UNCC @NeurIPS 2025 Embodied Agent Interface Challenge

Hilmi Demirhan[1*] Wlodek Zadrozny[1]

[1]University of North Carolina at Charlotte
hdemirha@charlotte.edu

Abstract

This paper presents our work for the Embodied Agent Interface Challenge, which evaluates the reasoning ability of language models in simulated environments. The challenge focuses on four skills: understanding the goal, breaking it into smaller steps, planning a sequence of actions, and predicting how the environment changes after each action. These tasks are carried out in the BEHAVIOR and VirtualHome simulators, both of which include well-defined scenarios and a uniform evaluation setup. Our system uses the OpenAI API with different models depending on the environment and stage of development: `gpt-4.1` for BEHAVIOR, `gpt-4.1-mini` for VirtualHome, and `gpt-5-mini` for both environments in later runs. The schemas for each task determine how the outputs are structured, and outputs are regenerated when they do not follow the specification. While building our system, we identified common issues, including unclear instructions and missing context, as well as discrepancies between the two environments. In the final public leaderboard, our system placed eighteenth overall with an overall score of 57.92, including 68.88 on the BEHAVIOR tasks and 46.96 on the VirtualHome tasks. In this paper, we describe our approach and discuss what these observations suggest about the strengths and limitations of current language models when used for embodied reasoning.

1 Introduction

The Embodied Agent Interface Challenge evaluates a system's ability to interpret a written goal, outline the steps needed to accomplish it, generate a sequence of actions, and predict the symbolic state changes that follow from those actions [7]. These four tasks cover the main steps involved in embodied reasoning, and the challenge separates them so each part can be examined on its own. This arrangement lets researchers look closely at how a system performs each step and see how mistakes in one part may carry into the next. By focusing on written goals and symbolic representations, the challenge provides a clear setting for analyzing reasoning without the added complexity of visual or physical inputs.

The challenge uses task definitions and symbolic interfaces derived from the BEHAVIOR [15] and VirtualHome [12] environments. These environments provide identified objects, permitted actions, and rules that describe how an action affects the symbolic state. Each task uses a fixed input format and output format to keep the evaluation process consistent. The provided annotations and scoring criteria support systematic examination and comparison of different systems. Since the task definitions align across the two environments, the challenge offers a controlled platform for evaluating reasoning approaches under different levels of task complexity.

We implemented the four-module architecture specified by the interface, applying the official schemas and input–output rules at each stage of the pipeline. The modular structure helps keep the process easy to follow and makes it possible to trace errors to a specific stage, since each part of the

*Corresponding author: hdemirha@charlotte.edu

pipeline produces a structured result. The system accommodates both BEHAVIOR and VirtualHome by using configurations that outline the objects, actions, and limits of each environment. In this paper, we present our approach, explain how it aligns with the challenge requirements, and discuss the observations made during development and testing.

2 Related Work

Early studies of embodied reasoning emerged from research on symbolic planning, instruction following, and hierarchical task networks. These systems showed that breaking a task into several smaller steps helps make the overall process clearer. Research on hierarchical task networks stressed the role of plans with well-defined preconditions and effects [2]. Shakey the Robot showed an early attempt to link symbolic planning with actions carried out in the real world [9]. Work on systems such as SHRDLU explored how natural language instructions could be interpreted and turned into actions in controlled settings [17]. Later work on grounded language understanding explored how robots could interpret language commands for navigation and manipulation [8]. Early learning-based systems such as ALVINN explored how perception and action could be connected in autonomous driving [11].

Simulation environments have been central to research on embodied tasks. VirtualHome introduced program-like task descriptions in a three-dimensional household setting, allowing activities to be studied with a fixed set of objects and actions [12]. The BEHAVIOR and BEHAVIOR-1K benchmarks expanded this approach by introducing more realistic scenes, a wider range of tasks, and detailed symbolic state information [15, 6].

The growth of large language models has led to renewed interest in using natural language for planning and higher-level control in robotics. Surveys have examined the ways language models assist with planning, tool use, activity descriptions, and multi-step instruction following [19, 16, 5]. Other studies have looked at how language relates to actions and perceptions, comparing symbolic, neural, and mixed approaches to robot reasoning [1, 4]. These works indicate that language models can give reasonable task steps when their outputs are limited to a known environment. However, they also document recurring issues, such as referring to missing objects, selecting actions that are not permitted, and producing state changes that do not follow the environment's rules.

Recent work on agentic systems looks at how language models can guide decisions over several steps and incorporate feedback from earlier outputs. The ReAct approach combined short reasoning notes with actions, letting a model decide what to do while explaining its choices [18]. Reflexion added a simple form of self-correction in which an agent reviews its previous attempts and adjusts its next move [14]. Other studies examined agents that call tools or external services to gather information during a task [13, 10]. In embodied tasks, researchers often use a layered design where the language model handles planning and a separate controller carries out the lower-level actions [3]. These agentic systems demonstrate the importance of structured interfaces, since they help keep reasoning aligned with the available actions and make intermediate decisions visible.

The Embodied Agent Interface was introduced to give researchers a common way to evaluate reasoning in embodied tasks [7]. The interface defines four modules—goal interpretation, subgoal decomposition, action sequencing, and transition modeling—each with its own schema and required fields. Our system follows this structure and uses the schemas to shape the input prompts and to check the format of the returned outputs. When an output does not follow the expected structure, it is regenerated so that the stages produce consistent intermediate results. This setup helps show how the reasoning process unfolds across the modules and makes it possible to identify where formatting issues and other irregularities appear during execution.

3 Methods

This section describes the system we built for the Embodied Agent Interface Challenge. The pipeline follows the four modules defined by the interface and uses the official prompt files for each task. The system does not train or fine-tune any models. Its purpose is to run the full set of inputs in a stable

way and to produce the required output files for the challenge evaluation.

3.1 System Overview

The system is divided into four modules corresponding to the interface: goal interpretation, subgoal decomposition, action sequencing, and transition modeling. For each module, the code loads the appropriate prompt, forwards it to the model, and saves the response to disk. The modules run independently, and each produces its own output file, which makes it easy to trace errors to the specific stage in which they appear.

The directory structure mirrors this workflow. Prompts, intermediate outputs, submission files, and checkpoints are stored in separate folders. Tasks are processed in batches, and the system writes checkpoint files at regular intervals so long runs can resume from the most recent completed item.

Different OpenAI models are used depending on the environment. BEHAVIOR tasks are run with `gpt-4.1`, VirtualHome tasks with `gpt-4.1-mini`, and later experiments evaluate `gpt-5-mini` for both environments. The pipeline logic remains unchanged across models.

3.2 Model Execution and Validation

An execution wrapper manages the interaction with the API, including timeouts, rate-limit responses, and similar transient failures. If a request returns an error, the wrapper pauses briefly before issuing another attempt. During extended runs, the system writes checkpoint files that allow it to continue from the most recent successful entry.

The response from the model is recorded in its original form. No additional processing or modification is performed. The implementation does not attempt to enforce the schema beyond checking that the response is non-empty. Any deeper validation is deferred to the official challenge evaluation tools.

3.3 Goal Interpretation

The goal interpretation module reads the natural-language description from the prompt file and forwards it to the model using the template defined by the challenge. The returned text is written directly to disk. The BEHAVIOR and VirtualHome versions differ only in the prompt content; the processing steps are the same.

3.4 Subgoal Decomposition

This module generates a list of intermediate steps based on the goal interpretation prompt. The prompt supplies the subgoal types permitted in the environment, and the model returns a sequence in the required format. The output is saved as returned. No additional checks are performed beyond confirming that the model produced a response.

3.5 Action Sequencing

The action sequencing module asks the model to expand each subgoal into an ordered list of actions. The prompt includes the allowed actions and object names for the environment. The model output is recorded as-is. The implementation does not perform planning or structural validation; all checking is handled by the challenge evaluator.

3.6 Transition Modeling

The transition modeling module asks the model to produce symbolic state updates for each action. The prompt includes the action list and the allowed state attributes for the environment. The returned text is saved without modification. No consistency checks are performed between steps; the module only records the generated output.

3.7 Environment-Specific Adaptations

Each environment provides its own prompt and configuration files describing objects, actions, subgoal types, and state attributes. The pipeline loads the appropriate files based on the task and forwards them to the model. The system does not attempt to merge BEHAVIOR and VirtualHome or translate between them. It applies the rules present in the prompt files, which allows the same implementation to run both environments without changing the underlying code.

4 Implementation Details

This work uses three OpenAI models: `gpt-4.1`, `gpt-4.1-mini`, and `gpt-5-mini`. The models are used as released, without any additional training or modification. All inputs come directly from the official prompt files. All experiments were run in Google Colab on CPU-based machines without GPU acceleration. Inference time depends on the length of the prompts and responses, but it was low enough to process the full set of tasks in large batches.

5 Evaluation Results

Our submission ranked 18th overall in the challenge with an overall score of 57.92. The system showed stronger performance in the BEHAVIOR environment, where the average score was 68.88. BEHAVIOR results were consistently higher across all four modules, including goal interpretation (78.70), action sequencing task success (75.00), execution success (83.00), subgoal decomposition (49.00 task-level and 54.00 execution-level), and transition modeling (58.60). The transition-modeling planner score reached 87.00, which was the strongest individual metric in our submission. These results indicate that the system handled BEHAVIOR's richer scene information and more detailed task structure more reliably.

Performance in VirtualHome was lower, with an average score of 46.96. Goal interpretation reached 22.10, and action sequencing produced stronger results at 68.90 for task success and 79.60 for execution success. Subgoal decomposition followed a similar trend with 61.80 at the task level and 79.50 at execution. Transition modeling showed a large decline in VirtualHome, with scores of 40.60 for state prediction and 29.50 for the planner metric. Many of the failures came from malformed JSON, missing fields, and object references that were not part of the environment. These issues appeared more often in VirtualHome than in BEHAVIOR. The difference between the two environments suggests that the system is sensitive to how much detail is provided in the prompt.

6 Analysis and Discussion

6.1 Error Analysis

Several types of errors appeared during evaluation. The main errors involved actions not included in the allowed vocabulary, missing or incorrect object references, invalid relations, and incomplete JSON outputs. Fewer errors occurred on BEHAVIOR tasks, where the prompts described the scenes and objects in more detail.

6.2 Task-Specific Observations

Goal interpretation worked reliably in BEHAVIOR but showed more problems in VirtualHome. Action sequencing was the most steady part of the system when the prompts used object names exactly as they appeared in the environment files. Transition modeling showed the widest performance gap between the two environments, with BEHAVIOR producing noticeably better results.

6.3 Cross-Environment Comparison

BEHAVIOR's higher performance reflects its more detailed prompts, larger number of object attributes, and clearer task descriptions. VirtualHome produced more malformed outputs and unsupported references. The model was more sensitive to formatting details in VirtualHome tasks.

6.4 Insights and Lessons Learned

Most errors were caused by schema violations, and structural problems were more common than mistakes in the reasoning itself. Regenerating outputs reduced the number of invalid cases but added to the total runtime. Tighter prompt specifications and decoding methods that enforce the expected format would likely improve the stability of the system. More controlled generation would help reduce malformed outputs, especially in environments with limited context, such as VirtualHome. In addition, methods that incorporate structural constraints during decoding could prevent many of the schema violations observed in the results.

7 Conclusion

We developed a modular system based on language model inference for the Embodied Agent Interface Challenge and evaluated it on all four tasks in both BEHAVIOR and VirtualHome. The system performed steadily in BEHAVIOR but showed more variability in VirtualHome, largely due to structural problems in the generated JSON. The results show that stricter schema control and more dependable output generation are important for improving overall stability. Many of the observed errors could be avoided if the system enforced the expected structure during generation rather than filtering after the fact. Future work may include decoding methods that restrict outputs to valid formats, the use of symbolic checks alongside model outputs, and adjustments to the prompt structure to reduce ambiguity.

References

[1] Vanya Cohen, Jason Xinyu Liu, Raymond Mooney, Stefanie Tellex, and David Watkins. A survey of robotic language grounding: Tradeoffs between symbols and embeddings. *arXiv preprint arXiv:2405.13245*, 2024.

[2] Kutluhan Erol, James Hendler, and Dana S Nau. Htn planning: Complexity and expressivity. In *AAAI*, volume 94, pages 1123–1128, 1994.

[3] 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.

[4] Hyeongyo Jeong, Haechan Lee, Changwon Kim, and Sungtae Shin. A survey of robot intelligence with large language models. *Applied Sciences*, 14(19):8868, 2024.

[5] Yeseung Kim, Dohyun Kim, Jieun Choi, Jisang Park, Nayoung Oh, and Daehyung Park. A survey on integration of large language models with intelligent robots. *Intelligent Service Robotics*, 17(5):1091–1107, 2024.

[6] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pages 80–93. PMLR, 2023.

[7] Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Erran Li Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking

llms for embodied decision making. *Advances in Neural Information Processing Systems*, 37:100428–100534, 2024.

[8] Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. Learning to parse natural language commands to a robot control system. In *Experimental robotics: the 13th international symposium on experimental robotics*, pages 403–415. Springer, 2013.

[9] Nils J Nilsson. Shakey the robot. 1984.

[10] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565, 2024.

[11] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[12] 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.

[13] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

[14] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

[15] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, Karen Liu, et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on robot learning*, pages 477–490. PMLR, 2022.

[16] Jiaqi Wang, Enze Shi, Huawen Hu, Chong Ma, Yiheng Liu, Xuhui Wang, Yincheng Yao, Xuan Liu, Bao Ge, and Shu Zhang. Large language models for robotics: Opportunities, challenges, and perspectives. *Journal of Automation and Intelligence*, 4(1):52–64, 2025.

[17] Terry Winograd. Understanding natural language. *Cognitive psychology*, 3(1):1–191, 1972.

[18] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

[19] Fanlong Zeng, Wensheng Gan, Yongheng Wang, Ning Liu, and Philip S Yu. Large language models for robotics: A survey. *arXiv preprint arXiv:2311.07226*, 2023.