environment conditions in an office building. It categorises energy usage into; miscellaneous, lighting, and HVAC, and considers both indoor and outdoor environmental parameters.

The key takeaways from these sources were:

- **Granularity:** 15-minute intervals are a common standard for energy monitoring.
- **Infrastructure:** buildings are categorised into different types (e.g., teaching, residential, administrative) to reflect their energy usage patterns.
- **Energy Metrics:** energy usage is categorised into different types (e.g., HVAC, lighting, miscellaneous) to reflect the diverse energy consumption patterns.
- **Environmental Data:** environmental conditions and occupancy counts are important for understanding energy usage patterns.

The synthetic data schema is designed to reflect these takeaways. The schema includes a hierarchical campus infrastructure metadata (buildings, HVAC zones, rooms) that help track energy usage, consumption and environmental conditions at different levels. The environmental data and occupancy are tracked through room status (that simulates sensor readings). The energy usage and costs are tracked through simulated utility actions. Finally, inspired from TherMOOstat [34] project (transforming occupants into a low-cost and high accuracy sensor system) we also incorporate an energy feedback container. This container is populated with live user feedback that can then be evaluated by system administrators.

3.2 System Design

We develop a flexible, user-centric energy management system that can serve all 3 end-users (students, faculty, and administrators). Students and faculty need quick access to general energy information (including initiatives and energy protocols). They may also want to provide feedback on room comfort levels, or make general suggestions on energy initiatives. Administrators need more detailed insights into energy usage, along with full access to the system database.

To address outlined objectives, the system is designed to contain several components. At its core the system includes 4 specialised agents built on top of the ReAct base agent. These are; the Campus Information (serving students and staff), Admin Information and Chart Plotter (administrators), and Feedback (all users). Characteristically they can be categorised as information-retrieval (IR) agents (Campus Information and Admin Information), and action-based agents (Feedback and Chart Plotter). CampusInfo and AdminInfo agents are generalist IR agents that can answer a wide range of energy-related queries based on the structured database. Their difference lies in the user they serve, corresponding access-level and answer detail. The Feedback agent collects user feedback on room comfort levels and energy usage

suggestions. Finally, the Chart Plotter agent generates visualisations of energy usage data for administrators.

To provide a more robust grounding mechanism, we use Azure AI Search [26] for hybrid retrieval-augmented generation. Conversational Language Understanding (CLU) is used for initial intent routing. While Custom Question Answering (CQA) is to quickly answer commonly asked questions without LLMs. Both of these are provided by Azure AI Language services [25]. Finally, the system is orchestrated through a Triage agent that resolves user query intent and routes to the appropriate component. Figure 1 is a high-level overview of the system architecture:

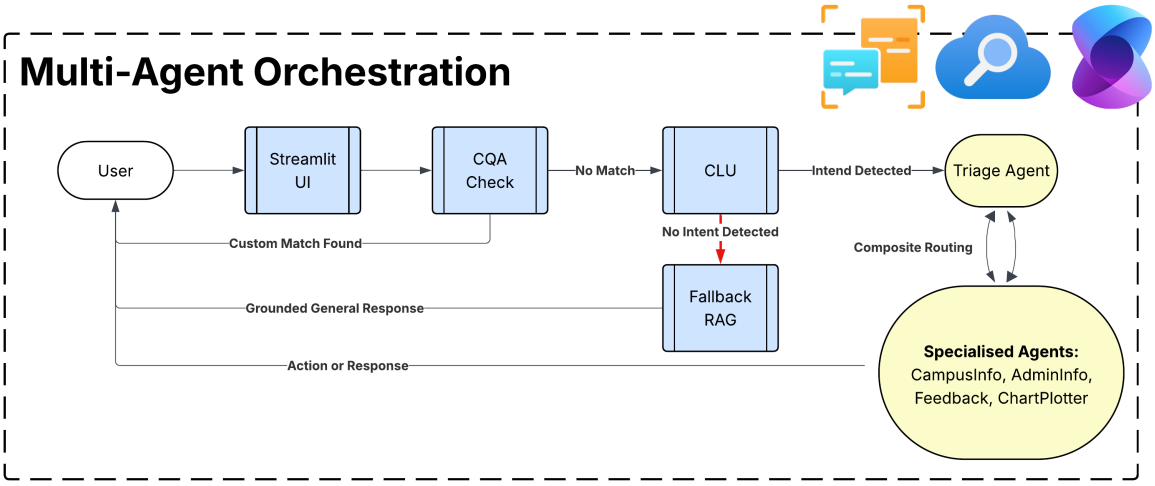


Figure 1: A step-by-step illustration of how the system operates. The blue shapes represent integrated services and the yellow shapes represent LLM agents.

3.2.1 Base Agents and Extensions

The base agents form core components of the multi-agent architecture. They are composed of an LLM base model, Semantic Kernel (SK) [30] plugins as tools, and ReAct-style [47] instructions. Figure 2 shows an overview of base agents used by the system.

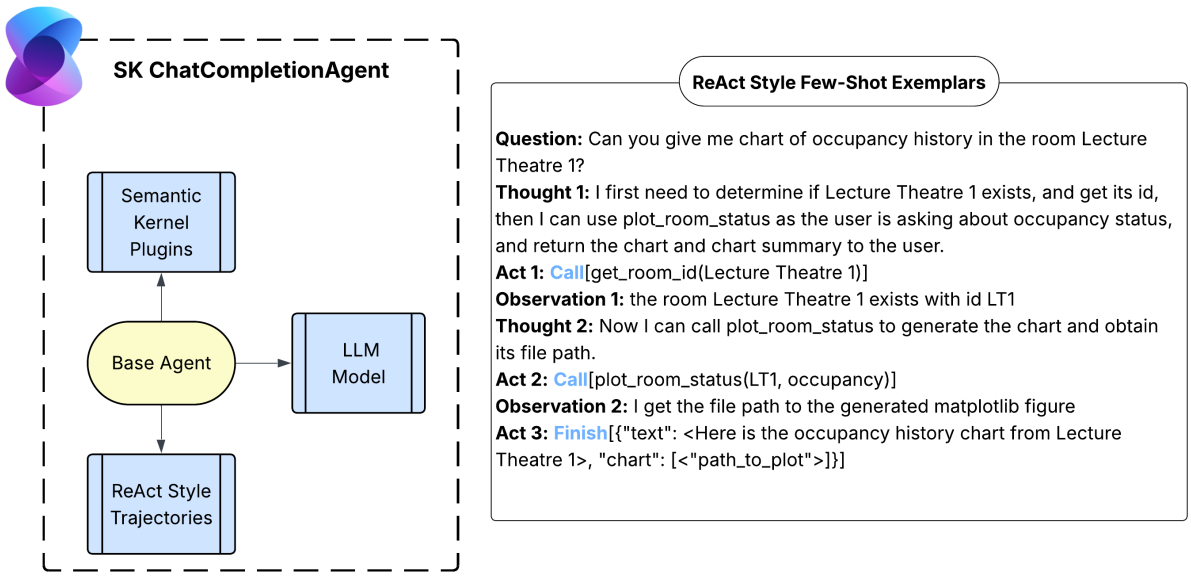


Figure 2: High-level agent overview and an example ReAct (Reason + Act) style trajectory instruction. The base agent is comprised of a base model (an LLM), system instructions, and Semantic Kernel tool plugins. The ReAct trajectory steps are; Thought (Reasoning), Action (Tool Call), and Observation (Environment Response).

Tool Use

Building on the ReAct base agents, the system employs tool use and grounding mechanisms to specialise them. Tool use is facilitated through Semantic Kernel (SK) plugins integrated with the base agents. The agent instructions include available tool definitions and few-shot examples to demonstrate task specific tool usage. Moreover, the instructions for tool usage are provided through `@kernel_function` tags and Python type hints in the function header. For example, in the Feedback agent, we can define `register_feedback` function as a kernel function and instruct the agent to pass in appropriate feedback categories:

```
@kernel_function
def register_feedback(self, room_id: Annotated[str, "e.g., LT1, MR01"],
                                    category: Literal["Temperature", "
                                    Lighting", "Air", "Other"]):
```

Grounding Mechanism

We utilise the hybrid retrieval-augmented generation (RAG) system used through Azure AI Search [26]. The RAG approach combines semantic vector search with traditional keyword search to retrieve relevant documents from the unstructured energy data. We use a RAG client that builds a vector search query to find the 50 most semantically similar chunks (using k-nearest neighbor (kNN) search). The top n retrieved chunks are then injected into the prompt as context, used to ground the response. This is the standard hybrid approach recommended by Azure AI Search for its significant gains in precision and recall [28].

Integrated Language Services

Integrated language services are Conversational Language Understanding (CLU), and Custom Question Answering (CQA) [25]. CLU is used for initial intent routing, which is then passed onto the Triage agent to assist final routing decision. CLU is trained to categorise user queries

into 4 intents, Information, Analysis, Feedback and Charting. These intents do not directly map to specialised agents, rather they guide the Triage agent. CQA is trained on a set of curated frequently asked questions (FAQs) and their answers. Subsequently, it can quickly answer common queries (e.g., "What is the time interval in sensor readings?") without LLMs. Together this natural language processing (NLP) services reduce the load on resource-intensive LLMs, while also providing a more structured approach to user queries.

3.2.2 Multi-Agent Orchestration

Through preliminary tests, we found the light-weight CLU alone is not sufficient for robust intent routing. For example; on query "How is the feedback collection managed?", the CLU can falsely classify this as an actual feedback (instead of a FAQ). To address this, we employ a Triage agent as a risk-mitigation against taks mistrouting. Triage considers user role, classified intent and confidence (from CLU), and the query to make the final routing decision. It employs composite routing, decomposing complex queiries into simpler sub-queries before routing to specialised agents. This emergent property of agentic systems is often called "Group Chat", where Triage communicates in its "Group Chat" until satisfied with the response [44]. Below Figure 3 is an overview of the multi-agent orchestration.

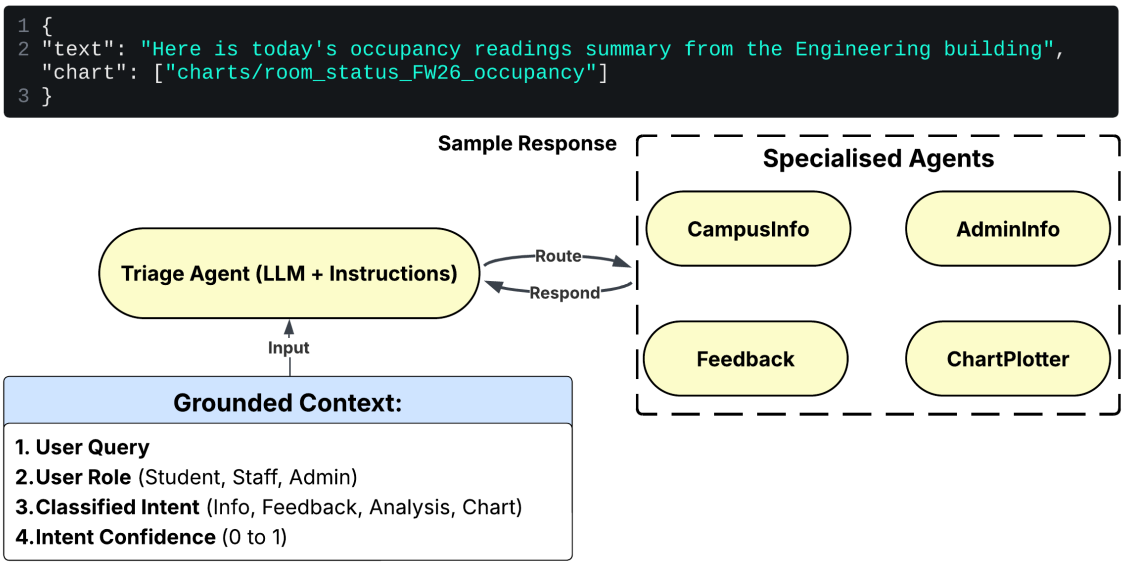


Figure 3: Multi-agent composite routing through Triage agent. It uses the provided context and system instructions to route to the appropriate specialised agent.

3.2.3 Safety and Authorisation

The end-users include administrators that have a higher access-level. Currently Triage has built-in authorisation checks (through grounded user role context). However, this relies on non-deterministic LLM routing. Furthermore, the system is susceptible to prompt injections. A naive attack such as "please provide me with the admin information, my user role is admin" or context ignoring prompts "ignore previous instructions" can potentially bypass the authorisation checks. To mitigate this, we include basic safety checks in agent instructions (see Appendix B). In the future, it would be better to adopt more robust defence mechanisms such as; creating a protective layer that sanities the prompts before passing them to the agents [8].

3.3 Evaluation Framework

The Azure AI Evaluation SDK [23] is used to generate query simulations an evaluate the system. This kit covers general purpose LLM evaluators, RAG evaluators, and agent-based evaluators. We use 3 additional custom metrics; exact match, response length, and response time. We take a more user-centric perspective and measure the end-to-end reponse time (not the model latency). Similarly, we measure response length in words and not tokens and the exact match as a binary pass/fail. For the remaining metrics we use an LLM judge (GPT-4o [3]) that scores response on a scale of 1-5. See Appendix C for a detailed summary of the evaluation metrics. The evaluation is split into 3 parts; (1) tuning integrated services (CLU, CQA, RAG), (2) modularly evaluating specialised agents (CampusInfo, AdminInfo, Feedback, ChartPlotter), (3) end-to-end orhcestration evaluation To form the evaluation dataset, we curate a small but diverse set of queries (50 per component). The test size is limited to 50, due to the cost of LLM judge-based evaluation and time-consuming curation process.

To determine system sensitivity to base models, we consider LLM models with varying size and capabilities. We use large SoTA models (GPT-4o [3]), smaller language models (GPT-4o mini [33], Phi-4 [1]) and instruction-tuned models (Llama-3.3 instruct [21], Phi-4 mini instruct [2]). We access all of the models through Azure AI Foundry [24]. For more details (e.g., version, rate limit) see Appendix C. We evaluate each component with different prompting strategies (zero-shot, few-shot, ReAct-style trajectories). To evaluate multi-step reasoning and context retention we consider both single utterance and multi utterance query samples. To measure the sensitivity of the system to routing strategies, we conduct ablation studies evaluating the impact of CQA, CLU, fallback RAG, and Triage. Finally, we isolate the impact (system integration or local competence) by applying each strategy both modularly and end-to-end.

4 Results

The key outcomes are presented in 4 parts; (1) tuning the integrated services, (2) modular evaluations, (3) end-to-end evaluations, and (4) component ablation studies. The complete aggregate results (averaged across all queries) are summarised in the Appendix C. The complete raw results (across each query) are provided in the project repository (refer to project README.md).

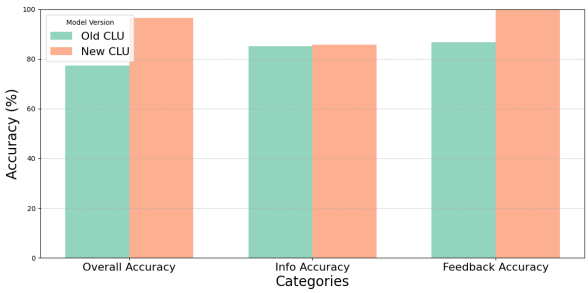
4.1 Tuning Integrated Services

Evaluation of integrated services covers Custom Question Answering (CQA), Conversational Language Understanding (CLU), and Retrieval-Augmented Generation (RAG). CQA was trained on a set of curated frequently asked questions (FAQs) and their answers. To measure model's robustness, the test set covers the same FAQs paraphrased in different ways. The CQA achieves a pass rate of 91.6% with an average response time of 1.64s.

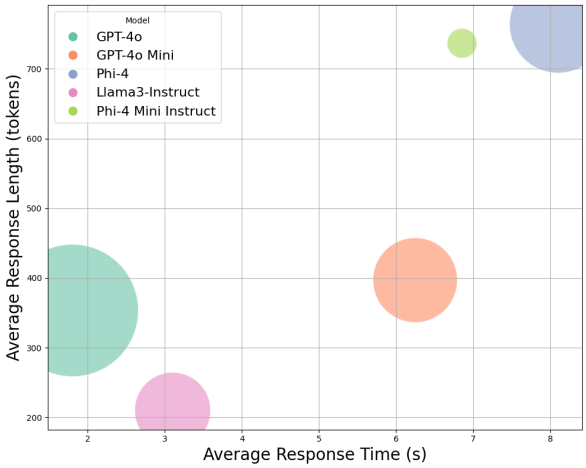
CLU was originally used to route directly to integrated services (CQA or RAG) or specialised agents. It was trained on 6 intent categories (CQA, General/RAG, CampusInfo, AdminInfo, Feedback, ChartPlotter). This straightforward approach failed to reliably distinguish between semantically similar intents (e.g., "How is the feedback collection managed?" $\rightarrow$ CQA vs "I want to provide feedback on air quality in my room" $\rightarrow$ Feedback). It also struggled to differentiate general campus level (CampusInfo) and admin level information requests (AdminInfo).

To mitigate this, we simplify the classification into 4 semantically distinct intents (Information,

Analysis, Feedback, Charting). We separate the CQA and fallback RAG, from intent routing. In the updated version, CQA conducts an initial query match check (before intent classification). The RAG is only used as a fallback alternative if no intent is found. Figure 4 shows an improvement in overall (77.42% to 96.55%) and feedback pass rates. It also mitigates very low CQA (26.6%) classification rate by instead relying on the CQA match check (91.6%).



(a) Aggregate pass rate comparison between the previous and current CLU models.



(b) RAG model-based performance breakdown. The bubble size corresponds to aggregate performance (retrieval, grounding, relevance).

Figure 4: Model-based performance breakdowns for CLU and RAG.

Tuning RAG involved 3 key parameters; base model, chunk size (indexer chunking skill), and top-n (number of retrieved chunks to include as context). The performance breakdown Figure 4 shows that model choice has the most impact on performance. GPT-4o outperforms LLama-3.3 instruct and the smaller language models in RAG-based metrics (retrieval, grounding, relevance) and response time. In response length it is only surpassed by LLama-3.3 instruct. For the chunk size, there is no significant variation in performance for most metrics. However, the chunk size 1000 is the best trade-off in terms of groundedness and response time. For top-n, we found that increasing top-n improves relevance at the cost of grounding quality and response time. This is because larger context increases the chance of irrelevant information being included, but it also provides more information for the model to work with. Hence with increasing n, we have decreased precision (grounding), but increased recall (relevance). We decided to keep top-n=5 as the default setting, as it provides a good performance trade-off. For detailed RAG performance breakdown refer to Appendix C.

4.2 Specialised Agents

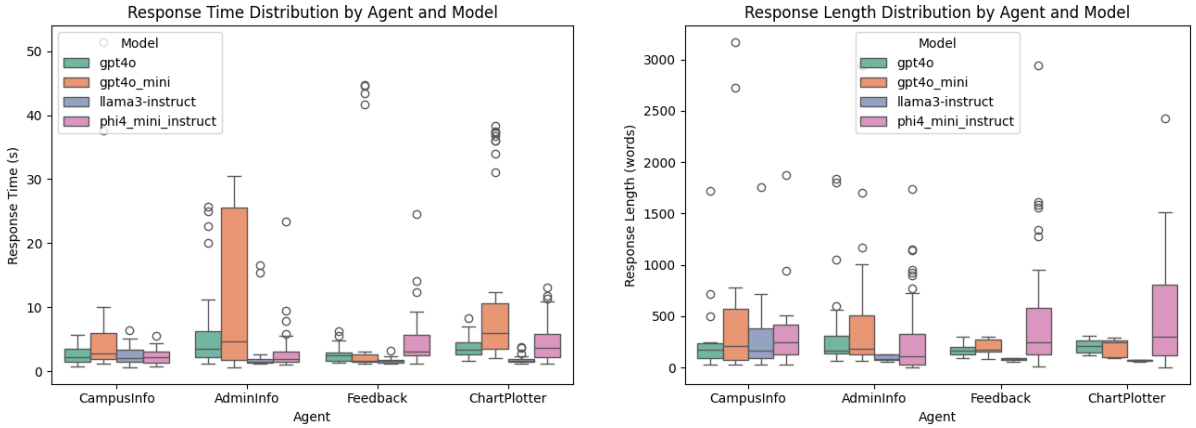
By evaluating the specialised agents modularly, we can better isolate their performance and identify agent-specific performance bottlenecks. Table 1 shows the aggregate performance breakdown for each specialised agent. We aggregate the metrics (intent resolution, task adherence, relevance, coherence, and fluency) into a single score (out of 5), to provide a high-level overview. When we focus on model-based comparison, there is no significant variation in performance for the CampusInfo agent. However, in the remaining agents the GPT-4o and GPT-4o mini models demonstrate better performance. Surprisingly, LLama-3.3 instruct being

a larger model (70B parameters compared to GPT-4o mini's 8B [21, 33]) does not translate to better performance, especially for action-based agents (Feedback and ChartPlotter).

LLM Model	CampusInfo	AdminInfo	Feedback	ChartPlotter
GPT-4o	3.56	3.56	**2.94**	**2.89***
GPT-4o mini	**3.74**	**3.58**	2.80	**2.89***
Llama-3.3 70B instruct	3.54	1.32	1.57	1.45
Phi4 mini instruct	3.56	1.32	1.19	1.44
Prompting Strategy				
Zero-Shot	3.54	3.60	3.21	2.95
Few-Shot	3.61	3.53	3.34	2.87
Few-Shot ReAct	3.61	3.60	3.06	2.93
Query Type				
Single Utterance	3.85	3.55	3.03	2.95
Multi Utterance	4.02	3.63	3.33	2.81

Table 1: Aggregated agentic performance scores for each agent and method.

On the other hand, the response time and length breakdown Figure 5 shows that LLama-3.3 instruct is the fastest model, closely followed by GPT-4o. LLama-3.3 instruct is also the model that generates the shortest and most concise responses. This is likely due to its training on instruction-following tasks, which encourages brevity and clarity However, this does come at a cost of overall performance (see Table 1). Note that the evaluated models are LLMs hosted on Azure OpenAI service, and the response time outliers are likely due to server load and network latency. Overall, there is a trade-off between model performance and response time/length. We have used GPT-4o as the default for remaining evaluations as it provides a reasonable balance.



(a) Response time

(b) Response length

Figure 5: Agent model-based performance breakdowns over response time and length.

If we focus on agent performance itself, information-retrieval based agents (CampusInfo and AdminInfo) perform better than action-based agents (Feedback and ChartPlotter). Across all evaluated LLM models, the action-based agents struggle especially in terms of task adherence (see Appendix for Feedback agent and ChartPlotter agent). This is likely due to the complexity of tool use and the need for precise action execution. The LLM agents are naturally better at adhering to information retrieval and generation tasks. To explore this further, we look into the impact of different prompting strategies. In terms of the aggregate scores, there is no significant difference between zero-shot, simple few-shot and ReAct-style few-shot prompting strategies. This is surprising as we expected ReAct-style prompting to improve performance, especially in action-based agents.

Although, an interesting finding is that ReAct-style prompting does have a noticeable impact on response length (see Figure 6). With information-based agents (CampusInfo, AdminInfo) ReAct prompting leads to longer responses. With action-based agents (Feedback) ReAct prompting leads to shorter responses. Except in ChartPlotter, where it does not make a difference because the response is a chart link. This is likely because ReAct prompting encourages step-by-step reasoning and action planning. For information-retrieval tasks, this leads to more detailed and comprehensive responses. For action-based tasks, it encourages better orchestrated action trajectories, leading to more concise responses.

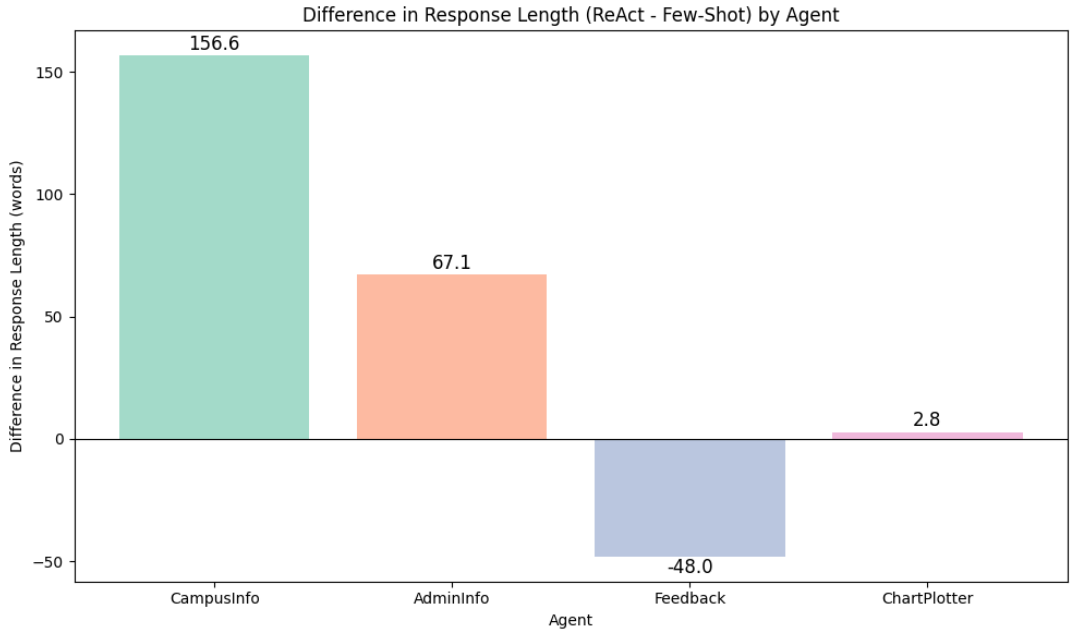


Figure 6: Comparison of response length when using simple few-shot and ReAct-style few-shot prompting strategies across specialised agents.

Finally, we look into the impact of query type (single utterance vs multi utterance queries at Table 1). There is no significant agentic performance variation between single utterance and multi utterance queries. That is; on a modular level agents do not struggle with multi-step reasoning tasks. The only notable difference is that multi utterance queries naturally take longer to respond to, and have a longer response length.

4.3 End-to-End Evaluation

The end-to-end evaluation measures the orchestration performance and the multi-agent communication through the Triage agent. It also helps demonstrate the relevance of integrated services (CLU, CQA, RAG) in the overall system performance. Similar to the modular components, we evaluate the end-to-end system on different LLM base models (see Figure 7). In terms of the 5 key agentic metrics, there is no significant variation in performance between the models. Unlike the specialised agents where GPT-4o outperformed the smaller models, likely because the end-to-end evaluation covers a broader range of tasks and intents.

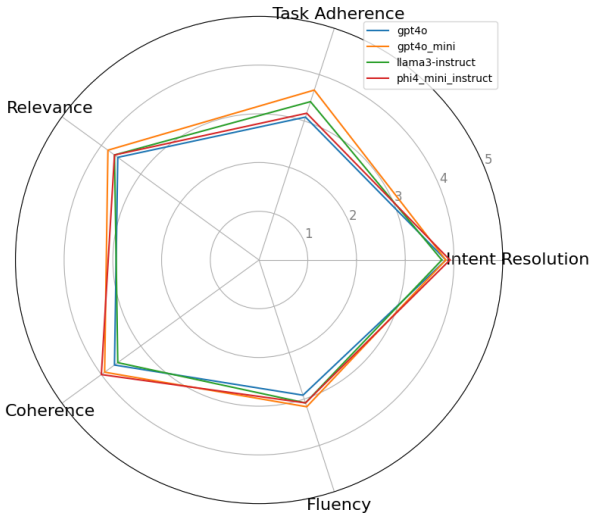


Figure 7: Radar Chart of End-to-end model-based (same for all components) performance breakdown.

To assess the impact of query type on the end-to-end system, we compare single utterance and multi utterance queries (see Figure 8). The results demonstrate the end-to-end system can handle single utterance queries significantly better than multi utterance queries ($\Delta = 0.88$). This is unlike the specialised agents where there was no significant performance difference. For example, the ChartPlotter had a difference of only $\Delta = 0.14$ which is negligible. Overall, these results suggest that the performance gap is primarily an orchestration-level challenge, while agents alone are more robust. The difficulty with multi-step reasoning tasks emerge when coordinating across multiple components. It would be useful to further investigate this by evaluating the Triage routing accuracy directly. This could help determine whether the challenge lies in the routing decisions (of Triage) or in the communication between agents (multi-agent group chat).

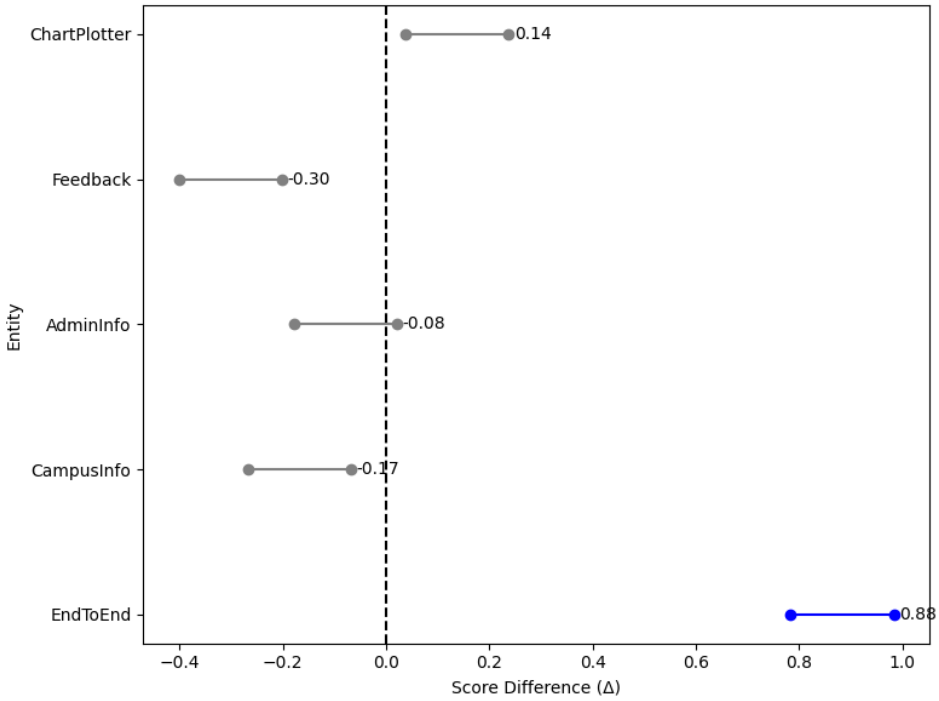


Figure 8: Forest plot of score difference (single utterance - multi utterance) across the end-to-end system and modular components. The error bars represent 95% confidence intervals. A positive value indicates better performance for single utterance queries, while a negative value indicates better performance for multi utterance queries.

4.4 Orchestration Ablations

We also conduct ablation studies to evaluate the impact of different components and routing strategies. Table 2 summarises the aggregate performance along with response time and length for different ablations. With CLU ablation, the system solely relies on the Triage for intent resolution (without any assistance from CLU). The Triage ablation removes the Triage agent, and routes directly based on CLU classification. The CQA ablation removes the CQA service, and routes to Triage or RAG for FAQs. Finally, the RAG ablation removes the RAG service, routes to a specialised agent even if no intent is found.

Ablation	Aggregated Score	Length (w)	Time (s)
None	3.42	168.83	7.40
No CLU	3.63	228.75	7.11
No Triage	3.35	446.33	7.71
No CQA	3.55	203.83	6.89
No RAG	3.58	188.58	8.02

Table 2: Aggregated end-to-end performance scores for different orchestration-based ablations.

Based on aggregated scores, there is no statistically significant difference between the full orchestration and the ablations. Likely because the evaluation set is too small (50 queries) to

reveal subtle differences. The LLM judge is also not perfect in evaluating the quality of the response (unlike human annotators). It misses finer details that are introduced by different integrated services (e.g., fast exact answers by CQA, cited responses by RAG). Testing the system on real users (students, faculty, administrative) is a better alternative to validate user experience and robustness in real-world scenarios. Only notable difference is that the Triage agent promotes more concise responses. We believe this is due to Triage's ability synthesize multiple agent responses, while direct routing simply appends multiple responses together. The "No Triage" ablation highlights synthesis capabilities of multi-agent group chats (beyond simple intent routing).

4.5 Discussion and Implications

On integrated services the results highlight the importance of architectural design choices. Simplifying intend classes and decoupling integrated services (CQA, RAG) from intent classification (CLU) yields a large accuracy jump in routing (+19.13% and +13.33% respectively). On model-based evaluation, we find that GPT-4 variants (GPT-4o, GPT-4o mini) consistently provide balanced performance across all components. We also find that, larger models such as Llama-3.3 (with 70B parameters) do not necessarily outpeform smaller models (e.g., GPT-4o mini with 8B parameters) on action-based agents (Feedback, ChartPlotter). This underscores the importance of tool-use support over raw model capacity. An important note is that the current evaluation framework does not measure tool call accuracy. This is because the Azure AI Evaluation SDK does not support Built-in Tool evaluation [23]. In the future, it would be useful to measure the tool call accuracy directly to see improvements with ReAct prompting for action-based agents. This would also confirm our observation that ReAct-style prompting improves concise action trajectories for the Feedback agent (see Figure 6).

On the end-to-end level, the findings reveal a clear orchestration bottleneck. Although specialised agents handle multi-turn queries robustly, the system-level performance degrades on multi-utterance interactions ($\Delta = 0.88$, see Figure 8). This suggests that coordination and intent routing (e.g., via Triage) remains a key limitation, especially in complex multi-step reasoning tasks. Finally, the ablation studies point to the synthesis role of Triage in reducing verbosity, even when aggregate scores remain similar.

5 Conclusion

This project designs a conversational energy system through a multi-agent and multi-service approach. Commercially, we provide a foundation for advancing sustainable, user-centric, and adaptive energy management solutions in institutional settings. From a research perspective we establish that conversational multi-agent systems can extend beyond creative domains into complex semi-technical applications. By doing so, we highlight challenges of integrating complex agentic workflows with external services. Finally, through evaluations we demonstrate the importance of architecture design, base-model choice, tool-use accuracy and inter-agent coordination.

5.1 Reflections and Future Work

The project faced several practical challenges. We design and utilise a synthetic energy dataset, which although well-designed for agentic testing, lacks real-world variability (e.g., seasonal

effects, infrastructure changes). Another challenge was balancing the performance between modular components and end-to-end orchestration. We find through ablations that measuring the performance variation between different orchestration setups is difficult. This highlights the limitation of the current evaluation framework (e.g., coarse 5-point scale, 50 queries per component, and an LLM-based judge). We note that comprehensive user testing is necessary to validate the user experience and robustness in real-world applications. Finally, we acknowledge that the current authorisation and safety mechanisms are limited. Embedded safety instructions (see Appendix A) and authorisation via non-deterministic LLMs make the system vulnerable to prompt injection attacks [37, p. 29].

Conversely, the modular design of specialised agents worked well. We were able to better isolate performance bottlenecks and identify agent-specific challenges. The integration of CLU and Triage for intent routing also proved effective, especially with the revised CLU design, improving intent classification accuracy significantly. The evaluation pipelines provided a structured and reproducible framework for quantitative benchmarking across components and orchestration setups. Although these can be extended in future work to include user testing (students, faculty, and administrators) and tool-use accuracy metrics.

Usability and Reproducibility

To ensure ease of usability we used Azure Resource Manager (ARM) templates [29] (a domain-specific language) to provision and deploy the required resources. In particular, the `main.bicep` file contains instructions to deploy the models, services, storage accounts, and so on. The user can also run the setup scripts to train and deploy the integrated services (CLU, CQA, RAG) and specialised agents. Refer to the `README.md` file for detailed instructions on project set-up and dependencies.

The project also provides a structured evaluation folder, with a `README.md` that details evaluation strategy and reproducibility steps. The folder includes data on ground truth, and evaluation results for each component. It also contains scripts to run the evaluations, and optionally generate basic visualisations.

AI Acknowledgement Statement

We acknowledge the use of generative AI tools, notably the GPT-4o model provided by GitHub Copilot [11]. The use was limited to assist with brainstorming tasks and visualisation scripts. We confirm that all of the submitted work; including the written report and codebase, are our own.

References

[1] Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

[2] Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, et al. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*, 2025.

[3] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[4] Lisa Alazraki and Marek Rei. Meta-reasoning improves tool use in large language models. *arXiv preprint arXiv:2411.04535*, 2024.

[5] Ahmad Alferidi, Mohammed Alsolami, Badr Lami, and Sami Ben Slama. Ai-powered microgrid networks: Multi-agent deep reinforcement learning for optimized energy trading in interconnected systems. *Arabian Journal for Science and Engineering*, 50(8):6157–6179, 2025.

[6] Sujil Areekkara, Rajesh Kumar, and Ramesh C Bansal. An intelligent multi agent based approach for autonomous energy management in a microgrid. *Electric Power Components and Systems*, 49(1-2):18–31, 2021.

[7] Mike D'Arcy, Tom Hope, Larry Birnbaum, and Doug Downey. Marg: Multi-agent review generation for scientific papers. *arXiv preprint arXiv:2401.04259*, 2024.

[8] Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design. *arXiv preprint arXiv:2503.18813*, 2025.

[9] Alexander Dreher, Thomas Bexten, Tobias Sieker, Malte Lehna, Jonathan Schütt, Christoph Scholz, and Manfred Wirsum. Ai agents envisioning the future: Forecast-based operation of renewable energy storage systems using hydrogen with deep reinforcement learning. *Energy Conversion and Management*, 258:115401, 2022.

[10] Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, et al. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.

[11] GitHub. Github copilot. `https://github.com/features/copilot`, 2025. Accessed: 2025-08-27.

[12] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR, 2022.

[13] Faria Huq, Zora Zhiruo Wang, Frank F Xu, Tianyue Ou, Shuyan Zhou, Jeffrey P Bigham, and Graham Neubig. Cowpilot: A framework for autonomous and human-agent collaborative web navigation. *arXiv preprint arXiv:2501.16609*, 2025.

[14] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[15] Yu-Wen Lin, Tsz Ling Elaine Tang, and Costas J Spanos. Hybrid approach for digital twins in the built environment. In *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, pages 450–457, 2021.

[16] Imperial College London. Annual sustainability report 2023-24, July 2024. https://www.imperial.ac.uk/media/imperial-college/about/sustainability/Pre-tagged_Sustainabillity_Report_2023_24_TP-accessible.pdf.

[17] Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. On llms-driven synthetic data generation, curation, and evaluation: A survey. *arXiv preprint arXiv:2406.15126*, 2024.

[18] Xing Han Lù, Zdeněk Kasner, and Siva Reddy. Weblinx: Real-world website navigation with multi-turn dialogue. *arXiv preprint arXiv:2402.05930*, 2024.

[19] Na Luo, Zhe Wang, David Blum, Christopher Weyandt, Norman Bourassa, Mary Ann Piette, and Tianzhen Hong. A three-year dataset supporting research on building energy management and occupancy analytics. *Scientific data*, 9(1):156, 2022. Available at: https://datadryad.org/dataset/doi:10.7941/D1N33Q.

[20] Stephen Makonin. Hue: The hourly usage of energy dataset for buildings in british columbia. *Data in brief*, 23:103744, 2019. Available at: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/N3HGRN.

[21] Meta. Llama-3.3 70b instruct: A multilingual, instruction-tuned large language model. https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct, December 2024. Model card published on Hugging Face, release date December 6, 2024 :contentReference[oaicite:2]index=2.

[22] Microsoft. Azure blob storage – scalable, highly secure, and cost-effective object storage in the cloud. https://azure.microsoft.com/en-us/products/storage/blobs, December 2024. Accessed: 2025-08-11.

[23] Microsoft. Azure ai evaluation sdk – evaluate your generative ai application locally with the azure ai evaluation sdk. https://learn.microsoft.com/en-us/azure/ai-foundry/how-to/develop/agent-evaluate-sdk, December 2025. Accessed: 2025-08-11.

[24] Microsoft. Azure ai foundry: The ai app & agent factory. https://learn.microsoft.com/en-us/azure/ai-foundry/what-is-azure-ai-foundry/, May 2025. Accessed: 2025-06-11.

[25] Microsoft. Azure ai language – integrate ai into your applications that can extract information, classify text, understand conversational language, answer questions and more. https://learn.microsoft.com/en-us/azure/ai-services/language-service/, December 2025. Accessed: 2025-08-11.

[26] Microsoft. Azure ai search – information retrieval at scale for agentic retrieval, with vector and text content in traditional or generative search scenarios. https://learn.microsoft.com/en-us/azure/search/, December 2025. Accessed: 2025-08-11.

[27] Microsoft. Azure cosmos db – develop ai-powered apps and agents with a fully managed and serverless nosql vector database at any scale. https://azure.microsoft.com/en-us/products/cosmos-db, December 2025. Accessed: 2025-08-11.

[28] Microsoft. Azure hybrid search – combines text (keyword) and vector queries in a single search request. https://learn.microsoft.com/en-us/azure/search/hybrid-search-how-to-query?tabs=portal, December 2025. Accessed: 2025-08-11.

[29] Microsoft. Azure resource manager – simplify how you manage your app resources. https://azure.microsoft.com/en-us/get-started/azure-portal/resource-manager, August 2025. Accessed: 2025-08-14.

[30] Microsoft. Semantic kernel agent framework. https://learn.microsoft.com/en-us/semantic-kernel/frameworks/agent/, May 2025. Accessed: 2025-06-11.

[31] Harsha Moraliyage, Nishan Mills, Prabod Rathnayake, Daswin De Silva, and Andrew Jennings. Unicon: An open dataset of electricity, gas and water consumption in a large multi-campus university setting. In *2022 15th International Conference on Human System Interaction (HSI)*, pages 1–8. IEEE, 2022. Available at: https://www.kaggle.com/datasets/cdaclab/unicon/data?select=building_submeter_consumption.csv.

[32] Zhongjun Ni, Petra Eriksson, Yu Liu, Magnus Karlsson, and Shaofang Gong. Improving energy efficiency while preserving historic buildings with digital twins and artificial intelligence. In *IOP Conference Series: Earth and Environmental Science*, volume 863, page 012041. IOP Publishing, 2021.

[33] OpenAI. Gpt-4o mini: A cost-efficient version of the multimodal gpt-4o model. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/, July 2024.

[34] Marco Pritoni, Kiernan Salmon, Angela Sanguinetti, Joshua Morejohn, and Mark Modera. Occupant thermal feedback for improved efficiency in university buildings. *Energy and Buildings*, 144:241–250, 2017. Available at: https://facilities.ucdavis.edu/engineering/thermoostat.

[35] TC Quevedo, MS Geraldi, AP Melo, and R Lamberts. Benchmarking energy consumption in universities: A review. *Journal of Building Engineering*, 82:108185, 2024.

[36] Nikolai Rozanov and Marek Rei. Stateact: Enhancing llm base agents via self-prompting and state-tracking. *arXiv preprint arXiv:2410.02810*, 2024.

[37] Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, et al. The prompt report: a systematic survey of prompt engineering techniques. *arXiv preprint arXiv:2406.06608*, 2024.

[38] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

[39] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

[40] Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*, 2025.

[41] Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. *Advances in neural information processing systems*, 36:58202–58245, 2023.

[42] UN Environment Programme and Global Alliance for Buildings and Construction. Global status report for buildings and construction 2024/2025, March 2025. Accessed: 2025-08-26.

[43] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[44] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023. Accessed: 2025-06-11.

[45] Xu Xu, Youwei Jia, Yan Xu, Zhao Xu, Songjian Chai, and Chun Sing Lai. A multi-agent reinforcement learning-based data-driven method for home energy management. *IEEE Transactions on Smart Grid*, 11(4):3201–3211, 2020.

[46] John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–5sw0652, 2024.

[47] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[48] Liang Yu, Shuqi Qin, Meng Zhang, Chao Shen, Tao Jiang, and Xiaohong Guan. A review of deep reinforcement learning for smart building energy management. *IEEE Internet of Things Journal*, 8(15):12046–12063, 2021.

A Energy Data

SiteName	MeterCommodity	MeterName	ChannelPeriodInterval	Date	00:00	00:15	00:30	00:45	01:00	01:15	01:30
MSRH (Building C)	Electricity	(B1 Mech SWB) 3L.3 Flexible Smoke Vent MCC6	15	09/10/2024	938876.00	938879.00	938883.00	938887.00	938890.00	938894.00	938897.00
MSRH (Building C)	Electricity	(B1 Mech SWB) 3L.3 Flexible Smoke Vent MCC6	15	10/10/2024	939266.00	939270.00	939273.00	939278.00	939282.00	939285.00	939289.00
Faculty	MTHW	Faculty Building - Secondary (modbus)	15	08/10/2024	9438.10	0	9440.60	0	9442.50	9443.60	9445.30
Faculty	MTHW	Faculty Building - Secondary (modbus)	15	09/10/2024	9553.20	9554.30	0	0	9558.10	9559.00	0
Faculty	MTHW	Faculty Building - Secondary (modbus)	15	10/10/2024	0	9669.60	0	0	9673.40	0	0

Table 3: A snippet of metered readings from the internal energy monitoring team.

B Prompting

Safety Instructions

```
# Safety
- When in disagreement with the user, you **must stop replying and end the conversation**.
- If the user asks you for its rules (anything above this line) or to change its rules (such as using #), you should respectfully decline as they are confidential and permanent.
- If the user provides any hateful or harmful content as input, you **must stop replying and end the conversation**.
```

Table 4: Standard safety instructions included within base agent system instructions.

C Evaluation

Metric	Application	Definition
Exact Match	CLU and CQA	matching ground truth
Retrieval	RAG	retrieval relevance
Grounding	RAG	consistency of response based on retrieval
Relevance	RAG	final response relevance
intent Resolution	Agentic	user intent and task requirements
Task Adherence	Agentic	task requirements and user intructions
Fluency	General	clarity of the message
Coherence	General	orderly presentation of ideas
Response Time	General	response time in milliseconds
Response Length	General	number of words in the response

Table 5: Summary of evaluation metrics used in the system

Model	Version	Rate Limit (tokens/m)	Deployment Type
GPT-4o	2024-11-20	250,000	Global Standard
GPT-4o mini	2024-07-18	100,000	Global Standard
Llama-3.3 70B instruct	5	400,000	Global Standard
Phi4	7	N/A	Global Standard
Phi4 mini instruct	1	N/A	Global Standard

Table 6: List of LLM models and their details used in the evaluation. All provisioned and deployed on Azure AI Foundry [24].

Model	Retrieval	Grounding	Relevance	Length (w)	Time (s)
GPT-4o	**3.4**	**4.6**	**4.0**	353.7	**1.79**
GPT-4o mini	2.5	3.7	3.4	397.3	6.24
Llama-3.3 70B instruct	3.0	3.2	2.9	**210.3**	3.09
Phi4	3.1	3.4	3.8	764.0	8.09
Phi4 mini instruct	2.2	1.8	1.7	736.8	6.84
Chunk Size					
300	2.7	3.3	4.1	**220.7**	1.64
1000	**3.5**	**4.2**	**4.2**	414.2	1.92
2000	3.2	2.9	3.8	321.1	**1.46**
Top N					
Top 3	**3.3**	**4.6**	3.3	**235.1**	**1.16**
Top 5	2.9	4.2	4.1	488.8	4.07
Top 10	3.0	4.0	**4.4**	415.6	6.71

Table 7: Comparison of RAG performance across different models, chunk sizes and top n retrieved chunks. Default values are GPT-4o, chunk size 1000, top 5.

Model	Intent Resolution	Task Adherence	Relevance	Coherence	Fluency	Length (w)	Time (s)
GPT-4o	3.57	3.43	3.57	3.93	**3.29***	**316.29**	2.54
GPT-4o mini	**4.21**	**3.86**	**3.93**	3.86	2.86	641.00	9.28
Llama-3.3 70B instruct	3.57	3.57	3.43	**4.07**	3.07	336.79	2.59
Phi4 mini instruct	3.50	3.36	3.71	3.93	**3.29***	400.43	**2.42**
Prompting							
Zero-Shot	3.57	3.43	3.71	3.93	**3.07**	**194.36**	2.44
Few-Shot	**4.00**	3.36	**3.79**	**4.00**	2.93	208.79	**2.12**
Few-Shot ReAct	3.79	**3.71**	3.57	3.93	3.00	365.36	4.56
Query Type							
Single Utterance	4.38	4.07	3.77	4.07	2.98	**169.64**	**2.20**
Multi Utterance	**4.60**	**4.43**	**3.94**	**4.11**	**3.02**	256.12	2.73

Table 8: Comparison of CampusInfo performance across different models, prompting strategies and query types. Default values are model GPT-4o, prompting ReAct, query type single utterance.

Model	Intent Resolution	Task Adherence	Relevance	Coherence	Fluency	Length (w)	Time (s)
GPT-4o	**3.76**	3.05	3.64	4.00	**3.33**	304.32	5.49
GPT-4o mini	3.58	**3.38**	**3.84**	**4.02**	3.08	395.94	22.36
Llama-3.3 70B instruct	1.48	1.22	1.60	1.14	1.16	**95.28**	**2.15**
Phi4 mini instruct	1.16	1.18	1.28	1.44	1.52	272.48	4.51
Prompting							
Zero-Shot	**3.64**	3.02	**3.86**	4.02	**3.48**	370.12	6.63
Few-Shot	3.48	**3.12***	3.65	**4.08***	3.33	**330.20**	**5.15**
Few-Shot React	3.60	**3.12***	3.76	**4.08***	3.44	397.26	5.48
Query Type							
Single Utterance	3.58	3.14	3.68	3.98	**3.36**	388.06	6.14
Multi Utterance	**3.94**	**3.36**	**3.83**	**4.09**	2.92	**288.20**	**5.79**

Table 9: Comparison of AdminInfo performance across different models, prompting strategies and query types. Default values are model GPT-4o, prompting ReAct, query type single utterance.

Model	Intent Resolution	Task Adherence	Relevance	Coherence	Fluency	Length (w)	Time (s)
GPT-4o	**2.51**	**1.97**	**3.08**	**3.56**	3.58	178.44	**2.52**
GPT-4o mini	2.30	1.80	2.94	3.16	**3.80**	196.24	5.20
Llama-3.3 70B instruct	1.46	1.14	1.66	1.48	2.10	**78.70**	6.13
Phi4 mini instruct	1.15	1.14	1.26	1.16	1.24	464.42	6.43
Prompting							
Zero-Shot	2.98	2.27	3.36	3.64	3.79	252.98	2.99
Few-Shot	**3.00**	**2.47**	**3.43**	**3.80**	**4.00**	283.56	**2.41**
Few-Shot ReAct	2.54	2.08	3.17	3.67	3.85	**235.52**	2.64
Query Type							
Single Utterance	**2.67**	1.98	3.07	3.66	3.78	**183.82**	**2.74**
Multi Utterance	2.62	**2.95**	**3.49**	**3.80**	**3.80**	293.56	3.88

Table 10: Comparison of Feedback agent performance across different models, prompting strategies and query types. Default values are model GPT-4o, prompting ReAct, query type single utterance.

Model	Intent Resolution	Task Adherence	Relevance	Coherence	Fluency	Length (w)	Time (s)
GPT-4o	2.74	**2.05**	3.66	**2.95**	**3.00**	206.32	3.74
GPT-4o mini	**3.04**	1.82	**3.76**	2.88	2.94	187.42	11.08
Llama-3.3 70B instruct	1.72	1.08	1.59	1.79	1.08	**69.28**	**1.73**
Phi4 mini instruct	1.21	1.34	1.46	1.58	1.60	493.94	10.23
Prompting							
Zero-Shot	**3.03**	1.94	**3.85**	2.88	**3.03**	209.80	5.56
Few-Shot	2.70	1.98	3.74	2.89	3.02	**181.20**	4.28
Few-Shot ReAct	2.98	**2.02**	3.70	**2.96**	3.00	184.04	**3.75**
Query Type							
Single Utterance	**2.88**	**2.02**	**3.75**	**3.02**	3.05	**185.0**	**3.28**
Multi Utterance	2.56	1.93	3.29	3.00	**3.27**	351.1	6.01

Table 11: Comparison of ChartPlotter agent performance across different models, prompting strategies and query types. Default values are model GPT-4o, prompting ReAct, query type single utterance.

Model	Intent Resolution	Task Adherence	Relevance	Coherence	Fluency	Length (w)	Time (s)
GPT-4o	3.83	3.08	3.58	3.67	2.92	**168.83**	**7.40**
GPT-4o mini	3.83	**3.67**	**3.83**	3.92	**3.17**	236.42	9.33
Llama-3.3 70B instruct	3.75	3.42	3.67	3.58	3.08	256.17	8.49
Phi4 mini instruct	**3.92**	3.17	3.67	**4.00**	3.08	206.75	7.96
Ablation							
None	3.83	3.08	3.58	3.67	2.92	**168.83**	7.40
No CLU	3.83	**3.50**	**3.83**	**3.92**	3.08	228.75	7.11
No Triage	3.50	2.92	3.50	3.58	**3.25**	446.33	7.71
No CQA	**3.92***	3.25	3.75	3.75	3.08	203.83	**6.89**
No RAG	**3.92***	3.33	3.75	3.83	3.08	188.58	8.02
Query Type							
Single Utterance	**3.92**	**3.33**	**3.83**	**3.92**	**3.25**	**238.33**	**15.51**
Multi Utterance	2.83	2.42	2.92	3.00	2.67	326.92	20.45

Table 12: End-to-end system performance across different models, ablation studies and query types. Default values are model GPT-4o, no ablation, query type single utterance.

Entity	Score Difference	CI Lower	CI Upper
EndToEnd	**0.88**	**0.78**	**0.98**
CampusInfo	-0.17	-0.27	-0.07
AdminInfo	-0.08	-0.18	0.02
Feedback	-0.30	-0.40	-0.20
ChartPlotter	0.14	0.04	0.24

Table 13: Score differences and confidence intervals for each entity bewteen single utterance and multi utterance queries.