SalsaBot: Towards a Robust and Generalizable Embodied Agent

Chan Hee Song*†, Jiaman Wu*, Ju-Seung Byeon, Zexin Xu, Vardaan Pahuja,
Goonmeet Bajaj, Samuel Stevens, Ziru Chen, Yu Su

The Ohio State University

Abstract

As embodied agents become more powerful, there arises a need for an agent to collaboratively solve tasks with humans. This paper introduces SalsaBot, an embodied agent designed for the Alexa Arena benchmark, which is a collaborative human-robot interaction benchmark. The primary aim of SalsaBot is to assist users in completing a game within a virtual environment by providing a consistent user-centric experience, which requires the agent to be capable of handling various types of user interactions. To ensure a great user experience, SalsaBot is equipped with robust macros, an explicit object memory, and a state-aware dialogue generation module. Our efforts and findings demonstrate that our SalsaBot is a robust interactive agent that can effectively collaborate with users.

1. Introduction

Building an embodied AI agent marks an exciting milestone in the field of artificial intelligence. While traditional embodied AI agents have focused on executing static tasks given by users, there is need for a conversational embodied agent that can collaborate with users to successfully complete the task. The recently released Alexa Arena benchmark [1] aims to test the embodied AI agent's ability to complete a human-robot interaction tasks. In Arena, the tasks are goal-oriented and formulated as games. At the beginning of each game session, the human users are given a high-level goal and a list of subgoals, and they engage with the agent in a multi-turn task-oriented dialogue via voice to command the agent to achieve the goal. Unlike the existing benchmarks [2] that present the complete task up front, Arena requires the robot to collaborate with users to fill in the missing information in user utterances. The unique collaborative aspect of the benchmark implies that no off-the-shelf method for embodied instruction following would work to a satisfactory degree or can be adapted with reasonable efforts.

*Team Leads. Equal Contribution.
†Corresponding author: song.1855@osu.edu

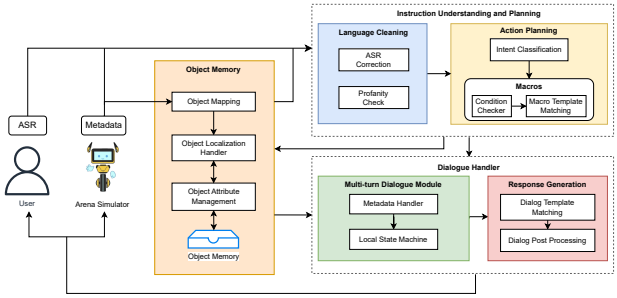

Figure 1. Overall system design.

As the instruction becomes conversational, it is much more important for the embodied AI agent to understand the diversity of the user's instructions. For static datasets, the language patterns are usually consistent between training and test data. However, for an conversational benchmark like Arena, user dialogues are vastly diverse. Not only does this manifest in the large number of different ways users may use to express the same thing, but it also manifests in the different levels of abstraction: Some users may prefer a *procedural* way and mostly use primitive actions, while other users may be more *declarative* and give high-level commands, e.g., *"Get milk from fridge"* entails a sequence of actions including going to the fridge, opening it, picking up milk, and closing the fridge.

We propose SalsaBot, a interactive embodied agent delivering a robust and consistent output regardless of their mental model about the agent or language usage patterns. We designed SalsaBot with 3 key aspects in mind: 1) The diversity in users' mental models, 2) The diversity in user commands, and 3) The high stakes of planning errors. To deliver robust and consistent performance, we propose a novel construct called *macros*, a high-level action that encompasses all the essential ingredients for achieving a certain goal, *e.g.*, picking up some object from a receptacle. Macros provide a powerful and flexible tool for robust and generalizable agent planning that caters to different levels of abstraction. Furthermore, we modularized our system to ensure that new features can easily be added, existing features can be extended, and allow separately trainable mod-

ules/features.

2. Approach

Our overall approach is shown in Figure 1. When user input is received, our SalsaBot performs initial processing through our instruction understanding and planning (IUP) module, which performs *language cleaning*, and *action planning*. The intent classification module, first point of entry of action planning module, identifies the responsible module to execute the user's intent.

SalsaBot categorizes intents into two types: *macros*, which abstract a sequence of actions, and *primitive actions*, which allow users to directly control the agent's actions. Because a single user utterance can contain a mixture of multiple intents, SalsaBot can detect and execute multiple intents simultaneously. Each intent is handled by our action planning module. Depending on the intent type, the action planning module invokes different sub-modules to execute the intent. Some intents require an action sequence (*e.g.*, macros) or dialogue with the users (*e.g.*, object disambiguation). In all cases, our response generation module generates appropriate responses. Furthermore, we use the response generation module to provide users with constructive feedback on failed actions and suggest suitable actions based on the environmental context.

Intent Classification: SalsaBot classifies intent into two types: primitive actions and macros. In order to do that, we elect a neural-symbolic approach. We first pass user utterances through a fine-tuned version of M-Track [3] which is trained only to output a set of actions that exist in the dataset. We add an additional output action that is labeled macro, which is used to trigger our macro module. Other primitive actions gets executed directly by the model.

Macro is a pivotal module in SalsaBot aimed at enhancing robustness, generalizability, and user experience. This module plays a critical role in bolstering the robustness of our action planning by leveraging pre-defined action sequences to guarantee that the requisite conditions for action execution are fulfilled. Moreover, it offers considerable potential for future development, thereby affording a high degree of extendibility. As illustrated in Figure 2, the planning model processes user utterance to generate a sequence of actions and user intents that are subsequently passed on to the macro module. This complementary component serves to verify whether the current environment and user intents trigger macro generation. In the context of our example, the action of *Repair* is considered a valid macro action. Subsequently, the condition checker examines whether *Holding* condition is satisfied. If the condition is not sufficiently met, *Macro* module generates a series of actions to fulfill the condition. Thereafter, the new metadata is submitted to the condition checker for reassessment. Once the *Holding* condition is met, our *Macro* module generates a pre-defined series of actions to replace the previously planned actions. In the event that a user prompts the presence of a *Seen* object, a Goto action is appended to the action sequence. Finally, with the aid of the condition checker, a comprehensive action query is sent to the arena to enable the user to complete all necessary steps required to execute the *Repair* action.

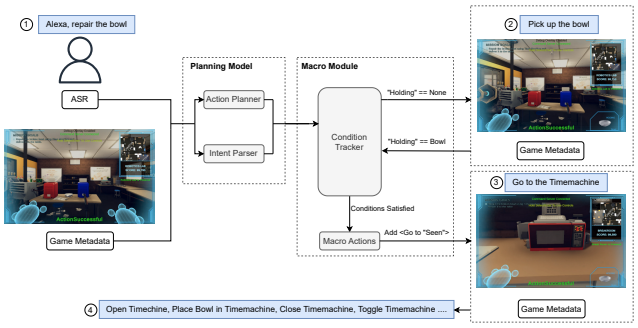

Figure 2. Overview of the macro workflow.

Model	Sucess Rate	Average Action Length
Arena Baseline [1]	34.20	8.82
SalsaBot	34.88	7.54

Table 1. **Task Success Rate on Arena Benchmark Test Split**

Object Memory: SalsaBot is equipped with a object memory module, which is an internal knowledge base of all the objects that it has seen during it's trajectory. The object memory module enables SalsaBot to efficiently go back to an object or an area that it has visited before without needing to explore again.

Response Generation: We have developed a response generation module that generates responses by selecting the pre-written templates and filling the slots in the templates.

3. Experiments

We develop SalsaBot (Figure 1) on the Arena framework [1] provided by Amazon Alexa. The dataset contains 7983 training episodes and 1149 test episodes over 6 scenes and 12 different task types. Each scene contains multiple rooms and the agent has to navigate between rooms to complete the task. Arena framework supports 15 action types: 4 navigation and 11 interaction. Apart from supporting primitive actions, Arena supports argument-based navigation, which the agent goes to an object (argument is a mask) or a room (argument is a room name) specified the argument. In addition to exceeding the baseline success rate, we demonstrate that our approach is much more effective in executing actions due to our macro and object memory module. We were able to decrease the average number of actions by more than 1 across all game sessions, resulting in a more efficient agent.

References

[1] Qiaozi Gao, Govind Thattai, Xiaofeng Gao, Suhaila Shakiah, Shreyas Pansare, Vasu Sharma, Gaurav Sukhatme, Hangjie Shi, Bofei Yang, Desheng Zheng, et al. Alexa arena: A user-centric interactive platform for embodied ai. *arXiv preprint arXiv:2303.01586*, 2023. 1, 2

[2] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[3] Chan Hee Song, Jihyung Kil, Tai-Yu Pan, Brian M. Sadler, Wei-Lun Chao, and Yu Su. One step at a time: Long-horizon vision-and-language navigation with milestones. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15461–15470, 2022. 2