2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Combining Ontological Knowledge and Large Language Model for User-Friendly Service Robots

Haru Nakajima and Jun Miura
Department of Computer Science and Engineering
Toyohashi University of Technology

Abstract— **Lifestyle support through robotics is an increasingly promising field, with expectations for robots to take over or assist with chores like floor cleaning, table setting and clearing, and fetching items. The growth of AI, particularly foundation models, such as large language models (LLMs) and visual language models (VLMs), is significantly shaping this sector. LLMs, by facilitating natural interactions and providing vast general knowledge, are proving invaluable for robotic tasks. This paper focuses on the benefits of LLMs for "bring-me" tasks, where robots fetch specific items for users, often based on ambiguous instructions. Our previous efforts utilized an ontology extended to handle environmental data to resolve such ambiguities, but faced limitations when unresolvable ambiguities required user intervention for clarity. Here, we enhance our approach by integrating LLMs for providing additional commonsense knowledge, pairing it with ontological data to mitigate the issue of hallucinations and reduce the need for user queries, thus improving system usability. We present a system that merges these knowledge bases and assess its efficacy on "bring-me" tasks, aiming to provide a more seamless and efficient robotic assistance experience.**

I. Introduction

Lifestyle support is one of the promising application areas of robotics technologies, and many chore tasks are expected to be replaced or supported by service robots. Possible chores include floor cleaning, setting and clearing tables, and finding and bringing objects, among others, and various systems have been developed that can interact with the users using knowledge about the task and environment [1], [2], [3]. Along with the rapid advancement of AI technologies, the expectations for such service robots are greatly rising. The biggest recent advancement is undoubtedly generative AI, especially those based on foundation models, such as large language models (LLMs) [4], [5] and visual language models (VLMs) [6].

Introducing LLMs into service robots is effective as they not only enable natural dialog between robot and human but also provide general knowledge that can be used for robot control [7], [8], [9]. This paper focuses on the latter advantage of LLMs. We deal with *bring-me* tasks, in which a robot brings a user-specified object from a distant place. Users' commands can be ambiguous, where complete information for robot action execution is partially missing. In our previous work [2], we used the ontology with extension for handling facts in the environment to resolve ambiguities. However, the recorded facts are limited and the robot always asked the user for necessary detailed information when the robot cannot resolve ambiguities completely by itself.

In this paper, we adopt a large language model (LLM) as another source of knowledge. In addition to being able to obtain commonsense-like knowledge from the LLM, combining two knowledge sources can reduce the hallucination problem by filtering outputs from the LLM using the ontological knowledge. This is effective to reduce the number of inquiries to the user, thereby increasing the usability of the system. This paper develops a system that combines the two knowledge sources and evaluates the performance using *bring-me* tasks. One of the crucial issues in applying LLMs to service robots is how to prevent hallucinations, as we cannot completely address it even if we tune prompts as much as possible. One approach is to introduce an explicit mechanism for grounding outputs. Recent trends in retrieval-augmented text generation (RAG) [10], [11] seek similar directions. Some work has proposed to use knowledge or experiences to generate feasible robotic plans [12], [13], [14], [15]. In this paper, we take this approach by combining ontological knowledge with LLMs. The contribution of the paper is twofold. First, we propose a new method of integrating two knowledge sources. Second, we evaluate the method for the "bring-me" task in terms of feasibility, efficiency, and usability.

II. Related Work

A. Service robots utilizing knowledge about the environment and dialog

Home service robots must have knowledge about the working environment and utilize it to automatically generate action plans. In "bring-me" tasks, for example, the type and the location of the objects are crucial information. Several studies have proposed the use of ontologies as basis of knowledge representation [16], [17], [18]. We have developed a robotic system that utilizes ontological knowledge supported by verbal interaction to resolve ambiguities in users' commands [2]. When knowledge bases are incomplete and uncertain, the robot may fail to retrieve correct or complete information for task execution and needs to ask the user for the missing information. Such a dialog-supported strategy is effective but might degrade the usability of the system by repeated inquiries.

Generating appropriate queries for increasing the possibility of obtaining informative answers is a key to developing user-friendly service robots. Morohashi and Miura

979-8-3503-7769-9/24/$31.00 ©2024 IEEE

[19] developed a method to generate queries that will yield the most informative answer from the user. Pramanick et al. [20] developed a method for ambiguity resolution by identifying the type of ambiguity using visual scene analysis and by generating appropriate inquiries for that type. Shin et al. [3] proposed a framework enabling social robots to generate questions that resolve uncertainties, taking into account knowledge categories, required cognitive processes, and types of questions. Generating appropriate queries will increase the usability of the system, but it would be nice if we could reduce the number of queries further.

B. Large language models for robots

Large language models (LLMs) have been more and more popular in robotics, in addition to many other domains, and there are many attempts to make robotic plans using LLMs [7], [8], [9], [13], [21], [22], [23], [24]. In these works, one of the common issues is how to make LLM outputs be consistent with (or grounded on) the current situation. Many of such works consider both the appropriateness and the feasibility of plans to choose the best plan. Example command-plan pairs, which are carefully-chosen or actually-held, are often input as prompts. They do not introduce a mechanism of explicitly filtering out ungrounded information.

Ocker et al. [23] explored the use of LLMs as commonsense knowledge base for the robot to understand the context of household tasks. They showed potential applicability of LLMs but suggested that combined use of LLMs and other knowledge sources would be effective. Lu et al. [12] developed a mechanism for extracting causal knowledge from the database, which is similar to the current planning problem, to include it within the next prompt. Ding et al. [13] developed a system that can generate feasible and efficient task plans by symbolic and geometric inference by LLM followed by feasibility checking. LLM-GROP enables the robot to achieve tasks with underspecified requests using LLM while guaranteeing feasibility using a separate checking module. Sakim et al. [14] proposed to use a knowledge network [25] for similar purpose.

III. Overview of the Proposed Approach

A. System configuration

Fig. 1 depicts the overview of the system structure. The system is based on our previous system which utilizes ontological knowledge and is extended to have a large language model (LLM) as another knowledge source. We use Llama 2 [5] as the LLM.

The figure also shows an example interaction between the user and the system. In the command "Find a fruit," the name of the fruit and its location are ambiguous. As preferences of fruits vary from person to person, the system directly asks the user for their preference (steps 7 to 9). On the other hand, the location of "apple" may belong to general knowledge (i.e., LLM), and the system refers to the LLM to obtain a list of possible locations and generates a robot plan for visiting these locations (steps 11 to 13). If the robot cannot find an apple at these locations, the system can pose another inquiry to complete the task (steps 15 to 19).

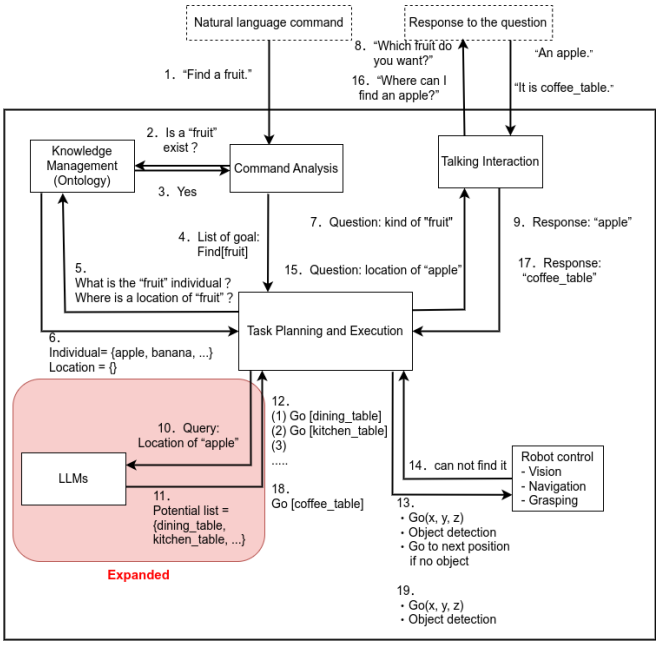


Fig. 1: System overview.

B. Addressing Hallucination

LLMs sometimes suffer from hallucinations, which results in generating incorrect outputs, or outputs inapplicable to the problem under consideration. In this work, the robot is intended to work in ordinary house environments, and we use an LLM to obtain information about object locations. We address the hallucination problem by (1) giving as a prompt a set of concrete knowledge of the environment, which is retreived from our ontological knowledge base, and (2) verifying the consistency between LLM outputs with the knowledge base. We will describe these steps in detail in the following sections.

IV. Proposed System

A. LangChain

LangChain is a framework for developing applications powered by language models, making the developed system context-aware and capable of reasoning using the model [26]. Here, we describe the use of LangChain in implementing our system.

1) Memory: Memory is used for incorporating history of dialog as contexts, by keeping the history and giving it along with inquiries in prompts. This enables the adoption of "Chain-of-Thought" reasoning [27]. We use ConversationTokenBufferMemory functionality, which automatically includes the history with a token size limitation. We limit the size of tokens such that the total text size does not exceed the maximum size allowed in Llama 2. We use the following expression as the token size limitation:

$$max\_token\_size = (max\_seq\_len/2 - sys\_token) \times 0.8,$$

```
from pydantic import BaseModel, Field
class Rank(BaseModel):
    name : List[str] = Field(description="A list of room or furniture that are given
                                examples above and ranked in order by potential.")
```

Fig. 2: Class definition of output format.

The output should be formatted as a JSON instance that conforms to the JSON schema below.

As an example, for the schema {"properties": {"foo": {"title": "Foo", "description": "a list of strings", "type": "array", "items": {"type": "string"}}}, "required": ["foo"]}
the object {"foo": ["bar", "baz"]} is a well-formatted instance of the schema. The object {"properties": {"foo": ["bar", "baz"]}} is not well-formatted.

Here is the output schema:
```
{"properties": {"name": {"description": "A list of room or furniture that are given examples above and ranked in order by potential.", "items": {"type": "string"}, "title": "Name", "type": "array"}}, "required": ["name"]}
```

Fig. 3: Embedding of the output format.

You are home assistant AI. Rooms in this house include ['bed_room', 'initial_position', 'kitchen', 'living_room', 'lobby'].
And Furnitures in this house include {'bed_room': ['bed', 'bookshelf_bedroom', 'desk', 'dresser', 'wagon_bedroom'], 'initial_position': [], 'kitchen': ['cabinet_kitchen', 'counter_wagon', 'high_table'], 'living_room': ['bookshelf_livingroom', 'coffee_table', 'dining_table', 'highshelf_livingroom', 'sideboard', 'sofa'], 'lobby': ['armchair_lobby', 'shelf_lobby', 'table_lobby']}.
The output should be formatted as a JSON instance that conforms to the JSON schema below.

As an example, for the schema {"properties": {"foo": {"title": "Foo", "description": "a list of strings", "type": "array", "items": {"type": "string"}}}, "required": ["foo"]}
the object {"foo": ["bar", "baz"]} is a well-formatted instance of the schema. The object {"properties": {"foo": ["bar", "baz"]}} is not well-formatted.

Here is the output schema:
```
{"properties": {"name": {"description": "A list of room or furniture that are given examples above and ranked in order by potential.", "items": {"type": "string"}, "title": "Name", "type": "array"}}, "required": ["name"]}
```

Fig. 4: System prompt used in the experiment.

where *max_seq_len* is the maximum allowed token size of Llama 2 and *sys_token* is the token size of the system prompt (see Sec. IV-B). This expression is for avoiding generating incomplete outputs for a large *max_seq_len* and for using 80% of the total token size for the context, so that Llama 2 can complete outputs even when the size of inquiry is large.

2) Output parser: Output parser is the functionality to define the output format. We use PydanticOutputParser and specify a JSON format. Fig. 2 is the definition of the output format class, indicating an output is a Python list of names of rooms or furniture. Fig. 3 shows the definition of the output JSON format.

B. Prompting

1) System prompt: We embed two types of information as prompts. One is the output format described in Sec. IV-A.2. The other is a set of concrete knowledge of the environment obtained from the ontological knowledge base. The latter includes the list of all rooms and furniture in the environment and the furniture-room relationships (i.e., in which room each furniture item exists). Giving such information contributes to avoiding hallucinations about room or furniture names. Additionally, giving it in the system prompt enables the system to refrain from providing it repeatedly, thereby reducing the size of tokens. Fig. 4 shows the system prompt used in our system.

2) Inquiry prompt: Table I shows the template of the prompt used for inquiry. Labels *general_pos* and *multiple_pos* are used for direct inquiries from the robot. Label *general_pos* is used when no default locations are recorded for an object in the ontological knowledge base. Label *multiple_pos* is used when an object has multiple default locations to constrain the output to such locations.

Labels *furniture*, *again*, and *summarize* are used for the consistency check step. Label *furniture* is for obtaining a probable set of furniture from the candidates. Label *again* is to repeat the inquiry in case Llama 2 fails to output the location list. Label *summarize* is used in case the output format is not appropriate.

TABLE I: Template for inquiry prompt.

Label	Inquiry
general_pos	I am searching for #object name. Which places are most probably where I can find it? Please tell me the top 5.
multiple_pos	I am searching for #object name. Which places are most probably where I can find it, #position? Please sort them in the order of likelihood.
furniture	I am searching for #object name. Which furniture are most probably where I can find it in #room name? Please tell me the top 3 furniture from furniture list, #furniture list.
again	You must choose potential places from the list given above.
summarize	Please summarize about potential #situation for #object name. You must follow the given output format above.

C. Consistency verification

Consistency verification is responsible for ensuring the appropriateness of the output, namely, the furniture list as possible locations of the target object. The verification is composed of the following three parts (see Fig. 5):

- **Initialization** part retrieves the concrete list of rooms and furniture from the ontological knowledge base.
- **Preprocessing** part extracts only possible locations from the JSON outputs from the Llama 2. When the output is not the JSON format[1], it orders Llama 2 to re-generate the output using *summarize* label (see Sec. IV-B.2). Then, it identifies the names of locations, which

[1]This could sometimes happen even though we give the prompt to output in the JSON format.

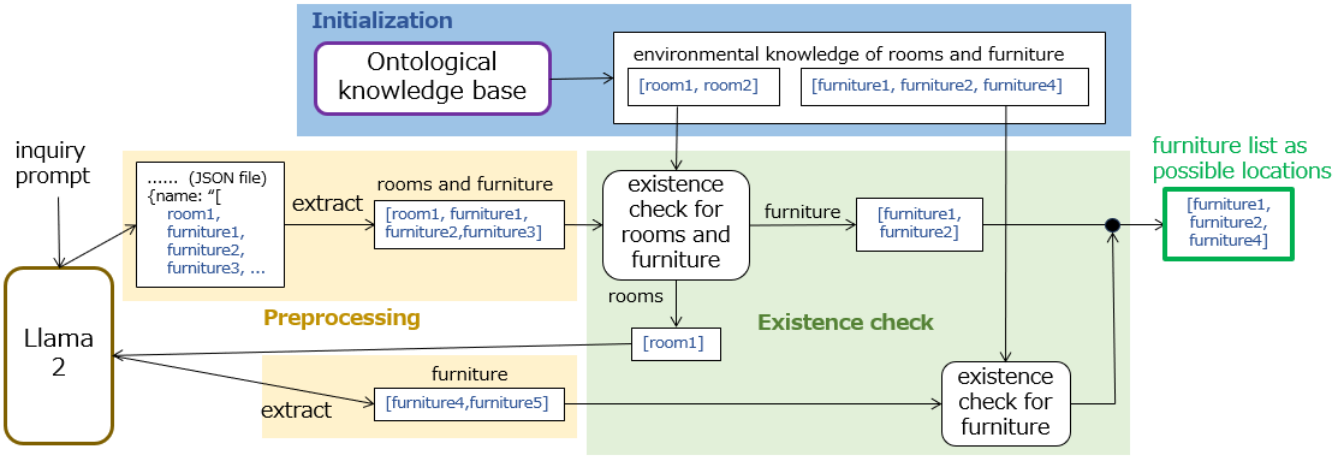


Fig. 5: Consistency verification.

are either rooms or furniture, and sends them to the next part.

- **Existence check** part examines the existence of elements in the generated list with the retrieved list to make a list of existing names of furniture for possible locations. If there is a room name in the generated list, this part orders Llama 2 to make a list of furniture in the room, and adds them to the generated furniture list. The final output is the list of furniture as possible object locations. If the list is empty, this part returns the *not_found* label, and the system resolves the ambiguity through dialog with the user.

D. Integrated use of ontological knowledge base and LLM

We develop the proposed system by extending the previous system [2] that uses an ontological knowledge base. The ontological knowledge base is constructed by adding concrete information about the environment to general ontology. On the other hand, LLMs are based on large language datasets and thus cover a wide variety of situations but may include information inapplicable to the current environment. We thus prioritize the ontological knowledge base over LLMs. Another point to note is that the system resolves the ambiguity about objects only with either reference to the ontological knowledge base or dialog with the user (as in the case of inquiry about "fruit" in Fig. 1), because object preference is personal in general and cannot be determined by LLMs' general knowledge.

Fig. 6 shows the flow of resolving the ambiguity about object location. The system first consults with the ontological knowledge base. If resolved, the system sends the location to the robot to find the target object. If the robot fails to find the object, the system asks the user about the location to search. If the system cannot get a single location from the ontological knowledge base, it then asks the LLM and, after the consistency verification, sends the candidate location list to the robot to visit. If the LLM does not return a location list, the system asks the user.

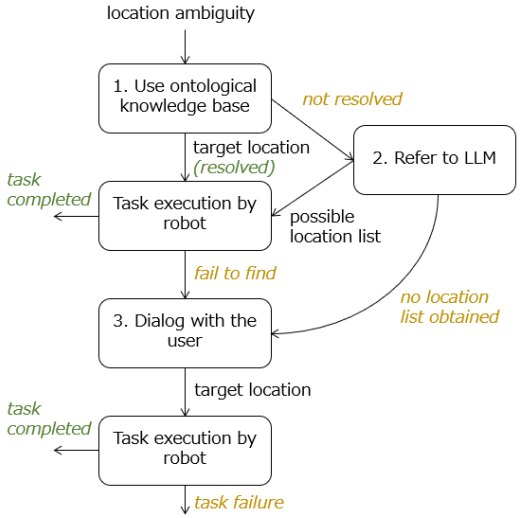


Fig. 6: Flow of location ambiguity resolution.

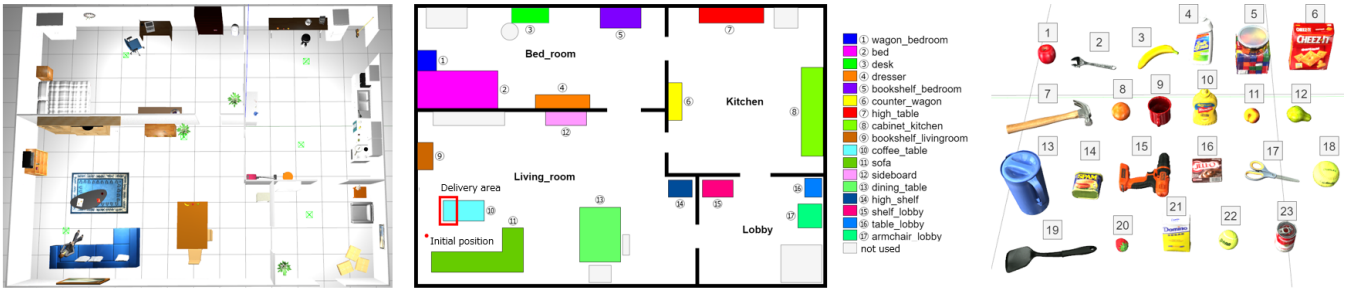
V. Experiments

A. Llama 2 implementation

We use the Llama2-13b-chat model on a PC with Intel Xeon Gold 5222 and Quadro RTX 800 on Ubuntu18.04 via Docker. The parameters used include: max_seq_len=4096, max_batch_size=1, max_gen_len=2048, temperature=0.6, and top_p=0.9. We developed a ROS1 (Melodic) interface to Llama 2 using ROS service. The service node on the Llama 2 server analyzes requests from the client and generates tokens for Llama 2. It also interprets the output from Llama 2 and sends the response to the client.

B. Experimental settings

We set up a four-room home environment (see Fig. 7(a)) using the Gazebo simulator. The placements of furniture are shown in Fig. 7(b). The red point is the initial robot position and the red rectangle is the delivery location in the case that the user's order is to bring an object. Fig. 7(c)



(a) Simulation environment (b) Furniture placements (c) Object set

Fig. 7: Testing environment and objects.

(1) The user ordered the system to find an apple.

(2) The system asked the ontological knowledge base, but it retuned no location information.

(3) The system asked Llama2 and got three candidate locations (high-table, dining-table, counter-wagon) after consistency verification.

(4) The robot visited the first candidate (high-table) but could not find an apple.

(5) The robot then visited the second candidate (dining-table), found an apple, and reported it to the user.

Fig. 8: Example run.

and Table II show the set of objects used, which is a part of the YCB object set [28]. The robot uses YOLOv7 [29] to recognize objects. We added information about all the furniture, rooms, and objects to the ontology. Additionally, each piece of furniture has an attribute indicating the room it is located in.

Fig. 8 shows an example run. The user ordered the system to find an apple. As the system could not find any location information in the ontological knowledge base, it asked Llama 2 for location information. After the consistency verification of the Llama 2's response, the system made

TABLE II: Object set.

No.	Name	No.	Name
1	apple	13	pitcher_base
2	adjustable_wrench	14	potted_meat_can
3	banana	15	power_drill
4	bleach_cleanser	15	power_drill
5	colored_wood_blocks	16	pudding_box
6	cracker_box	17	scissors
7	hammer	19	spatula
8	orange	20	strawberry
9	mug	21	sugar_box
10	mustard_bottle	22	tennis_ball
11	peach	23	tomato_soup_can
12	pear		

a robot plan for visiting the candidate locations one after another. The robot found an apple at the second candidate (dining_table) and came back to the user to report.

C. Experimental conditions and metrics

We compare the following three approaches:

- (1) The system with only the ontological knowledge base (OKB) [2]. It uses OKB to get location information and asks the user when either no location information is available or multiple locations are obtained (denoted as OKB).
- (2) The system with both OKB and LLM, where LLM does not use the dialog history (denoted as OKB+LLM).
- (3) The system with both OKB and LLM, where LLM uses the dialog history (denoted as OKB+LLM+MEM).

We also consider two situations: (a) OKB has default location information, and (b) OKB does not have default location information. Default location information is used for generating location candidates in OKB, and for consistency verification in approaches (2) and (3). We conducted a survey with sixteen subjects to make the location distribution for each object, and extract high probability locations as default ones. Fig. 9 shows an example histogram for "fruit" class, and Table III shows the complete list of object-location relationships. Some objects are categorized into one class (e.g., fruits class).

We test all six combinations of the approaches and the situations against a set of ten user commands shown in Table IV. We execute each command ten times. We also simulate

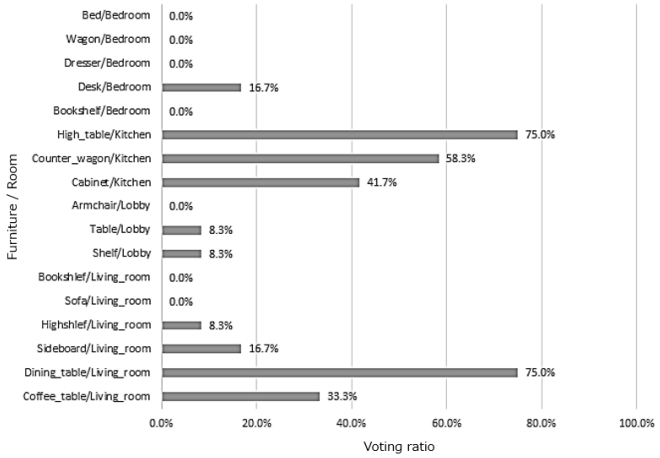


Fig. 9: Example histogram for fruits.

TABLE III: Default locations for each object class.

Object	Default location list
mug	[dining_table, coffee_table, counter_wagon]
cracker_box	[cabinet_kitchen, dining_table, coffee_table, counter_wagon]
fruits	[dining_table, counter_wagon, high_table]
mustard_bottle	[cabinet_kitchen, dining_table, counter_wagon, high_table]
spatula	[cabinet_kitchen, counter_wagon, high_table]
power_drill	[shelf_lobby]
pitcher_base	[cabinet_kitchen, dining_table, counter_wagon, high_table]
ball	[shelf_lobby]
pudding_box	[cabinet_kitchen, dining_table, coffee_table, counter_wagon]
colored_wood_blocks	(not available)
bleach_cleanser	[cabinet_kitchen, counter_wagon]
potted_meat_can	[cabinet_kitchen, counter_wagon, high_table]
tomato_soup_can	[cabinet_kitchen, counter_wagon, high_table]
sugar_box	[cabinet_kitchen, counter_wagon, high_table]
hammer	[shelf_lobby]
adjustable_wrench	[shelf_lobby, sideboard]
scissors	[sideboard, desk]

the user in the experiments such that the user always gives correct answers when asked. The results are evaluated by the following metrics: task completion rate, task completion time, number of user interaction, and the number of furniture pieces visited.

D. Results

1) Task completion rate: Table V summarizes the task completion rates for each of the approaches. OKB and OKB+LLM+MEM exhibit remarkable performance, while OKB+LLM was not competitive. Failure cases occur due to object recognition failures. Adding context is obviously effective in using LLM.

TABLE IV: Set of commands for testing.

Command	Object location
Find an apple.	dining_table
Find a power_drill.	shelf_lobby
Find a colored_wood_blocks.	bookshelf_bedroom
Find a pitcher.	counter_wagon
Find a potted_meat_can.	cabinet_kitchen
Take a peach.	high_table
Take a mug.	coffee_table
Bring a mustard_bottle.	cabinet_kitchen
Bring a sugar_box.	counter_wagon

TABLE V: Task completion rate

OKB	OKB+LLM	OKB+LLM+MEM
98.0%	80.0%	97.0%

TABLE VI: Average task completion time [sec].

	without default location			with default location		
	OKB	OKB +LLM	OKB +LLM +MEM	OKB	OKB +LLM	OKB +LLM +MEM
Find	83.0	311.8	338.3	**81.8**	143.8	157.4
Take	**99.3**	369.9	395.4	105.4	123.4	177.3
Bring	**224.7**	301.7	304.4	225.1	415.7	258.0

2) Time for task completion: Table VI compares the times for completing tasks (only for success cases) for every combination of approach, situation, and command type. The time increases as the number of locations visited increases. OKB can achieve the task most efficiently compared to the others. This is because in OKB, the system asks the user every time there are any ambiguities and, therefore, can get a single, correct target location, thereby minimizing the number of visits. However, this merit must be considered with the demerit of asking the user many times.

3) Number of inquiries to the user: Table VII shows the number of inquiries to the user for each case, and Fig. 10 shows the results indicating statistically significant differences. In these statistics, we assume that object recognition and robot action execution are always successful to focus on the capability of generating location candidates. From the table, where the results are analyzed separately for cases with and without default locations, it is evident that introducing an LLM significantly reduces the number of inquiries. This suggests a considerable improvement in usability. In addition, using the dialog history also has a positive effect.

4) Number of visited pieces of furniture: Table VIII shows the number of visited pieces of furniture for each case, and Fig. 11 shows the results indicating statistically significant differences. Here, we also assume that recognition and action always succeed and have analyzed cases with and without default locations separately. The OKB approach always visits only one piece of furniture because the correct answer is obtained either as ontological knowledge or from the user. In contrast, more pieces of furniture are visited for the other cases (i.e., ones with LLM). Among the approaches with LLM, using the dialog history is effective. More interestingly, irrespective of using the dialog history, the cases with default location information give better results than the ones without it. This is because default location information is utilized to effectively generate candidate location lists with more promising furniture.

5) Execution Cost: Table IX summarizes the average execution time and count for each case. With default location information, the time and the number of inquiries to LLM

TABLE VII: Average number of inquiries to the user.

	without default location			with default location		
	OKB	OKB +LLM	OKB +LLM +MEM	OKB	OKB +LLM	OKB +LLM +MEM
Mean	1.0	0.33	**0.28**	0.8	0.14	**0.08**
SD	**0.0**	0.47	0.45	0.40	0.35	**0.27**

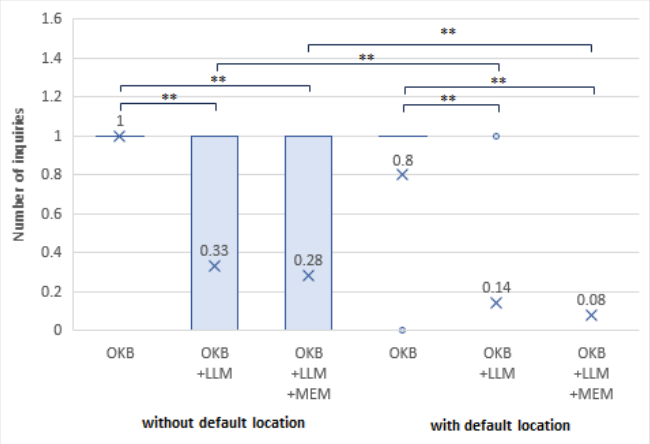


Fig. 10: Statistical analysis of the number of inquiries to the user. (*: p <0.05, **: p <0.01.)

TABLE VIII: Average number of visited furniture.

	without default location			with default location		
	OKB	OKB +LLM	OKB +LLM +MEM	OKB	OKB +LLM	OKB +LLM +MEM
Mean	**1.0**	3.68	3.70	**1.0**	2.45	1.96
SD	**0.0**	2.35	2.41	**0.0**	1.84	1.45

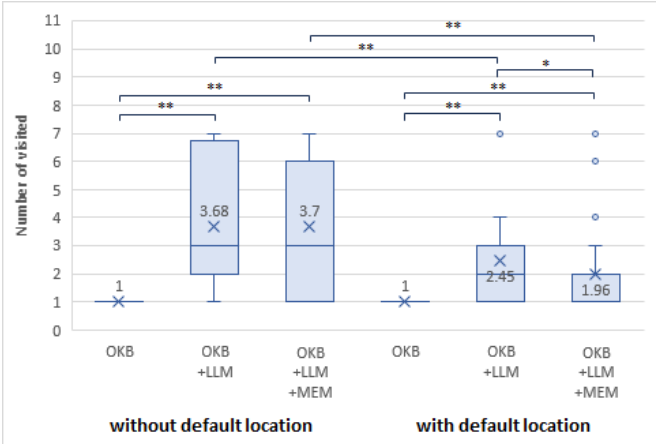


Fig. 11: Statistical analysis of the number of visited furniture. (*: p <0.05, **: p <0.01.)

become shorter. Time for generating outputs is almost the same for cases with and without default location, while the approach using the dialog history needs extra time for managing the memory, with generating more tokens.

TABLE IX: Average execution time and count.

	without default location		with default location	
	OKB +LLM	OKB +LLM +MEM	OKB +LLM	OKB +LLM +MEM
Time for list generation [s]	35.27	30.79	10.81	**9.95**
Number of inquiries to LLM	4.63	3.14	1.80	**1.13**
Time for each LLM output [s]	**7.62**	9.81	7.51	8.75
Size of generated tokens	201.2	246.4	**198.8**	223.4

VI. Summary

This paper has shown a new approach to combining ontological knowledge with a large language model (LLM) for the "bring-me" task. The developed system leverages the LLM, which has vast commonsense knowledge about the household environment. As a result, it can retrieve appropriate knowledge about object locations without overly asking the user, thereby increasing usability. We developed a framework to filter out infeasible solutions by repeatedly applying firm knowledge in the ontological knowledge base. Although the proposed approach seeks to prioritize generating feasible plans, further improvements are needed for better handling the variety of tasks as well as applying the proposed approach to real service robot scenarios.

Acknowledgment

This work is in part supported by the Kayamori Foundation of Information Science Advancement.

References

[1] R. Deits, S. Tellex, P. Thaker, D. Simeonov, T. Kollar, and N. Roy, "Clarifying commands with information-theoretic human-robot dialog," *J. of Human-Robot Interaction*, vol. 1, no. 1, pp. 78–95, 2012.

[2] L. V. Gómez and J. Miura, "Ontology-based knowledge management with verbal interaction for command interpretation and execution by home service robots," *Robotics and Autonomous Systems*, vol. 140, 2021.

[3] M. Shin, M. Jang, M. Cho, and J.-K. Ryu, "Uncertainty-resolving questions for social robots," in *Companion of the 2023 ACM/IEEE Int. Conf. on Human-Robot Interaction*, 2023.

[4] OpenAI, "Gpt-4 technical report," *arXiv preprint arXiv:2303.087745*, 2024.

[5] H, Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[6] A. R. *et al.*, "Learning transferable visual models from natural language supervision," *arXiv preprint arXiv:2103.00020*, 2021.

[7] M. Ahn *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[8] M. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *Proceedings of 2022 Int. Conf. on Machine Learning*, 2022.

[9] I. Singh *et al.*, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE Int. Conf. on Robotics and Automation*, 2023, pp. 11 523–11 530.

[10] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-T. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *arXiv preprint arXiv:2005.11401*, 2020.

[11] H. Li, Y. Su, D. Cai, Y. Wang, and L. Liu, "A survey on retrieval-augmented text generation," *arXiv preprint arXiv:2202.01110*, 2022.
[12] Y. Lu, W. Feng, W. Zhu, W. Xu, X. E. Wang, M. Eckstein, and W. Y. Wang, "Neuro-symbolic procedural planning with commonsense prompting," *arXiv preprint arXiv:2206.02928*, 2022.
[13] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," in *Proceedings of 2023 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2023.
[14] M. S. Sakib and Y. Sun, "From cooking recipes to robot task trees – improving planning correctness and task efficiency by leveraging llms with a knowledge network," *arXiv preprint arXiv:2309.09181*, 2023.
[15] T. Kagaya, T. Yuan, Y. Lou, J. Karlekar, S. Pranata, A. Kinose, K. Oguri, F. Wick, and Y. You, "Rap: Retrieval-augmented planning with contextual memory for multimodal llm agents," *arXiv preprint arXiv:2402.03610*, 2024.
[16] J. Olszewska *et al.*, "Ontology for autonomous robots," in *Proceedings of 26th Int. Symp. on Robot and Human Interactive Communication (RO-MAN)*, 2017, pp. 189–194.
[17] D. Beßler, M. Pomarlan, and M. Beetz, "Owl-enabled assembly planning for robotic agents," in *Proceedings of 17th Int. Conf. on Autonomus Agents and MultiAgent Systems (AAMAS)*, 2018, pp. 1684–1692.
[18] G. Ricardez, Y. Osaki, M. Ding, J. Takamatsu, and T. Ogasawara, "Estimating the operation of unknown appliances for service robots using cnn and ontology," in *Proceedings of 2nd IEEE Int. Conf. on Robotic Computing (IRC)*, 2018, pp. 181–182.
[19] K. Morohashi and J. Miura, "Query generation for resolving ambiguity in user's command for a mobile service robot," in *Proceedings of 2019 European Conf. on Mobile Robots*, 2019.
[20] P. Pramanick, C. Sarkar, S. Banerjee, and B. Bhowmick, "Talk-to-resolve: Combining scene understanding and spatial dialog to resolve granular task ambiguity for a collocated robot," *Robotics and Autonomous Systems*, vol. 155, 2022.
[21] J. Liang *et al.*, "Code as policies: Language model programs for embodied control," *arXiv preprint arXiv:2209.07753*, 2022.
[22] M. Gramopadhye and D. Szafir, "Generating executable action plans with environmentally-aware language models," *arXiv preprint arXiv:2210.04964*, 2022.
[23] F. Ocker, J. Deigmöller, and J. Eggert, "Exploring large language models as a source of common-sense knowledge for robots," *arXiv preprint arXiv:2311.08412*, 2023.
[24] M. Gramopadhye and D. Szafir, "Generating executable action plans with environmentally-aware language models," *arXiv preprint arXiv:2210.04964*, 2022.
[25] D. Paulius, R. M. Y. Huang, W. Buchanan, J. Sam, and Y. Sun, "Functional object-oriented network for manipulation learning," in *Proceedings of 2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2016, pp. 2655–2662.
[26] "Langchain," https://github.com/langchain-ai/langchain/.
[27] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," *arXiv preprint arXiv:2201.11903*, 2022.
[28] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *Proceedings of 2015 IEEE Int. Conf. on Advanced Robotics*, 2015.
[29] C. Wang, A. Bochkovsky, and H. Liao, "Yolov7: Trainable beg-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.