PEC-Home: Interpretation of Progressively Elliptical Commands in Smart Homes

Yingyu Shan[1], Zeming Liu[2], Silin Li[1], Boao Qian[1], Jiashu Yao[1], Yuhang Guo[1†], Haifeng Wang[3]
[1]Beijing Institute of Technology [2]Beihang University [3]Baidu Inc.
[†]Corresponding author Email: shanyingyu@bit.edu.cn, guoyuhang@bit.edu.cn

Abstract

Recent advancements in Large Language Models (LLMs) have empowered home assistants with natural language interaction capabilities. However, current assistants overlook the progressive omission that occurs in human dialogue as shared context accumulates, leading to more elliptical expressions for efficient communication. Thus, current assistants still struggle to interpret such elliptical expressions accurately, which limits their effectiveness in real-world applications. In practical smart home scenarios, assistants face two major challenges caused by elliptical commands: (1) referential ambiguity caused by different environmental expectations among multiple users; and (2) intention ambiguity resulting from user preferences that evolve over time or change with the environment. To address these challenges, we introduce PEC-Home, the first simulated home dataset specifically designed for interpreting progressively elliptical commands in smart homes. Extensive experiments on various LLMs, including GPT-4o, show that existing home assistants struggle to execute user-intended operations based solely on elliptical commands. Even when equipped with tools for storing and retrieving user dialogue history, execution accuracy remains below that achieved with complete commands. Our code and dataset are available at https://github.com/BITHLP/PEC-Home.

1 Introduction

Home assistants automate routine household tasks and enhance user interaction through intuitive, context-aware support, seamlessly integrating into daily life (Weiser, 1999). Specifically, their automation leads to increased convenience (Ur et al., 2014), optimized energy usage (Sepasgozar et al., 2020; Gupta et al., 2020) and so on.

In the pre-LLM era, home assistants primarily focused on executing predefined rules and pattern recognition tasks like household routine automation (Dey et al., 2006; Ur et al., 2014) and activity prediction (Tax, 2018; Kim et al., 2017; Khraief et al., 2019). While these assistants achieved home automation, they fundamentally lacked the capability to interact with users through natural language.

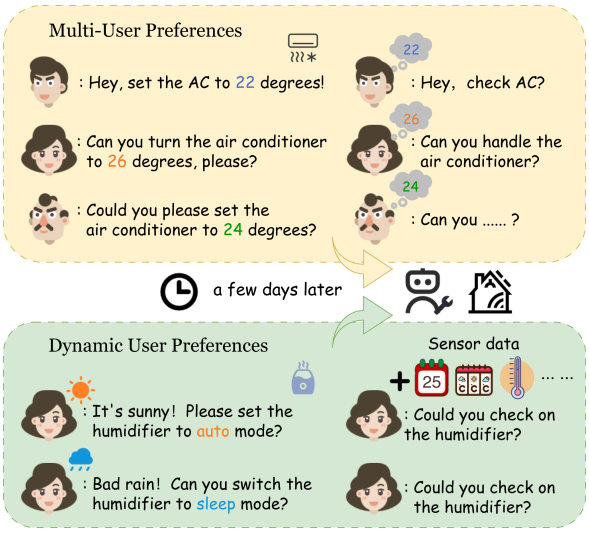

Figure 1: An example of PEC-Home. Multi-User Preferences presents the referential ambiguity from conflicting "comfortable temperature" definitions between family members and Dynamic User Preferences indicates the intention ambiguity caused by environment changes.

The advent of LLMs (Vaswani et al., 2017) revolutionized home assistants by enabling natural language understanding across domains (Achiam et al., 2023; Touvron et al., 2023; DeepSeek-AI et al., 2024). Recent LLM-based home assistants like Sasha (King et al., 2024) and SAGE (Rivkin et al., 2024) demonstrate improved device control capability facing commands like "Make it less chilly". However, these LLM-based assistants operate under static, single-user assumptions or oversimplify dynamic user command shifts as merely explicit or ambiguous states, disregarding the progressive shift from explicit to elliptical

Findings of the Association for Computational Linguistics: ACL 2026, pages 19989–20006
July 2-7, 2026 ©2026 Association for Computational Linguistics

through long-term interactions. Cognitive science researches have shown that humans naturally develop idiosyncratic and increasingly elliptical conventions with familiar partners over repeated interactions (Krauss and Weinheimer, 1964; Zwaan and Radvansky, 1998; Hawkins et al., 2020). By neglecting this phenomenon, current home assistants fall short in handling naturally elliptical commands, preventing users from interacting with them as efficiently and intuitively as they would with human partners. This limitation fundamentally undermines home assistants' ultimate goal of providing intuitive, context-aware support.

To address this limitation, we introduce PEC-Home, the first simulated home dataset that is specifically designed for simulating progressively elliptical commands in smart homes. The interpretation of progressively elliptical commands encounters two core challenges in practical scenarios (as shown in Figure 1): referential ambiguity arising from conflicts in multi-user preferences (e.g., conflicting "comfortable temperature" definitions between family members), and intention ambiguity in dynamic user preferences, where previously explicit commands (e.g., "set the humidifier to auto mode") gradually lose specificity as the environment changes. PEC-Home comprises 1,780 dialogues from 1,424 personas (Zhang et al., 2018), providing a novel and more practical perspective for enhancing and evaluating the performance of LLM-based home assistants.

To evaluate the effectiveness current LLM-based home assistants in interpreting progressively elliptical commands, we conduct extensive experiments on PEC-Home. We investigate a range of LLMs using both zero-shot prompting and in-context learning to assess their ability to execute user-intended operations based solely on elliptical commands. The results show that none of the evaluated LLMs are able to reliably interpret these elliptical instructions. Furthermore, we enhance LLMs with external tools, retrieval-augmented generation (RAG) for accessing dialogue history, or fine-tuning LLMs aimed at improving command interpretation capability. However, our experiments demonstrate that even state-of-the-art models such as GPT-4o fail to maintain their performance achieved on complete commands when handling elliptical ones, highlighting the limitations of existing methods.

- To the best of our knowledge, we are the first to identify the task of progressively elliptical commands interpretation in human–home assistant interactions.
- To facilitate the study of this task, we introduce PEC-Home, the first simulated home dataset modeling the progressive shift from explicit to elliptical commands.
- Our experimental results on 10 distinct LLMs demonstrate that all models experience substantial performance drops when interpreting progressively elliptical commands. Even with enhancements such as tool integration, RAG, and fine-tuning, these models including GPT-4o still fail to achieve reliable execution accuracy on highly elliptical commands.

2 Related Work

2.1 Pre-LLM Home Automation Systems

Home assistants in the pre-LLM era were essentially home automation systems that relied on rule-based or machine learning algorithms. Among the earliest approaches, Dey et al. (2006) proposed iCAP, a rule-based system enabling users to create rules to automate home devices. Similarly, Ur et al. (2014) explored trigger-action programming, enabling users to create custom automation rules (e.g., "If it is 6 p.m., then turn the lights on").

The advancement of machine learning has enabled home automation systems to integrate diverse algorithms, including SVM for emotion-based automation (Jaihar et al., 2020), CNN for elderly fall detection (Khraief et al., 2019), and LSTM for next-activity prediction in multi-user environments (Kim et al., 2017). Suman et al. (2022) investigated how RL-based smart homes can influence human behavior. Similarly, Gupta et al. (2020) proposed a multi-objective RL framework aiming to optimize power consumption in smart homes.

PEC-Home fundamentally differs from pre-LLM automation systems by shifting from rule-based automation and single-task pattern recognition to natural language driven progressive elliptical commands resolution.

2.2 LLM-Based Home Assistants

Humans often bridge abstract concepts (e.g., 'comfort') and device-specific actions (e.g., 'turning on the air conditioner') through intuitive semantic associations (King et al., 2024). LLMs demonstrate the ability to understand the underlying concrete actions intended by humans (He et al., 2024). This

Ellipsis Level	Example	Components
Level 1	"Hey, could you turn on the bedroom air conditioner and set the temperature to 26 degrees at 10:00 PM tonight? I'm heading to bed soon and want it comfortable."	*All components clear*
Level 2	"Hey, can you set the air conditioner to 26 degrees at 10:00 PM? Getting ready for bed and want it nice and cool."	*Room omitted*
Level 3	"Can you do something with the air conditioner before I sleep? Make it comfortable for tonight."	*Operation & Parameters omitted*
Level 4	"Hey, you know the air conditioner, right? Could you take care of it for me tonight?"	*Only device specified*

Components: room device operation operation parameters

Figure 2: Examples of progressively elliptical user commands across four levels (Lv1–Lv4) illustrate the defined standards and the systematic omission of four core components (room, device, operation, and operation parameters). We color room, device, operation, and operation parameters. We emphasize that the examples in the figure are simplified for better understanding.

capability has inspired recent works to integrate LLMs in resolving ambiguous commands.

Sasha (King et al., 2024), an LLM-based smart home assistant that generates action plans for underspecified user commands through iterative reasoning. Rivkin et al. (2024) proposes an LLM-based agent SAGE that dynamically generates prompt trees and toolchains to flexibly handle user requests. Yin et al. (2024) introduces Harmony, a framework leveraging the locally deployable LLM to address user needs. AwareAuto (Shi et al., 2024), the first end-user programming system leveraging LLMs to bridge user expressions with smart home automation.

In contrast to prior research focus on ambiguous commands, our work identifies the task of progressively elliptical commands interpretation, which is intuitively emerging through long-term interactions.

3 Problem Definition

The ultimate task of home assistants is to interpret a user's command u_t at turn t within a smart home environment H. Each command, which can be explicit or elliptical, must be accurately mapped to a specific, executable assistant response a_t. This response typically takes the form $r_i.d_j.m_k(\theta)$, indicating a target room r_i, a device-method $d_j.m_k$, and parameters θ.

While baseline approaches (e.g., prompt-based methods) perform this mapping using only the current command and home state, $f_{\text{baseline}} : (u_t, H) \rightarrow a_t$, the challenge of interpreting elliptical commands necessitates leveraging contextual information. Primarily, this involves the dialogue history, $\mathcal{C} = \{(u_1, a_1), \ldots, (u_{t-1}, a_{t-1})\}$, leading to advanced interpretation approaches such as $f_{\text{advanced}} : (u_t, \mathcal{C}, H) \rightarrow a_t$.

In practical smart home scenarios, assistants face two major challenges caused by elliptical commands: (1) Multi-User Preferences, which primarily involves mapping $f_{\text{multi}}(u_t, \mathcal{C}_l, H) \rightarrow a_t$ using selected user-specific dialogue histories $\mathcal{C}_l$, and (2) Dynamic User Preferences, which adapts to environmental states $\mathcal{C}_l^t$ and sensor data S_t via the function $f_{\text{dyn}}(u_t, \mathcal{C}_l^t, H, S_t) \rightarrow a_t$. We provide more detailed problem definition in the Appendix C.1.

4 PEC-Home

PEC-Home (**P**rogressively **E**lliptical **C**ommands Dataset for Smart Homes) comprises 1,780 dialogues involving 1,424 unique personas from PersonaChat (Zhang et al., 2018). Each dialogue includes a sequence of 4 chats that become progressively more elliptical. PEC-Home is designed to capture and simulate the progressive ellipsis of user commands over collaborative communication, which is a well-studied phenomenon in cognitive science (Krauss and Weinheimer, 1964; Clark and Wilkes-Gibbs, 1986; Hawkins et al., 2020). To capture different aspects of this progressive shift from explicit to elliptical, PEC-Home is divided

into two parts: 1) **Multi-User Preferences** and 2) **Dynamic User Preferences**. In this section, we provide a comprehensive description of the collection process of PEC-Home, a detailed comparison with existing datasets, and a thorough presentation of its statistics.

4.1 Dataset Collection

Virtual Environment Construction We constructed a virtual home environment that includes 12 types of devices, covering common device categories such as lighting, humidifiers, etc. Each device is equipped with multiple personalized executable methods, totaling over 50 distinct methods. A comprehensive list of all devices and their executable methods is provided in the Appendix B.1. To simulate real-world scenarios, these devices are distributed across multiple rooms in our virtual environment. A detailed list of devices in each room is available in the Appendix B.1. In total, more than 350 personalized methods are allocated to different rooms, ensuring diversity and comprehensive scenario coverage in our dataset.

User Commands Generation Based on this virtual environment, we first generated function-call style device operations. Based on above, we utilized unique personas for each user and generated personalized parameters aligned with their preferences. For multi-user preferences, we randomly selected three distinct users to form a household. For dynamic user preferences, we generated two preference parameters for each user within an device method, each corresponding to different environmental states. We ensured that all operations included memorable personalized parameters.

The increasing ellipsis in PEC-Home's user commands is designed to reflect patterns observed in real human communication. Prior study (Krauss and Weinheimer, 1964) have shown that when interlocutors repeatedly refer to the same object, their references becomes more elliptical in collaborative process. Initial detailed descriptions are gradually shortened by omitting elements which mutually established within the shared context, converging on brief labels. To capture this progressive shift from explicit to elliptical commands in user-assistant interactions, PEC-Home's ellipsis levels are constructed by gradually reducing the amount of explicit information in commands. Inspired by King et al. (2024), which identifies four core components of smart home commands:**room**, **device**, **operation**, and **operation parameters** (e.g., 'brightness set to 3'), we define four ellipsis levels by omitting these components. To ensure data quality and minimize manual effort, we utilize GPT-4o to generate natural language commands based on function-call style operations and user personas, ensuring clarity, grammatical accuracy, and alignment with casual conversation style. The four defined levels are:

- **Level 1:** The user command includes all four core components.
- **Level 2:** As fundamental contextual elements such as time and space are usually established before introducing events in human communication (Zwaan and Radvansky, 1998), we assume users first establish spatial location during interactions. Following prior work on conceptual pacts and common ground (Brennan and Clark, 1996; Clark and Wilkes-Gibbs, 1986), once this spatial reference is mutually established through previous commands, repeating it for subsequent commands within the same space becomes redundant. Thus, we omit the **room** information at this level.
- **Level 3:** Previous research (Carroll, 1980) has found that descriptive modifiers in user language are gradually omitted over time. After removing location information, the remaining modifiers mainly include explicit **operation** and **operation parameters**. Therefore, we omit these descriptive components at this level.
- **Level 4:** Following the findings that interactions often converge towards idiosyncratic, shared references (Krauss and Weinheimer, 1964), commands at this level only explicitly specify the `device`, simulating the ultimate reference collaborators have achieved.

The prompt used to guide LLMs in generating commands at each level was manually reviewed and adjusted to ensure generated commands met the defined standards. Detailed examples of progressively elliptical commands are provided in Figure 2, and specific prompts used to guide GPT-4o in generating commands are detailed in Appendix B.2.

Quality Assessment Inspired by Zeng et al. (2024), we apply manual sampling and validation procedures to validate the reliability of our operation generation process. Three graduate researchers conducted a manual evaluation of 500 randomly selected function-call style device operations. The results demonstrated that 100% of the operations were correct, and 96% were properly

Dataset	Prog. Ellipsis	Multi Pref.	Dynamic Pref.	Long-Inter.	Persona	Pers. Size	Total
IFTTT (Yu et al., 2021)	✗	✗	✗	✗	✗	0	50,000+*
Sasha (King et al., 2024)	✗	✗	✗	✗	✗	0	60
SAGE (Rivkin et al., 2024)	✗	✗	✗	✗	✔	3	50
PEC-Home(Ours)	✔	✔	✔	✔	✔	1,424	7,120

Table 1: Comparison of existing simulated home datasets. Key features include Progressively Elliptical Commands (Prog. Ellipsis), Multi-User Preferences (Multi Pref.), Dynamic User Preferences (Dynamic Pref.), Long-Term Interaction Support (Long-Inter.), and Persona Size (Pers. Size). * The IFTTT dataset uses a hard-coded 'If This Then That' format, which differs from the natural language instructions used in the other datasets (Sasha, SAGE, PEC-Home).

	Multi-User Preferences				Dynamic User Preferences			
Statistics	Lv1	Lv2	Lv3	Lv4	Lv1	Lv2	Lv3	Lv4
Avg. Tokens	42.08	34.54	33.88	28.43	42.10	34.70	33.87	28.11
Num. Commands	1,068	1,068	1,068	1,068	712	712	712	712
Room (%)	99.34	1.59	1.03	12.36	99.02	1.26	0.98	11.10
Device (%)	98.97	95.69	82.30	91.29	98.60	96.63	79.92	91.57
Operation (%)	91.48	79.49	26.40	4.49	90.73	77.25	26.54	6.04

Table 2: Statistics of PEC-Home across different ellipsis levels (Lv1-Lv4) for two tasks: Multi-User Preferences and Dynamic User Preferences. Core component percentages reflect the presence of Room, Device, and Operation specifications in user commands.

aligned with environmental states, highlighting the high accuracy and reliability of our operation generation approach. To confirm the credibility of our user commands generation framework, the same three researchers manually assessed 200 dialogues which involving multiple progressively elliptical commands. The results revealed that 94.5% of these commands met the defined standards, demonstrating the reliability of the entire process.

4.2 Comparison

Table 1 compares PEC-Home with existing simulated home datasets, highlighting its unique contributions. Unlike existing datasets, PEC-Home is specifically designed to model the shift from explicit to elliptical that naturally arises from long-term human-assistant interactions. In practical smart home scenarios, interpreting such progressively elliptical commands presents two core challenges (as illustrated in Figure 1). To address these, PEC-Home not only captures these ambiguities but also provides personalized long-term interaction data to facilitate the study of progressively elliptical commands interpretation.

4.3 Statistics

Table 2 summarizes the statistics for the PEC-Home dataset across two tasks: Multi-User Preferences and Dynamic User Preferences. Both tasks show a clear trend where the average token count decreases from Lv1 to Lv4, indicating that the user commands become progressively shorter and elliptical. Additionally, the percentages of Room, Device, and Operation components also decrease as the commands become more elliptical, further supporting that the commands are becoming more elliptical and less specific.

The lack of statistics for **Operation parameters** component is due to the presence of highly common parameters, such as "up", "sleep", etc. These words frequently appear in user commands making them difficult to track accurately. Additionally, time-related parameters often appear in the 12-hour format (e.g., "11:30 PM") rather than the 24-hour format, further complicating the precise identification of time-related parameters. As a result, we didn't provide statistics for the operation parameters component.

5 Experiment

5.1 Setup

Models We select several open-source and closed-source LLMs for a comprehensive evaluation (Zhuang et al., 2023; Wang et al., 2024a). Specifically, the open-source models including: Llama-3.1-8B-Instruct (AI@Meta, 2024), Mistral-7B-

Method	Multi-User Preferences								Dynamic User Preferences							
	Lv1		Lv2		Lv3		Lv4		Lv1		Lv2		Lv3		Lv4	
	EA	F1	EA	F1	EA	F1	EA	F1	EA	F1	EA	F1	EA	F1	EA	F1
LLaMA3-8B																
0-Shot	45.04	38.94	7.87	6.80	1.12	1.94	0.56	2.21	36.66	29.42	6.32	5.83	0.56	0.95	0.28	0.96
ICL	79.21	75.57	10.49	10.71	2.81	4.29	1.40	5.49	74.58	72.18	7.87	9.23	1.40	2.94	0.84	3.93
RAG	73.97	72.04	71.82	69.97	58.71	58.41	42.88	43.19	74.30	66.73	58.29	58.54	35.67	38.46	24.02	28.51
Mistral-7B																
0-Shot	58.43	47.68	6.84	5.98	1.22	1.06	1.31	1.56	57.16	40.24	6.60	5.03	1.26	0.49	0.56	0.62
ICL	90.36	77.77	16.11	15.63	2.81	3.79	0.66	4.57	88.76	79.32	19.52	17.83	2.25	4.59	1.12	4.89
RAG	88.76	83.46	83.15	79.92	63.76	62.20	47.47	49.75	79.92	73.05	70.93	67.44	44.24	44.94	33.99	37.51
Gemma2-9B																
0-Shot	48.41	51.76	8.33	9.30	0.84	3.36	0.28	3.43	44.94	50.62	8.15	9.02	0.56	1.82	0.28	3.26
ICL	93.26	75.33	16.20	13.18	2.53	3.63	0.66	4.25	91.85	77.55	15.87	12.57	1.97	3.73	0.56	4.57
RAG	92.60	90.76	82.68	82.24	55.24	59.75	38.30	47.73	92.13	89.89	79.78	77.19	42.42	47.62	32.72	42.69
Gemma2-27B																
0-Shot	48.60	57.06	6.93	7.93	1.03	2.77	0.94	2.70	45.51	47.27	7.44	8.22	0.56	1.47	0.28	2.40
ICL	93.63	77.80	15.36	14.02	2.81	4.82	0.84	5.25	91.01	78.25	13.90	11.01	1.97	3.21	0.84	4.32
RAG	91.95	78.46	87.36	82.58	62.27	65.60	45.04	53.14	92.98	76.98	88.37	78.59	57.87	59.55	43.12	51.06
Qwen2.5-7B																
0-Shot	54.59	55.00	9.46	9.65	0.94	2.43	0.66	3.16	54.21	45.44	11.38	9.76	0.70	1.03	0.28	1.51
ICL	88.58	83.95	15.45	12.42	2.62	3.61	1.22	4.85	86.94	80.10	17.42	13.69	2.25	3.85	1.40	4.89
RAG	91.76	87.25	89.79	85.18	64.04	65.05	48.50	53.94	93.68	86.88	91.57	85.97	52.67	57.17	43.40	50.56
Qwen2.5-14B																
0-Shot	50.94	58.26	8.71	9.50	2.06	3.38	1.31	3.60	38.48	48.35	8.00	9.71	0.98	2.97	0.84	3.36
ICL	85.86	74.83	10.67	10.87	2.62	3.98	1.69	4.79	81.88	71.18	12.08	12.68	2.67	4.14	1.97	3.98
RAG	88.11	88.79	86.61	87.13	69.19	73.50	48.41	57.86	79.21	81.65	66.57	68.68	47.33	53.79	40.73	49.33
Qwen2.5-32B																
0-Shot	55.06	61.27	10.21	11.98	2.15	5.24	1.12	4.75	50.14	55.97	9.27	11.19	1.40	3.73	1.12	4.33
ICL	92.13	89.89	14.23	14.14	2.15	4.52	1.69	4.22	88.20	85.35	6.46	5.93	1.55	2.20	1.26	2.91
RAG	89.14	84.01	87.92	75.08	68.07	58.54	48.97	47.68	90.31	84.01	79.92	75.08	58.43	58.54	43.68	47.68
Qwen2.5-72B																
0-Shot	39.32	51.76	7.68	9.87	2.62	4.67	1.97	3.83	34.27	47.82	7.16	9.61	2.39	4.44	1.69	3.74
ICL	89.42	88.11	15.72	13.17	4.03	4.41	2.25	4.46	88.90	75.76	16.29	15.90	2.81	4.58	2.11	3.91
RAG	95.51	92.47	92.32	89.48	69.10	70.27	52.25	57.73	84.55	85.43	79.21	80.55	54.49	61.41	45.65	55.07
GPT-4o																
0-Shot	55.99	61.57	5.06	6.20	1.03	3.07	1.31	2.68	52.81	65.14	5.62	7.21	1.12	4.03	0.70	4.63
ICL	92.98	90.25	14.75	14.94	3.37	5.74	1.40	5.25	93.26	92.02	17.98	17.86	3.46	5.85	1.69	6.18
RAG	93.07	90.98	92.70	90.72	76.05	77.69	55.71	64.21	91.57	82.23	88.34	78.97	61.24	65.53	50.70	58.47
DeepSeek-V3																
0-Shot	55.15	62.58	11.24	13.00	2.81	4.72	2.06	5.18	52.81	59.97	6.74	8.69	1.40	2.59	1.69	3.37
ICL	92.79	89.42	13.02	11.84	3.18	4.41	1.78	5.13	92.84	88.43	4.78	4.85	1.69	2.01	0.98	4.55
RAG	94.10	81.72	92.88	80.46	76.78	70.57	53.37	54.68	80.62	62.91	78.65	61.71	62.08	52.52	49.58	46.12

Table 3: Performance comparison across ellipsis levels for Multi-User Preferences and Dynamic User Preferences. Results are evaluated using Execution Accuracy (EA) and F1 Score (F1), where orange indicates the best performance and purple denotes the second-best performance. 0-Shot, ICL, and RAG refer to Zero-Shot Prompting, In-Context Learning, and Retrieval-Augmented Generation, respectively.

Instruct-v0.3 (Jiang et al., 2023), Google's Gemma series (gemma-2-9b/27b-it) (Team et al., 2024), the Qwen2.5 Series (Qwen2.5-7B/14B/32B/72B-Instruct) (Yang et al., 2024), and the DeepSeek-V3 (DeepSeek-AI et al., 2024). For closed-source models, we choose GPT-4o (Achiam et al., 2023). We exclude reasoning models as LLM-based assistants require rapid responses, and such models typically

introduce delays by first generating reasoning trajectory.

Methods We experiment with Zero-Shot Prompting, In-Context Learning, and Retrieval-Augmented Generation on LLMs mentioned above. For specific implementation details, please refer to the Appendix C.

5.2 Metrics

Following previous LLM-based home assistants (Shi et al., 2024; Rivkin et al., 2024) and code generation tasks (Chen et al., 2021; Yu et al., 2018), we use **Execution Accuracy (EA)** as the primary metric to evaluate the performance of the home assistant in executing user instructions. Specifically, the correct execution of the home assistant is defined as generating the precise device control operations required to accurately fulfill the user's command within our virtual environment.

To evaluate the operational accuracy of home assistants in executing operations accurately, we utilize the **F1** score (Devlin et al., 2019; Wang et al., 2024b). The specific formula for calculating F1 is provided in the Appendix C.2.

5.3 Results

Table 3 shows the results of the experiments on PEC-Home. Several conclusions can be drawn from the results.

The results indicate that no models can accurately execute user-intended operations based solely on elliptical commands. Even when augmented with RAG to access dialogue history, the performance of these models on elliptical commands does not match that achieved with complete commands. Specifically for RAG methods, while dialogue history offers some assistance, the performance in correctly executing user operations still declines as the commands become more elliptical.

While increased model parameters can enhance a model's ability to leverage dialogue history for command interpretation, this advantage sharply declines when addressing highly elliptical commands. Experimental results demonstrate that although larger models (e.g., Qwen2.5 and Gemma2 series) show improved performance on less elliptical commands (Lv2), these gains rapidly diminish facing highly elliptical commands. Even SOTA model like GPT-4o, under In-Context Learning, underperforming some smaller RAG-equipped models on Lv4 commands. This emphasizes that **merely increasing parameter size is not a sufficient strategy for accurately executing user-intend operations facing elliptical commands.**

EA and F1 reveal divergent failure modes across ellipsis levels. On less elliptical commands, EA is often higher than F1. This discrepancy is primarily attributed to two factors: firstly, as detailed in Appendix C.2, EA does not require output format to be strictly correct, unlike F1, which highlights the current limitations of LLMs in adhering to precise required formats. Secondly, models may exhibit "over-execution", where they correctly complete the core instruction but also generate extra operations, thereby lowering the F1 score but not affecting the EA. Conversely, on highly elliptical commands, EA tends to be lower than F1. This phenomenon often occurs because models take "shortcuts". For instance, in response to a command like "turn on the light and set brightness to 4," a model might only generate the simpler "turn_on" operation. Such partially matched simple operations can boost F1 scores, but the EA score remains low, leading to the observed divergence between these two metrics.

6 Analysis

In this section, we extend our analysis by conducting three additional research questions (RQs) to further investigate elliptical command interpretation. These RQs are designed to systematically explore the key factors that influence the assistants' ability to resolve elliptical command, focusing on **memory management (RQ1)**, **model optimization (RQ2)**, and **external tools integration (RQ3)**. For experiment details, please refer to Appendix C and a comprehensive error analysis is provided in Appendix D.

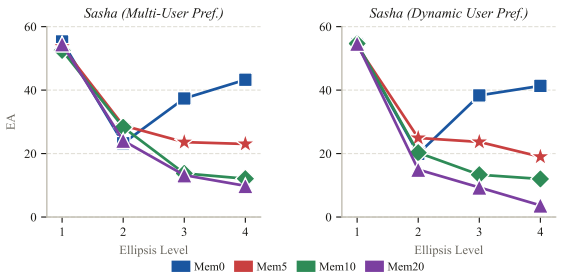

Figure 3: Execution Accuracy of Sasha on Qwen2.5-7B-Instruct across varying amounts of preloaded memory in multi-user preferences and dynamic user preferences scenarios. 'Mem number' indicates the amount of preloaded irrelevant memory.

6.1 RQ1: How Does Preloaded Irrelevant Memory Affect Interpretation Accuracy?

In practical applications, home assistants often need to handle a large number of commands, and external memory tools are considered to enhance model performance (Lewis et al., 2020; Gui et al., 2022). However, the effectiveness of utilizing memory is influenced by the amount of preloaded irrelevant memory. Figure 3 shows the EA of Sasha and Figure 5, 6, and 7 in Appendix show the EA of SAGE, RAG (Qwen2.5-7B) and RAG (Gemma2-9B) across varying amounts of preloaded memory.

This section investigates how the amount of preloaded irrelevant memory impacts interpretation performance. For Sasha when no irrelevant memory is present in its database, Sasha can effectively leverage dialogue history to enhance its performance on highly elliptical commands. However, when irrelevant information is preloaded into the database, a general trend is observed across all evaluated methods that their EA declines as the commands become more elliptical. This phenomenon underscores the challenges current methods face in filtering and utilizing relevant history dialogue from a noisy memory environment when interpreting increasingly elliptical commands.

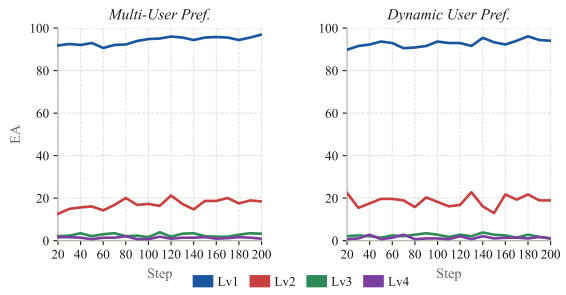

Figure 4: Execution Accuracy of Qwen2.5-7B-Instruct model in test dataset under different training steps.

6.2 RQ2: Does Fine-tuning Enhance Elliptical Commands Resolution?

Fine-tuning is a common approach to adapting LLMs to specific domains. We split the dataset into training, validation, and test sets in a 5:1:4 ratio, and then used the training dataset to fine-tune the Qwen2.5-7B and Gemma2-9B models. These models are selected as our main experiments involve evaluations of both the Qwen and Gemma model series. Figure 4 shows the results on Qwen2.5-7B and figure 8 in Appendix shows the results on Gemma2-9B.

We use the same prompt as the one used for the ICL method to guide fine-tuned model inference. The results show that while fine-tuning significantly boost performance on low-ellipsis, its performance collapse on highly elliptical commands. This performance on highly elliptical tasks mirrored that of ICL methods in Table 3, demonstrating that fine-tuning, despite aiding in interpreting complete commands, fails to resolve the core challenge of interpreting progressively elliptical commands.

Method	Multi-User Preference			
	Lv1	Lv2	Lv3	Lv4
SAGE	51.40	30.25	15.73	4.49
Sasha	55.34	23.25	37.36	43.26
Method	**Dynamic User Preference**			
	Lv1	Lv2	Lv3	Lv4
SAGE	52.00	34.55	18.67	5.33
Sasha	54.33	19.93	38.33	41.33

Table 4: Execution Accuracy of SAGE and Sasha on Qwen2.5-7B-Instruct across varying levels of ellipsis in multi-user preferences and dynamic user preferences scenarios.

6.3 RQ3: How Do External Tools Improve Ellipsis Command Handling?

The integration of external tools has become mainstream in the current home assistants and autonomous agents (Qiao et al., 2024; Chen et al., 2024; Zhang et al., 2024). For instance, **Sasha** and **SAGE** represent two mainstream approaches among current LLM-based assistants. For specific implementation details, please refer to the Appendix C.1. Thus, this section explores the performance of current tool-based home assistants in PEC-Home. Table 4 shows the results.

Both Sasha and Sage fall short compared to ICL approaches facing complete commands. This suggests that tool-based methods struggle with basic user commands. Among Lv2 commands, where instructions are moderately elliptical, Sasha fails to decide whether invoke external memory, leading to lower EA scores compared to more elliptical commands. In contrast, SAGE exhibits consistently poor performance across all command levels. These results show that while tool-based methods can handle some elliptical commands, but they fail to consistently resolve elliptical commands.

7 Conclusion

We identify the task of interpreting progressively elliptical commands in smart homes, which is naturally arises from human communication and represents a well-studied phenomenon in cognitive science. To facilitate the study of this crucial task, we introduced PEC-Home, the first simulated home dataset specifically designed for interpreting such progressively elliptical commands. Our experimental results demonstrate a significant challenge for current LLM-based assistants. All evaluated models struggle to accurately execute user-intended operations based solely on elliptical commands. Furthermore, even augmented with advanced techniques such as RAG, external tools, and fine-tuning, these LLMs, including state-of-the-art models like GPT-4o, still exhibit substantial performance degradation as commands become more elliptical compared to their performance when facing complete commands.

Limitations

Our virtual environment encompasses a diverse range of devices and each device is associated with numerous methods. These methods often come with detailed descriptions. When these descriptions are provided as input to the model, the context becomes very long, leading to higher computational costs.

Furthermore, due to strict privacy constraints inherent in smart home environments, we are unable to collect large-scale, real-world user interaction data. While real production data is the ideal standard, large-scale access is currently unfeasible. To mitigate this limitation, we utilized LLMs for data generation but took extensive measures to ensure the simulated dialogues faithfully reflect real human communication habits. Prior to dataset construction, we conducted a pilot study on real human-home interactions, observing that users naturally tend to omit room and operation information as shared context grows. These empirical findings, firmly grounded in cognitive science and linguistic theories, directly guided our simulation design. Finally, we implemented a rigorous human-in-the-loop verification process, where experts manually assessed a random sample of 50 dialogues. The results showed that 96.5% of the generated commands were contextually natural and met human linguistic patterns. Through these efforts, we ensure that PEC-Home serves as a reliable, highly realistic, and privacy-safe proxy for actual human-system interactions.

Ethical Statement

During the construction of PEC-Home, we adhered to strict ethical standards and ensured that all procedures complied with ethical standards. The virtual environment and the devices within it were manually constructed, carefully reviewed, and meticulously validated to ensure their accuracy and reliability. Following the generation of user commands, we conducted strict quality assessments on the dataset to verify its integrity and ensure it complies with high standards. As a result, we are confident that PEC-Home does not contain any offensive or biased content. Furthermore, all research was performed with a strong commitment to ethical principles, ensuring transparency, and fairness.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No. U21B2009).

References

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

AI@Meta. 2024. Llama 3 model card.

Susan E Brennan and Herbert H Clark. 1996. Conceptual pacts and lexical choice in conversation. *Journal of experimental psychology: Learning, memory, and cognition*, 22(6):1482.

John M Carroll. 1980. Naming and describing in social communication. *Language and Speech*, 23(4):309–322.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Zehui Chen, Weihua Du, Wenwei Zhang, Kuikun Liu, Jiangning Liu, Miao Zheng, Jingming Zhuo, Songyang Zhang, Dahua Lin, Kai Chen, and Feng Zhao. 2024. T-eval: Evaluating the tool utilization capability of large language models step by step. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9510–9529, Bangkok, Thailand. Association for Computational Linguistics.

Herbert H Clark and Deanna Wilkes-Gibbs. 1986. Referring as a collaborative process. *Cognition*, 22(1):1–39.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, , and et al. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Anind K Dey, Timothy Sohn, Sara Streng, and Justin Kodama. 2006. icap: Interactive prototyping of context-aware applications. In *Pervasive Computing: 4th International Conference, PERVASIVE 2006, Dublin, Ireland, May 7-10, 2006. Proceedings 4*, pages 254–271. Springer.

Liangke Gui, Borui Wang, Qiuyuan Huang, Alexander Hauptmann, Yonatan Bisk, and Jianfeng Gao. 2022. KAT: A knowledge augmented transformer for vision-and-language. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 956–968, Seattle, United States. Association for Computational Linguistics.

Saurabh Gupta, Siddhant Bhambri, Karan Dhingra, Arun Balaji Buduru, and Ponnurangam Kumaraguru. 2020. Multi-objective Reinforcement Learning based approach for User-Centric Power Optimization in Smart Home Environments . In *2020 IEEE International Conference on Smart Data Services (SMDS)*, pages 89–96, Los Alamitos, CA, USA. IEEE Computer Society.

Robert D Hawkins, Michael C Frank, and Noah D Goodman. 2020. Characterizing the dynamics of learning in repeated reference games. *Cognitive science*, 44(6):e12845.

Qianyu He, Jie Zeng, Wenhao Huang, Lina Chen, Jin Xiao, Qianxi He, Xunzhe Zhou, Jiaqing Liang, and Yanghua Xiao. 2024. Can large language models understand real-world complex instructions? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18188–18196.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

John Jaihar, Neehal Lingayat, Patel Sapan Vijaybhai, Gautam Venkatesh, and Kishor P Upla. 2020. Smart home automation using machine learning algorithms. In *2020 international conference for emerging technology (INCET)*, pages 1–4. IEEE.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Chadia Khraief, Faouzi Benzarti, and Hamid Amiri. 2019. Convolutional neural network based on dynamic motion and shape variations for elderly fall detection. *International Journal of Machine Learning and Computing*, 9(6):814–820.

Younggi Kim, Jihoon An, Minseok Lee, and Younghee Lee. 2017. An activity-embedding approach for next-activity prediction in a multi-user smart space. In *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–6. IEEE.

Evan King, Haoxiang Yu, Sangsu Lee, and Christine Julien. 2024. Sasha: Creative goal-oriented reasoning in smart homes with large language models. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 8(1).

Robert M Krauss and Sidney Weinheimer. 1964. Changes in reference phrases as a function of frequency of usage in social interaction: A preliminary study. *Psychonomic Science*, 1:113–114.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Jiang, Chengfei Lv, and Huajun Chen. 2024. AutoAct: Automatic agent learning from scratch for QA via self-planning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3003–3021, Bangkok, Thailand. Association for Computational Linguistics.

Dmitriy Rivkin, Francois Hogan, Amal Feriani, Abhisek Konar, Adam Sigal, Xue Liu, and Gregory Dudek. 2024. Aiot smart home via autonomous llm agents. *IEEE Internet of Things Journal*.

Samad Sepasgozar, Reyhaneh Karimi, Leila Farahzadi, Farimah Moezzi, Sara Shirowzhan,

Sanee M. Ebrahimzadeh, Felix Hui, and Lu Aye. 2020. A systematic content review of artificial intelligence and the internet of things applications in smart home. *Applied Sciences*, 10(9):3074.

Yingtian Shi, Xiaoyi Liu, Chun Yu, Tianao Yang, Cheng Gao, Chen Liang, and Yuanchun Shi. 2024. Bridging the gap between natural user expression with complex automation programming in smart homes. *Preprint*, arXiv:2408.12687.

Shashi Suman, Ali Etemad, and Francois Rivest. 2022. Potential impacts of smart homes on human behavior: A reinforcement learning approach. *IEEE Transactions on Artificial Intelligence*, 3(4):567–580.

Niek Tax. 2018. Human activity prediction in smart home environments with lstm neural networks. In *2018 14th international conference on intelligent environments (IE)*, pages 40–47. IEEE.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, and et al. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L Littman. 2014. Practical trigger-action programming in the smart home. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 803–812.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Hongru Wang, Rui Wang, Boyang Xue, Heming Xia, Jingtao Cao, Zeming Liu, Jeff Z. Pan, and Kam-Fai Wong. 2024a. AppBench: Planning of multiple APIs from various APPs for complex user instruction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15322–15336, Miami, Florida, USA. Association for Computational Linguistics.

Huiming Wang, Liying Cheng, Wenxuan Zhang, De Wen Soh, and Lidong Bing. 2024b. Order-agnostic data augmentation for few-shot named entity recognition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7792–7807, Bangkok, Thailand. Association for Computational Linguistics.

Mark Weiser. 1999. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

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

Ziqi Yin, Mingxin Zhang, and Daisuke Kawahara. 2024. Harmony: A home agent for responsive management and action optimization with a locally deployed large language model. *Preprint*, arXiv:2410.14252.

Haoxiang Yu, Jie Hua, and Christine Julien. 2021. Analysis of ifttt recipes to study how humans use internet-of-things (iot) devices. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, SenSys '21, page 537–541. ACM.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Li Zeng, Yingyu Shan, Zeming Liu, Jiashu Yao, and Yuhang Guo. 2024. FAME: Towards factual multi-task model editing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15992–16011, Miami, Florida, USA. Association for Computational Linguistics.

Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. 2024. CodeAgent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13643–13658, Bangkok, Thailand. Association for Computational Linguistics.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you

have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36:50117–50143.

Rolf A Zwaan and Gabriel A Radvansky. 1998. Situation models in language comprehension and memory. *Psychological bulletin*, 123(2):162.

A Detailed Problem Definition

Let a virtual smart home environment be represented as:

$$H = \{r_1, r_2, \ldots, r_n\} \tag{1}$$

where each room $r_i \in H$ contains operable devices:

$$D_i = \{d_1, d_2, \ldots, d_m\} \tag{2}$$

Each device $d_j \in D_i$ is defined by:

$$d_j = \langle s_j, M_j \rangle \tag{3}$$

where $s_j = [s_j^1, \ldots, s_j^p]$ is the device state vector and $M_j = \{m_j^1(\theta_1), \ldots, m_j^k(\theta_k)\}$ is the executable method set.

Given the current user instruction u_t at turn t, the dialogue history is:

$$\mathcal{C} = \{(u_1, a_1), \ldots, (u_{t-1}, a_{t-1})\} \tag{4}$$

where each pair (u_k, a_k) (for $k < t$) represents the user's instruction u_k and the system's corresponding response a_k.

Mapping user instruction u_t to system response a_t varies:

- Baseline systems (e.g., prompt-based methods) map u_t to a_t using only the home state H:

$$f_{\text{baseline}} : (u_t, H) \rightarrow a_t \tag{5}$$

- Advanced systems (e.g., RAG) also use dialogue history $\mathcal{C}$ (Eq. 4) for contextual interpretation of u_t:

$$f_{\text{advanced}} : (u_t, \mathcal{C}, H) \rightarrow a_t \tag{6}$$

The model's output $a_t = r_i.d_j.m_k(\theta)$ (target room r_i, device-method $d_j.m_k$, parameters θ) is the system's response. The new pair (u_t, a_t) updates the dialogue history.

A.1 Multi-User Preferences Management

Let $\mathcal{P} = \{p_1, \ldots, p_k\}$ denote the user persona set where each user p_l maintains a distinct dialogue history:

$$\mathcal{C}_l = \{(u_1^l, a_1^l), \ldots, (u_{t-1}^l, a_{t-1}^l)\} \tag{7}$$

Advanced systems dynamically select relevant history through:

$$\mathcal{C}_l = \text{LLM\_Select}(u_t, \mathcal{P}) \in \{\mathcal{C}_1, \ldots, \mathcal{C}_k\} \tag{8}$$

The operational mapping then becomes:

$$f_{\text{multi}} : (u_t, \mathcal{C}_l, H) \rightarrow r_i.d_j.m_k(\theta) \tag{9}$$

A.2 Dynamic User Preferences Adaptation

For each user p_l, their dialogue history contains two environment-specific preference contexts:

$$\mathcal{C}_l = \{\mathcal{C}_l^{e_1}, \mathcal{C}_l^{e_2}\} \tag{10}$$

where e_1 and e_2 represent distinct environmental states. Given real-time sensor data S_t, advanced systems dynamically select the relevant context:

$$\mathcal{C}_l^t = \text{LLM\_Select}(S_t, \mathcal{C}_l) \in \{\mathcal{C}_l^{e_1}, \mathcal{C}_l^{e_2}\} \tag{11}$$

The operational mapping then becomes:

$$f_{\text{dyn}} : (u_t, \mathcal{C}_l^t, H, S_t) \rightarrow r_i.d_j.m_k(\theta) \tag{12}$$

B Dataset Details

B.1 Virtual Environment Details

This section provides comprehensive statistics on our virtual environment. Table 5 provides a detailed overview of all rooms in the virtual environment, along with the corresponding devices located within each room. Table 6 presents a complete list of devices and their associated executable methods, detailing the executable methods available for each device. These tables provide a comprehensive understanding of the virtual environment's layout and the operational interfaces of devices.

B.2 Commands Generation Details

In this section, we detail the specific prompts used for guiding the LLMs to generate user commands. Table 8 presents the prompt designed for generating the complete and most clear (Le 1) user commands. Additionally, the system prompts used for generating higher-elliptical commands are provided in Table 9. We also provide an example of device methods description in the prompt in Table 11.

C Experiments

C.1 Implementation Details of Baselines

For SAGE and Sasha, we use the framework provided by Rivkin et al. (2024). Since SAGE and

Room Name	Devices
Master Bedroom	light, air_conditioner, heating, fan, air_purifiers, aromatherapy, trash, humidifier, dehumidifiers, tv
Guest Bedroom	light, air_conditioner, heating, fan, air_purifiers, trash, humidifier, dehumidifiers, tv
Living Room	light, air_conditioner, heating, fan, air_purifiers, aromatherapy, trash, humidifier, dehumidifiers, media_player
Dining Room	light, fan air_purifiers, humidifier, dehumidifiers, trash
Study Room	light, air_conditioner, heating, fan, air_purifiers, humidifier, dehumidifiers, trash, aromatherapy
Kitchen	light, fan, trash, water_heater
Bathroom	light, heating, trash, water_heater
Balcony	light, aromatherapy, trash, media_player
Store Room	light, air_purifiers, humidifier, dehumidifiers

Table 5: Distribution of devices across all rooms in the virtual environment.

Device Name	Methods
light	turn_on; turn_off; set_brightness; set_mode
air_conditioner	turn_on; turn_off; set_temperature; set_mode; set_fan_speed; set_swing;
heating	turn_on; turn_off; set_temperature; set_mode; set_fan_speed
fan	turn_on; turn_off; set_speed; set_swing
air_purifiers	turn_on; turn_off; set_mode; set_fan_speed
water_heater	turn_on; turn_off; set_temperature; set_mode
media_player	turn_on; turn_off; play; pause; stop; set_volume; set_song; set_artist; set_style
trash	pack
aromatherapy	turn_on; turn_off; set_intensity; set_interval
humidifier	turn_on; turn_off; set_intensity; set_mode
dehumidifiers	turn_on; turn_off; set_intensity; set_mode
tv	turn_on; turn_off; set_volume; change_channel;

Table 6: List of devices and their associated executable methods in the virtual environment.

Sasha require the LLM to possess the capability of tool calling, and only Qwen2.5 supports native function-call ability among open-source LLMs, we conducted evaluations on the Qwen2.5 series.

SAGE As a central hub, SAGE interfaces with multiple specialized tools: smart_device_tool for device control, memory_tool for command history, and weather_tool for environmental data collection. The smart_device_tool, functioning as an independent intelligent agent, provides essential device management functions including get_methods(), list_devices(), list_rooms(), and execute_command(). Based on these multiple tools, SAGE dynamically generates prompt trees and toolchains to handle user commands. To optimize system performance, we have streamlined the toolset by removing non-essential components such as code tools and external interaction capabilities, thereby reducing potential interference with the model's decision-making process.

Sasha An intelligent agent designed for smart home control. It communicates with devices using JSON format and integrates memory and environmental data to enhance device control capabilities. Sasha uses its plan generation tool to generate high-quality action plans for under-specified user commands. Once the agents complete their operations, we verify the success of the execution by checking the final state of the devices. In contrast with Sage, Sasha only has ReAct-style (Yao et al., 2023) planner tool while Sage has multiple tools such as smart_device_tool, memory_tool, and so on.

Zero-Shot Prompting We utilize the same prompt as the In-Context Learning prompt but remove the examples and incorporate specific format requirements. In this approach, the model does not have access to the user's dialogue history.

In-Context Learning We provide the prompt used in In-Context Learning in Table 10. For each level of elliptical commands, we include one illustrative example. In this approach, the model does not have access to the user's dialogue history neither.

RAG We implement retrieval-augmented generation using vector database. All user commands and the corresponding generated device operations are stored in a database. The user's current instruction is used as a query to search through the

Error Type	User Command	Generated (red) / Golden (green)
Room Missing	"Lights on, cool white, please! . . . "	light.turn_on(), light.set_mode("cool white") bathroom.light.turn_on(), bathroom.light.set_mode("cool white")
Room Error	"Hey, can you handle the aromatherapy setup? . . . "	living_room.aromatherapy.turn_on(), living_room.aromatherapy.set_intensity(80) balcony.aromatherapy.turn_on(), balcony.aromatherapy.set_intensity(0)
Parameter Missing	"Hey, could you manage that trash thing we discussed?"	dining_room.trash.pack() dining_room.trash.pack(18:46)
Parameter Error	"Could you make sure the humidifier is off by 11:30 PM? . . . "	guest_bedroom.humidifier .turn_off("23:30") guest_bedroom.humidifier .turn_off(23:30)
Ignoring History	"Hey, can you handle that thing with the air conditioner? . . . " *History:* "Could you have the air conditioning kick . . . " *Hist. resp.:* guest_bedroom .air_conditioner.turn_on(07:13)	guest_bedroom.air_conditioner .turn_on(07:00) guest_bedroom.air_conditioner .turn_on(07:13)

Table 7: Five representative error types in PEC-Home: Room Missing, Room Error, Parameter Missing, Parameter Error, and Ignoring History. The core component mentioned in user commands is shown in orange, the error response generated by LLMs is shown in red, and the golden answer is shown in green.

dialogue history, with a fixed setting of top$k$ = 3 for all experiments. To evaluate the impact of preloaded irrelevant memory, we additionally collected fifty natural language instructions from users and randomly introduced 5, 10, and 20 of these as irrelevant memory.

C.2 Metrics

F1 is calculated as follows:

$$Precision = \frac{\text{operation\_correct\_num}}{\text{operation\_pred\_num}} \quad (13)$$

$$Recall = \frac{\text{operation\_correct\_num}}{\text{operation\_gold\_num}} \quad (14)$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (15)$$

Here, Precision measures the proportion of correctly predicted operations out of all predicted operations, while Recall measures the proportion of correctly predicted operations out of all ground truth operations.

In contrast to F1, **Execution Accuracy (EA)** is measured by extracting the operations generated by the model without requiring the model's output format to be strictly correct. (In this paper, the model is required to enclose its generated instructions within {}; however, for EA calculation, we consider whether the generated operations are functionally correct, assuming format issues can be addressed as an engineering problem later.) For F1, we apply a stricter requirement that the format of the generated instructions must also be correct.

Execution Accuracy (EA) assesses the functional correctness of model-generated operations without enforcing strict output formatting. In this work, although models are instructed to enclose generated instructions within curly braces ({}), EA considers only whether the predicted operations would achieve the user intended operation execution in the environment, treating syntactic inconsistencies as engineering issues that can be addressed post hoc. In contrast, F1 applies a more strict criterion where outputs must not only be functional correct but also strictly adhere to the prescribed format.

This two-tiered evaluation strategy enables a nuanced analysis by decoupling semantic understanding from surface-level formatting compliance. EA reflects the model's core capacity to interpret natural language instructions and identify appropriate device operations, independent of format errors. Meanwhile, F1 captures practical deployability by measuring conformity to interface standards required for downstream execution. The combination of these metrics provides a more comprehensive assessment of the model's core interpretation capabilities and practical utility, enabling evaluation of both its semantic understanding and its ability to generate outputs suitable for real-world deploy-

ment.

C.3 Implementation Details

The experiments with open-source models were conducted on NVIDIA A100 GPUs and NVIDIA A800 GPUs. For GPT-4o, we utilized the APIs provided by OpenAI. When fine-tuning Qwen2.5-7B-Instruct, we performed the fine-tuning experiments on NVIDIA A800 GPUs using the LoRA (Low-Rank Adaptation) (Hu et al., 2022) technique. The hyperparameters for LoRA are configured as follows: r=16, $lora\_alpha$=32, $learning\_rate$=1e-5.

D Error Analysis

We conduct a comprehensive error analysis to identify the challenges in interpreting progressively elliptical commands. Inspired by Yin et al. (2024); Yu et al. (2018), we categorize five representative error types, as shown in Table 7. To ensure accuracy beyond automated metrics and provide a quantitative understanding, we conducted a manual root-cause analysis on 100 randomly sampled error cases from the Qwen2.5-7B (RAG) results. The distribution of the primary failure modes is detailed below:

Ignoring History (38%): The most frequent error. It occurs when the model fails to leverage dialogue history retrieved from the database. The model treats the elliptical command in isolation, failing to retrieve entities (e.g., air conditioner) or states established in previous turns.

Room Missing (22%): Occurs when user instructions specify a device's operation but omit its location. The operation is correct, but the model fails to infer the spatial context (e.g., outputting `turn_on` without specifying `bedroom`).

Room Error (17%): Occurs when the model infers the wrong room due to ambiguous context switching.

Parameter Error (13%): Involves incorrect parameter formatting or value assignment. While some type conversions (e.g., string to integer) might seem trivial, models often hallucinate formats (e.g., generating "11:30 PM" instead of the required "23:30") or misinterpret vague values (e.g., "a bit warmer"). In strict API-based home automation, adhering to the exact schema is critical; a "close enough" parameter usually results in execution failure (API Error). Thus, strictly capturing these as errors is vital for evaluating an agent's robustness.

Parameter Missing (10%): Results from incomplete parameter extraction for device operations.

It is important to note that structural formatting errors are not separately categorized in this analysis. This is because, during the Execution Accuracy (EA) calculation, models' responses that are operationally correct but have syntax-level formatting errors are still considered correct. Such common structural errors include missing the required curly braces {} for function-call style operations, or the erroneous use of other delimiters such as single quotes ‘ ’ or angle brackets <>. This leniency strictly applies to the outer syntax, whereas API parameter values must remain precise as discussed above.

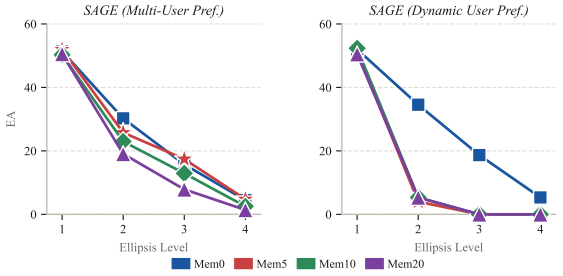

Figure 5: Execution Accuracy of SAGE on Qwen2.5-7B-Instruct across varying amounts of preloaded memory in multi-user preferences and dynamic user preferences scenarios. 'Mem number' indicates the amount of preloaded irrelevant memory.

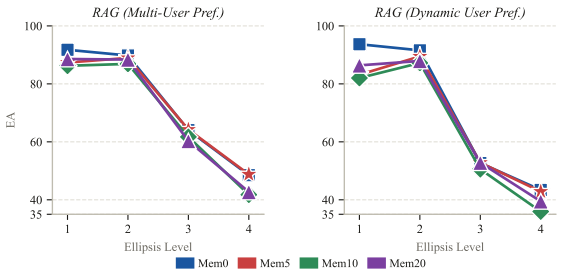

Figure 6: Execution Accuracy of RAG on Qwen2.5-7B-Instruct across varying amounts of preloaded memory in multi-user preferences and dynamic user preferences scenarios. 'Mem number' indicates the amount of preloaded irrelevant memory.

Prompt

Act as a homeowner interacting casually with a smart home assistant. Your task is to generate natural language commands that you would use to instruct your virtual assistant. A clear command should include the device, operation, room, and trigger time or setting and you should ensure that the command you generated should have those four parts. You should incorporate the environment parameters, and the generated command should correlate with the environment. Remember, you are the user issuing commands, not the assistant responding. Ensure your commands are varied and reflect how a real person would naturally communicate with their virtual assistant. You just need to generate one command.

<examples>
Here are some examples:
device : dehumidifier
required instruction : <turn_on(06:53)>
given persona : <my mom is my best friend. I have four sisters. I believe that mermaids are real. i love iced tea.>
environment parameters : <"humidity": 70>
respond content: <"Hey, can you turn on the dehumidifier in the living room at 6:53, please? It's getting a bit too humid in here. Thanks!">
...
</examples>
required instruction :
given persona :
environment parameters :
respond content :

Table 8: The prompt used to guide LLM to generate the most clear (Lv1) user commands. The illustration of the manually defined ellipsis level is shown in orange and a few shots are shown in blue.

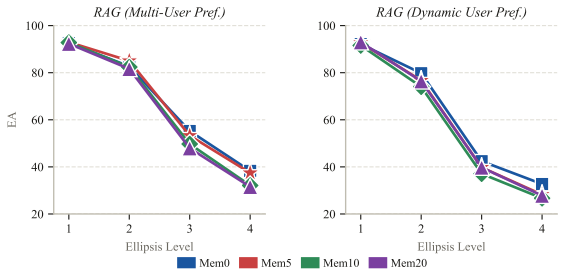

Figure 7: Execution Accuracy of RAG on Gemma2-9b-it across varying amounts of preloaded memory in multi-user preferences and dynamic user preferences scenarios. 'Mem number' indicates the amount of preloaded irrelevant memory.

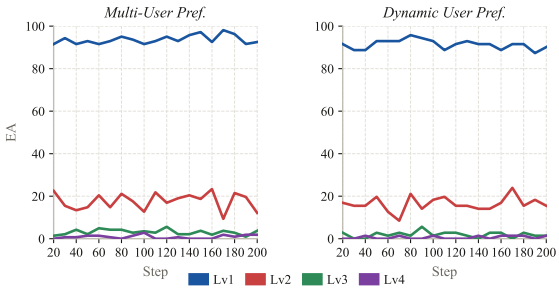

Figure 8: Execution Accuracy of Gemma2-9b-it model in test dataset under different training steps.

Prompt

The system prompt used for generating Lv2 user commands:

Act as a homeowner interacting casually with a smart home assistant. You have given a full command, specifying the operation, device, and room. Then, follow up with shorter commands that assume context, using natural language and varying sentence structures. You should exclude the room in your commands, while still specifying the operation, device, and trigger time/settings. The chat history is provided below. Remember, you are the user issuing commands, not the assistant responding. Ensure your commands are varied and reflect how a real person would naturally communicate with their virtual assistant. You just need to generate one command.

The system prompt used for generating Lv3 user commands:

... The third command should be a more vague version compared to the second command, using natural language and varied sentence structures. In that command, you should not include the room, and ensure that the trigger time or setting remains ambiguous. Additionally, make the operation itself somewhat ambiguous, based on the second command. Assume the assistant has interacted with the user multiple times and has learned their preferences and habits. The chat history is provided below. ...

The system prompt used for generating Lv4 user commands:

... The fourth command should be the most vague version compared to the others, using natural language and varied sentence structures. This time, you should avoid including the operation, room, or trigger time/setting in the command, and only reference the device. Assume the assistant has interacted with the user multiple times and has learned their preferences and habits. The chat history is provided below. ...

Table 9: The prompts used to guide LLM to generate user commands of other ellipsis levels. The illustration of manually defined ellipsis levels is shown in orange.

Prompt

You are a helpful AI Assistant that controls the devices in a house. Complete the following task as instructed or answer the following question with the information provided only. The devices and the methods devices possess are provided below, please only use the methods provided. Only output assistant instructions and enclose them in {}. Please ensure that any parameters involving time are expressed in a 24-hour format. For example: Use 14:00 to represent 2:00 PM, not 2:00 PM. Use 08:30 to represent 8:30 AM, not 8:30 AM.

<device_method>

The following provides the methods to control each device in the current house.

... (Device methods, Table 11 shows an example.)

</device_method>

<example>

Here are a few examples, your output format should be consistent with the results provided in the example:

user_instruction: "Hey, can you turn on the air conditioner in the study room and set the temperature to 23 degrees? I'm getting ready to finalize some paperwork from my recent fair, and I need a comfortable spot to focus."

assistant_instruction: study_room.air_conditioner.turn_on(),
study_room.air_conditioner.set_temperature(23)

...

</example>

<environment>

The following provides the environment information of the current room.

</environment>

Here are the user instructions you need to reply to.

user instructions :

assistant_instruction :

Table 10: The prompt used to guide LLM to generate device operation using In-Context Learning. Few shots are shown in blue.

Example

light
light.turn_on(time: Optional[str] = None, format: '%H:%M');
light.turn_off(time: Optional[str] = None, format: '%H:%M');
light.set_brightness(brightness: int) (range: 0 – 4);
light.set_mode(mode: str) (options: ["soft warm", "neutral", "cool white", "daylight", "cool"]);

Table 11: Examples of home device methods.