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

Zirui Zhao, Wee Sun Lee, and David Hsu
School of Computing, National University of Singapore
{ziruiz, leews, dyhsu}@comp.nus.edu.sg

***Abstract*—Natural language provides a natural interface for human communication, yet it is challenging for robots to comprehend due to its abstract nature and inherent ambiguity. Large language models (LLMs) contain commonsense knowledge that can help resolve language ambiguity and generate possible solutions to abstract specifications. While LLMs have shown promise as few-shot planning policies, their potential for planning complex tasks is not fully tapped. This paper shows that LLMs can be used as both the *commonsense model of the world* and the *heuristic policy* in search algorithms such as Monte Carlo Tree Search (MCTS). MCTS explores likely world states sampled from LLMs to facilitate reasoned decision-making. The commonsense policy from LLMs guides the search to relevant parts of the tree, substantially reducing the search complexity. We demonstrate the effectiveness of our method in daily task-planning experiments and highlight its advantages over using LLMs solely as policies.**

I. Introduction

Natural language provides an intuitive and user-friendly interface for humans to specify goals for robots. However, the abstraction and ambiguity inherent in natural language often result in incomplete information regarding both the goal state and how the goal can be achieved. For example, a human might instruct a robot to "bring me a fruit," requiring the robot to have commonsense knowledge of the types of objects that can be considered fruits, as well as understand the goal state indicated by that instruction. Furthermore, instructions such as "bring me a fruit" describe high-level tasks that require multiple actions to complete. For instance, the robot may need to look in the fridge for a fruit (recognizing that apples, oranges, peaches, etc. are fruits), and if the fridge has no fruit, look further in the pantry. A typical household environment comprises hundreds of moveable items and containers, resulting in a huge search space that makes the planning intractable. Utilizing commonsense knowledge to deduce missing information and reduce search space is crucial for the robot to achieve the natural language-specified goal successfully.

Recently, large language models (LLMs) [5, 26, 6] have become prevalent in AI research. LLMs encode vast commonsense knowledge and exhibit impressive performance in information retrieval and text generation. Li et al. [23] suggested utilizing the fine-tuned LLMs as policies to provide instructions for achieving abstract, incomplete goals specified by natural language. Huang et al. [18] and Ahn et al. [2] proposed to use the pre-trained LLM directly as a few-shot or zero-shot policy for planning without fine-tuning, with only a few examples as prompts. Furthermore, Huang et al. [19] proposed translating sensor observations into language feedback to provide to the pre-trained LLMs so that it is able to provide the next instruction conditioned on the outcomes of the previous instructions. These works use LLMs as policies, exploiting the commonsense knowledge of the correct action to take given the history of actions and observations previously encountered.

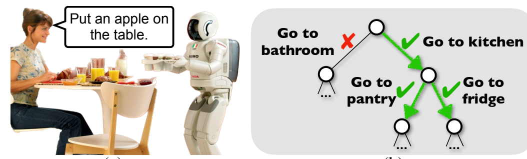


Fig. 1. We focus on language-instructed object rearrangement tasks as shown in (a). We use the commonsense knowledge to guide the search algorithms to explore potentially promising actions and make reasoned decisions (b).

In this study, we leverage the vast commonsense knowledge encapsulated in LLMs by incorporating it into an online search algorithm, i.e., Monte Carlo Tree Search (MCTS) [9]. MCTS utilizes LLMs' rich world knowledge as a model for reasoned decision-making, which is not fully exploited when LLMs are used solely as policies. Specifically, LLMs provide prior common sense beliefs of the world that can be used to sample likely states. These generated samples encompass various common scenarios, such as fruits being present on the kitchen counter, inside the fridge, or in the pantry. MCTS summarizes the useful information in searching the likely states through the estimated Q value (the expected reward after taking action) so as to make a reasonable decision. The algorithm progressively updates its belief of the world as it acts in the world and receives observations to rectify model errors. In addition, instead of directly providing instructions, we employ the LLM as a search heuristic to guide exploration only toward promising parts of the search tree. For instance, when given the instruction "bring me a fruit," the LLM uses commonsense knowledge to prioritize opening the fridge or pantry rather than opening the trash can. By utilizing LLM policies as search heuristics, we transform an otherwise intractable search task into a computationally practical one.

In contrast to solely using LLMs as policies, our method is able to exploit knowledge about likely states of the world in the LLMs to facilitate reasoned planning through tree search[1]. MCTS enables LLM to leverage its world modeling knowledge and explore new combinations of actions to tackle novel tasks.

[1]For the purpose of the search, we assume that the action set is known with known deterministic transitions. In the experiments, the physical-level motion planning is processed separately when executing an action.

By employing the LLM policy as a heuristic, we substantially reduce the search complexity required to identify good actions. We demonstrate the advantages through our experiments conducted in large, complex household environments, specifically in the context of daily task planning.

II. Problem statement

We aim to solve object rearrangement task-planning problems in a household environment. In this task, a human user uses natural language to instruct the robot to find an item and move it to a target location. The problem can be formulated as a Partially Observable Markov Decision Process (POMDP): $(S, A, \Omega, T, O, R, \gamma)$. The state space S is object-centric and consists of the robot's position and the positions of moveable items and containers. The actions space A defines the action that the robot can do. In our case, we predefine a set of high-level actions: object picking, object placing, moving, opening a container, and closing a container. The observation space, Ω, is the space for natural language observations. T defines the transition function of states, which we assume to be deterministic. O is the observation function at the current state s. We assume it is deterministic, but the robot will only observe items in the same room or in an opened container at the robot's location. $R(s, a)$ is the reward function determined by the action a taken at the state s. In our setting, the robot will receive a high positive reward if all the items are at the goal position, and the task will be terminated. The discount factor is specified by γ. The history trajectory h_t at time step t consists of a sequence of executed actions and received observations up to time $t-1$, $h_t = (o_0, a_0, o_1, a_1, \ldots, o_{t-1}, a_{t-1})$. The objective is to find an optimal policy $\pi^*(h_t)$ that maximize the expected cumulative rewards $\pi^*(h_t) = \arg\max_{a \in A} \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i R(s_{t+i}, a_{t+i}) | a_t = a\right]$.

III. LLM-MCTS: Monte Carlo planning with commonsense knowledge

As shown in Fig 2, our core idea is to use LLMs as the commonsense world model and the heuristic policy within the MCTS framework, enabling reasoned and informed decision-making for daily tasks. MCTS enables LLM to leverage its world modeling knowledge and explore new combinations of actions to tackle novel tasks. LLM helps MCTS through the biased sampling of states and action selection, improving its efficiency in resolving large, complex task-planning problems.

A. Large Language Models as commonsense world model

A commonsense prior belief of states can improve the effectiveness of object and location searches by prioritizing the search to appropriate locations. Our approach utilizes LLM's commonsense knowledge to generate the initial belief of states, which is updated with each action and observation in the real world. MCTS samples from the belief in simulation to estimate the value of the action.

We use object-centric state representation and categorize the objects in the house as moveable objects (e.g., apples), containers (e.g., fridge), and surfaces (e.g., kitchen table). The states of a moveable object might be inside the containers or on the surfaces. The containers and surfaces should be inside a room. Similar to [23, 29], we maintain the belief in object-centric graphs, where nodes are objects and edges describe abstract-level relationships (e.g., an apple is inside the fridge, and the fridge is inside the kitchen) between objects and rooms. Details are in the Appendix.

Assume a dataset $\mathcal{D}$ is accessible, containing expert actions and observations in similar household environments to solve daily tasks. LLMs can use the observations in the data to know what are the objects in the house and predict their positions, forming the commonsense belief of the state. To achieve this, we find all the objects, containers, and surfaces that appeared in the dataset $\mathcal{D}$ to form a list of objects $\mathcal{D}_{\text{obj}}$ using a unique name for all of them. To approximate $b(s_0)$, we ask the LLMs to sample the positions of objects M times. For each sample, we ask the LLM to predict the position of objects using $\mathcal{D}_{\text{obj}}$ and a fixed prompt. We use three prompt examples to provide example formats of the response. The exact prompts we used are provided in the appendix. As the responses from LLM are free-form natural language, we have to precisely map those expressions to $\mathcal{D}_{\text{obj}}$ for consistent state representation. Thus, we encode the names of objects in the LLM's response into embeddings using sentence-BERT $f(\cdot)$ [30] and examine their cosine similarity to the unique name of objects in $\mathcal{D}_{\text{obj}}$: $\text{CosineSim}(e_i, e) = \frac{f(e_i)f(e)}{\|f(e_i)\|\|f(e)\|}$, where e is the name of objects, containers, or surfaces in the LLM's response, and $e_i \in \mathcal{D}_{\text{obj}}$ are the unique names in the object list. We select the most similar expressions in $\mathcal{D}_{\text{obj}}$ to form the sampled state. For example, when querying the position of an apple, the LLM's response is "on the kitchen table," we use the above technique to translate "the kitchen table" to "kitchentable," a unique name in $\mathcal{D}_{\text{obj}}$.

Similar to prior works [41], we use LLMs to translate the natural language goal into a formal representation for the search algorithm. We use a fixed set of prompt examples for LLM to interpret natural language goals, such as "put one apple into the fridge" is translated as a tuple "(*apple*, *inside*, *fridge*)." For compositional instructions, it will translate it into multiple tuples, such as "put one apple on the kitchen table and one plate inside the dishwasher" is translated as "(*apple, on, kitchentable*), (*plate, inside, dishwasher*)." Similar to initial belief generation, we precisely map the LLM-generated goal into the admissible expressions in $\mathcal{D}_{\text{obj}}$ for search using the same representation as the state. In MCTS, the goal is used to identify the reward. As the representations are the same, we can directly check whether the object's state is the same as the goal by string matching. If the goal is reached, it will receive a large positive reward, or 0 otherwise.

B. Large Language Models as heuristic policy

We use LLMs to play the role of $\pi(a|h)$ in PUCT to guide the action selection in the simulation procedure. In this procedure, the LLM takes as input the examples in the dataset, the goal description, the current observation, and the history of actions, and then outputs the suggested action plan.

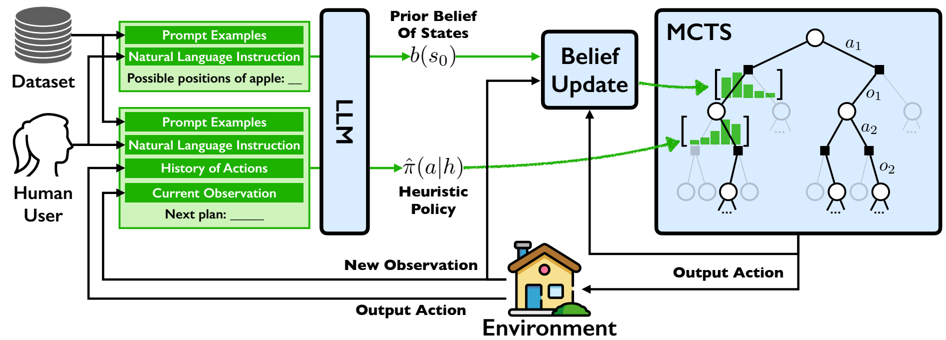


Fig. 2. Overview of LLM-MCTS. We query the large language model (LLM) for the commonsense belief of states and the commonsense planning policy for action selection. For each simulation in the MCTS algorithm, we sample from the commonsense belief to obtain an initial state of the world and use the LLM as heuristics to guide the trajectory to promising parts of the search tree.

Similar to [23], the observations and goal description are translated into English sentences. As the answer of LLM is from the conditional distribution of the following words given the context, it can also be viewed as a commonsense policy of actions to take conditioned on the context of tasks, observations, and completed actions. However, direct implementation and access to the probability value of the GPT-3.5 is not available. Thus, we propose an empirical policy distribution $\hat{\pi}$ that uses sampling to approximate the policy distribution.

We sample the LLM for M times to approximate the policy probability distribution. For each sample, we query the LLM with prompt and trajectory history h and receive an answer of the following actions to take $\alpha_i \sim \mathrm{LLM}(h, \mathrm{prompt})$, where α_i is the first action of the answer. The prompt examples are retrieved from the dataset according to the similarity to the current language instruction ℓ. We use [30] to translate the instructions in the dataset $\ell_i \in \mathcal{D}$ into embedding and examine their cosine similarity to the current instruction: $\mathrm{CosineSim}(\ell_i, \ell)$. In experiments, we use a subset of $\mathcal{D}$ to show its performance when restricted to a small training set. We select the top K similar instructions and use the corresponding expert trajectories as a K-shot prompt. However, the answer α_i is a free-formed natural language sentence that cannot be mapped to admissible actions for the agent directly. To ensure that the action can be executed, we follow the method in prior works [18] to represent the actions and admissible actions by embeddings from [30] and evaluate their cosine similarity $\mathrm{CosineSim}(\alpha_i, a)$. The empirical policy distribution is formulated as follows: $\hat{\pi}(a|h) = \lambda\frac{1}{|A|} + (1-\lambda)\mathrm{Softmax}\{\sum_{i=1}^{M}\mathrm{CosineSim}(\alpha_i, a) - \eta\}$, where η is the average value of $\sum_i \mathrm{CosineSim}(\alpha_i, a)$ and $|A|$ is the size of the admissible action space. λ is a hyper-parameter that adds randomness to the belief, as the sampled actions from LLM could be very deterministic. Therefore, the empirical policy distribution is a mixture of approximated policy from LLM and uniform distribution.

C. Search with commonsense world model and heuristic policy

We integrate the world model and policy from LLM in MCTS, presented in Alg 1 in Appendix B. For each simulation, MCTS samples a state from the belief $b(s)$ at the root (line 4). It independently samples one position for each object to construct a state s. This sampled state s is then employed in the simulation, generating a new tree trajectory. An action a^* is chosen during the simulation based on the Q value, visit counts, and LLM policy (lines 28 and 29). The observation and transition function, denoted as $\mathcal{G}$ (lines 15 and 30), predict the next state s' given the selected action a^* and the sampled state s, thus progressing to the subsequent step in the simulation (lines 30 and 31). When encountering leaf nodes in the tree, MCTS expands the tree and performs a random rollout for the corresponding node (lines 23 to 26). A uniform policy is employed to sample actions in the rollout, and the discounted reward is then returned (lines 14 to 17). Upon completing the task or reaching the maximum depth, the accumulated rewards are backpropagated, updating each node's estimated Q value (lines 32 to 35). Following N simulations, the output action is determined based on the estimated Q value (lines 3 to 8). Upon completion of the search process, the agent will execute an action and receive a new observation. For simplicity, we assume that the observation and transition functions are deterministic and known. In cases where an object is detected, its corresponding position within the belief will be updated with the observed position. Conversely, if the object remains undetected at certain positions, the belief regarding its presence in those positions will be rendered null, denoted by a zero value.

IV. Experiments

A. Experimental settings

We proceed with our experiments in the VirtualHome [28], a large household simulated environment with partial observation, large action space, and long planning horizon. The house contains hundreds of interactive items and containers with various types of rooms. It is a well-suited platform for evaluating embodied decision-making for solving daily tasks in household environments.

The tasks we use to evaluate our method are object rearrangement tasks, such as table setup, food preparation, snack preparation, and bathroom cabinet organization. To generate training data, we follow the method described in [29], creating 2000 tasks with randomly initialized scenes and expert action trajectories. There are several settings for the evaluation.

Simple tasks are the tasks that only require the rearrangement of one item generated from the same distribution as the training dataset. *Comp.* refers to the composition of simple tasks in order to rearrange multiple objects sampled from the same distribution as the dataset. The composition of tasks increases the planning horizon, making it more challenging to complete. In evaluation, we also use the *Novel Simple* tasks with seen items. For compositional tasks, we include *Novel Comp*ositional tasks, with 2 or 3 primary tasks composed. We also generate scenes at a *Novel Apartment* for testing, where the distribution of object positions differs from the training dataset.

We evaluate the success rate of completing the tasks within 30 steps, while a typical task can be finished within at most 15 steps. The task is considered successful if all the requirements of object positions are satisfied. For example, given the instruction "Put one apple inside the fridge," the task is successful if any apple is in the fridge. For simplicity, we don't consider the task of rearranging a very specific object, e.g., putting the leftmost apple in the fridge.

We evaluate the following baselines as comparisons. *UCT* [21]: We use the UCT algorithm to conduct planning without commonsense knowledge and use the ground-truth reward function in simulation. We use uniform distribution as the initial belief for states of objects. For the *finetuned GPT2 policy* [23], we use the collected training dataset with 10000 trajectories to fine-tune a language model (GPT-2) as the planning policy. *GPT3.5 Policy*: This is an improved version of [18]. When querying for actions, it not only takes as input the instructions and history of actions but also the currently visible objects. We use the LLM as the policy only, with a few examples as prompts to interact with the environments.

In Appendix C we show the result of ablation studies within the GPT3.5-MCTS framework, as well as the limitation and failure case analysis.

B. Results

The main results of the experiments are shown in Table I, reporting the success rate of our method and baselines in completing the tasks in VirtualHome environments. In this result, GPT3.5-MCTS outperforms all the compared baselines, especially for unseen situations. UCT works poorly in all conditions, as the poor model and the huge search tree make the planning intractable. Thus, we focus our discussion on comparing the finetuned GPT2 policy and GPT3.5 policy.

For *Simple*, in-distribution tasks, the planning horizon is relatively short. Finetuned GPT2 policy, GPT3.5 Policy, and our method work reasonably well, but our method still outperforms the baselines. For *Novel Simple* tasks, finetuned GPT2 policy works significantly worse than GPT3.5 Policy and GPT3.5-MCTS. This is because the fine-tuning of narrow tasks results in a biased distribution of the policy and compromises generalizability. GPT3.5 Policy and GPT3.5-MCTS work better due to the LLM's few-shot planning capability. GPT3.5-MCTS works better for both situations. It benefits from the MCTS' look-ahead search that explore commonsense states of the world for potential outcomes in order to make reasoned decisions. It utilizes the additional commonsense knowledge of the world encoded in LLM, compared to using LLM solely as policy.

TABLE I

SUCCESS RATE (%): MEAN ± STANDARD ERROR

Method	Seen Apartment: Simple	Comp.	NovelSimple	NovelComp.(2)	NovelComp.(3)
UCT	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
finetuned GPT2 policy	81.3±2.4	59.0±6.7	41.2±7.1	30.9±2.8	2.3±1.5
GPT3.5 Policy	83.4±6.8	47.0±7.8	74.3±4.0	48.2±8.8	5.4±2.0
GPT3.5-MCTS (Ours)	91.4±3.3	71.2±6.2	88.1±4.3	72.6±6.9	33.6±3.1
Method	**Unseen Apartment: Simple**	**Comp.**	**NovelSimple**	**NovelComp.(2)**	**NovelComp.(3)**
UCT	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
finetuned GPT2 policy	65.5±3.4	39.9±5.2	33.4±6.4	12.8±3.9	1.1±0.9
GPT3.5 Policy	74.3±5.0	43.3±4.0	67.8±4.9	54.0±3.0	6.9±2.1
GPT3.5-MCTS (Ours)	82.9±3.2	71.9±5.6	79.3±3.3	70.4±6.4	38.8±3.4

For the *Comp*ositional, in-distribution tasks, the finetuned GPT2 policy and GPT3.5 policy get significantly worse performance, while GPT3.5-MCTS works far better. The finetuned GPT2 policy is trained by behavior cloning that suffers from compounding errors. Therefore, when the planning horizon gets longer, the influence of the errors accumulates and compromises the overall performance significantly. As for GPT3.5 Policy, the longer horizon potentially introduces more possible errors during planning, which might not be included in the prompt examples. Without suitable guidance from prompt, we cannot guarantee the GPT3.5 Policy will carry out suitable replanning when encountering errors or mistakes. MCTS encourages exploration to a certain extent of different possible actions during searching, introducing additional guidance to the GPT3.5 policy to look into other possible solutions. This is because the action selection procedure in GPT3.5-MCTS is not purely determined by GPT3.5 Policy but also by the Q value and visit counts. Therefore, MCTS encourages GPT3.5 Policy to explore other possible search directions instead of excessively applying certain actions sampled by itself.

V. CONCLUSION

We use Large Language Models as the commonsense world model and the heuristic policy within the Monte Carlo Tree Search framework, enabling better-reasoned decision-making for daily tasks. MCTS enables LLM to leverage its world modeling knowledge for informed reasoning and explore new combinations of actions to tackle novel tasks. LLM helps MCTS through the biased sampling of states and actions, improving its efficiency in resolving complex task-planning problems. The runtime of our method is currently hindered by computational constraints, resulting in sluggish performance. Nonetheless, we are optimistic about the potential of our approach. Future advancements in edge computing devices may allow our method to be applied in robotic systems to enhance decision-making capabilities. Our analysis and empirical evidence suggest that, for some domains, the knowledge possessed by LLM in world modeling exhibits a higher degree of comprehensiveness than policies, owing to the significant discrepancy in their respective description complexities. Thus, leveraging LLM's world knowledge holds considerable promise in decision-making disciplines and beyond.

Acknowledgments

This research is supported in part by the National Research Foundation (NRF), Singapore and DSO National Laboratories under the AI Singapore Program (AISG Award No. AISG2-RP-2020-016) and the Agency of Science, Technology and Research, Singapore, under the National Robotics Program (Grant No. 192 25 00054).

References

[1] Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins SRI, Anthony Barrett, Dave Christianson, et al. Pddl| the planning domain definition language. *Technical Report, Tech. Rep.*, 1998. 7

[2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 1, 7

[3] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018. 7

[4] Dhruv Batra, Angel X Chang, Sonia Chernova, Andrew J Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, et al. Rearrangement: A challenge for embodied ai. *arXiv preprint arXiv:2011.01975*, 2020. 7

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020. 1

[6] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023. 1

[7] Panpan Cai, Yuanfu Luo, Aseem Saxena, David Hsu, and Wee Sun Lee. Lets-drive: Driving in a crowd by learning from tree search. *arXiv preprint arXiv:1905.12197*, 2019. 7

[8] Zhenfang Chen, Peng Wang, Lin Ma, Kwan-Yee K Wong, and Qi Wu. Cops-ref: A new dataset and task on compositional referring expression comprehension. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10086–10095, 2020. 7

[9] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Computers and Games: 5th International Conference, CG 2006, Turin, Italy, May 29-31, 2006. Revised Papers 5*, pages 72–83. Springer, 2007. 1

[10] Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. *arXiv preprint arXiv:2302.06692*, 2023. 7

[11] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 440–448, 2020. 7

[12] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021. 7

[13] Jiayuan Gu, Devendra Singh Chaplot, Hao Su, and Jitendra Malik. Multi-skill mobile manipulation for object rearrangement. *arXiv preprint arXiv:2209.02778*, 2022. 7

[14] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006. 7

[15] Jörg Hoffmann. Ff: The fast-forward planning system. *AI magazine*, 22(3):57–57, 2001. 7

[16] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 804–813, 2017. 7

[17] Ronghang Hu, Anna Rohrbach, Trevor Darrell, and Kate Saenko. Language-conditioned graph networks for relational reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10294–10303, 2019. 7

[18] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR, 2022. 1, 3, 4, 7

[19] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022. 1, 7

[20] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019. 7

[21] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006: 17th European Conference on Machine Learning Berlin,*

Germany, September 18-22, 2006 Proceedings 17, pages 282–293. Springer, 2006. 4

[22] Belinda Z Li, William Chen, Pratyusha Sharma, and Jacob Andreas. Lampp: Language models as probabilistic priors for perception and action. *arXiv e-prints*, pages arXiv–2302, 2023. 7

[23] Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, et al. Pre-trained language models for interactive decision-making. *Advances in Neural Information Processing Systems*, 35:31199–31212, 2022. 1, 2, 3, 4, 9, 10

[24] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *arXiv preprint arXiv:2005.07648*, 2020. 7

[25] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *arXiv preprint arXiv:2210.06407*, 2022. 7

[26] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023. 1

[27] Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023. 7

[28] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018. 3, 9

[29] Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Yuan-Hong Liao, Joshua B Tenenbaum, Sanja Fidler, and Antonio Torralba. Watch-and-help: A challenge for social perception and human-ai collaboration. *arXiv preprint arXiv:2010.09890*, 2020. 2, 3, 10

[30] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019. 2, 3

[31] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017. 7

[32] Raeid Saqur and Karthik Narasimhan. Multimodal graph networks for compositional generalization in visual question answering. *Advances in Neural Information Processing Systems*, 33:3070–3081, 2020. 7

[33] Mohit Shridhar and David Hsu. Interactive visual grounding of referring expressions for human-robot interaction. *arXiv preprint arXiv:1806.03831*, 2018. 7

[34] Mohit Shridhar, Dixant Mittal, and David Hsu. Ingress: Interactive visual grounding of referring expressions. *The International Journal of Robotics Research*, 39(2-3):217–232, 2020. 7

[35] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023. 7

[36] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017. 7

[37] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419): 1140–1144, 2018. 7

[38] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems*, 34:251–266, 2021. 7

[39] Stefanie Tellex, Nakul Gopalan, Hadas Kress-Gazit, and Cynthia Matuszek. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3 (1), 2020. 7

[40] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6629–6638, 2019. 7

[41] Yaqi Xie, Chen Yu, Tongyao Zhu, Jinbin Bai, Ze Gong, and Harold Soh. Translating natural language to planning goals with large-language models. *arXiv preprint arXiv:2302.05128*, 2023. 2

[42] Danfei Xu, Roberto Martín-Martín, De-An Huang, Yuke Zhu, Silvio Savarese, and Li F Fei-Fei. Regression planning networks. *Advances in Neural Information Processing Systems*, 32, 2019. 7

[43] Xiaofeng Yang, Guosheng Lin, Fengmao Lv, and Fayao Liu. Trrnet: Tiered relation reasoning for compositional visual question answering. In *European Conference on Computer Vision*, pages 414–430. Springer, 2020. 7

[44] Bowen Zhang and Harold Soh. Large language models as zero-shot human models for human-robot interaction. *arXiv preprint arXiv:2303.03548*, 2023. 7

[45] Zirui Zhao, Wee Sun Lee, and David Hsu. Differentiable parsing and visual grounding of verbal instructions for object placement. *arXiv preprint arXiv:2210.00215*, 2022. 7

APPENDIX

A. Related work

Grounding robot behaviors from natural language instructions is critical to human-robot interactions. It is useful in a broad range of tasks, such as language-conditioned manipulation [34, 33, 35, 45, 24, 25, 39], visual language navigation [3, 40], and visual question answering [31, 16, 17, 43, 32, 20, 8]. However, those tasks lack a large, complex state and action space or a long planning horizon. Our research mainly focuses on object rearrangement tasks [4] in household environments to demonstrate the method. Object rearrangement is a typical and comprehensive instruction-following problem with large-scale, long-horizon task planning. The comprehensiveness comes from its coverage of many instruction-following subtasks, such as object searching, navigation, and object pick-and-place. It also requires long-horizon planning to determine how to pick an object, navigate, and place it somewhere. Some prior works [2, 13, 38] focus on grounding low-level robot actions in object rearrangement tasks, therefore using a limited number of objects or a small room environment for simplicity. The resulting planning tasks are simpler. We select the large-scale house environment for experiments, where the scale of task planning is significantly more complex.

The execution of natural language instructions entails long-horizon task planning due to the inherent abstraction of the language. For task planning, early attempts [1, 15, 14, 11] used symbolic representation and search algorithms to efficiently carry out small-scale or short-horizon problems; these methods do not work well for large-scale, long-horizon problems. Recently, researchers [36, 37, 42, 7] have used deep learning and reinforcement learning methods to learn search heuristics to accelerate planning. Those learned policies or heuristics are not generalizable to other unseen settings. Most recently, the pre-trained LLMs have been exhibiting impressive ability for format following and content generation, allowing them to be applied as policies for task planning [18, 2, 19]. However, the planning policy may suffer from hallucination issues of LLMs. Our method uses the LLM's knowledge of the world together with a search algorithm for better-reasoned decision-making. To operate in the real world, task planning should be integrated with physical-level motion planning, i.e., task and motion planning (TAMP) [12, 11]. Limited by the scope, we consider the influence of physical-level motion planning as future work.

Pre-trained large language models (LLMs) have recently been dominating AI research due to their vast encoded knowledge. That knowledge has been leveraged in many tasks successfully. It has been used as a few-shot policy for language-conditioned task planning [18, 2, 19]. LLMs can also be viewed as probabilistic priors for non-linguistic perception and decision-making tasks [22]. In reinforcement learning, LLMs are applied as heuristics to provide background knowledge for providing exploration bias [10]. Studies also leverage LLMs as a human model to simulate human behaviors [27] or facilitate better human-robot interactions [44]. However, the literature has not discussed utilizing LLMs' commonsense knowledge of modeling the world in depth. In this paper, we show that LLMs' commonsense knowledge of the world can be exploited in a search algorithm such as Monte Carlo Tree Search to facilitate more reasoned planning for daily tasks.

B. Pseudocode of LLM-MCTS

The pseudocode of LLM-MCTS is shown in Alg 1. For each simulation, MCTS samples a state from the belief $b(s)$ at the root (line 4). It independently samples one position for each object to construct a state s. This sampled state s is then employed in the simulation, generating a new tree trajectory. An action a^* is chosen during the simulation based on the Q value, visit counts, and LLM policy (lines 28 and 29). The observation and transition function, denoted as $\mathcal{G}$ (lines 15 and 30), predict the next state s' given the selected action a^* and the sampled state s, thus progressing to the subsequent step in the simulation (lines 30 and 31). When encountering leaf nodes in the tree, MCTS expands the tree and performs a random rollout for the corresponding node (lines 23 to 26). A uniform policy is employed to sample actions in the rollout, and the discounted reward is then returned (lines 14 to 17). Upon completing the task or reaching the maximum depth, the accumulated rewards are backpropagated, updating each node's estimated Q value (lines 32 to 35). Following N simulations, the output action is determined based on the estimated Q value (lines 3 to 8).

C. Additional results and discussion

1) Ablation study: We conduct ablation studies to see the individual contributions of different components within the GPT3.5-MCTS framework. The *No Heuristic Policy* version of GPT3.5-MCTS refers to the absence of PUCT guided by the GPT3.5 Policy for action selection. Instead, it solely relies on UCT with an initial commonsense belief derived from LLM. The variant employing the *Uniform State Prior* utilizes a uniform prior belief regarding states, in contrast to the LLM-generated initial belief employed during the search process. Lastly, the variant operating in a *Fully Observable* environment aims to assess the accuracy of LLM's knowledge in modeling the world.

Table II presents the results of our ablation experiments. The outcomes obtained under the *No Heuristic Policy* version highlight the significance of heuristic policies in facilitating MCTS to conduct efficient searches for complex and large-scale planning tasks. Conversely, the results of the *Uniform State Prior* row indicate that imperfect world models compromise search performance. This is because the model of the world determines the Q value. The wrong model results in an inaccurate estimation of the Q value, misleading the search process toward irrelevant locations. The *Fully Observable* results demonstrate that GPT3.5-MCTS with perfect knowledge of the environment only slightly outperforms its counterpart without it, implying that the commonsense knowledge of LLM regarding world modeling suffices for practical purposes.

Algorithm 1 LLM-MCTS

procedure SEARCH($h, b, \mathcal{T}, N$)
  $n \leftarrow 0$
  while $n < N$ **do**
    $s \sim b(s)$
    SIMULATE($s, h, \text{False}, 0, \mathcal{T}$)
    $n \leftarrow n + 1$
  end while
  return $\text{argmax}_{a \in A} Q(h, a)$
end procedure
procedure ROLLOUT(s, h, done, d)
  if $\gamma^d < \epsilon$ or $\text{done} = \text{True}$ **then**
    return 0
  end if
  $a \sim \pi_{\text{rollout}}(h, \cdot)$
  $(s', o, r, \text{done}) \sim \mathcal{G}(s, a)$
  $h' \leftarrow$ PUSHBACK($h, [a^*, o]$), $d' \leftarrow d + 1$
  return $r + \gamma \cdot$ROLLOUT(s, h', done, d')
end procedure
procedure SIMULATE($s, h, \text{done}, d, \mathcal{T}$)
  if $\gamma^d < \epsilon$ or $\text{done} = \text{True}$ **then**
    return 0
  end if
  if h is not in $\mathcal{T}$ **then**
    $\mathcal{T} \leftarrow \mathcal{T} \cup h, N(h) \leftarrow 0$
    $\forall a \in A, N(h, a) \leftarrow 0, Q(h, a) \leftarrow 0$
    return ROLLOUT(s, h, done, d)
  end if
  $\hat{\pi}(a|h) \leftarrow$QUERYLLMPOLICY($h$)
  $a^* \leftarrow \underset{a \in A}{\text{argmax}}\, Q(h, a) + c\hat{\pi}(a|h) \frac{\sqrt{N(h)}}{N(h,a)+1}$
  $(s', o, r, \text{done}) \sim \mathcal{G}(s, a^*)$
  $h' \leftarrow$ PUSHBACK($h, [a^*, o]$), $d' \leftarrow d + 1$
  $R \leftarrow r + \gamma \cdot$SIMULATE($s', h', \text{done}, d', \mathcal{T}$)
  $N(h, a^*)$ += 1, $N(h)$ += 1
  $Q(h, a^*) \leftarrow Q(h, a^*) + \frac{R - Q(h, a^*)}{N(h, a^*)}$
  return R
end procedure

TABLE II
RESULTS ON ABLATION STUDY: MEAN $\pm$ STANDARD ERROR OF SUCCESS RATE (%)

Method	Seen Apartment				Unseen Apartment			
	Simple	Comp.	NovelSimple	NovelComp.(2)	Simple	Comp.	NovelSimple	NovelComp.(2)
GPT3.5-MCTS (No Heuristic Policy)	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
GPT3.5-MCTS (Uniform State Prior)	3.2±1.1	0.0±0.0	1.1±0.4	0.0±0.0	1.1±0.2	0.0±0.0	0.0±0.0	0.0±0.0
GPT3.5-MCTS (Fully Observable)	94.0±2.1	80.7±3.3	94.3±2.4	78.5±4.0	85.1±5.0	77.5±3.2	82.2±3.3	76.6±3.1
GPT3.5-MCTS (Ours)	91.4±3.3	71.2±6.2	88.1±4.3	72.6±6.9	82.9±3.2	71.9±5.6	79.3±3.3	70.4±6.4

2) Failure analysis: 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 cutleryfork*" where the "*cutleryfork*" is not in the observation. 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. 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).

3) Discussion: We analyze the complexity of the approaches in terms of the description length of the world model and the policy. Assume a house has n moveable objects, m containers, and k rooms. We further assume that each of the n objects and m containers can be positioned inside, at most, a constant number of the $m+k$ containers or rooms and that the positions of each object and container are independent. Each nonzero probability location of the $m+k$ containers or rooms requires $\log(m+k)$ bits to specify. We further assume that we use a constant number of bits to represent each nonzero probability coefficient. Overall the prior distribution of objects in the home requires $O((m+n)\log(m+k))$ bits to describe. For the policy, we can use a pair of objects and containers to specify a task, and a solution to a task involves a path where each edge specifies an action with a target object, container, or room (e.g., "grab the apple, walk to the kitchen, walk to the fridge, open the fridge, put the apple inside the fridge"). To describe all these paths for the mn pairs would require an order of $mn\log(m+n+k)$ bits assuming all the policies have a short bounded number of steps. These are open-loop policies but can be modified into closed-loop ones by stopping the search for the object once it has been found and starting to search for the target location. The analysis suggests that learning the model could be easier than learning the policies for this domain. However, the model-based approach also requires a goal test, which we assume to be similar to recognizing which policy to take. Furthermore, clever decomposition and sharing among the policies can reduce their complexity, e.g., policies can be decomposed into searching for an object and a target location, reducing its description complexity. Whether the LLM successfully learned these shared policies is less clear.

Composing tasks increases the description complexity of policies to $O((mn)^N \log(m+n+k))$, where N is the number of composed tasks if done naively. The goal recognition component of the model-based approach similarly becomes more complex. The composed tasks can also be decomposed into individual tasks done sequentially. How much the LLMs can decompose the problems is unclear, although our experiments on this problem show better results for the model-based approach. Decomposition can help reduce the descriptive complexity, and we expect it would accordingly reduce the sample complexity of learning. Decomposition is automatically done in the tree search at the expense of more computation, although we still depend on the LLM to handle the goal test. For the policy, the LLM must learn to do the decomposition, which may make the learning problem computationally more difficult.

D. Experimental environments

We use the VirtualHome simulator [28] 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.

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.

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.

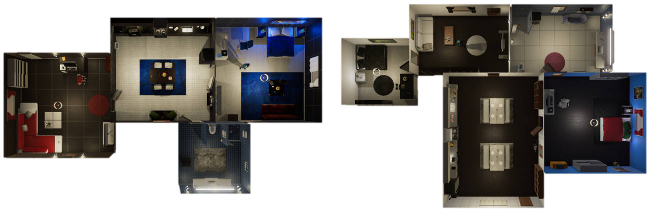
The tasks are randomly sampled from different distributions. We define various types of object rearrangement tasks for evaluation:

- *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:

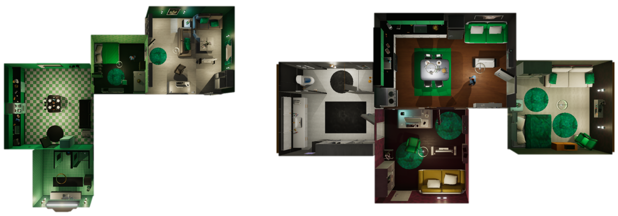
- *Seen Apartment*: the map of the apartment is shown in Figure 3. These household environments are the same as the ones in the training set, while the object positions are randomly initialized according to a pre-defined commonsense distribution in VirtualHome [28].
- *Unseen Apartment*: the map of the apartment is shown in Figure 4. 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 [28].

3) Goal specification: Similar to prior works [23], 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) (b)

Fig. 3. 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) (b)

Fig. 4. 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.

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 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.

5) Observations: We use the same representation as [23] 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 [23] 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."

E. Data gathering

Similar to prior works [29, 23], we collect expert trajectories in VirtualHome using regression planning with handcrafted heuristics[2]. 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.

F. Implementation details of belief in LLM-MCTS

This section introduces our implementation details for the belief of states in GPT3.5-MCTS. We use GPT-3.5 as the LLM. The temperature parameter is set to 0.6, and the top-p parameter is 0.9. The sample number for querying the LLM is $M = 10$, and the number of prompts is $K = 1$. The mixture parameter for heuristic policy is $\lambda = 0.5$. We select 200 instances uniformly at random from the dataset as prompt candidates for the LLM model and policy. The source code will be released at https://llm-mcts.github.io before the publication.

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

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.

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.

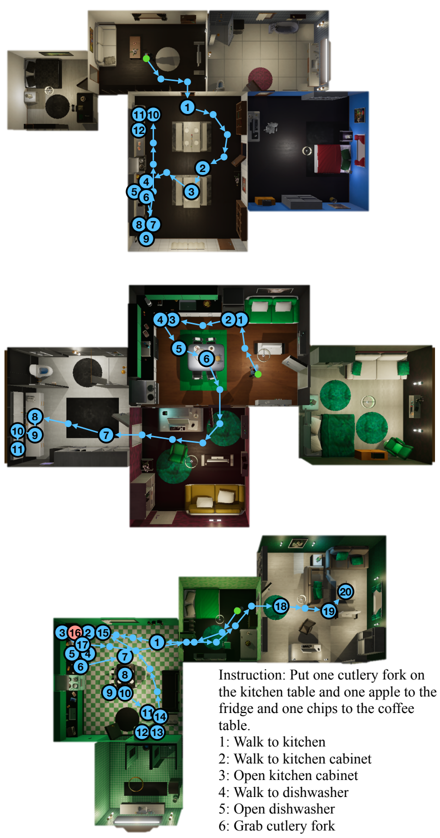


Fig. 5. Successful examples

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

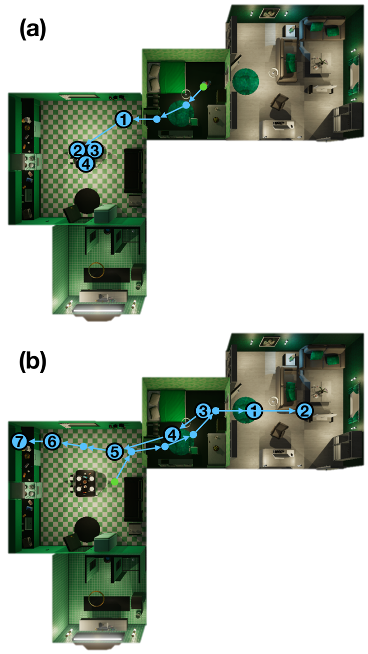


Fig. 6. Failed examples. (a) Policy error and translation error. LLM outputs walk to the cutlery fork, but the cutlery fork is not in observation. We use embeddings to evaluate the most similar valid actions. Therefore it translates the action to one similar action "walk to cutlery knife." The action 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.

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.

G. Visualized examples

We provide a set of successful (shown in Figure 5) and failed trajectories (shown in Figure 6) 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 6 (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 6 (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).

H. 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.

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:
```