

Revisiting ALFRED: Refining commands for evaluating language-guided task planning

Minsu Jang

¹Electronics and Telecommunications Research Institute
218 Gajeong-ro, Yuseong-gu
Daejeon-si, 34129, South Korea
minsu@etri.re.kr

Abstract

Language-guided task planning has seen significant advances with the emergence of large language models (LLMs). This paper presents a refinement of the ALFRED benchmark, a widely-used dataset for evaluating embodied AI in household environments. We identify and categorize errors in the original crowd-sourced commands and propose refined commands that maintain alignment with expert demonstrations. We further augment these commands through paraphrasing, contextualization, and cross-lingual translation to create a more comprehensive evaluation suite. Our experimental results across multiple LLM-based planners demonstrate that while state-of-the-art models achieve high success rates on base commands, performance degrades with increased linguistic complexity. The refined benchmark provides a more reliable foundation for evaluating language-guided planning systems while introducing controlled challenges through linguistic variations. Refined command sets are available at https://github.com/zebehn/revisiting_alfred.

1 Introduction

Recent trends in large language model-based task planning have demonstrated significant improvements in the ability of LLMs to understand and execute complex multi-step tasks. Robotics has greatly benefited from these advancements, particularly in task automation, where vast common-sense and procedural knowledge embedded in LLMs enable robotic agents to generate task plans across a variety of domains based on informal natural language commands. This has also facilitated intuitive language-based interactions between robots and users, as demonstrated in works such as Zero-Shot Planner (Huang et al. 2022), Google Say-Can (Ahn et al. 2022), and Google PaLM-E (Driess et al. 2023). These advancements are driven by increased model scale, improved training data quality, and enhanced integration with perceptual modules, enabling agents to navigate, manipulate objects, and adapt to dynamic environments. Additionally, the shift towards incorporating contextual reasoning and uncertainty management has made these models more robust in handling real-world conditions and ambiguous instructions.

Simulations for embodied artificial intelligence, such as AI2Thor (Kolve et al. 2017), VirtualHome (Puig et al. 2018), iGibson (Li et al. 2021), and IsaacSim (NVIDIA 2023), are increasingly employed for conducting experiments and benchmarking the performance of LLM-based task planners. These simulation environments provide photo-realistic and interactive settings, enabling researchers to evaluate the effectiveness of language-guided task planning in a controlled and repeatable manner. This systematic evaluation capability has been a driving force in advancing LLM-based task planners by allowing rapid iteration and development.

Several domain-specific benchmarks built on top of the aforementioned embodied AI simulators serve as critical tools for evaluating and advancing language model-based task planners across different task domains. These benchmarks are tailored to assess planners within specific types of environments, such as household tasks (Shridhar et al. 2020; Puig et al. 2021; Nasiriany et al. 2024) and tabletop manipulations (Zeng et al. 2021; Jiang et al. 2022), allowing systematic evaluation and fostering improvements in both generalization and task-specific skills.

In this paper, we examine the ALFRED benchmark, a widely-adopted simulation framework for evaluating embodied task planning in domestic environments. While ALFRED has become a standard testbed for language-guided planning systems, particularly those leveraging LLMs, its reliance on crowd-sourced commands introduces significant challenges. Our analysis reveals that these natural language commands often contain errors and ambiguities that compromise the benchmark’s effectiveness for evaluation. To address these limitations, we conduct a systematic review of the command dataset and propose new command sets that more accurately align with the expert-generated task demonstrations. These refined commands maintain the natural language characteristics desired for real-world applications while ensuring precise correspondence with the intended actions, thereby enabling more reliable evaluation of language-guided planning systems.

2 Background

2.1 Language Model-based Task Planning

Large Language Models (LLMs) have emerged as a transformative technology for task planning, offering three key ad-

vantages: natural language understanding that enables conversational task specification, zero-shot generalization for handling novel scenarios without explicit training, and pre-trained world knowledge for common-sense reasoning about objects and actions. These capabilities significantly enhance the flexibility and accessibility of planning systems compared to traditional approaches.

The process of LLM-based task planning involves conditioning a language model with contextual descriptions to generate action sequences that accomplish goals specified in natural language. This approach has been implemented through various methodologies: one-shot plan generation that produces complete plans in a single inference step (Huang et al. 2022), iterative action selection that determines actions sequentially (Ahn et al. 2022; Hao et al. 2023), and deliberative tree search that explores multiple action trajectories (Ao et al. 2024; Hu et al.).

2.2 ALFRED Benchmark

ALFRED (Action Learning From Realistic Environments and Directives) is a benchmark for embodied AI that evaluates agents’ ability to execute household tasks based on natural language instructions (Shridhar et al. 2020). Built on the AI2Thor simulator, ALFRED presents agents with complex scenarios requiring object manipulation, spatial navigation, and instruction understanding. The benchmark encompasses six fundamental task categories: object relocation, stacking, heating, cooling, cleaning, and examination, all of which require interpreting natural language directives and executing appropriate actions in a 3D environment.

The benchmark’s evaluation dataset comprises 8,055 expert demonstrations derived from 2,685 unique task definitions. These definitions are systematically generated by combining 7 task types, 58 object classes, and 26 receptacle classes across 120 distinct scenes, creating a comprehensive test suite for embodied task learning and execution.

2.3 LoTa-Bench

LoTa-Bench is an automated evaluation framework designed specifically for language model-based task planners (Choi et al. 2024). The framework provides two key components: a standardized implementation interface for language model-based planners and a domain abstraction layer that enables world state querying, action execution, and goal verification through natural language. This architecture facilitates systematic evaluation of planning capabilities across multiple domains, currently supporting both ALFRED and Watch-And-Help (Puig et al. 2021) environments. The benchmark’s modular design allows for consistent assessment of different planning approaches while abstracting domain-specific implementation details.

3 Method

Our systematic approach to analyzing and refining the ALFRED dataset consists of four primary phases:

1. **Error Analysis:** Comprehensive examination of annotation errors in the original ALFRED dataset to identify patterns and categories of inconsistencies

2. **Command Generation:** Development of refined task commands to create more precise and consistent natural language instructions that accurately reflect expert demonstrations
3. **Command Augmentation:** Generation of four variant command sets to evaluate robustness across different linguistic expressions: base refined commands that strictly align with demonstrations, paraphrased variations that maintain semantic equivalence, contextualized commands incorporating additional environmental details, and Korean-language commands enabling cross-lingual evaluation
4. **Planner Evaluation:** Comparative assessment of language-guided planners using both original and augmented command datasets to measure performance across different linguistic variations

For the analysis, generation, and augmentation phases (steps 1-3), we employ OpenAI’s `o1-preview` model (OpenAI 2024), leveraging its advanced reasoning capabilities and robust multilingual understanding. The prompts for these phases are presented in Appendix A.1 through A.4. The generated and augmented commands were examined through human inspection, and errors were corrected. The evaluation phase uses multiple state-of-the-art language models to ensure a comprehensive assessment of planning performance across the refined command datasets.

4 Results

4.1 Error Analysis

The ALFRED dataset’s natural language commands are derived from crowd-sourced annotations, where workers provided instructions based on demonstrated tasks. Each demonstration, generated using a PDDL-compatible task planner, was presented to crowd-sourced workers who were tasked with inferring the demonstrated actions’ intent and goal, then formulating appropriate commands to guide an agent in replicating these actions.

Our analysis revealed systematic errors in the crowd-sourced annotations, stemming from variations in workers’ reasoning capabilities and diverse interpretations of task intentions. We identify three primary categories of errors that significantly impact the dataset’s reliability:

- **Object Identification Errors:** These errors manifest when annotators misidentify or incorrectly specify objects in their commands. For instance, a demonstration showing an agent placing an egg in a microwave might be annotated as ”Place a potato in the microwave.” Such errors fundamentally alter the task’s objective and can lead to incorrect training signals for learning systems.
- **Spatial Reference Errors:** These errors involve inaccurate or ambiguous specifications of locations and receptacles. A representative example is when a demonstration showing toilet paper being placed in a toilet receives the annotation ”Put two toilet papers on the toilet tank lid.” These errors can result in spatial navigation failures and incorrect object placement during task execution.

- **Action Sequence Misalignment:** This category encompasses discrepancies between the demonstrated action sequence and the annotated command. For example, a demonstration of placing a sliced tomato directly into a refrigerator might receive the annotation "Put a bowl with a tomato in it in the fridge." Such errors introduce phantom steps or omit crucial actions, potentially leading to task execution failures.

Table 1 shows the statistics of the errors in the crowd-sourced commands in the ALFRED dataset for the valid-seen and valid-unseen tasks. These annotation errors present significant challenges for developing and evaluating language-guided planning systems, as they introduce noise into both training and evaluation processes.

Error Type	Valid-Seen	Valid-Unseen
No of Commands	820	821
Object Id. Errors	176	112
Space Ref. Errors	160	146
Action Seq. Misalignment	128	136

Table 1: Statistics of errors in the ALFRED dataset commands for valid-seen and valid-unseen tasks.

4.2 Command Generation

The ALFRED benchmark represents each expert demonstration through a JSON document containing comprehensive task specifications: scene descriptions with object poses and coordinates, task type identifiers, goal state definitions, and both high-level action sequences and low-level directives, along with multiple natural language command annotations. To generate refined commands, we extracted two key components from each demonstration: the task type and the high-level action sequence. Using OpenAI’s `o1-preview` model, we provide a prompt structure combining:

- Descriptions on the command generation task
- Explanations of ALFRED task types
- Five in-context examples covering primary task categories (*Pick & Place*, *Pick Two & Place*, *Stack & Place*, *Heat & Place*)

An illustrative in-context example from our prompt:

- **Task Type:** *pick_and_place_simple*
- **Action Sequence:**
 1. GotoLocation(countertop)
 2. PickupObject(peppershaker)
 3. GotoLocation(drawer)
 4. PutObject(peppershaker,drawer)
 5. Done
- **Command:** *Put the pepper shaker to the drawer*

Using this approach, we generated refined commands for both validation splits: 251 commands for *valid_seen* tasks and 255 commands for *valid_unseen* tasks, ensuring one precise command per expert demonstration.

	Set	Unique Words	Diversity	Entropy
valid-seen	B	129	0.060	5.172
	P-1	165	0.080	5.689
	P-2	171	0.083	5.734
	C	521	0.065	6.974
	K	144	0.123	6.137
valid-unseen	B	97	0.045	5.024
	P-1	150	0.071	5.605
	P-2	154	0.073	5.692
	C	530	0.065	7.136
	K	122	0.102	6.272

Table 2: Linguistic complexity metrics across command variants (B: Base, P-1: Paraphrased-1, P-2: Paraphrased-2, C: Contextual, and Korean translations). The metrics include vocabulary size (unique words), lexical diversity, and entropy.

4.3 Command Augmentation

To evaluate the robustness of language model-based planners across diverse linguistic expressions, we developed four variant sets of task commands. These variants test planners’ ability to handle semantic variations, contextual reasoning, and cross-lingual understanding.

The first variant, Paraphrased-1, introduces verb variations while maintaining other linguistic elements constant, testing robustness to action verb synonyms. The second variant, Paraphrased-2, incorporates alternative terms for both objects and locations, evaluating comprehension of broader semantic variations. The Contextual variant presents a two-sentence structure where contextual background precedes the actual command, assessing planners’ ability to integrate contextual information with direct instructions, identify and resolve uncertainties through question-answering when context is absent, and reference implicit information in command execution. Finally, the Korean variant provides cross-lingual evaluation through direct translation of base commands, capturing variations in linguistic nuances and semantic interpretations across languages.

Using OpenAI `o1-preview` model, we generated four variant sets for each base command, resulting in 2,024 commands in total (506 commands \times 4 variants) across the *valid_seen* and *valid_unseen* splits. Sample commands for each variant set are illustrated below:

- **Base:** Put the cloth in the bathtub basin.
- **Paraphrased-1:** Set the cloth in the bathtub basin.
- **Paraphrased-2:** Place the cloth in the bathtub.
- **Contextual:** In our household, we usually place cloths in the bathtub basin for soaking or washing. Once you have the cloth ready, please ensure it’s placed in the usual spot.
- **Korean:** 천을 욕조에 넣으세요.

We analyzed the linguistic complexity of the command sets using three quantitative metrics: vocabulary size (unique word count), lexical diversity, and entropy. As shown in Table 2, both paraphrased and contextualized variants demon-

Command Set	gpt-4o-mini	gpt-4o	o1-preview	Llama 3.1 70B
Base	58.7	75.2	69.7	61.4
Paraphrased-1	54.1	71.5	66.9	56.9
Paraphrased-2	50.4	69.7	70.6	54.1
Contextual	39.4	58.7	57.8	58.7
Contextual(<i>without context</i>)	11.9	19.2	20.1	18.6
Korean	48.6	59.6	69.7	51.4

Table 3: Plan execution success rates (%) across different LLMs on the refined ALFRED *valid-seen* tasks: comparison over command variants (Base, Paraphrased-1, Paraphrased-2, Contextual, and Korean)

strate increased vocabulary diversity and semantic complexity compared to the base commands. This increased linguistic variation introduces additional challenges for language understanding and task planning, making these command sets particularly valuable for evaluating LLMs’ robustness to natural language variations. These characteristics align with our goal of creating a more comprehensive and challenging benchmark for language-guided planning systems.

4.4 Planner Evaluations

We conducted comprehensive evaluations of various LLM-based planners using our refined ALFRED command sets for *valid-seen* tasks through the LoTa-Bench framework¹. The evaluation follows a one-shot planning approach, where each planner generates a complete action sequence from a single user command. The prompt structure includes 14 carefully selected in-context examples covering all ALFRED task types, ensuring comprehensive coverage of possible planning scenarios.

Our experimental setup focuses exclusively on language-based planning without visual inputs. Planners generate action sequences solely based on the command text and a pre-defined set of executable actions provided in the prompt. The environment interaction is simplified by using language identifiers for objects and locations, where the simulator teleports the agent to the target positions based on these identifiers. This setup enables the execution of embodied plans without requiring visual grounding, allowing us to isolate and evaluate the language understanding and planning capabilities of the models.

For each generated plan, LoTa-Bench executes the action sequence step-by-step in the AI2-Thor simulator. A plan is considered successful only when the final world state after execution precisely matches the specified goal conditions for the given command. The success rates for different models across our command variants are presented in Table 3.

OpenAI’s *o1-preview* and *gpt-4o* models achieved the highest success rates, while *Llama 3.1 70B* demonstrated competitive performance. Notably, success rates declined across all models when tested on the augmented command sets compared to the base set, confirming that

¹Due to computational constraints with the *Llama 3.1 70B* model, experiments were conducted on a selected 50% subset of the task commands while maintaining the original distribution across task types.

these variants introduce increased linguistic complexity to the planning task. The contextual command set presents a particularly interesting challenge: when the contextual background sentence is excluded, planners must demonstrate sophisticated uncertainty reasoning to successfully complete tasks, providing a valuable benchmark for evaluating LLMs’ ability to identify and handle ambiguous instructions.

Table 4 categorizes plan execution failures into three distinct error types. Command Misalignment (Type 1) occurs when there is a discrepancy between the commanded action and the target goal state. Execution Failure (Type 2) represents failures in navigation or manipulation within the environment. Planning Failure (Type 3) refers to the generation of action sequences that are insufficient or inappropriate for achieving the target goal.

Analysis of the error distribution reveals that our refined command set effectively eliminates command misalignment, validating that the generated commands maintain strict alignment with their corresponding demonstrated tasks. The remaining failures are attributed to environmental interaction challenges and planning complexities, suggesting that the performance bottleneck lies in the execution and planning capabilities rather than command interpretation.

Command Set	# Type 1	# Type 2	# Type 3
ALFRED(original)	27	49	24
Base	0	66	34
Paraphrased-1&2	0	65	35
Contextual	0	56	44
Korean	9	52	39

Table 4: Rates(%) of errors according to the types

5 Conclusions

In this work, we presented a systematic approach to refining and augmenting the ALFRED benchmark dataset for language-guided planning. Our analysis revealed significant inconsistencies in the original crowd-sourced commands, which we addressed through LLM-based command generation and augmentation. The refined command sets not only eliminate semantic misalignments present in the original dataset but also introduce controlled variations in linguistic complexity through paraphrasing, contextualization, and cross-lingual translation.

Our evaluation results demonstrate that while state-of-the-art LLMs achieve promising success rates on the base command set, their performance degrades with increased linguistic complexity in the augmented variants. This performance gap, particularly evident in contextual and cross-lingual scenarios, highlights important challenges in developing robust language-guided planning systems. The refined ALFRED benchmark, with its diverse command variants, provides a more reliable and comprehensive framework for evaluating future advances in language-guided planning systems.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. RS-2022- IIT20951, Development of Uncertainty-Aware Agents Learning by Asking Questions, 50%, No. RS-2024-00336738, Development of Complex Task Planning Technologies for Autonomous Agents, 50%).

References

Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Ao, J.; Wu, F.; Wu, Y.; Swikir, A.; and Haddadin, S. 2024. LLM as BT-Planner: Leveraging LLMs for Behavior Tree Generation in Robot Task Planning. *arXiv preprint arXiv:2409.10444*.

Choi, J.-W.; Yoon, Y.; Ong, H.; Kim, J.; and Jang, M. 2024. LoTa-Bench: Benchmarking Language-oriented Task Planners for Embodied Agents. In *The Twelfth International Conference on Learning Representations*.

Driess, D.; Xia, F.; Sajjadi, M. S.; Lynch, C.; Chowdhery, A.; Ichter, B.; Wahid, A.; Tompson, J.; Vuong, Q.; Yu, T.; et al. 2023. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*.

Hao, S.; Gu, Y.; Ma, H.; Hong, J.; Wang, Z.; Wang, D.; and Hu, Z. 2023. Reasoning with Language Model is Planning with World Model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 8154–8173.

Hu, M.; Mu, Y.; Yu, X. C.; Ding, M.; Wu, S.; Shao, W.; Chen, Q.; Wang, B.; Qiao, Y.; and Luo, P. 2023. Tree-Planner: Efficient Close-loop Task Planning with Large Language Models. In *The Twelfth International Conference on Learning Representations*.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, 9118–9147. PMLR.

Jiang, Y.; Gupta, A.; Zhang, Z.; Wang, G.; Dou, Y.; Chen, Y.; Fei-Fei, L.; Anandkumar, A.; Zhu, Y.; and Fan, L. 2022. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2(3): 6.

Kolve, E.; Mottaghi, R.; Han, W.; VanderBilt, E.; Weihs, L.; Herrasti, A.; Deitke, M.; Ehsani, K.; Gordon, D.; Zhu, Y.; et al. 2017. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*.

Li, C.; Xia, F.; Martín-Martín, R.; Lingelbach, M.; Srivastava, S.; Shen, B.; Vainio, K.; Gokmen, C.; Dharan, G.; Jain, T.; et al. 2021. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272*.

Nasiriany, S.; Maddukuri, A.; Zhang, L.; Parikh, A.; Lo, A.; Joshi, A.; Mandlekar, A.; and Zhu, Y. 2024. RoboCasa: Large-Scale Simulation of Everyday Tasks for Generalist Robots. *arXiv preprint arXiv:2406.02523*.

NVIDIA. 2023. NVIDIA Isaac Sim.

OpenAI. 2024. Introducing OpenAI o1-preview.

Puig, X.; Ra, K.; Boben, M.; Li, J.; Wang, T.; Fidler, S.; and Torralba, A. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8494–8502.

Puig, X.; Shu, T.; Li, S.; Wang, Z.; Liao, Y.-H.; Tenenbaum, J. B.; Fidler, S.; and Torralba, A. 2021. Watch-And-Help: A Challenge for Social Perception and Human-AI Collaborations. In *International Conference on Learning Representations*.

Shridhar, M.; Thomason, J.; Gordon, D.; Bisk, Y.; Han, W.; Mottaghi, R.; Zettlemoyer, L.; and Fox, D. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10740–10749.

Zeng, A.; Florence, P.; Tompson, J.; Welker, S.; Chien, J.; Attarian, M.; Armstrong, T.; Krasin, I.; Duong, D.; Sindhwani, V.; et al. 2021. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, 726–747. PMLR.

A Prompts

A.1 Error Analysis

You are a reasoning agent specialized in analyzing an intent of a user command and its alignment with the action sequence. The given action sequence is `**the ground truth of the task**`, so you need to `**find alignment problems only in the commands**`. Suppose you operate a robotic agent with the capability to perform the following seven classes of tasks:

1. Pick and Place, 2. Stack and Place, 3. Pick Two and Place, 4. Clean and Place, 5. Heat and Place, 6. Cool and Place, 7. Examine in Light.

Sometimes tasks involve slicing objects like apples. In this world, actions like HeatObject, CoolObject, SliceObject, CleanObject assume that you have already picked up the object after the actions. Sometimes the task involves putting something into a movable receptacle like a bowl and then placing the receptacle in a specific location. If heating, cooling, cleaning, or slicing action is involved in the process, it should be mentioned in the command .

In this world, the robot perceives the world and performs actions purely through a language and symbolic world model and reasoning. The integrity check needs to consider that the robot cannot find an object with visual descriptions like color, shape, etc., because the robot cannot actually perceive visual features of the space and objects. But if an object property specified in the command is achieved in the action sequence, it is consistent. For example, given a command "Put a clean knife on the countertop." and an action sequence involving cleanobject(knife), the command is consistent with the action sequence.

Integrity errors can be categorized into the following three types:

1. Type 1: Spatial Interpretation Errors

- Explanation: Problems with correctly interpreting or locating specified spatial instructions.
- Example 1:
 - Command: Put a plate on the counter left of the sink.
 - Action Sequence: 1. GotoLocation(countertop), 2. PickupObject(plate), 3. GotoLocation(countertop), 4. PutObject(plate,countertop), 5. Done.
 - Issue: The command mistakenly specifies the position relative to the sink but the action sequence places the plate simply on the countertop.
- Example 2:
 - Command: Put a spray bottle on the toilet tank lid.
 - Action Sequence: 1. GotoLocation(countertop), 2. PickupObject(spraybottle), 3. GotoLocation(toilet), 4. PutObject(spraybottle,toilet), 5. Done.
 - Issue: The command mistakenly directs to put a spray bottle on the toilet tank lid but the action sequence places it simply in the toilet.

2. Type 2: Object Specification Errors

- Explanation: Instances where the robot substitutes the intended object with a different one, either due to perception limitations or task misinterpretation.
- Example 1:
 - Command: Place a potato in the microwave.
 - Action Sequence: 1. GotoLocation(countertop), 2. PickupObject(egg), 3. GotoLocation(microwave), 4. PutObject(egg,microwave), 5. Done.
 - Issue: The command mistakenly directs to put a potato in the microwave but the action sequence puts an egg instead.
- Example 2:
 - Command: Move a bottle of lotion into a drawer.
 - Action Sequence: 1. GotoLocation(countertop), 2. PickupObject(soapbottle), 3. GotoLocation(drawer), 4. PutObject(soapbottle,drawer), 5. Done.
 - Issue: The command mistakenly directs to move a lotion bottle into a drawer but the action sequence involves a soap bottle instead.
- Example 3:
 - Command: Put a yellow-handled knife into a drawer.
 - Action Sequence: 1. GotoLocation(countertop), 2. PickupObject(knife), 3. GotoLocation(drawer), 4. PutObject(knife,drawer), 5. Done.
 - Issue: The command directs to find a yellow-handled knife, but the robot cannot perceive the color and cannot fulfill the task based on the command.

3. Type 3: Action Sequence Completeness Issues

- Explanation: Key actions or details are omitted, leading to incomplete task fulfillment.
- Example 1:

- Command: Put a plate in a fridge.
- Action Sequence: 1. GotoLocation(countertop), 2. PickupObject(plate), 3. GotoLocation(sinkbasin), 4. CleanObject(plate), 5. GotoLocation(fridge), 6. PutObject(plate,fridge), 7. Done.
- Issue: The command mistakenly directs to put a plate in a fridge but the action sequence includes washing the plate.
- Example 2:
 - Command: Put a bowl with a tomato in it in the fridge.
 - Action Sequence: 1. GotoLocation(diningtable), 2. PickupObject(butterknife), 3. GotoLocation(tomato), 4. SliceObject(tomato), 5. PutObject(butterknife,countertop), 6. PickupObject(tomato), 7. GotoLocation(bowl), 8. PutObject(tomato,bowl), 9. PickupObject(bowl), 10. GotoLocation(fridge), 11. PutObject(bowl,fridge), 12. Done.
 - Issue: The command mistakenly directs to put a bowl with a tomato in it in the fridge but the action sequence includes slicing the tomato.

4. Type 4: Other Errors

- Explanation: Other errors that do not fit into the above categories.

5. Type 5: No Error

- Explanation: The command is consistent with the action sequence.

You will be given a Query consisting of a command and a sequence of actions that accomplish the command. You check the integral correspondence between the command and the action sequence and decide whether the robot can accomplish the goal of the command under the perceptual constraints.

Answer only in JSON format, without any compartments like """"json"" or """""".

Here are some samples:

Query:

- Task Type: pick_and_place_simple
- Action Sequence: 1. GotoLocation(countertop), 2. PickupObject(peppershaker), 3. GotoLocation(drawer), 4. PutObject(peppershaker,drawer), 5. Done.
- Command: Put the pepper shaker to the drawer.

Answer:

```
{"Integrity": true, "Reason": "The command is consistent with the action sequence.", "Type": 5}
```

Query:

- Task Type: pick_two_obj_and_place
- Action Sequence: 1. GotoLocation(garbagecan), 2. PickupObject(toiletpaper), 3. GotoLocation(cabinet), 4. PutObject(toiletpaper,cabinet), 5. GotoLocation(countertop), 6. PickupObject(toiletpaper), 7. GotoLocation(cabinet), 8. PutObject(toiletpaper,cabinet), 9. Done.
- Command: Put two toilet papers on the yellow table.

Answer:

```
{"Integrity": false, "Reason": "The command mistakenly specifies to put two toilet papers on the yellow table, but the action sequence places the toilet paper in the cabinet.", "Type": 1}
```

Now, answer the following query.

Query:

- Task Type: {task_type}

– Action Sequence: {action_sequence}
– Command: {command}
Answer:

A.2 Command Generation

You are a reasoning agent specialized in intent prediction. There are seven types of tasks that can be commanded: 1. Pick and Place, 2. Stack and Place, 3. Pick Two and Place, 4. Clean and Place, 5. Heat and Place, 6. Cool and Place, 7. Examine in Light.

Sometimes tasks involve slicing objects like apples. In this world, when you put down something in the fridge and then pick it up again, it is assumed that the object is cooled. As such, when you put down something in the microwave and then pick it up again, it is assumed that the object is heated. Also, when you put down something in the sink and open a facet and then close it, it is assumed that the object is cleaned. When you want to put something into a receptacle, you should open it first and then close it after you put the object in. Sometimes the task involves to put something into a movable receptacle like a bowl and then place the receptacle in a specific location.

If heating, cooling, cleaning, or slicing action is involved in the process, it should be mentioned in the command. You will be given with a Query consisted of a sequence of actions that accomplishes an unknown task. You infer the goal of the action sequence and answer with a natural language command that properly addresses the goal. Answer with only the command without any additional rationale or explanation.

Here are some samples.

Query:
– Task Type: pick_and_place_simple
– Action Sequence: 1. GotoLocation(countertop), 2. PickupObject(peppershaker), 3. GotoLocation(drawer), 4. PutObject(peppershaker,drawer), 5. Done.
Answer: Put the pepper shaker to the drawer.

Query:
– Task Type: pick_two_obj_and_place
– Action Sequence: 1. GotoLocation(garbagecan), 2. PickupObject(toiletpaper), 3. GotoLocation(cabinet), 4. PutObject(toiletpaper,cabinet), 5. GotoLocation(countertop), 6. PickupObject(toiletpaper), 7. GotoLocation(cabinet), 8. PutObject(toiletpaper,cabinet), 9. Done.
Answer: Put two toilet paper to the cabinet.

Query:
– Task Type: pick_and_place_with_movable_recep
– Action Sequence: 1. GotoLocation(diningtable), 2. PickupObject(knife), 3. SliceObject(tomato), 4. GotoLocation(diningtable), 5. PutObject(knife,diningtable), 6. GotoLocation(diningtable), 7. PickupObject(tomato), 8. GotoLocation(pot), 9. PutObject(tomato,pot), 10. PickupObject(pot), 11. GotoLocation(fridge), 12. PutObject(pot,fridge), 13. Done.
Answer: Put a sliced tomato in the pot and then put the pot in the fridge.

Query:
– Task Type: pick_and_place_with_movable_recep
– Action Sequence: 1. GotoLocation(microwave), 2. PickupObject(cup), 3. GotoLocation(sinkbasin), 4. PutObject(cup, sinkbasin), 5. PickupObject(butterknife), 6. PutObject(butterknife, cup), 7. Done.
Answer: Put a cup containing a butter knife in it to the sink.

Query:

– Task Type: pick_heat_then_place_in_recep
– Action Sequence: 1. GotoLocation(sinkbasin), 2. PickupObject(butterknife), 3. SliceObject(tomato), 4. GotoLocation(microwave), 5. PutObject(butterknife,microwave), 6. GotoLocation(sinkbasin), 7. PickupObject(tomato), 8. GotoLocation(microwave), 9. HeatObject(tomato), 10. GotoLocation(sinkbasin), 11. PutObject(tomato,sinkbasin)
Answer: Put a slice of heated tomato in the sink.
Now, answer to the following query.

Query:
– Task Type: {task_type}\
– Action Sequence: {action_sequence}\
Answer:

A.3 Command Augmentation

Listing 1: Prompt for Rephrased-1

Please rephrase each of the following commands. Use everyday conversational expressions. The nouns need to be retained, but verbs, adjectives, adverbs and countable expressions can be paraphrased. The meaning of the original commands should be retained! Please punctuate each sentence with a new line. And good deal of diversity is welcomed, e.g. variations on 'cooled', 'cleaned' etc. with different synonyms.

{commands}

Listing 2: Prompt for Rephrased-2

Please rephrase each the following commands. Use everyday conversational expressions. The nouns referring to objects and locations can be paraphrased. The meaning of the original commands should be retained! Please punctuate each sentence with a new line. And good deal of diversity is welcomed.

{commands}

Listing 3: Prompt for Contextual commands

The original commands are somewhat direct in that it directly indicates what to do. Let's make it somewhat indirect so that an agent needs to do more elaborate reasoning to accomplish the task.

First, provide context information rich enough to make an agent able to come up with a concrete task plan. Second, give an indirect command that is clearly meaningful under the given context. An example:

– Original Command: "Put the spray bottle in the toilet."
– Altered Command: "In our house, cleaning supplies like spray bottles are always kept in the bathroom. Thank you for completing your task with the spray bottle. Please be sure to return it to its usual spot."

{commands}

Listing 4: Prompt for Korean commands

Please translate the following commands in plain and clear Korean.

{commands}

{commands} is replaced with a list of commands, each separated by a newline.

A.4 Failure Analysis

The following prompt is designed to analyze planning logs in order to identify and categorize failure cases.

The following is a log of a session for task execution consisted of action steps and its outcome indicating the success or failure of the execution.

When an action fails, a clue message or two are provided. Please analyze the log and let me know which step has failed on which reason.

And finally, please provide the type of the failure which is categorized as follows:

1. Misleading Command: The command directed the robot to perform an action, but the target goal was different from the command.
2. Action Failure: The robot failed to perform the action due to navigation or manipulation errors in the given environment.
3. Plan Failure: The steps are not appropriate to achieve the target goal.

Sometimes there could be two error types observed in the log. For example, the command is misleading based on the target goal, and some step of the execution fails due to navigation or manipulation errors in the environment.

In this case, you should indicate the two error types in the Failure Type field.

Please provide the type of the failure which is categorized as follows:

You will be given a log with a tag <LOG>, and you provide the analysis with a tag <ANALYSIS>.

<ANALYSIS> should be in the following JSON format:

```
{
  "Command": "The command",
  "Steps": [
    {"action": "The action", "succeeded": true or false},
    {"action": "The action", "succeeded": true or false},
    ...
  ],
  "succeeded": true or false,
  "Analysis": "The analysis",
  "Failure Types": [1, 2] # a list of failure types, 1 is misleading
  command, 2 is action failure, 3 is plan failure
}
```

Here are some samples:

```
<LOG>
{'repeat_idx': 0, 'task': 'pick_and_place_simple-SoapBar-None-
Cart-401/trial_T20190907_054906_608944'}Evaluating (1/208):
alfred/data/json_2.1.0/pick_and_place_simple-SoapBar-None-
Cart-401/trial_T20190907_054906_608944
Task: To place the soap on the rack.
#####Whole.Sequence mode
HTTP Request: POST https://api.openai.com/v1/chat/completions
"HTTP/1.1 200 OK"
[]
1. find a soap bar
object navigation (SoapBar)
Last action succeeded
```

```
2. pick up the soap bar
pick SoapBar|-02.39|+01.30|+03.92
Last action succeeded
3. find a rack
object navigation (Rack)
Last action succeeded
4. put down the soap bar
Last action succeeded
5. done
target goal: {"object": "SoapBar", "parent": "Cart", "toggle": "",
"mrecep": ""}
success: False
</LOG>

<ANALYSIS>
{ "Command": "To place the soap on the rack.",
  "Task ID": "pick_cool.then_place_in_recep-Plate-None-
Cabinet-27/trial_T20190906_173120_350651",
  "Steps": [
    {"action": "Find a soap bar", "succeeded": true},
    {"action": "Pick up the soap bar", "succeeded": true},
    {"action": "Find a rack", "succeeded": true},
    {"action": "Put down the soap bar", "succeeded": true},
    {"action": "Done", "succeeded": true}
  ],
  "succeeded": false,
  "Analysis": "The command directed the robot to place the soap
bar on the rack, but the target goal was to place the soap bar on
the cart. The task command is misleading which led to the
failure. The command and the target goal are mismatched.",
  "Failure Types": [1]
}
</ANALYSIS>

<LOG>
{'repeat_idx': 5, 'task': 'pick_cool.then_place_in_recep-Plate-
None-Cabinet-27/trial_T20190906_173120_350651'}Evaluating
(2/208): alfred/data/json_2.1.0/pick_cool.then_place_in_recep-
Plate-None-Cabinet-27/trial_T20190906_173120_350651
Task: Put a plate in a cabinet.
#####Whole.Sequence mode
HTTP Request: POST https://api.openai.com/v1/chat/completions
"HTTP/1.1 200 OK"
1. find a plate
object navigation (Plate)
Last action succeeded
2. pick up the plate
pick Plate|-00.31|+00.78|+00.41
Last action succeeded
3. find a cabinet
object navigation (Cabinet)
Last action succeeded
4. open the cabinet
open Cabinet
OpenObject action failed: object not found: Cabinet
|+01.76|+00.39|+00.87, moving backward and trying again...
Last action succeeded
5. put down the plate
put Plate|-00.31|+00.78|+00.41 on Cabinet
|+01.76|+00.39|+00.87
PutObject action failed: No valid positions to place object found,
trying again...
put Plate|-00.31|+00.78|+00.41 on Cabinet
|+01.76|+00.39|+00.87
PutObject action failed: No valid positions to place object found,
```



```

trying again...
put Plate|-00.31|+00.78|+00.41 on Cabinet
|+01.76|+00.39|+00.87
PutObject action failed: No valid positions to place object found,
trying again...
put Plate|-00.31|+00.78|+00.41 on Cabinet
|+01.76|+00.39|+00.87
PutObject action failed: No valid positions to place object found,
trying again...
put Plate|-00.31|+00.78|+00.41 on Cabinet
|+01.76|+00.39|+00.87
PutObject action failed: No valid positions to place object found,
trying again...
put Plate|-00.31|+00.78|+00.41 on Cabinet
|+01.76|+00.39|+00.87
PutObject action failed: No valid positions to place object found,
trying again...
put Plate|-00.31|+00.78|+00.41 on Cabinet
|+01.76|+00.39|+00.87
PutObject action failed: No valid positions to place object found,
trying again...
put Plate|-00.31|+00.78|+00.41 on Cabinet
|+01.76|+00.39|+00.87
PutObject action failed: No valid positions to place object found,
trying again...
put Plate|-00.31|+00.78|+00.41 on Cabinet
|+01.76|+00.39|+00.87
PutObject action failed: No valid positions to place object found,
trying again...
put Plate|-00.31|+00.78|+00.41 on Cabinet
|+01.76|+00.39|+00.87
PutObject action failed: No valid positions to place object found,
trying again...
put Plate|-00.31|+00.78|+00.41 on Cabinet
|+02.04|+01.81|+00.28
PutObject action failed: No valid Receptacle found, trying again...
put Plate|-00.31|+00.78|+00.41 on Cabinet
|+01.76|+00.39|+00.87
PutObject action failed: No valid positions to place object found,
trying again...
put Plate|-00.31|+00.78|+00.41 on Cabinet
|+01.76|+00.39|+00.87
PutObject action failed: No valid positions to place object found,
trying again...
drop
llm_skill_interact failed
errorMessage:
returned msg: put down failed
6. close the cabinet
close Cabinet
Last action succeeded
7. done
target goal: {"object": "Plate", "parent": "Cabinet", "toggle": "",
"mrecep": ""}
success: False
</LOG>

<ANALYSIS>
{"Command": "Put a plate in a cabinet.",
"Task ID": "pick_cool_then_place_in_recep-Plate-None-
Cabinet-27/trial_T20190906_173120_350651",
"Steps": [

```

```

{"action": "Find a plate", "succeeded": true},
{"action": "Pick up the plate", "succeeded": true},
{"action": "Find a cabinet", "succeeded": true},
{"action": "Open the cabinet", "succeeded": true},
{"action": "Put down the plate", "succeeded": false},
{"action": "Close the cabinet", "succeeded": true},
{"action": "Done", "succeeded": false} ],
"succeeded": false,
"Analysis": "The robot failed at step 5 while attempting to put
down the plate into the cabinet. Multiple 'PutObject' action
failures occurred with warnings such as 'No valid positions to
place object found' and 'No valid Receptacle found'. This
indicates manipulation errors in the environment prevented the
robot from completing the action.",
"Failure Types": [2]
}
</ANALYSIS>

<LOG>
{"repeat_idx": 2, 'task': 'pick_clean_then_place_in_recep-
DishSponge-None-Cart-401/trial_T20190907_024634_972453'}
Evaluating (4/208): alfred/data/json_2.1.0/
pick_clean_then_place_in_recep-DishSponge-None-Cart-401/
trial_T20190907_024634_972453
Task: Put a clean sponge on a metal rack.
#####Whole.Sequence mode
HTTP Request: POST https://api.openai.com/v1/chat/completions
"HTTP/1.1 200 OK"
1. find a sponge
object navigation (Sponge)
Last action succeeded
2. pick up the sponge
pick None
Last action succeeded
3. find a sink
object navigation (Sink)
Last action succeeded
4. put down the sponge
drop
llm_skill_interact failed
errorMessage: nothing in hand to drop!
returned msg: Robot is not holding any object
5. find a faucet
object navigation (Faucet)
Object is already visible
llm_skill_interact failed
errorMessage: nothing in hand to drop!
returned msg:
6. turn on the faucet
toggle on Faucet
Last action succeeded
7. turn off the faucet
toggle off Faucet
Last action succeeded
8. find a sponge
object navigation (Sponge)
Last action succeeded
9. pick up the sponge
pick None
Last action succeeded
10. find a metal rack
object navigation (MetalRack)
Last action succeeded
11. put down the sponge
drop

```

```

llm_skill_interact failed
errorMessage: nothing in hand to drop!
returned msg: Robot is not holding any object
12. done
target goal: {"object": "DishSponge", "parent": "Cart", "toggle":
"", "mrecep": ""}
success: False
</LOG>

<ANALYSIS>
{ "Command": "Put a clean sponge on a metal rack.",
  "Task ID": "pick_clean_then_place_in_recep-DishSponge-None
-Cart-401/trial_T20190907_024634_972453",
  "Steps": [
    {"action": "Find a sponge", "succeeded": true},
    {"action": "Pick up the sponge", "succeeded": false},
    {"action": "Find a sink", "succeeded": true},
    {"action": "Put down the sponge", "succeeded": false},
    {"action": "Find a faucet", "succeeded": false},
    {"action": "Turn on the faucet", "succeeded": true},
    {"action": "Turn off the faucet", "succeeded": true},
    {"action": "Find a sponge", "succeeded": true},
    {"action": "Pick up the sponge", "succeeded": false},
    {"action": "Find a metal rack", "succeeded": true},
    {"action": "Put down the sponge", "succeeded": false},
    {"action": "Done", "succeeded": false} ],
  "succeeded": false,
  "Analysis": "The robot was instructed to 'Put a clean sponge on
a metal rack,' but the target goal was to place the 'DishSponge'
on the 'Cart'. This mismatch between the command and the
target goal led to the failure. Additionally, in steps 2 and 9, the
robot attempted to 'pick None,' indicating it did not correctly
pick up the sponge.",
  "Failure Types": [1, 2]
}
</ANALYSIS>

<LOG>
{'repeat_idx': 1, 'task': 'pick_cool_then_place_in_recep-
BreadSliced-None-DiningTable-27/
trial_T20190908_075813_148407'}Evaluating (5/208): alfred/data
/json.2.1.0/pick_cool_then_place_in_recep-BreadSliced-None-
DiningTable-27/trial_T20190908_075813_148407
Task: Slice bread, chill it in the fridge and put it on the dining
table.
#####Whole_Sequence mode
HTTP Request: POST https://api.openai.com/v1/chat/completions
"HTTP/1.1 200 OK"
1. find a bread
object navigation (Bread)
Last action succeeded
2. pick up the bread
pick Bread|-00.20|+00.82|+00.81
Last action succeeded
3. slice the bread
slice Bread
llm_skill_interact failed
errorMessage: target object cannot be sliced if it is in the agent's
hand
returned msg: Slice action failed
4. put down the bread
drop
Last action succeeded
5. find a fridge
object navigation (Fridge)

```

```

Last action succeeded
6. open the fridge
open Fridge
Last action succeeded
7. pick up the bread
pick Bread|-00.20|+00.82|+00.81
llm_skill_interact failed
errorMessage: Bread|-00.20|+00.82|+00.81 is not visible.
returned msg: Robot is not holding any object
8. put down the bread
drop
drop
llm_skill_interact failed
errorMessage: nothing in hand to drop!
returned msg: Robot is not holding any object
9. close the fridge
close Fridge
Last action succeeded
10. done
target goal: {"object": "BreadSliced", "parent": "DiningTable", "
toggle": "", "mrecep": ""}
success: False
</LOG>

<ANALYSIS>
{ "Command": "Slice bread, chill it in the fridge and put it on the
dining table.",
  "Task ID": "pick_cool_then_place_in_recep-BreadSliced-None
-DiningTable-27/trial_T20190908_075813_148407",
  "Steps":
  [
    {"action": "Find a bread", "succeeded": true},
    {"action": "Pick up the bread", "succeeded": true},
    {"action": "Slice the bread", "succeeded": false},
    {"action": "Put down the bread", "succeeded": true},
    {"action": "Find a fridge", "succeeded": true},
    {"action": "Open the fridge", "succeeded": true},
    {"action": "Pick up the bread", "succeeded": false},
    {"action": "Put down the bread", "succeeded": false},
    {"action": "Close the fridge", "succeeded": true},
    {"action": "Done", "succeeded": false}
  ],
  "succeeded": false,
  "Analysis": "The robot failed at step 3 by attempting to slice
the bread while holding it, which is not permitted ('target object
cannot be sliced if it is in the agent's hand'). Additionally, after
placing the bread down, the robot did not attempt to slice it
again. At step 7, the robot failed to pick up the bread because it
was not visible, indicating a navigation or perception error. The
plan did not account for slicing the bread while it was on a
surface, leading to a failure in task planning and execution.",
  "Failure Types": [2, 3]
}
</ANALYSIS>

<LOG>
{session_log}
</LOG>

<ANALYSIS>

```

{session_log} is replaced with a log from a plan evaluation session generated by Lota-Bench (Choi et al. 2024).