# RAG-Modulo: Solving Sequential Tasks using Experience, Critics, and Language Models

Abhinav Jain, Chris Jermaine\*, Vaibhav Unhelkar\*

Abstract—Large language models (LLMs) have recently emerged as promising tools for solving challenging robotic tasks, even in the presence of action and observation uncertainties. Recent LLM-based decision-making methods (also referred to as LLM-based agents), when paired with appropriate critics, have demonstrated potential in solving complex, long-horizon tasks with relatively few interactions. However, most existing LLM-based agents lack the ability to retain and learn from past interactions—an essential trait of learning-based robotic systems. We propose RAG-Modulo, a framework that enhances LLM-based agents with a memory of past interactions and incorporates critics to evaluate the agents' decisions. The memory component allows the agent to automatically retrieve and incorporate relevant past experiences as in-context examples, providing context-aware feedback for more informed decisionmaking. Further by updating its memory, the agent improves its performance over time, thereby exhibiting learning. Through experiments in the challenging BabyAI and AlfWorld domains, we demonstrate significant improvements in task success rates and efficiency, showing that the proposed RAG-Modulo framework outperforms state-of-the-art baselines.

## I. Introduction

Solving goal-driven sequential tasks is a core problem in robotics, with a wide array of challenges [1], [2], [3], [4], [5], [6]. Due to imperfect actuation, real-world robots operate in stochastic environments. Their sensors often provide only a partial view of the surroundings, requiring decision-making under partial observability and limited knowledge of the world model. To reduce the programming burden for end-users, even complex, long-horizon tasks are frequently defined by sparse reward functions or natural language descriptions of the robot's goal.

Various paradigms and corresponding methods have been explored to address this fundamental challenge [7], [8], [9], [10], [11]. The planning paradigm assumes access to a task model, which is often unavailable in real-world applications. While reinforcement learning can operate without a task model, it typically requires a prohibitively large number of exploratory interactions and significant manual effort for reward design. This challenge is further compounded in partially observable environments, where sparse rewards and safety concerns limit the feasibility of extensive exploration.

To complement these long-standing paradigms, language models have recently emerged as promising tools for solving

Corresponding author: abhinav.jain@rice.edu

All authors are affiliated with the Department of Computer Science, Rice University, Houston, TX. This work was supported in part by the NSF and Rice University funds.

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

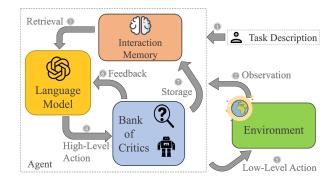


Fig. 1. The RAG-Modulo framework incorporates a language model to generate candidate actions and a set of critics to evaluate them. Importantly, it features mechanisms for storing and retrieving past interactions, which enable learning from experience and improve decision-making over time.

long-horizon tasks in robotics [12], [13], [14], [15], [16], [17], [18], [19]. They can approximate world knowledge [20], [21], [22] and use few-shot reasoning to decompose high-level tasks into mid-level plans [23], [24], [25]. Additionally, they can function as dynamic planners, adjusting their strategies based on environmental feedback, which is especially useful in partially observable settings [17]. Moreover, their performance is shown to improve when integrated with formal systems that evaluate decisions based on criteria such as correctness, executability, and user preferences [26].

Despite their promise, most existing LLM-based decisionmaking methods (also referred to as LLM-based agents) lack the ability to learn from experience. To effectively solve complex, long-horizon tasks, a robotic agent must demonstrate the ability to learn: meaning it should improve its performance over time as it gains more experience in its environment. A prevalent approach to realize such "learning" for LLM-based robotic agents is to tune prompts using in-context examples [27], but this method is constrained by the selection of examples, requires domain knowledge, and demands manual effort. Another option is to fine-tune language models based on past interactions [15], [28], but this approach can be computationally expensive and resource intensive. To address these gaps, we propose RAG-Modulo: a framework which augments a language model with a memory that stores past interactions, retrieving relevant experience at each step of the task to guide robot decision-making.

As shown in Figs. 1 and 2, RAG-Modulo extends the LLM-Modulo framework [26] with memory, where formal verifiers or critics evaluate the feasibility of actions at each step based on criteria like syntax, semantics, and executability.

<sup>\*</sup>Equal advising, authors listed alphabetically.

The interactions, along with feasibility feedback, are stored in memory and retrieved as in-context examples, enabling automatic prompt tuning for future tasks. By leveraging these past interactions, the agent can generalize from its experiences, avoid repeated mistakes, and make more accurate decisions—much like how humans learn from their past errors. In summary, building on the insight of memory-augmented behavior generation, this paper makes three key contributions:

- RAG-Modulo: A framework with LLM-based agents that learns not through back-propagation, but by building up a database of experiences (Interaction Memory) that it then accesses.
- A retrieval mechanism that enable LLM-based agents to access context-aware interactions from memory as in-context examples, automatically tuning prompts and reducing manual effort.
- A suite of experiments on challenging tasks from AlfWorld and BabyAI, where RAG-Modulo outperforms recent baselines and demonstrates improved performance with minimal environment interactions.

## II. PROBLEM FORMULATION

In this section, we formally model the tasks of interest and define the problem, followed by an explanation of how language models can be prompted to function as agents.

## A. Task Model

We focus on object-centric, goal-driven sequential robotic tasks that may involve uncertainties in both actions and observations [29]. More specifically, we denote  $\mathbb{S}_o$  as the set of all possible objects in the robot's environment and  $\mathbb{S}_p$  as the set of object properties. We formally define the task model with the tuple  $(\mathbb{S}, \mathbb{G}, \mathbb{A}, \mathbb{O}, T, R_g, h, \gamma)$ . Given  $\mathbb{S}_o$  and  $\mathbb{S}_p$ , a state  $s \in \mathbb{S}$  is defined as an assignment of object properties. A is the set of low-level physical actions and  $\mathbb{G}$  is the set of all goals. A goal  $g \in \mathbb{G}$  is the natural language description of the goal state.  $\mathbb{O}$  is the set of observations retrieved from states via an observation function  $O: (\mathbb{S} \times \mathbb{A}) \mapsto \mathbb{O}$ , and  $T: (\mathbb{S} \times \mathbb{A}) \mapsto \mathbb{S}$  is the transition function.  $R_g$  is the goal-conditioned reward function, which = 1 when goal is achieved, else 0. Finally,  $\gamma$  denotes the discount factor and h represents the task horizon.

Following prior work [2], [16], the agent is also equipped with a set of high-level text actions, denoted by  $\mathbb{C}$ . In reinforcement learning (RL) literature, these can be interpreted as macro actions or options [30], [31]. Each action  $c \in \mathbb{C}$  is composed of a function and its corresponding set of arguments, i.e., c = FUNCTION(ARGUMENT), such as OPEN(TYPE.DOOR, COLOR.RED). We assume that the robot can execute this high-level action by breaking it down into a sequence of primitive actions  $(a_1, a_2, \ldots)$ , governed by its low-level policy  $\pi_c$ , until a termination condition  $\beta_c$  is met. For the remainder of the paper, we simply refer to high-level actions as actions.

## B. Problem Statement

We can now formally define the problem statement. Given the initial state,  $s_0$ , generate the shortest sequence of actions  $(c_1, c_2, \ldots, c_t)$  to reach the goal state described as g.

# C. Language Models as Agents

As shown in recent works [16], [17], large language models (LLM) can be prompted at each time step to generate a sequence of actions using the following prompt:

$$PROMPT_t = p_{env}; \{g^k, o^k, c^k\}_{k=1}^K; g; \{o_{1:t-1}, c_{1:t-1}\}; o_t$$

where, at time-step t, the prompt consists of (i) fixed prefix  $p_{env}$  describing the environment; (ii) K in-context examples comprised of goal-observation-action tuples,  $\{g^k, o^k, c^k\}_{k=1}^K$ ; (iii) goal description g; (iv) the history of actions for previously visited states,  $\{o_{1:t-1}, c_{1:t-1}\}$ ; (v) the current observation,  $o_t$ . The in-context examples demonstrate how to solve similar tasks and the history of past interactions provides the language model with context about how the agent has interacted with the environment so far.

#### III. RELATED WORK

In this section, we discuss related methods that utilize language models as agents, use memory components and incorporate retrieval-augmented generation (RAG).

Language Models as Agents. Recent works have explored using language models as agents for solving long-horizon tasks by generating plans [14], [16], [17], [19], [18]. Approaches like ProgPrompt [16], [32] generate static plans offline, which may fail when encountering unforeseen object interactions in a partially observable environment. LLM-Planner-like approaches [17], [19], [18] offer a more online approach, allowing for plan updates if an action fails, but it does not store past successes and failures to guide future decisions. The method in [19] involves a human in the loop to prompt and verify. [18] generates feasible plans but relies on precise model dynamics estimation to assess plan feasibility. More recently, [26] have shown that language models should be coupled with verifiers or critics to generate sound plans. These recent methods have informed our work; however, in contrast to these works, RAG-Modulo stores and retrieves past interactions from memory to inform and improve decision-making.

Learning with Experience. Reinforcement learning agents typically use a replay buffer to store experiences for policy optimization. However, solving complex long-horizon tasks often demands millions of trajectories or environment interactions to learn effectively [1]. In contrast, our approach requires only a few hundred experiences to enable meaningful learning. Very recently, some LLM-based approaches have introduced memory modules that store past experiences and expand as the agent interacts with the environment [33], [34], [35], [36], [37]. These methods store experiences at the skill level, retrieving them when needed, but lack the ability to track past successes and failures at the interaction level. Moreover, they often require multiple LLMs to reason, relabel

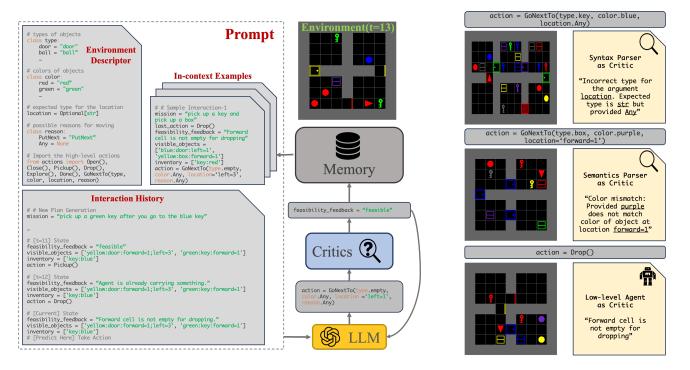


Fig. 2. (Left) The prompt in RAG-Modulo consists of an environment descriptor, a history of past interactions, and in-context examples to guide the LLM in selecting a feasible action. Here, the agent can be carrying a blue key, which it needs to drop before picking up the green key. The retrieved in-context example shows a similar scenario where the agent is unable to drop an object in an occupied cell. Based on this, the agent generates an action to move to an empty cell before completing the task. (Right) Illustration of how each critic provides feedback for the infeasible action shown on top.

and abstract primitive skills into more complex composite ones.

In contrast, the proposed RAG-Modulo stores experiences at the interaction level, removing the need for LLM-guided relabeling, and retrieves these experiences at every decision-making step to offer more informed guidance to the language model. Importantly, our work is complementary to these methods, as they tackle different aspects of continual learning — one focuses on learning library of skills, while the other emphasizes learning from past mistakes and successes.

RAG systems for Robotics. Retrieval Augmented Generation (RAG) systems enhance language model predictions by retrieving relevant information from external databases [38], [39]. For example, [40] employs RAG to collect exemplars for solving sub-tasks with web agents, while [41] retrieves driving experiences from a database for autonomous vehicle planning. In robotics, [42] explores retrieval for deep RL agents, but it does not use LLMs, limiting its adaptability and scalability. [43] employs a policy retriever to extract robotic policies from a large-scale policy memory. In contrast, our approach integrates a RAG system within an LLM-Modulo framework, where past interactions and feedback from critics is stored and continuously expanded. This enables the retrieval of interaction-level experiences, including mistakes and corrections, providing more detailed and contextaware guidance for sequential decision-making.

# IV. PROPOSED APPROACH

We now describe RAG-Modulo, summarized in Alg. 1, which is composed of an LLM, a bank of critics, and an

# **Algorithm 1** RAG-Modulo

```
1: INPUT: (g, h, LLM, \mathbb{M})
 2: t \leftarrow 1
                                              ▶ Initialize the time-step
 3: M \leftarrow \{\}
 4: while t \le h or (g \text{ is satisfied}) do
         o_t \leftarrow \text{Observe the environment}
 5:
          (I^k, c^k) \leftarrow \text{Retrieve interactions from memory (Eq. 2)}
 6:
 7:
         PROMPT_t \leftarrow Construct the prompt (Eq. 1)
         c_t \leftarrow LLM(PROMPT_t)
                                                          ▶ Predict action
 8:
          f_t \leftarrow \text{CHECKFEASIBILITY } (o_t, c_t)
 9:
         if f_t is SUCCESS then \triangleright Keep track of interactions
10:
              M \leftarrow M \cup \{I_t \doteq (g, c_{t-1}, f_{t-1}, o_t), c_t\}
11:
12:
          end if
13: end while
14: if q is satisfied then
         \mathbb{M} \leftarrow M \cup \mathbb{M}
                                                       ▶ Update memory
16: end if
17: return (c_{1:t}, \mathbb{M})
```

interaction memory  $(\mathbb{M})$  coupled with mechanisms for storing and retrieving interaction experience. At each step t of a task specified by natural language goal g and horizon h, RAG-Modulo first retrieves interactions I from the memory that are relevant to the task and current observation  $o_t$ , using them to guide the LLM's decision-making (line 6 in Alg. 1). The LLM selects action  $c_t$  based on this context and receives feedback (lines 8-9) from a bank of critics (Alg. 2). If feasible, the interaction is stored (lines 10-12). Once the

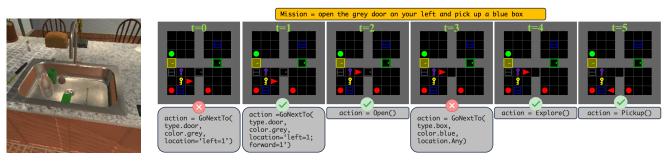


Fig. 3. (Left) AlfWorld Domain where the agent is shown in a household environment. (Right) Execution trace while solving a task from BabyAI. Ticks and Crosses show feasible and infeasible actions respectively.

# Algorithm 2 CHECKFEASIBILITY

```
1: INPUT: o_t, c_t
 2: f_t \leftarrow \text{SUCCESS}, \text{REASON} \leftarrow \text{NONE}
 3: try:
       Parse c_t using \varphi_{syntax}

⊳ Syntax Critic

 4:
                                                  ▶ Semantics Critic
 5:
       Parse c_t using \varphi_{semantics}
 6:
          Execute \pi_{c_{+}}
                                          7:
       until \beta_{c_t} is True
 8:
 9: except Exception as REASON:
       f_t \leftarrow \text{FAILURE}(\text{REASON})
10:
11: return f_t
```

goal is achieved, the interaction memory is updated for future retrieval (lines 14-16), enabling learning from experience.

## A. Critics and Feedback

Informed by the [26], RAG-Modulo includes a bank of critics  $(\varphi_{syntax}, \varphi_{semantics}, \varphi_{low-level})$  who provide feedback on actions selected by the LLM. F denotes the set of feedbacks described in natural language. The syntax parser  $arphi_{syntax}: \mathbb{C} \mapsto \mathbb{F}$  returns feedback based on syntactical correctness. It ensures that the LLM's response adheres to the grammar rules of the environment. The semantics parser  $\varphi_{semantics}: (\mathbb{C} \times \mathbb{O}) \mapsto \mathbb{F}$  returns feedback based on semantic correctness. It verifies that the predicted action is meaningful and logically consistent with the current observation o, e.g., ensuring the agent has the correct key before opening a door. The low-level policy critic  $\varphi_{low-level}$  :  $(\mathbb{C} \times \mathbb{O}) \mapsto \mathbb{F}$  checks if c is executable from o. It runs the execution using  $\pi_c(o)$  until  $\beta_c(o)$  is satisfied. For example, while traversing a path it can determine if an obstacle is encountered. As summarized in Alg. 2, each critic  $\varphi$ , either returns SUCCESS or FAILURE along with the corresponding REASON. This mimics how programmers receive feedback from compilers during debugging. We now formally define the overall feasibility feedback  $f \in \mathbb{F}$  as a function of the feedback from all critics:

$$f = \begin{cases} \text{SUCCESS}, & \text{if } \varphi_{syntax} \land \varphi_{semantics} \land \varphi_{low-level} \\ \text{FAILURE(REASON)}, & \text{otherwise} \end{cases}$$

Given the feedback, the prompt has the following structure:

$$PROMPT_t = \tag{1}$$

$$p_{env}; \{g^k, c_{-1}^k, f^k, o^k, c^k\}_{k=1}^K; g; (o_{1:t-1}, c_{1:t-1}, f_{1:t-1}); o_t\}$$

where, in-context examples and history now include the previous action  $c_{-1}$  and its feasibility feedback.

# B. Interaction Memory, Storage and Retrieval

RAG-Modulo considers a database of past interactions representing the agent's memory  $\mathbb{M}$  of solving prior tasks and their outcomes. We represent such interaction with the tuple (I,c), where  $I=(g,c_{-1},f,o)$ . Formally, the memory includes a set of interactions  $\mathbb{M}=\{(I^1,c^1),\ldots,(I^m,c^m)\}$ , where m represents the memory size.

**Retrieval.** At every decision-making step of a given task, RAG-Modulo retrieves from the memory the top-K most relevant interactions  $\{I^k,c^k\}_{k=1}^K$  that resemble the current task and situation and uses them as in-context examples as shown in Fig. 2. Formally, this is represented as:

$$I_{1:K} = \underset{I \in \mathbb{M}}{\operatorname{argmax}_{K}} \cos(e(I_{t}), e(I))$$
 (2)

 $\operatorname{argmax}_K$  returns the top-K samples from the memory that have the highest cosine similarity with  $I_t$ . e(I) represents the fixed-size embedding of I generated by the encoder model e. As detailed in Sec. V, we use OpenAI's TEXT-EMBEDDING-3-LARGE [44] as the encoder model e for realizing RAG-Modulo in our experiments.

**Storage.** For every successfully completed task,  $\{g, c_{1:h}, f_{1:h}, o_{1:h}, c_{1:h}\}$ , RAG-Modulo fills the memory with its interactions  $(I_t, c_t)$  for which the current option  $c_t$  is always feasible (i.e.  $f(c_t) = \text{SUCCESS}$ ). Thus, every stored tuple is a successful interaction that includes rectifications when  $f_{t-1} = \text{FAILURE}$ , which can be used by the LLM when planning future actions.

## V. EXPERIMENTAL SETUP

We evaluate the performance of RAG-Modulo in AlfWorld [2] and BabyAI [1], [15] benchmarks, depicted in Fig. 3. These benchmarks include several features representative of challenges in robot decision-making, thereby making them suitable benchmarks for evaluating RAG-Modulo. For instance, both benchmarks include a suite of sequential tasks that need to be performed by situated agents. The

| Approach             |                                | BabyAI-Synth                        |                  | BabyAI-BossLevel                |                                 |                                  |  |
|----------------------|--------------------------------|-------------------------------------|------------------|---------------------------------|---------------------------------|----------------------------------|--|
|                      | SR ↑                           | InExec ↓                            | Len ↓            | SR ↑                            | InExec ↓                        | Len ↓                            |  |
| Expert               | 0.96                           | 0.79                                | 8.34             | 0.98                            | 0.43                            | 8.41                             |  |
| ProgPrompt           | $0.24 \pm 0.08$                |                                     |                  | $0.11 \pm 0.06$                 |                                 |                                  |  |
| LLM-Planner          | $\textbf{0.48}\pm\textbf{0.1}$ | $12.02\pm2.17$                      | $14.72\pm2.13$   | $0.24 \pm 0.08$                 | $13.98 \pm 1.48$                | $16.16\pm1.35$                   |  |
| Ours                 | $\textbf{0.48}\pm\textbf{0.1}$ | $\textbf{5.18} \!\pm \textbf{1.18}$ | $14.82 \pm 2.14$ | $\textbf{0.57}\pm\textbf{0.10}$ | $\textbf{3.74}\pm\textbf{0.78}$ | $\textbf{12.48}\pm\textbf{1.49}$ |  |
|                      |                                |                                     |                  |                                 |                                 |                                  |  |
|                      |                                | AlfWorld-Seen                       |                  |                                 | AlfWorld-Unseen                 | 1                                |  |
|                      | SR ↑                           | AlfWorld-Seen<br>InExec ↓           | Len ↓            | SR ↑                            | AlfWorld-Unseer<br>InExec ↓     | Len ↓                            |  |
| Expert               | SR ↑ 0.82                      |                                     | Len ↓<br>18.18   | SR ↑ 0.81                       |                                 |                                  |  |
| Expert<br>ProgPrompt |                                | InExec ↓                            | •                | •                               | InExec ↓                        | Len ↓                            |  |
|                      | 0.82                           | InExec ↓                            | •                | 0.81                            | InExec ↓                        | Len ↓                            |  |

Table I. Baseline comparison on BabyAI (top) and AlfWorld (bottom) environments. ↑ denotes higher is better. ↓ denotes lower is better. For a fair few-shot comparison, each approach uses 10 in-context examples in BabyAI-Synth and 5 in the remaining environments. The best performing approach is shown in **Bold**. Error bars are computed using bootstrapped sampling with 10k trials. (−) indicates the metric is not applicable for the approach.

environments are partially observable to the agent, requiring the agent to explore, navigate and interact with objects to complete tasks described in natural language. Solving these tasks require reasoning over long-horizon in presence of a sparse reward signal, which is challenging for planning, RL and LLM-based decision-making algorithms [1], [2], [15].

Tasks. AlfWorld offers a diverse set of household tasks across various difficulty levels. We conduct experiments using the seen and UNSEEN validation sets, which include 140 and 132 task instances, respectively. The SEEN set is designed to measure in-distribution generalization, whereas the unseen set measures out-of-distribution generalization. BabyAI, a 2D grid world environment, features 40 levels of varying complexity. We focus on the Synth and BossLevel levels. The Synth level includes single-step instructions, such as "pick up a ball" or "go to the red key," while the BossLevel provides more complex, multi-step instructions, such as "put the yellow ball next to the purple ball, then open the purple door." Each level contains 100 evaluation task instances.

**Prompt Design.** We represent robot decisions as Python programs [16]. Fig. 2 illustrates our prompt. The high-level actions are imported as Python functions. Each action is further defined with the types of arguments it requires. Finally, each argument type is defined as a class whose attributes represent the environment objects and their attributes. The interaction at each step is represented by key variables like feasibility\_feedback, visible\_objects and inventory. We task the LLM to predict the next action as the value of the variable action.

**Language Model.** We use GPT-40 [45] as the large language model LLM for generating actions and OpenAI's TEXT-EMBEDDING-3-LARGE [44] as the embedding model e for encoding instructions into 3072 dimensional vectors. Greedy decoding is applied with a maximum token limit of 200 for the LLM-Planner and 50 for the other approaches. The horizon (h) for high-level actions is set to 30 for AlfWorld, 20 for BabyAI-BossLevel, and 25 for BabyAI-Synth.

Baselines. We consider the following baselines for compar-

ison, each using language models as high-level planners: (i) ProgPrompt [16] is a powerful static planner for robotic tasks that generates a complete plan at the start of a task and uses assertion checks to ground the plan to the current state. It is representative of LLM-based agents that do not involve memory or learning from experience. (ii) LLM-Planner [17] is a method that employs grounded replanning, dynamically updating the plan throughout the task. It is a representative approach of more recent LLMbased agents that also utilize retrieval-augmented generation; however, in a different manner than that of RAG-Modulo. For each environment, all baselines have access to 100 training tasks with expert-provided demonstrations. We initialize the memory in RAG-Modulo using these expert demonstrations. We refer to the initial memory as prior experience, which is updated online based on experience of solving new tasks.

**Metrics.** To measure the decision-making performance, we consider three evaluation metrics. (i) Success Rate (SR) measures the fraction of tasks that the planner completed successfully. (ii) Average In-Executability (InExec) is the average number of selected actions that cannot be executed in the environment. (iii) Average Episode Length (Len) is the average number of planning actions that are required to complete a given task. As ProgPrompt is an offline approach, (InExec) and (Len) metrics are not applicable for it.

## VI. RESULTS AND DISCUSSION

How does RAG-Modulo compare against other LLM as Agents baselines? In Table I, we report comparison with the baselines. RAG-Modulo demonstrates a higher success rate than ProgPrompt across both domains. This can be attributed to ProgPrompt's lack of memory and critics, which means it does not benefit from interactive learning. By interacting with the environment and retrieving relevant experience, RAG-Modulo enables more informed decision-making.

RAG-Modulo also outperforms LLM-Planner in terms of success rate, in-executability, and average episode length. Notably, the success rate improvements range from +0.33

|                                 | BabyAI-Synth                     |                                 |                  | BabyAI-BossLevel |                                 |                  |
|---------------------------------|----------------------------------|---------------------------------|------------------|------------------|---------------------------------|------------------|
|                                 | SR ↑                             | InExec ↓                        | Len ↓            | SR ↑             | InExec ↓                        | Len ↓            |
| RAG-Modulo                      | $0.48 \pm 0.1$                   | $5.18 \pm 1.18$                 | $14.82 \pm 2.14$ | $0.57\pm0.1$     | $\textbf{3.74}\pm\textbf{0.78}$ | $12.48 \pm 1.49$ |
| with trajectory-level retrieval | $\textbf{0.50}\pm\!\textbf{0.1}$ | $5.30 \pm 0.93$                 | $15.42\pm2.04$   | $0.52 \pm 0.1$   | $4.22 \pm 0.76$                 | $13.24\pm1.43$   |
| without prior experience        | $0.44 \pm 0.1$                   | $\textbf{4.67}\pm\textbf{0.88}$ | $15.92\pm2.05$   | $0.54 \pm 0.1$   | $4.68 \pm 0.88$                 | $13.16\pm1.48$   |
| without memory                  | $0.43 \pm 0.1$                   | $6.72 \pm 1.13$                 | $16.23\pm2.03$   | $0.37 \pm 0.09$  | $6.07 \pm 0.99$                 | $14.48\pm1.47$   |

Table II. Ablating different components of RAG-Modulo.  $\uparrow$  denotes higher is better.  $\downarrow$  denotes lower is better. The best performing approach is shown in **Bold**. Error bars are computed using bootstrapped sampling with 10k trials. The first row presents the results of the complete RAG-Modulo framework. The second row corresponds to a variant of RAG-Modulo with an alternate retrieval function, which retrieves the most similar task trajectory. The third row shows performance of RAG-Modulo when starting with no prior experience. The last row represents a variant that does not involve a memory component.he tuple  $(g^k, c^k_{-1}, f^k, o^k)$ .

to +0.37 in more challenging environments like BabyAI-BossLevel and AlfWorld-Unseen. Additionally, RAG-Modulo has lower in-executability (approx. 7 lower in Synth and 16 in Alfworld-Seen), and achieves shorter average episode lengths. While both systems are interactive and utilize retrieval-augmented generation, the key advantage of RAG-Modulo is the memory of past interactions that includes critics' feedback. By leveraging this memory, RAG-Modulo can avoid infeasible actions and accomplish tasks with fewer steps. Additionally, the lower episode length achieved by the RAG-Modulo facilitates a reduction in the overall cost of using LLMs, such as API expenses for closed-source models.

What is the optimal number of interactions K to use as in-context examples? We ablate the number of interactions retrieved from memory and evaluate performance on BabyAI environments. The results reported in Fig. 4 show that the success rate improves as K increases, peaking at K=5 for BossLevel and K = 10 for SynthLevel, before beginning to decline. Similarly, in Fig. 5 we observed that in-executability and average episode length decrease initially but start to rise as K continues to grow. The initial boost in performance can be attributed to the inclusion of more informative interactions, enhancing the LLM's decision-making capabilities. The subsequent decline likely stems from the LLM's sensitivity to irrelevant or noisy context [46], [47]. As K increases, the chance of introducing less relevant or low-quality interactions also rises, which can distract the model and degrade its output quality [48]. These trends suggest retrieving a modest number of interactions (between 5 and 10) while solving tasks using the RAG-Modulo framework.

How does the choice of retrieval function affect performance? We examine how retrieving interactions at different levels of granularity impacts performance. Specifically, we compare against an ablation of our approach that utilizes a trajectory-level retrieval function. This ablation first identifies the most relevant task in the memory by computing the cosine similarity between goals, and then extract top K interactions from that task's trajectory. We represent the performance of this variable in the second row of Table II. We observe that retrieving at the interaction level generally yields better results, with lower in-executability and shorter episode lengths, while maintaining similar or higher success rates across both BabyAI domains. This suggests that retrieving interactions from a diverse set of tasks provides

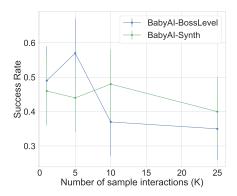


Fig. 4. Success Rate as a function of K

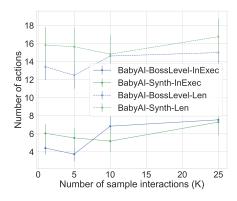


Fig. 5. In-Executability and Episode Length as a function of K

the language model with richer information than simply retrieving interactions from the most relevant single task.

How does the presence of memory affect performance? To study the role of memory, we consider a variant of the proposed approach that does not include any interaction memory. This variant is representative of the LLM-Modulo framework [26], which includes interaction and critics but no mechanisms for storage or retrieval of experience. As reported in the last row of Table II, completely removing the memory component leads to a significant drop in performance, with a 0.20 decrease in success rate on BabyAI-BossLevel and an increase in average episode length by 1.4 to 2.0 steps. This demonstrates that storing and retrieving past interactions and feedback significantly improves the decision-making capabilities of the critic-aided language model.

How does prior experience affect performance? Lastly, in the third row of Table II, we report results of RAG-Modulo when it is not seeded with any prior expert-generated experience. Unsurprisingly, we find that prior experience generally helps in sequential decision-making. Interestingly, even starting our approach with an empty memory (third row, Table II) still outperforms the variant that does not include a memory component (fourth row, Table II), as the agent can gradually collect experiences of successes and failures, allowing it to learn and improve its decision-making.

# VII. CONCLUSION

This paper introduces RAG-Modulo, a framework for solving sequential decision-making tasks by providing LLM-based agents memory of past interactions. Extending the recent LLM-Modulo framework, RAG-Modulo not only incorporates critic feedback regarding the feasibility of generated actions but also enables agents to remember successes and mistakes and learn from them. RAG-Modulo demonstrates superior performance on the challenging BabyAI and AlfWorld benchmarks, achieving higher success rates while requiring fewer actions to complete sequential tasks.

In future work, we plan to utilize RAG-Modulo to solve tasks in other environments involving physical robots, such as FurnitureBench with the Panda robot [6]. We also see potential in integrating RAG-Modulo with existing continual learning frameworks, such as BOSS and Voyager [33], [34], to enable learning from experience at multiple layers of abstractions: namely, skills and interactions. Another avenue is to explore tunable retrieval models that can anticipate future needs to further enhance the agent's performance [49], [50]. Finally, we are interested in studying how RAG-Modulo can enhance end-user programming of complex robot behaviors by leveraging user commands, experience and critiques.

## REFERENCES

- M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio, "Babyai: A platform to study the sample efficiency of grounded language learning," arXiv preprint arXiv:1810.08272, 2018.
- [2] M. Shridhar, X. Yuan, M.-A. Côté, Y. Bisk, A. Trischler, and M. Hausknecht, "Alfworld: Aligning text and embodied environments for interactive learning," arXiv preprint arXiv:2010.03768, 2020.
- [3] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10740–10749.
- [4] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu, "robosuite: A modular simulation framework and benchmark for robot learning," arXiv preprint arXiv:2009.12293, 2020.
- [5] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan, "Vima: Robot manipulation with multimodal prompts," 2023.
- [6] M. Heo, Y. Lee, D. Lee, and J. J. Lim, "Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation," in *Robotics: Science and Systems*, 2023.
- [7] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.
- [8] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

- [9] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of* control, robotics, and autonomous systems, vol. 3, no. 1, pp. 297–330, 2020.
- [10] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, no. 1, pp. 265–293, 2021.
- [11] B. Singh, R. Kumar, and V. P. Singh, "Reinforcement learning in robotic applications: a comprehensive survey," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 945–990, 2022.
- [12] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," arXiv preprint arXiv:2210.03629, 2022.
- [13] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, et al., "Do as i can, not as i say: Grounding language in robotic affordances," arXiv preprint arXiv:2204.01691, 2022.
- [14] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International conference on machine learning*. PMLR, 2022, pp. 9118–9147.
- [15] T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P.-Y. Oudeyer, "Grounding large language models in interactive environments with online reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 3676–3713.
- [16] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2023, pp. 11523–11530.
- [17] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "Llm-planner: Few-shot grounded planning for embodied agents with large language models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 2998–3009.
- [18] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *Autonomous Robots*, vol. 47, no. 8, pp. 1345–1365, 2023.
- [19] S. H. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, "Chatgpt for robotics: Design principles and model abilities," *IEEE Access*, 2024.
- [20] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?" arXiv preprint arXiv:1909.01066, 2019.
- [21] A. Roberts, C. Raffel, and N. Shazeer, "How much knowledge can you pack into the parameters of a language model?" arXiv preprint arXiv:2002.08910, 2020.
- [22] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know?" *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 423–438, 2020.
- [23] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in neural information* processing systems, vol. 35, pp. 22199–22213, 2022.
- [24] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al., "Chain-of-thought prompting elicits reasoning in large language models," Advances in neural information processing systems, vol. 35, pp. 24824–24837, 2022.
- [25] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le, et al., "Least-to-most prompting enables complex reasoning in large language models," arXiv preprint arXiv:2205.10625, 2022.
- [26] S. Kambhampati, K. Valmeekam, L. Guan, K. Stechly, M. Verma, S. Bhambri, L. Saldyt, and A. Murthy, "Llms can't plan, but can help planning in llm-modulo frameworks," arXiv preprint arXiv:2402.01817, 2024.
- [27] M. G. Arenas, T. Xiao, S. Singh, V. Jain, A. Ren, Q. Vuong, J. Varley, A. Herzog, I. Leal, S. Kirmani, et al., "How to prompt your robot: A promptbook for manipulation skills with code as policies," in 2024 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2024, pp. 4340–4348.
- [28] V. Pallagani, B. Muppasani, K. Murugesan, F. Rossi, L. Horesh, B. Srivastava, F. Fabiano, and A. Loreggia, "Plansformer: Generating symbolic plans using transformers," arXiv preprint arXiv:2212.08681, 2022.
- [29] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox, "Mimicgen: A data generation system for scalable

- robot learning using human demonstrations," in Conference on Robot Learning. PMLR, 2023, pp. 1820–1864.
- [30] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," Artificial intelligence, vol. 112, no. 1-2, pp. 181–211, 1999.
- [31] C. Daniel, H. Van Hoof, J. Peters, and G. Neumann, "Probabilistic inference for determining options in reinforcement learning," *Machine Learning*, vol. 104, pp. 337–357, 2016.
- [32] R. Hazra, P. Z. Dos Martires, and L. De Raedt, "Saycanpay: Heuristic planning with large language models using learnable domain knowledge," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 18, 2024, pp. 20123–20133.
- [33] J. Zhang, J. Zhang, K. Pertsch, Z. Liu, X. Ren, M. Chang, S.-H. Sun, and J. J. Lim, "Bootstrap your own skills: Learning to solve new tasks with large language model guidance," arXiv preprint arXiv:2310.10021, 2023.
- [34] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv:2305.16291*, 2023.
- [35] G. Tziafas and H. Kasaei, "Lifelong robot library learning: Bootstrapping composable and generalizable skills for embodied control with language models," in 2024 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2024, pp. 515–522.
- [36] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," Advances in Neural Information Processing Systems, vol. 36, 2024.
- [37] A. Zhao, D. Huang, Q. Xu, M. Lin, Y.-J. Liu, and G. Huang, "Expel: Llm agents are experiential learners," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 17, 2024, pp. 19632–19642.
- [38] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave, "Atlas: Fewshot learning with retrieval augmented language models," *Journal of Machine Learning Research*, vol. 24, no. 251, pp. 1–43, 2023.
- [39] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, et al., "Improving language models by retrieving from trillions of

- tokens," in *International conference on machine learning*. PMLR, 2022, pp. 2206–2240.
- [40] M. Kim, V. Bursztyn, E. Koh, S. Guo, and S.-w. Hwang, "Rada: Retrieval-augmented web agent planning with llms," in *Findings of the Association for Computational Linguistics ACL 2024*, 2024, pp. 13511–13525.
- [41] X. Dai, C. Guo, Y. Tang, H. Li, Y. Wang, J. Huang, Y. Tian, X. Xia, Y. Lv, and F.-Y. Wang, "Vistarag: Toward safe and trustworthy autonomous driving through retrieval-augmented generation," *IEEE Transactions on Intelligent Vehicles*, 2024.
- [42] A. Goyal, A. Friesen, A. Banino, T. Weber, N. R. Ke, A. P. Badia, A. Guez, M. Mirza, P. C. Humphreys, K. Konyushova, et al., "Retrievalaugmented reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 7740–7765.
- [43] Y. Zhu, Z. Ou, X. Mou, and J. Tang, "Retrieval-augmented embodied agents," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17985–17995.
- [44] OpenAI, "https://openai.com/index/new-embedding-models-and-api-updates/," 2024.
- [45] \_\_\_\_, "https://openai.com/index/hello-gpt-4o," 2024.
- [46] F. Shi, X. Chen, K. Misra, N. Scales, D. Dohan, E. H. Chi, N. Schärli, and D. Zhou, "Large language models can be easily distracted by irrelevant context," in *International Conference on Machine Learning*. PMLR, 2023, pp. 31210–31227.
- [47] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2024.
- [48] T. Merth, Q. Fu, M. Rastegari, and M. Najibi, "Superposition prompting: Improving and accelerating retrieval-augmented generation," arXiv preprint arXiv:2404.06910, 2024.
- [49] S. Weijia, M. Sewon, Y. Michihiro, S. Minjoon, J. Rich, L. Mike, and Y. Wen-tau, "Replug: Retrieval-augmented black-box language models," *ArXiv*: 2301.12652, 2023.
- [50] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig, "Active retrieval augmented generation," arXiv preprint arXiv:2305.06983, 2023.