Large Language Models as Commonsense Knowledge for Large-Scale Task Planning

Anonymous Author(s)
Affiliation
Address
`email`

Appendix

A Experimental environments

We use the VirtualHome simulator [1] to evaluate our approach as well as the baseline methods. VirtualHome is a 3D household environment with partial observation, large action space, and long planning horizon. It contains hundreds of interactive objects and containers, allowing it to perform various household object rearrangement tasks. This section introduces details of the tasks, the goal specifications, the actions, and the observations in our experimental settings.

A.1 List of objects, containers, surfaces, and rooms in the apartment

We list all the objects that are included in our experimental environment. Here, we can put *moveable objects* into the *Containers* or on the *Surfaces*. The *Containers* and *Surfaces* are located at a *Room* in the apartment.

- *Containers*: bathroom cabinet, kitchen cabinet, bathroom counter, fridge, oven, dishwasher, microwave, stove, bathroom cabinet
- *Surfaces*: bed, bookshelf, cabinet, coffee table, cutting board, floor, fryingpan, kitchen counter, kitchen table, nightstand, sofa, stove
- *moveable objects*: alcohol, apple, banana, bar soap, bell pepper, boardgame, book, box, bread slice, bucket, candle, candy bar, carrot, cellphone, cereal, chicken, Chinese food, chips, chocolate syrup, clock, clothes pants, clothes pile, clothes shirt, coatrack, coffeepot, condiment bottle, condiment shaker, cooking pot, crackers, crayons, creamy buns, cupcake, cutlery fork, cutlery knife, cutlets, cutting board, dish bowl, dishwashing liquid, face cream, folder, fryingpan, glasses, globe, hair product, hanger, juice, keyboard, lime, lotion bottle, magazine, milk, milkshake, minced meat, mouse, mug, notes, oven tray, pancake, paper, pear, pie, pillow, plate, plum, poundcake, pudding, radio, remote control, salad, salmon, slippers, sports ball, sundae, teddybear, toilet paper, toothbrush, toothpaste, towel, towel rack, toy, washing sponge, water glass, whipped cream, wine, wineglass
- *Rooms*: bedroom, bathroom, living room, kitchen.

A.2 Tasks

We use the object rearrangement tasks for evaluation. The task is to search for one or more objects in the house and move them to the desired positions. We use natural language as the interface to specify the tasks. Thus, the agent should take as input the natural language instruction and observations, and then output actions.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

The tasks are randomly sampled from different distributions. We define various types of object rearrangement tasks for evaluation:

- *Simple*: this task is to move one object in the house to the desired location. The combination of the object and desired location has appeared in the training dataset.
- *Novel Simple*: this task is to move one object in the house to the desired location. The combination of the object and desired location has**not** appeared in the training dataset.
- *Comp.*: this task is composed of 2 *Simple* tasks, moving more than one object in the house to their desired location. This kind of task has a longer planning horizon as it requires moving multiple objects to complete. The combinations of *Simple* tasks have appeared in the training dataset.
- *Novel Comp. (2)*: this task is composed of 2 *Simple* tasks, moving more than one object in the house to their desired location. The combinations of *Simple* tasks have not appeared in the training dataset.
- *Novel Comp. (3)*: this task is composed of 3 *Simple* tasks, moving more than one object in the house to their desired location. This kind of task has the longest planning horizon. The combinations of *Simple* tasks have not appeared in the training dataset.

We also have different household environments:

- *Seen Apartment*: the map of the apartment is shown in Figure 1. These household environments are the same as the ones in the training set, while the object positions are randomly initialized according to a pre-defined commonsense distribution in VirtualHome [1].
- *Unseen Apartment*: the map of the apartment is shown in Figure 2. These household environments are not the same as the ones in the training set. The object positions are also sampled from a different pre-defined commonsense distribution in VirtualHome [1].

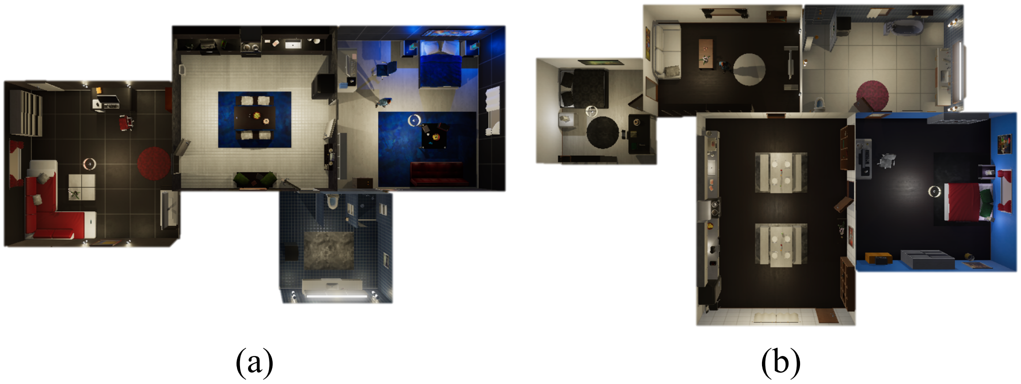


Figure 1: The map of the *seen apartments* in our setting. These household environments are the same as the ones in the training set, while the object positions are randomly initialized according to a commonsense distribution.

A.3 Goal specification

Similar to prior works [2], we define the goal in the VirtualHome system by a set of predicates. For instance, a goal can be defined by *Inside(apple, fridge):2; Inside(plate, dishwasher):1*, meaning "put two apples inside the fridge and put one plate inside the dishwasher." For *Simple* and *Novel Simple* tasks, it only requires moving one object, while *Comp.* and *Novel Comp.* have more than one object to move.

A.4 Actions

In VirtualHome, the agent is able to navigate in the environment, grab an object, put an object inside the containers (e.g., fridge) or on the surfaces (e.g., table), open and close the container, etc. The

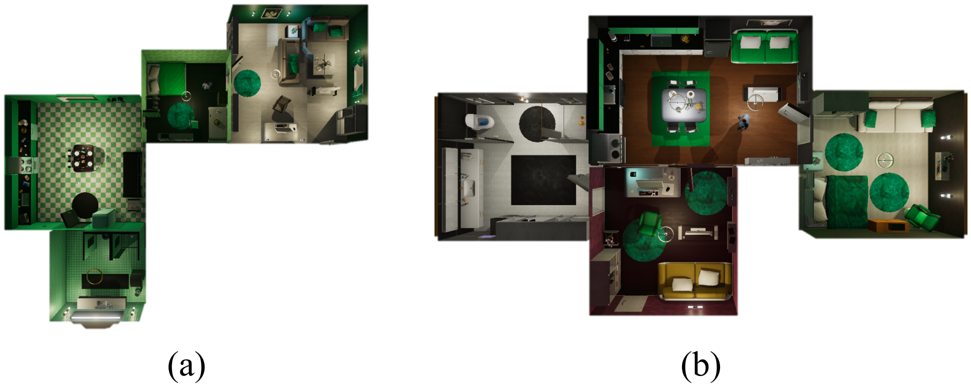


Figure 2: The map of the *unseen apartments* in our setting. These household environments are not the same as the ones in the training set. The object positions are also sampled from a different commonsense distribution.

actions in VirtualHome are grounded to moveable objects, containers, or rooms in the environment. For example, `Open(5)` is to open an object with index (5). The list of available actions in our setting are listed below:

- `Walk(<item>)`: walk to the `<item>`. The `<item>` can be a moveable object, a container, or a room. The precondition of this action is that the `<item>` is visible. The effect of this action is that the agent is close to the `<item>` if the `<item>` is an object or inside the `<item>` if the `<item>` is a room. The action is translated into the sentence "walk to the `<name of item>`" when feeding into LLMs.
- `Open(<item>)`: open the `<item>`. The `<item>` can be a moveable object or a container. The precondition of this action is that the agent should be close to `<item>`. The effect of this action is that the `<item>` is opened. The action is translated into the sentence "open the `<name of item>`" when feeding into LLMs.
- `Close(<item>)`: close the `<item>`. The `<item>` can be a moveable object or a container. The precondition of this action is that the agent should be close to `<item>`. The effect of this action is that the `<item>` is closed. The action is translated into the sentence "close the `<name of item>`" when feeding into LLMs.
- `Grab(<item>)`: grab the `<item>`. The `<item>` should be a moveable object. The precondition of this action is that the agent should be close to the `<item>`, and the agent is not holding any objects. The effect of this action is that the agent will hold the `<item>`. The action is translated into the sentence "grab the `<name of item>`" when feeding into LLMs.
- `PutIn(<item1>, <item2>)`: put the moveable object `<item1>` inside the container `<item2>`. The precondition of this action is that the agent should be close to the `<item2>` and holding `<item1>`. The effect of this action is that the agent is not holding any objects, and the `<item1>` is inside the `<item2>`. The action is translated into the sentence "put the `<name of item1>` inside the `<name of item2>`" when feeding into LLMs.
- `PutBack(<item1>, <item2>)`: put the moveable object `<item1>` on the surface `<item2>`. The precondition of this action is that the agent should be close to the `<item2>` and holding `<item1>`. The effect of this action is that the agent is not holding any objects, and the `<item1>` is on the `<item2>`. The action is translated into the sentence "put the `<name of item1>` on the `<name of item2>`" when feeding into LLMs.

A.5 Observations

We use the same representation as [2] for partial observation. The observation is a list of visible objects and relationships between those objects. Each object or container has a state: *open* or *close*. The fine-tuned GPT2 policy [2] also uses the 3d coordinates of the object. We also use relationships to connect different objects, such as `Inside(apple, fridge)`. Those relationships are translated to natural language descriptions when feeding into LLMs, such as "an apple is inside the fridge."

B Data gathering

Similar to prior works [3, 2], we collect expert trajectories in VirtualHome using regression planning with handcrafted heuristics[1]. The expert has full observation of the environment. Given the goal predicates and full observation, the agent will use the handcrafted heuristics for each task to effectively search for the solutions. The expert also has a handcrafted mechanism for compositional tasks to decompose one task into subtasks and finish them progressively. For each trajectory, we include the goal predicates (used by the VirtualHome system and the expert agent), the goal instruction (used by the agent), the partial observation for each time step (not used by the expert agent, the expert agent uses full observation), and the expert actions.

C Implementation details of belief in LLM-MCTS

This section introduces our implementation details for the belief of states in GPT3.5-MCTS. The source code will be released at `https://llm-mcts.github.io` before the publication.

C.1 State representation

We represent the states by a list of objects and their relationships. Each object has a unique name and id in the simulator, as well as the state of the object. We use the same unique name and id in our state representation. The relationships connect different objects, containers, surfaces, and rooms. The VirtualHome contains 59 different types of relationships, including `Inside, On, Close, Facing`, etc. We use the same type of relationships in our state representation.

C.2 Belief

The belief of the state also contains a list of objects and their relationships. However, we parameterize the relationships by a vector, representing the probability that the relationship is true. This vector is affiliated with the object representation. For simplicity, we only include the relationships `Inside, On` in our belief, as we only query LLM about the object positions to build up the commonsense belief of the state.

When building up a state's belief, we query LLM to predict the position of each moveable object, container, and surface. The position of a moveable object is specified by the relationships (i.e., `Inside` or `On`) between itself and a container or surface. The position of a container or a surface is specified by its relationship (i.e., `Inside`) to the room. We use sampling to approximate the distribution of the position. The moveable objects' belief of position is represented by a vector whose dimension is the same as the total number of containers and surfaces in the house. Each vector entry denotes the probability that whether the object is inside a specific container or on a specific surface is true. When asking LLM to predict the object positions, we asked LLM for M times and received multiple responses from LLM. We then count each entry's total number of predictions and normalize them to become a probability distribution. We initialize the value of other unsampled entries in the vector by a lower bound of the probability 1×10^{-3} to ensure that the model will not eliminate other possibilities when the commonsense model is wrong.

The agent will receive new observations to update their belief when interacting with the environment. We will first predict the next state of the agent by the transition function and then update the belief of the object positions by new observations. Suppose the object is inside the current observation. In that case, the other entry of the relations between objects will be masked out by zero, and the entry of the relationships in observation will be replaced by the value of one. However, if a relationship is not inside the observation, the value of the corresponding entry will be replaced by zero, and the vector will be normalized again.

[1] Their implementation is available at the GitHub repository: `https://github.com/xavierpuigf/watch_and_help.git`

D Visualized examples

We provide a set of successful (shown in Figure 3) and failed trajectories (shown in Figure 4) to give a better understanding of the tasks and our method. Policy, model, and translation errors are the primary causes of failures. Among these, policy errors are responsible for the majority of the failures. Often time, the policy produces unreasonable behaviors that mislead the search procedure. For example, it usually outputs inadmissible actions, such as "*walk to the cutlery fork*" where the "*cutlery fork*" is not in the observation (shown in Figure 4 (a)). It also produces back-and-forth behaviors, resulting in an unreasonable heuristic and slowing the search procedure. For example, when putting objects inside the microwave, it is sometimes struck by repeatedly opening and closing the microwave. For model error, the predicted positions of objects are not always correct. Since a random rollout policy is employed, incorrect object states can result in higher Q-values than correct states, leading to misguided exploration (shown in Figure 4 (b)). The wrong translation also compromises the performance as we translate the response from LLM to admissible action or object names to ensure executability. This is caused in part by the VirtualHome environments, as the policy might not understand the underlying logic of the actions in VirtualHome, such as you have to walk close to interact with the object. Thus, if the LLM outputs "open fridge" but is not close enough to the fridge, the action will be translated to other admissible actions ("open fridge" is not inside the admissible actions for this case as it is invalid due to the setting of VirtualHome).

E Prompts

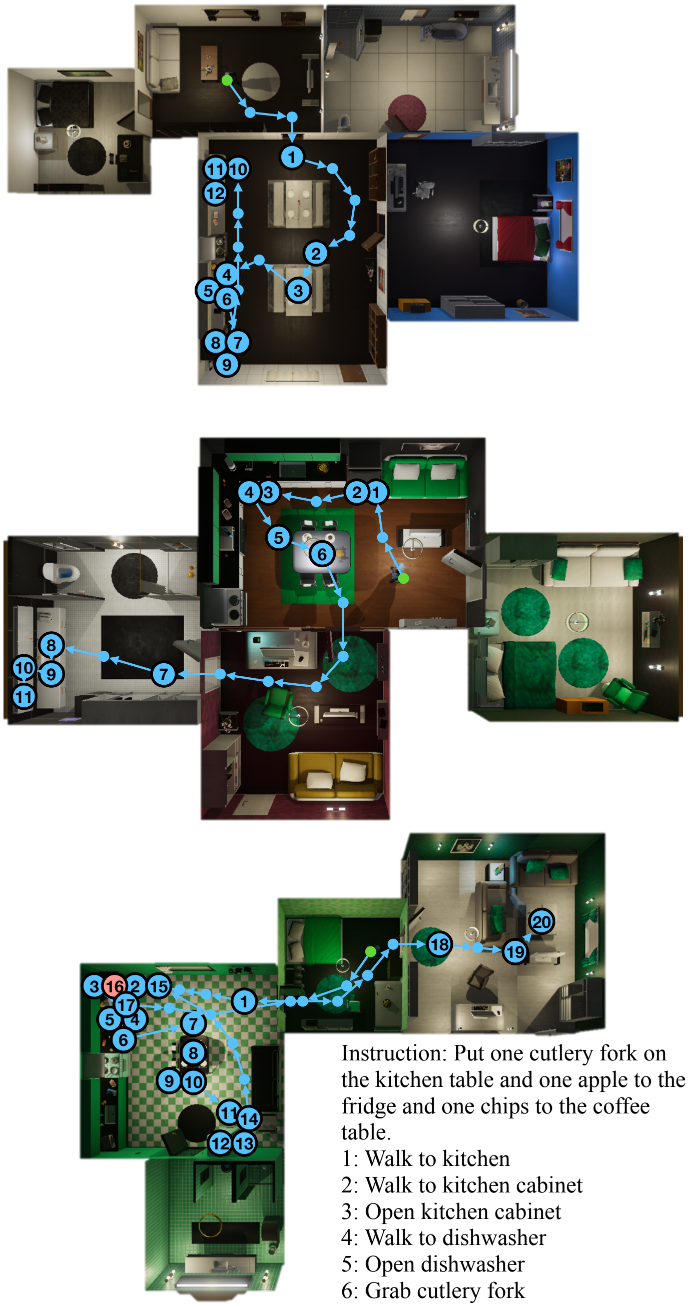
One example prompt for the LLM policy is shown in Listing 1. The exact prompt we used for building up the commonsense belief is shown in Listing 2. Please note that the final question in Lising 2 is one example of the questions for demonstration.

References

[1] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8494–8502.

[2] S. Li, X. Puig, C. Paxton, Y. Du, C. Wang, L. Fan, T. Chen, D.-A. Huang, E. Akyürek, A. Anandkumar *et al.*, "Pre-trained language models for interactive decision-making," *Advances in Neural Information Processing Systems*, vol. 35, pp. 31 199–31 212, 2022.

[3] X. Puig, T. Shu, S. Li, Z. Wang, Y.-H. Liao, J. B. Tenenbaum, S. Fidler, and A. Torralba, "Watch-and-help: A challenge for social perception and human-ai collaboration," *arXiv preprint arXiv:2010.09890*, 2020.



Instruction: Put one plate inside the kitchen cabinet and one chicken inside microwave.

1: Walk to kitchen
2: Walk to plate
3: Grab plate
4: Walk to kitchen cabinet
5: Open kitchen cabinet
6: Put plate inside kitchen cabinet
7: Move fridge
8: Open fridge
9: Grab chicken
10: Walk to microwave
11: Open microwave
12: Put chicken inside microwave

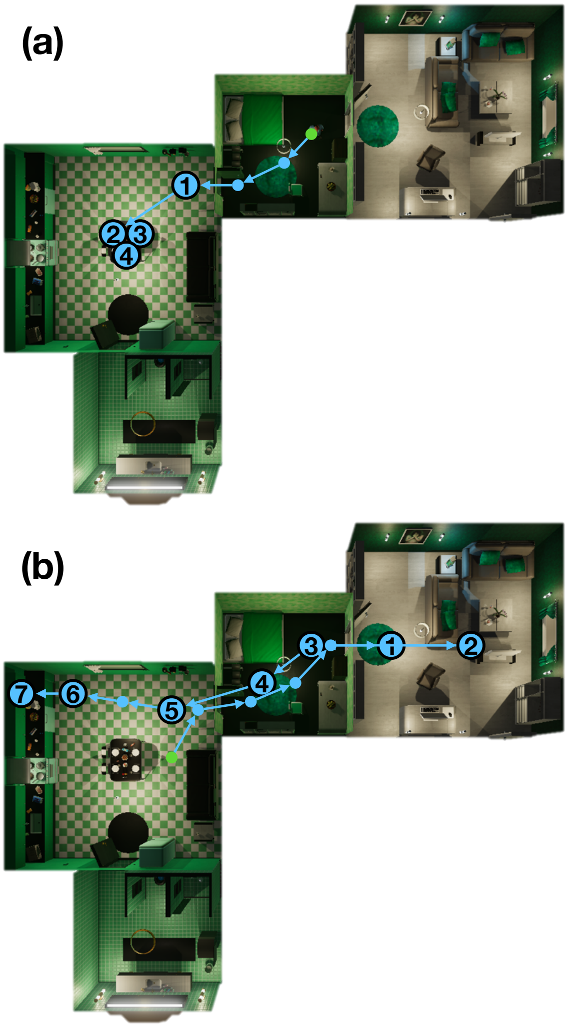
Instruction: Put one apple on the kitchen table and one toothbrush inside the bathroom cabinet.

1: Walk to fridge
2: Open fridge
3: Walk to apple
4: Grab apple
5: Walk to kitchen table
6: Put apple on kitchen table
7: Walk to bathroom
8: Walk to toothbrush
9: Grab toothbrush
10: Open bathroom cabinet
11: Put toothbrush inside bathroom cabinet

Instruction: Put one cutlery fork on the kitchen table and one apple to the fridge and one chips to the coffee table.

1: Walk to kitchen
2: Walk to kitchen cabinet
3: Open kitchen cabinet
4: Walk to dishwasher
5: Open dishwasher
6: Grab cutlery fork
7: Walk to kitchen table
8: Put cutlery fork on kitchen table
9: Walk to apple
10: Grab apple
11: Walk fridge
12: Open fridge
13: Put apple inside fridge
14: Close fridge
15: Walk to kitchen cabinet
16: Open kitchen cabinet (Failed)
17: Grab chips
18: Walk to living room
19: Walk to coffee table
20: Put chips on coffee table

Figure 3: Successful examples



Instruction: Put one cutlery fork on the coffee table
1: Walk to kitchen
2: Walk to cutlery knife
3: Walk to cutlery knife
4: Walk to cutlery knife
…

Instruction: Put one apple inside the fridge
1: Walk to living room
2: Walk to coffee table
3: Walk to bedroom
4: Walk to nightstand
5: Walk to kitchen
6: Walk to kitchen cabinet
7: Open kitchen cabinet
…

Figure 4: Failed examples. (a) Policy error and translation error. LLM outputs walk to the cutlery fork, but the cutlery fork is not in observation. We use embeddings to evaluate the most similar valid actions. Therefore it translates the action to one similar action "walk to cutlery knife." The action has an incorrect semantic meaning and causes failure. (b) model error. The LLM predicts the apple is on the nightstand in the bedroom and on the coffee table in the living room. As we are using random rollout to get the estimation of the reward, there will be situations when the incorrect actions result in a higher estimated Q value, thereby misleading the exploration.

Listing 1 Example prompt for the heuristic policy

```
You need to generate a high-level plan for completing a household task using the
allowed actions and visible objects.
Allowed actions:  walk to <object>, walk to <room>, walk to <container>, walk to
<surface>, grab <object>, open <container>, close <container>, put <object> on
<surface>, put <object> inside <container>.
Rooms in the house:  bedroom, bathroom, living room, kitchen
You need to strictly follow the format in the following examples:

Goal:  Put one apple inside the fridge
Completed actions:  walk to the kitchen, walk to the apple
Current Observation:  a kitchen table is inside the kitchen, a kitchen counter is
inside the kitchen, an apple is on the kitchen counter, a plate is on the kitchen
table, a banana is on the kitchen counter, a fridge is inside the kitchen and fridge
is closed, a kitchen cabinet is inside the kitchen and kitchen cabinet is closed,
a cutlery knife is on the kitchen table, a microwave is inside the kitchen and
microwave is closed, a dishwasher is inside the kitchen and dishwasher is closed.
Next actions:  grab the apple, walk to the fridge, open the fridge, put the apple
inside the fridge, close the fridge, done.

Now, finish the next following task.

Goal:  Put one apple on the kitchen table
Completed actions:  walk to the kitchen
Current observation:  a kitchen table is inside the kitchen, an apple is on the
kitchen table, a kitchen counter is inside the kitchen, an apple is on the kitchen
counter, a cutlery knife is on the kitchen counter, a fridge is inside the kitchen
and fridge is closed, a kitchen cabinet is inside the kitchen and kitchen cabinet
is closed, a kitchen table is inside the kitchen, a plate is on the kitchen table,
a pounding cake is on the kitchen table, a microwave is inside the kitchen and
microwave is closed, a dishwasher is inside the kitchen and dishwasher is closed.
Next actions:
```

Listing 2 Example prompt for the commonsense world model

```
You need to predict the positions of the moveable objects, containers, and surfaces
in the apartment according to the commonsense.
Rooms in the apartment:  bedroom, bathroom, living room, kitchen.
Containers in the apartment:  bathroom cabinet, kitchen cabinet, bathroom counter,
fridge, oven, dishwasher, microwave, stove, bathroom cabinet.
Surfaces in the apartment:  bed, bookshelf, cabinet, coffee table, cutting board,
floor, fryingpan, kitchen counter, kitchen table, nightstand, sofa, stove.

You need to strictly follow the format in the following examples:

Question:  what are the possible positions of strawberry?
Answer:  Inside fridge, On kitchen table.
Question:  what are the possible positions of soap?
Answer:  On bathroom counter.
Question:  what are the possible positions of water cup?
Answer:  On kitchen table, Inside dishwasher.

Now, answer the next following question.

Question:  what are the possible positions of apple?
Answer:
```