

ARMAP: AUTONOMOUS AGENTS FROM AUTOMATIC REWARD MODELING AND PLANNING

Anonymous authors

Paper under double-blind review

ABSTRACT

Large language models (LLMs) have demonstrated remarkable capabilities across a range of text-generation tasks. However, LLMs still struggle with problems requiring multi-step decision-making and environmental feedback, such as online shopping, scientific reasoning, and mathematical problem-solving. Unlike pure text data, collecting large-scale decision-making data is challenging. Moreover, many powerful LLMs are only accessible through APIs, which hinders their fine-tuning for agent tasks due to cost and complexity. To address LLM agents' limitations, we propose a framework that can automatically learn a reward model from the environment without human annotations. This model can be used to evaluate the action trajectories of LLM agents and provide heuristics for task planning. Specifically, our approach involves employing one LLM-based agent to navigate an environment randomly, generating diverse action trajectories. Subsequently, a separate LLM is leveraged to assign a task intent and synthesize a negative response alongside the correct response for each trajectory. These triplets (task intent, positive response, and negative response) are then utilized as training data to optimize a reward model capable of scoring action trajectories. This reward model can be integrated with LLM-based agents and various planning algorithms to enhance task-solving performance. The effectiveness and generalizability of our framework are demonstrated through evaluations conducted on different agent benchmarks. In conclusion, our proposed framework represents a significant advancement in enhancing LLM agents' decision-making capabilities. By automating the learning of reward models, we overcome the challenges of data scarcity and API limitations, potentially revolutionizing the application of LLMs in complex and interactive environments. This research paves the way for more sophisticated AI agents capable of tackling a wide range of real-world problems requiring multi-step decision-making.

1 INTRODUCTION

Developing AI agents capable of perceiving environments, understanding instructions, and acting to accomplish a wide range of tasks in interactive settings (Brooks, 1986) have many real-world applications, including virtual human assistants (Reed et al., 2022; Casheekar et al., 2024), business process management (Kirchdorfer et al., 2024), and robotic process automation (Rana et al., 2023; Ahn et al., 2022; Palo et al., 2023).

The recent advent of large generative models has revolutionized numerous applications, such as question answering (Rajpurkar et al., 2016), text summarization (Hermann et al., 2015), and multi-modal understanding (Chen et al., 2015; Goyal et al., 2017; Yu et al., 2016). However, while these models excel in text comprehension and generation tasks, their performance in decision-making scenarios—such as online shopping and scientific reasoning falls relative short of human capabilities. This disparity likely stems from the nature of the training data. Large generative models are typically pre-trained on readily available image and text corpora from the internet. In contrast, trajectory data for agent tasks, which require multi-step interaction with the environment, is more challenging to collect and does not naturally occur on the internet. Furthermore, current state-of-the-art commercial Language Learning Models (LLMs), such as GPT-4V (OpenAI et al., 2024) and Gemini (Reid et al., 2024), often provide only limited APIs for general users. This restriction renders it either infeasible

054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100
101
102
103
104
105
106
107



Figure 1: In Fig. 1 (a), we show that it is difficult for LLM agents to generate multi-step plans in an interactive environment to achieve the instruction goal. However, it is relatively easy for an LLM to learn a reward model that can evaluate whether the trajectories meet the task instructions, as shown in Fig. 1 (b). In Fig. 1 (c), we show that a learned reward model can be used to guide the default policy models to improve action planning.

or cost-prohibitive to fine-tune these models for specific agent tasks, further impeding progress in this field.

Previous studies have explored the development of autonomous agents for decision-making tasks using large language models (LLMs). Early research (Yao et al., 2023a; Zheng et al., 2024; Deng et al., 2024) utilized instruction prompts with few-shot examples to direct LLMs in handling various agent tasks. These methods do not require task-specific fine-tuning but have shown limited performance on benchmarks requiring interaction with environments and precise action prediction. A different research direction involves collecting human preference data (Hong et al., 2023) or distilling trajectory data from advanced commercial LLM APIs (Zeng et al., 2023; Deng et al., 2024) and fine-tuning smaller open-source LLMs to create new policy models for agent tasks. However, this distillation process relies on advanced pre-trained agent models for trajectory data extraction, which are often unavailable, expensive, or subject to commercial restrictions. For instance, data from models such as GPT-4 or Gemini cannot be used for commercial purposes.

A fundamental premise of our approach is that, in most agent applications, evaluation is easier than generation (Karp, 1975; Naor, 1996). As illustrated in Fig. 1 (a), **generating** a correct multi-step solution to navigate to the target page is challenging since it needs to predict multiple actions and interact with the environment. However, it is relatively simple to **evaluate** whether the output action trajectories and environment states meet the provided intent to find a "vans sport canvas fashion sneaker". Building on this premise, we suggest that developing a reward model is more feasible than creating a policy model for agent tasks. With an effective reward model, it becomes possible to guide LLMs in planning tasks both effectively and efficiently. For instance, as depicted in Fig. 1 (c), by integrating the reward model with an LLM-based agent and the Monte Carlo Tree Search (MCTS) algorithm (Silver et al., 2017; Coulom, 2006), we can simulate and evaluate the future states of agent tasks, thereby making better decisions for subsequent actions. This approach is analogous to mental simulation (Hegarty, 2004; Lake et al., 2017) in cognitive science, where humans envision the outcomes of potential actions to make better decisions in problem-solving.

While reward models can assist LLM agents in planning, developing these reward models presents significant challenges. Some prior studies have utilized powerful commercial LLM APIs as evaluators for tasks (Kwon et al., 2023a). Although these approaches have demonstrated effectiveness in certain applications, they rely on state-of-the-art LLM models for evaluation, which are often expensive and difficult to scale. In this paper, we introduce an automated method to learn multi-modal reward models without relying on state-of-the-art LLMs for guidance. Furthermore, previous work has not considered integrating the learned reward models with various planning algorithms for problem-solving.

The process of learning the reward model involves three steps. Initially, we utilize an LLM-based agent (e.g., Dubey et al. (2024)) to navigate in the environments, aiming to achieve a randomly

108 proposed intent while collecting extensive action trajectory demonstrations. Subsequently, the LLM
109 model examines the collected trajectories and proposes a refined intent that the sampled trajectories
110 actually accomplish. Additionally, we prompt the LLM to generate negative trajectories that fail
111 to achieve the intended task. Finally, based on the synthetic data (intents, positive trajectories, and
112 negative trajectories) collected, we train a customized reward model using widely adopted vision-
113 language models such as VILA (Lin et al., 2023) to evaluate whether the user’s intent has been
114 fulfilled by the action trajectories. With this automatic reward model, we enhance the performance of
115 LLM-based agents in conjunction with various planning algorithms such as best of n, reflection, and
116 MCTS.

117 In summary, we introduce a novel framework ARMAP (autonomous Agents from automatic Reward
118 Modeling And Planning) for LLM-based agents incorporating an automatic reward model that
119 evaluates task completion, analogous to mental simulation in human cognition. This framework offers
120 several advantages: (1) Effectiveness: It enhances the performance of various LLM agents across
121 different tasks. (2) Flexibility: It eliminates the need for fine-tuning the LLMs themselves and allows
122 for optimization of custom reward targets during inference, enabling more controllable generation.
123 (3). Practicality: The training of the automatic reward model does not rely on labor-intensive labeling
124 or state-of-the-art commercial LLMs, making it more feasible and widely applicable.

125 126 127 2 RELATED WORK

128
129 **LLMs for Agent tasks.** Our research is related to deploying large language models (LLMs) as
130 agents for decision-making tasks in interactive environments (Liu et al., 2023; Zhou et al., 2023;
131 Shridhar et al., 2020; Toyama et al., 2021). Earlier works, such as (Yao et al., 2023a), fine-tuned
132 models like BERT (Devlin et al., 2019) for decision-making in simplified environments, such as
133 online shopping or mobile phone manipulation. With the advent of large language models (Brown
134 et al., 2020; OpenAI et al., 2024), it became feasible to perform decision-making tasks through
135 zero-shot or few-shot in-context learning. To better assess the capabilities of LLMs as agents, several
136 models have been developed (Deng et al., 2024; Xiong et al., 2024; Hong et al., 2023; Yan et al.,
137 2023). Most approaches (Zheng et al., 2024; Deng et al., 2024) provide the agent with observation and
138 action history, and the language model predicts the next action via in-context learning. Additionally,
139 some methods (Zhang et al., 2023; Li et al., 2023; Song et al., 2024) attempt to distill trajectories
140 from state-of-the-art language models to train more effective policy models. In contrast, our paper
141 introduces a novel framework that automatically learns a reward model from LLM agent navigation,
142 using it to guide the agents in making more effective plans.

143 **LLM Planning.** Our paper is also related to planning with large language models. Early re-
144 searchers (Brown et al., 2020) often prompted large language models to directly perform agent tasks.
145 Later, Yao et al. (2022) proposed ReAct, which combined LLMs for action prediction with chain-
146 of-thought prompting (Wei et al., 2022). Several other works (Yao et al., 2023b; Hao et al., 2023;
147 Zhao et al., 2023) have focused on enhancing multi-step reasoning capabilities by integrating LLMs
148 with tree search methods. Our model differs from these previous studies in several significant ways.
149 First, rather than solely focusing on text generation tasks, our pipeline addresses multi-step action
150 planning tasks in interactive environments, where we must consider not only historical input but also
151 multimodal feedback from the environment. Additionally, our pipeline involves automatic learning of
152 the reward model from the environment without relying on human-annotated data, whereas previous
153 works rely on prompting-based frameworks that require large commercial LLMs like GPT-4 (Open-
154 AI et al., 2024) to learn action prediction. Furthermore, ARMAP supports a variety of planning
155 algorithms beyond tree search.

156 **Learning from AI Feedback.** In contrast to prior work on LLM planning, our approach also
157 draws on recent advances in learning from AI feedback (Bai et al., 2022; Lee et al., 2023; Yuan
158 et al., 2024; Sharma et al., 2024; Pan et al., 2024; Koh et al., 2024). These studies initially prompt
159 state-of-the-art large language models to generate text responses that adhere to predefined principles
160 and then potentially fine-tune the LLMs with reinforcement learning. Like previous studies, we also
161 prompt large language models to generate synthetic data. However, unlike them, we focus not on
fine-tuning a better generative model but on developing a classification model that evaluates how well
action trajectories fulfill the intended instructions. This approach is simpler, requires no reliance on

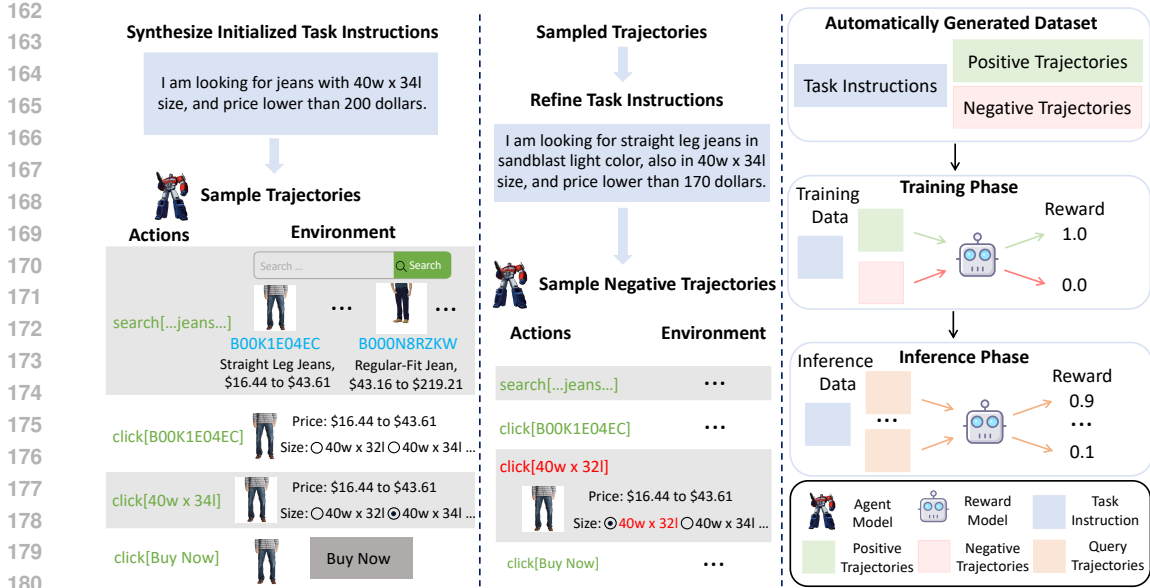


Figure 2: The pipeline of our ARMAR framework. We first generate an initial task instruction using LLMs with in-context learning and sample trajectories aligned with the initial language instructions in the environment. Next, we use the LLM to summarize the sampled trajectories and generate refined task instructions that better match these trajectories. We then modify specific actions within the trajectories to perform new actions in the environment, collecting negative trajectories in the process. Using the refined task instructions, along with both positive and negative trajectories, we train a lightweight reward model to distinguish between matching and non-matching trajectories. The learned reward model can then collaborate with various LLM agents to improve task planning.

state-of-the-art LLMs, and is more efficient. We also demonstrate that our learned reward model can integrate with various LLMs and planning algorithms, consistently improving their performance.

3 MODEL

In this section, we provide a detailed introduction to our framework, autonomous Agents from automatic Reward Modeling And Planning (ARMAR). The framework includes automated reward data generation in section 3.2, reward model design in section 3.3, and planning algorithms in section 3.4.

3.1 BACKGROUND

The planning tasks for LLM agents can be typically formulated as a Partially Observable Markov Decision Process (POMDP): $(\mathcal{X}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T})$, where:

- \mathcal{X} is the set of text instructions;
- \mathcal{S} is the set of environment states;
- \mathcal{A} is the set of available actions at each state;
- \mathcal{O} represents the observations available to the agents, including text descriptions and visual information about the environment in our setting;
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the transition function of states after taking actions, which is given by the environment in our settings.

Given a task instruction $x \in \mathcal{X}$ and the initial environment state $s_0 \in \mathcal{S}$, planning tasks require the LLM agents to propose a sequence of actions $\{a_n\}_{n=1}^N$ that aim to complete the given task, where $a_n \in \mathcal{A}$ represents the action taken at time step n , and N is the total number of actions executed

in a trajectory. Following the n -th action, the environment transitions to state s_n , and the agent receives a new observation o_n . Based on the accumulated state and action histories, the task evaluator determines whether the task is completed.

An important component of our framework is the learned reward model \mathcal{R} , which estimates whether a trajectory h has successfully addressed the task:

$$r = \mathcal{R}(x, h), \quad (1)$$

where $h = \{\{a_n\}_{n=1}^N, \{o_n\}_{n=0}^N\}$, $\{a_n\}_{n=1}^N$ are the actions taken in the trajectory, $\{o_n\}_{n=0}^N$ are the corresponding environment observations, and r is the predicted reward from the reward model. By integrating this reward model with LLM agents, we can enhance their performance across various environments using different planning algorithms.

3.2 AUTOMATIC REWARD DATA GENERATION.

To train a reward model capable of estimating the reward value of history trajectories, we first need to collect a set of training language instructions $\{x_m\}_{m=1}^M$, where M represents the number of instruction goals. Each instruction corresponds to a set of positive trajectories $\{h_m^+\}_{m=1}^M$ that match the instruction goals and a set of negative trajectories $\{h_m^-\}_{m=1}^M$ that fail to meet the task requirements. This process typically involves human annotators and is time-consuming and labor-intensive (Christiano et al., 2017; Rafailov et al., 2024). As shown in Fig. 8 of the Appendix. we automate data collection by using Large Language Model (LLM) agents to navigate environments and summarize the navigation goals without human labels.

Instruction Synthesis. The first step in data generation is to propose a task instruction for a given observation. We achieve this using the in-context learning capabilities of LLMs. The prompt for instruction generation is shown in Fig. 9 of the Appendix. Specifically, we provide some few-shot examples in context along with the observation of an environment state to an LLM, asking it to summarize the observation and propose instruction goals. In this way, we collect a set of synthesized language instructions $\{x_m^{raw}\}_{m=1}^M$, where M represents the total number of synthesized instructions.

Trajectory Collection. Given the synthesized instructions x_m^{raw} and the environment, an LLM-based agent is instructed to take actions and navigate the environment to generate diverse trajectories $\{x_m^{raw}, h_m\}_{m=0}^M$ aimed at accomplishing the task instructions. Here, h_m represents the m -th history trajectory, which consists of N actions $\{a_n\}_{n=1}^N$ and $N + 1$ environment observations $\{o_n\}_{n=0}^N$. Due to the limited capabilities of current LLMs, the generated trajectories h_m may not always align well with the synthesized task instructions x_m . To address this, we ask the LLM to summarize the completed trajectory h_m and propose a refined goal x_m^r . This process results in a set of synthesized demonstrations $\{x_m^r, h_m\}_{m=0}^{M_r}$, where M_r is the number of refined task instructions.

Pairwise Data Construction. To train a reward model capable of distinguishing between good and poor trajectories, we also need trajectories that do not satisfy the task instructions. To create these, we sample additional trajectories that differ from $\{x_m^r, h_m\}$ and do not meet the task requirements by modifying actions in h_m and generating corresponding negative trajectories $\{h_m^-\}$. For clarity, we refer to the refined successful trajectories as $\{x_m, h_m^+\}$ and the unsuccessful ones as $\{x_m, h_m^-\}$. These paired data will be used to train the reward model described in Section 3.3, allowing it to estimate the reward value of any given trajectory in the environment.

3.3 REWARD MODEL DESIGN.

Reward Model Architectures. Theoretically, we can adopt any vision-language model that can take a sequence of visual and text inputs as the backbone for the proposed reward model. In our implementation, we use the recent VILA model (Lin et al., 2023) as the backbone for reward modeling since it has carefully maintained open-source code, show strong performance on standard vision-language benchmarks like (Fu et al., 2023; Goyal et al., 2017; Hudson & Manning, 2019), and support multiple image input.

The goal of the reward model is to predict a reward score to estimate whether the given trajectory (x_m, h_m) has satisfied the task instruction or not, which is different from the original goal of VILA models that generate a series of text tokens to respond to the task query. To handle this problem, we

270 additionally add a fully-connected layer for the model, which linearly maps the hidden state of the
271 last layer into a scalar value.

272 **Optimization Target.** Given the pairwise data that is automatically synthesized from the envi-
273 ronments in Section 3.2, we optimize the reward model by distinguishing the good trajectories
274 (x_m, h_m^+) from bad ones (x_m, h_m^-) . Following standard works of reinforcement learning from human
275 feedback (Bradley & Terry, 1952; Sun et al., 2023b;a), we treat the optimization problem of the
276 reward model as a binary classification problem and adopt a cross-entropy loss. Formally, we have
277

$$278 \mathcal{L}(\theta) = -\mathbf{E}_{(x_m, h_m^+, h_m^-)}[\log \sigma(\mathcal{R}_\theta(x_m, h_m^+) - \mathcal{R}_\theta(x_m, h_m^-))], \quad (2)$$

279 where σ is the sigmoid function and θ are the learnable parameters in the reward model \mathcal{R} . By
280 optimizing this target, the reward model is trained to give higher value scores to the trajectories that
281 are closer to the goal described in the task instruction.
282

283 3.4 PLANNING WITH LARGE VISION-LANGUAGE REWARD MODEL.

284 After getting the reward model to estimate how well a sampled trajectory match the given task
285 instruction, we are able to combine it with different planning algorithms to improve LLM agents’
286 performance. Here, we summarize the typical algorithms we can adopt in this paper.
287

288 **Best of N.** This is a simple algorithm that we can adopt the learned reward model to improve the
289 LLM agents’ performances. We first prompt the LLM agent to generate n different trajectories
290 independently and choose the one with the highest predicted reward score as the prediction for
291 evaluation. Note that this simple method is previously used in natural language generation (Zhang
292 et al., 2024) and we adopt it in the context of agent tasks to study the effectiveness of the reward
293 model for agent tasks.
294

295 **Reflexion.** Reflexion (Shinn et al., 2024) is a planning framework that enables large language
296 models (LLMs) to learn from trial-and-error without additional fine-tuning. Instead of updating
297 model weights, Reflexion agents use verbal feedback derived from task outcomes. This feedback is
298 converted into reflective summaries and stored in an episodic memory buffer, which informs future
299 decisions. Reflexion supports various feedback types and improves performance across decision-
300 making, coding, and reasoning tasks by providing linguistic reinforcement that mimics human
301 self-reflection and learning.

302 **MCTS.** We also consider tree search-based planning algorithms like Monte Carlo Tree Search
303 (MCTS) (Coulom, 2006; Silver et al., 2017) to find the optimal policy. There is a tree structure
304 constructed by the algorithm, where each node represents a state and each edge signifies an action.
305 Beginning at the initial state of the root node, the algorithm navigates the state space to identify action
306 and state trajectories with high rewards, as predicted by our learned reward model.

307 The algorithm tracks 1) the frequency of visits to each node and 2) a value function that records the
308 maximum predicted reward obtained from taking action a in state s . MCTS would visit and expand
309 nodes with either higher values (as they lead to high predicted reward trajectory) or with smaller visit
310 numbers (as they are under-explored). We provide more details in the implementation details and the
311 appendix section.
312

313 4 EXPERIMENTS

314 In this section, we conduct a series of experiments to demonstrate the effectiveness of the proposed
315 framework for agent tasks. First, we evaluate the framework’s performance on standard agent
316 benchmarks (Yao et al., 2023a; Wang et al., 2022; Yao et al., 2023b), detailed in Section 4.2. Next,
317 we show how customizing the reward target during inference allows us to generate more tailored
318 action plans, as described in Section 4.3. Finally, we conduct ablation studies in Section 4.4. Before
319 delving into the experimental results, we provide an overview of our experimental setup.
320

321 4.1 EXPERIMENTAL SETUP

322 **Environments.** We evaluate the ARMAP framework in three different environments:
323

- **Webshop** is a well-known environment for online shopping (Yao et al., 2023a), where the agent must search for and select products on the website to obtain a final result. Following the setup of AgentBench (Liu et al., 2023) for LLM evaluation, we test the model on the validation split, using the default matching reward as the evaluation metric.
- **ScienceWorld** (Wang et al., 2022) is an interactive benchmark designed for embodied science experiments. It places agents in a simulated text-based environment where they must perform elementary science experiments by navigating the environment, manipulating objects, and observing outcomes. The aim is to assess whether AI models can apply scientific knowledge, rather than merely retrieve or assemble information. We evaluate the framework on both seen and unseen splits.
- **Game of 24** is a mathematical game where the agent is given four numbers and must use arithmetic operations (addition, subtraction, multiplication, and division) to make the number 24. For instance, given the input '3, 5, 7, 11,' one possible solution is ' $(7-3)*(11-5) = 24$ '. Following Yao et al. (2023b), we selected 100 challenging puzzles, specifically those indexed from 901 to 1,000, and the performance metric is the success rate across these puzzles. As shown in Fig. 7 of the Appendix, we use the chain-of-thought prompting technique, prompting the LLM agents to output intermediate steps followed by the final answer. Each step of the solution is considered an action.

LLM Setup. Our framework requires LLM models to act as agents, generating synthetic task instructions from the environment along with few-shot examples in the prompt context. We also deploy agents to perform these synthetic tasks in the environment, collecting diverse trajectories for further analysis. In this paper, we primarily use the Llama3-70b-instruct model (Dubey et al., 2024) to synthesize training data for the automatic reward models, as it is open-source, easy to deploy locally, and delivers robust performance. We avoid state-of-the-art commercial models like GPT-4 or Gemini due to their high costs and the complexity of reproducing results caused by frequent model updates, making them less suitable for our research objectives.

To evaluate the performance of various LLM agents, we serve a representative set of LLM APIs locally, balancing model diversity with affordable serving costs. We identify the LLMs by their model family and size. Specifically, these are Llama70B, Llama8B, Mistral7B, and Phi3.8B. We note that these open-source model families are frequently updated, and we provide the current model links in the Appendix A.3. All models can be easily set up using the vLLM library (Kwon et al., 2023b) and a single H100 GPU.

Baselines. We implement our ARMAP framework using different planning algorithms, including Reflexion, Best-of-N, and MCTS, which we denote as **ARMAP-R**, **ARMAP-B**, and **ARMAP-M**, respectively. We limit the maximum number of trajectories our ARMAP can explore to 10 in the ScienceWorld and Webshop environments to systematically evaluate the pipeline’s effectiveness across different LLM agent backbones. We also compare the model with two baselines that do not use reward model guidance: **Sampling** and **Greedy**. For the Game of 24 environment, we follow the setup of a previous study (Yao et al., 2023b) and set the maximum number of explored trajectories to 100. For **Sampling**, we set the model temperature to 1 and sample action trajectories using chain-of-thought prompting (Wei et al., 2023). For **Greedy**, we set the temperature to 0, generating the action sequence with the highest probability. Further implementation details are provided in the Appendix. We will release all the code, model, and data for easy reproduction upon acceptance.

4.2 EFFECTIVENESS FOR REWARD PLANNING.

In this section, we investigate the effectiveness of the framework across different language models (Dubey et al., 2024; Jiang et al., 2023; Abdin et al., 2024) and various planning algorithms. The results are shown in Table 1. Based on the table, we have the following observations. First, our proposed pipeline is effective, as it consistently outperforms the **Sampling** and **Greedy** baselines across different planning algorithms. Additionally, we observe that the average improvement is more significant on weaker models, such as Phi (Abdin et al., 2024) and Mistral-7B (Jiang et al., 2023), compared to stronger models like Llama3-1-70B (Dubey et al., 2024). We believe this is because weaker models explore more low-reward trajectories, providing greater opportunities for the reward model to improve performance.

378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400

Backbone	Algorithms	Webshop	ScienceWorld seen	ScienceWorld unseen	Game24	Average
Llama70B	Sampling	52.0	53.9	50.6	9.6	38.0
	Greedy	50.4	57.2	55.1	6.0	37.5
	ARMAP-R	56.5	59.0	56.7	16.0	43.5
	ARMAP-B	62.0	57.3	57.0	19.0	46.1
	ARMAP-M	66.8	58.2	55.9	24.0	49.3
Llama8B	Sampling	56.4	24.5	20.6	2.0	27.0
	Greedy	57.7	29.9	23.8	2.0	28.9
	ARMAP-R	58.3	31.2	28.0	6.0	31.3
	ARMAP-B	59.3	35.7	28.1	11.0	34.1
	ARMAP-M	60.2	32.5	24.9	9.0	32.6
Mistral7B	Sampling	17.7	18.4	17.1	1.0	12.2
	Greedy	37.2	21.1	19.6	1.0	19.5
	ARMAP-R	54.1	21.7	19.7	2.0	25.6
	ARMAP-B	54.4	24.5	21.2	2.0	26.4
	ARMAP-M	58.2	30.0	23.4	4.0	29.6
Phi3.8B	Sampling	34.7	10.0	7.6	2.0	15.2
	Greedy	42.4	9.5	6.5	2.1	17.5
	ARMAP-R	53.3	9.6	7.2	4.0	21.9
	ARMAP-B	52.1	20.0	17.0	9.0	26.5
	ARMAP-M	53.7	28.3	24.3	10.0	30.0

401 Table 1: Effectiveness of the proposed method on different benchmarks. Our ARMAP framework
402 consistently outperforms the baselines across different language models.
403

404
405
406
407
408
409
410
411
412
413

Algorithms	Action↓	Price ↓	Reward ↑
Greedy	4.6	102.4	50.4
ARMAP-B	4.7	102.2	62.0
ARMAP-M	4.5	97.9	66.8
ARMAP-B+Length-Penalty	3.9	98.8	60.3
ARMAP-M+Length-penalty	4.0	102.1	65.5
ARMAP-B+Price-penalty	5.0	65.5	57.5
ARMAP-M+Price-penalty	4.3	69.0	62.4

414
415
416
417
418
419
420
421
422
423
424
425
426

414 Table 2: Controllable Trajectory Generation. We show that we can generate controllable trajectories
415 like shorter action lengths and lower prices by customizing reward targets. We use Llama70B as the
416 default API for action prediction.

417 Among the three planning algorithms, MCTS performs the best on average, likely due to its superior
418 mechanisms for identifying higher-reward trajectories and searching less-explored trajectories. We
419 also notice that Reflexion performs the worst on weaker models like Mistral7B and Phi3.8B. We
420 suspect this is because Reflexion was designed for ChatGPT-family-based agents and requires the
421 LLM agent to possess strong capabilities for learning from trial and error. Finally, we present
422 qualitative results of different methods in Fig. 3, where it is clear that our ARMAP generates better
423 trajectories than the baselines, aided by the guidance of automatic reward models. In Appendix A.5,
424 we analyze several failure cases, offer more detailed insights into the limitations of the current
425 approach, and suggest potential improvements in reward modeling.

427
428
429
430
431

4.3 CONTROLLABLE GENERATION.

429 Another benefit of our ARMAP pipeline is that we can customize our reward targets during inference,
430 allowing us to generate more controllable action sequences, rather than solely maximizing the
431 predicted rewards. Agent fine-tuning methods (Li et al., 2023; Zeng et al., 2023) find it challenging
to achieve this goal since agent behaviors are typically fixed during inference. We conducted

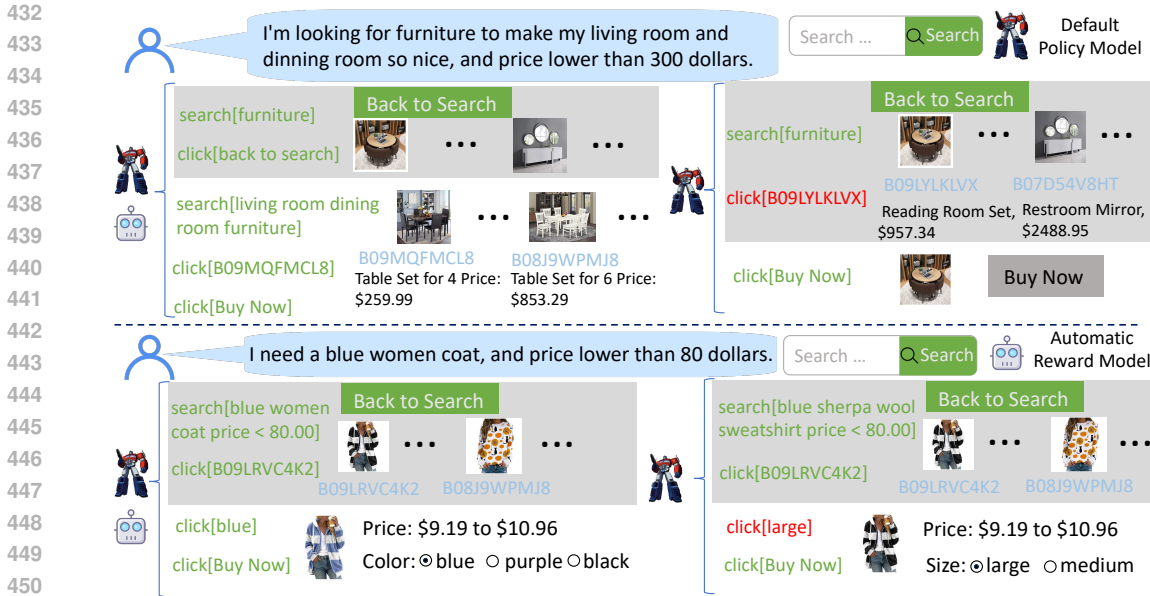


Figure 3: Two qualitative results of the Webshop task. The figure shows two examples utilizing the advantages of our ARMAP framework and we are able to correct errors made by existing methods. In the top example, when the search results do not meet the requirements, our ARMAP method leverages the advantage of the tree structure to backtrack and search again, thereby retrieving the appropriate target item. In contrast, existing methods fail to backtrack when the target item is not found. In the bottom example, by using the ARMAP to evaluate different states in the environment, our method is able to select the color that offers a higher reward and better meets the requirements when choosing between size and color, rather than mistakenly selecting the wrong size. These two examples sufficiently demonstrate the advantages of our method compared to traditional approaches.

Models	Model Base	ScienceWorld (seen)
Greedy	Phi3.8B	9.6
SFT-Policy	VILA3B	18.6
ARMAP-B w/o R	Llama70B and Phi3.8B	16.0
ARMAP-M w/o R		26.5
ARMAP-B	VILA3B and Phi3.8B	20.0
ARMAP-M		28.3

Table 3: Ablation study of the proposed framework. Our ARMAP framework is more effective than directly finding a policy model and using the general LLM for reward generation.

experiments in the Webshop environment to evaluate the impact of customizable reward targets. In addition to the original objective of maximizing the predicted reward $\mathcal{R}(x, h)$, we defined two additional optimization targets. First, we aimed to minimize the number of actions in the trajectory history, defining the reward target as $\mathcal{R}(x, h) - \text{NumberOfAction}(h)$. Second, we sought to minimize the price of the target product, with a customized target of $\mathcal{R}(x, h) - \text{PriceOfProduct}(h)$. Table 2 presents the results. By applying a length penalty on the reward target for ARMAP-M, we reduced the average action length from 4.5 to 4 and the average product price from 97.9 to 69.0, while maintaining comparable performance on the default matching reward. Similar performance was observed for ARMAP-B. Additionally, we provide a qualitative example in Fig. 4. From this example, we can see that our customized reward target successfully guided the LLM agent to purchase products with fewer action steps while still finding the target product.

486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539

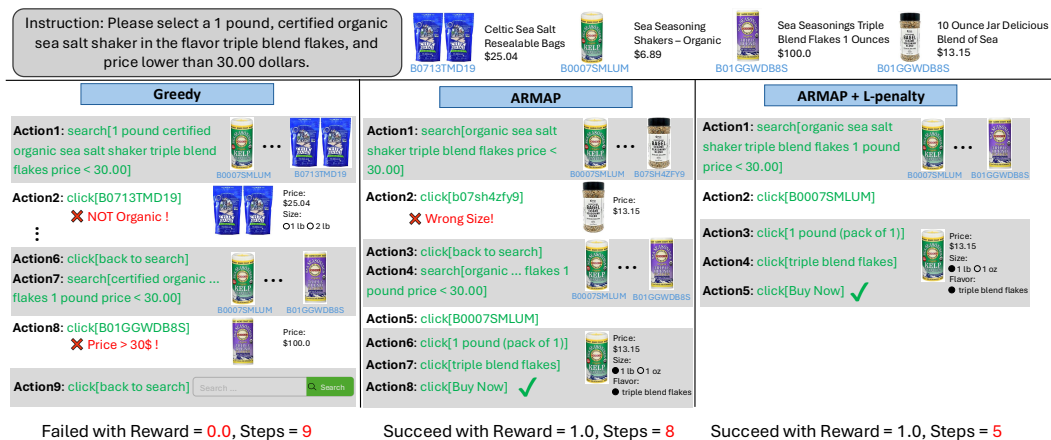


Figure 4: A typical example of customized reward target for shorter trajectory generation. On the left, we show the default greedy decoding generates a long trajectory without finding the target product. In the middle, we show our default reward can guide the LLM agent to generate a correct but long trajectory. On the right, we show our framework with a customized reward target for shorter trajectories, which finds a correct and short trajectory for the target product.

4.4 ABLATION STUDIES.

We conduct ablation studies to investigate the effectiveness of the framework. Specifically, we aim to answer the following questions: **Q1**. Can we train a policy model with fully supervised learning to handle multi-step tasks from the synthesized trajectory data? **Q2**. Can a large, general language model be used as the reward model to perform guidance without automatic reward learning?

We conducted experiments using the ScienceWorld benchmark, and the results are shown in Table 3. When comparing our pipeline to the SFT model trained using our reward backbone VILA3B, we observed that although the policy model trained through fully supervised learning performed reasonably well (18.6), it still lagged behind the performance of our planning framework (28.3). This suggests that learning a policy model is more challenging than learning a reward model, highlighting the effectiveness of our proposed ARMAP pipeline (answering **Q1**).

Next, we replaced our smaller 3B reward model with a much larger language model, Llama3-1-70B, and used few-shot prompting to predict the reward of the extracted trajectories. We found that this larger model also improved performance compared to the default greedy model, demonstrating the effectiveness of our planning framework. However, it still performed worse than our pipeline using automatic reward learning, despite the Llama3-1-70B being about 20 times larger, further showcasing the efficiency and effectiveness of our approach (answering **Q2**).

We provide additional ablation experiments in the Appendix A.2, including the data quality from various LLMs, reward modeling target and computational efficiency.

5 CONCLUSION

We propose a framework, ARMAP, for large language model (LLM) agents to manage tasks that require multi-step decision-making and environmental feedback, such as online shopping or scientific reasoning. This framework allows LLM-based agents to enhance task planning by autonomously learning a reward model from the environment, without the need for human labeling. The method utilizes pre-trained LLM agents to generate diverse action trajectories within an environment, which are then evaluated by a separate LLM based on the task’s intent. These evaluations help train a reward model that strengthens the agents’ decision-making capabilities. The framework enhances the performance of LLM agents in addressing complex tasks and mitigates issues related to data scarcity and API limitations. Its effectiveness is demonstrated across various benchmarks, representing a significant advancement in the development of AI agents for real-world, multi-step problem-solving.

REFERENCES

- 540
541
542 Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach,
543 Amit Bahree, and et al. Phi-3 technical report: A highly capable language model locally on your
544 phone, 2024. URL <https://arxiv.org/abs/2404.14219>.
- 545 Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, and
546 et al. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint*
547 *arXiv:2204.01691*, 2022.
- 548
549 Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna
550 Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness
551 from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.
- 552
553 Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method
554 of paired comparisons. *Biometrika*, 1952.
- 555
556 R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and*
557 *Automation*, 2(1):14–23, 1986. doi: 10.1109/JRA.1986.1087032.
- 558 Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, and
559 et al. Language models are few-shot learners, 2020. URL <https://arxiv.org/abs/2005.14165>.
- 560
561 Avyay Casheekar, Archit Lahiri, Kanishk Rath, Kaushik Sanjay Prabhakar, and Kathiravan Srinivasan.
562 A contemporary review on chatbots, ai-powered virtual conversational agents, chatgpt: Appli-
563 cations, open challenges and future research directions. *Computer Science Review*, 52:100632,
564 2024.
- 565
566 Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and
567 C. Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server, 2015. URL
568 <https://arxiv.org/abs/1504.00325>.
- 569 Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep
570 reinforcement learning from human preferences. *Advances in neural information processing*
571 *systems*, 2017.
- 572 Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International*
573 *conference on computers and games*. Springer, 2006.
- 574
575 Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su.
576 Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing*
577 *Systems*, 2024.
- 578
579 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep
580 bidirectional transformers for language understanding, 2019.
- 581
582 Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, and et al.
583 The llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.
- 584
585 Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu
586 Zheng, Ke Li, Xing Sun, et al. Mme: A comprehensive evaluation benchmark for multimodal
587 large language models. *arXiv preprint arXiv:2306.13394*, 2023.
- 588
589 Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V
590 in VQA matter: Elevating the role of image understanding in Visual Question Answering. In
591 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- 592
593 Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu.
Reasoning with language model is planning with world model. *arXiv*, 2023.
- Mary Hegarty. Mechanical reasoning by mental simulation. *Trends in cognitive sciences*, 2004.

- 594 Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Su-
595 leyman, and Phil Blunsom. Teaching machines to read and comprehend. In C. Cortes, N. Lawrence,
596 D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*,
597 volume 28. Curran Associates, Inc., 2015. URL [https://proceedings.neurips.cc/paper_](https://proceedings.neurips.cc/paper_files/paper/2015/file/afdec7005cc9f14302cd0474fd0f3c96-Paper.pdf)
598 [files/paper/2015/file/afdec7005cc9f14302cd0474fd0f3c96-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2015/file/afdec7005cc9f14302cd0474fd0f3c96-Paper.pdf).
- 599
600 Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan
601 Wang, Yuxiao Dong, Ming Ding, and Jie Tang. Cogagent: A visual language model for gui agents,
602 2023.
- 603 Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning
604 and compositional question answering. *Conference on Computer Vision and Pattern Recognition*
605 *(CVPR)*, 2019.
- 606
607 Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot,
608 Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier,
609 L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas
610 Wang, Timoth  e Lacroix, and William El Sayed. Mistral 7b, 2023. URL [https://arxiv.org/](https://arxiv.org/abs/2310.06825)
611 [abs/2310.06825](https://arxiv.org/abs/2310.06825).
- 612 Richard M Karp. On the computational complexity of combinatorial problems. *Networks*, 1975.
- 613
614 Lukas Kirchdorfer, Robert Bl  mel, Timotheus Kampik, Han Van der Aa, and Heiner Stuckenschmidt.
615 Agentsimulator: An agent-based approach for data-driven business process simulation. In *2024*
616 *6th International Conference on Process Mining (ICPM)*, pp. 97–104. IEEE, 2024.
- 617
618 Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language
619 model agents. *arXiv*, 2024.
- 620
621 Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language
622 models, 2023a. URL <https://arxiv.org/abs/2303.00001>.
- 623
624 Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E.
625 Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model
626 serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating*
Systems Principles, 2023b.
- 627
628 Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building
629 machines that learn and think like people. *Behavioral and brain sciences*, 2017.
- 630
631 Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor
632 Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with
ai feedback. *arXiv*, 2023.
- 633
634 Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem.
635 Camel: Communicative agents for "mind" exploration of large language model society. In *Thirty-*
636 *seventh Conference on Neural Information Processing Systems*, 2023.
- 637
638 Ji Lin, Hongxu Yin, Wei Ping, Yao Lu, Pavlo Molchanov, Andrew Tao, Huizi Mao, Jan Kautz,
639 Mohammad Shoeybi, and Song Han. Vila: On pre-training for visual language models, 2023.
- 640
641 Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding,
642 Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui
643 Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang.
Agentbench: Evaluating llms as agents. *arXiv preprint arXiv: 2308.03688*, 2023.
- 644
645 Moni Naor. Evaluation may be easier than generation. In *Proceedings of the twenty-eighth annual*
ACM symposium on Theory of computing, pp. 74–83, 1996.
- 646
647 OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, and et al. Gpt-4 technical
report, 2024.

- 648 Norman Di Palo, Arunkumar Byravan, Leonard Hasenclever, Markus Wulfmeier, Nicolas Heess,
649 and Martin Riedmiller. Towards a unified agent with foundation models, 2023. URL <https://arxiv.org/abs/2307.09668>.
- 650
651 Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous
652 evaluation and refinement of digital agents, 2024.
- 653
654 Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea
655 Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances*
656 *in Neural Information Processing Systems*, 2024.
- 657
658 Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for
659 machine comprehension of text, 2016. URL <https://arxiv.org/abs/1606.05250>.
- 660
661 Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf.
662 Sayplan: Grounding large language models using 3d scene graphs for scalable task planning. In
663 *7th Annual Conference on Robot Learning*, 2023. URL <https://openreview.net/forum?id=wMp0M00Ss7a>.
- 664
665 Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, and
666 et al. A generalist agent, 2022. URL <https://arxiv.org/abs/2205.06175>.
- 667
668 Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste
669 Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5:
670 Unlocking multimodal understanding across millions of tokens of context. *arXiv*, 2024.
- 671
672 Samuel Schmidgall, Rojin Ziaei, Carl Harris, Eduardo Reis, Jeffrey Jopling, and Michael Moor.
673 Agentclinic: a multimodal agent benchmark to evaluate ai in simulated clinical environments,
674 2024.
- 675
676 Archit Sharma, Sedrick Keh, Eric Mitchell, Chelsea Finn, Kushal Arora, and Thomas Kollar. A
677 critical evaluation of ai feedback for aligning large language models. *arXiv*, 2024.
- 678
679 Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion:
680 Language agents with verbal reinforcement learning. *Advances in Neural Information Processing*
681 *Systems*, 36, 2024.
- 682
683 Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi,
684 Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions
685 for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern*
686 *recognition*, pp. 10740–10749, 2020.
- 687
688 Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew
689 Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In
690 *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL
691 <https://arxiv.org/abs/2010.03768>.
- 692
693 David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez,
694 Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi
695 by self-play with a general reinforcement learning algorithm. *arXiv*, 2017.
- 696
697 Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error:
698 Exploration-based trajectory optimization of LLM agents. In Lun-Wei Ku, Andre Martins, and
699 Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2024.
- 700
701 Zhiqing Sun, Sheng Shen, Shengcao Cao, Haotian Liu, Chunyuan Li, Yikang Shen, Chuang Gan,
Liang-Yan Gui, Yu-Xiong Wang, Yiming Yang, et al. Aligning large multimodal models with
factually augmented rlhf. *arXiv*, 2023a.
- Zhiqing Sun, Yikang Shen, Hongxin Zhang, Qinhong Zhou, Zhenfang Chen, David Cox, Yiming
Yang, and Chuang Gan. Salmon: Self-alignment with principle-following reward models. *arXiv*,
2023b.

- 702 Daniel Toyama, Philippe Hamel, Anita Gergely, Gheorghe Comanici, Amelia Glaese, Zafarali Ahmed,
703 Tyler Jackson, Shibl Mourad, and Doina Precup. Androidenv: A reinforcement learning platform
704 for android. *arXiv*, 2021.
- 705
706 Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is
707 your agent smarter than a 5th grader?, 2022. URL <https://arxiv.org/abs/2203.07540>.
- 708 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny
709 Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in*
710 *neural information processing systems*, 2022.
- 711
712 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le,
713 and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.
714 URL <https://arxiv.org/abs/2201.11903>.
- 715 Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng,
716 and Sujian Li. Watch every step! Llm agent learning via iterative step-level process refinement.
717 *arXiv*, 2024.
- 718
719 An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu
720 Zhong, Julian McAuley, Jianfeng Gao, et al. Gpt-4v in wonderland: Large multimodal models for
721 zero-shot smartphone gui navigation. *arXiv*, 2023.
- 722 Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao.
723 React: Synergizing reasoning and acting in language models. *arXiv*, 2022.
- 724
725 Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable
726 real-world web interaction with grounded language agents, 2023a. URL <https://arxiv.org/abs/2207.01206>.
- 727
728 Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik
729 Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023b.
730 URL <https://arxiv.org/abs/2305.10601>.
- 731
732 Licheng Yu, Patrick Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg. Modeling context
733 in referring expressions, 2016. URL <https://arxiv.org/abs/1608.00272>.
- 734
735 Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason
736 Weston. Self-rewarding language models. *arXiv*, 2024.
- 737
738 Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning:
739 Enabling generalized agent abilities for llms. *arXiv*, 2023.
- 740
741 Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin
742 Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language
743 models. *arXiv*, 2023.
- 744
745 Shun Zhang, Zhenfang Chen, Sunli Chen, Yikang Shen, Zhiqing Sun, and Chuang Gan. Improving
746 reinforcement learning from human feedback with efficient reward model ensemble. *arXiv*, 2024.
- 747
748 Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for
749 large-scale task planning. In *Thirty-seventh Conference on Neural Information Processing Systems*,
750 2023.
- 751
752 Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web
753 agent, if grounded. In *Forty-first International Conference on Machine Learning*, 2024.
- 754
755 Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng,
Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building
autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. URL <https://webarena.dev>.

A APPENDIX

In this section, we provide supplementary material for the main paper.

A.1 EXPERIMENTS ON ALFWORLD AND AGENTCLINIC.

We extend our experiment on ALFWorld (Shridhar et al., 2021), a classic environment for House-Holding, where the agent must accomplish tasks in physical house-holding environments, like “Put a pan on the dining table”. Following the setup of AgentBench (Liu et al., 2023) for LLM evaluation, we test the model on the dev and std split, using the default success rate as the evaluation metric. Specifically, we used LLaMa-3.1-70B to generate around 1600 pairs of positive and negative samples with our data generation pipeline. Then we train a reward model with these synthesized data. We evaluate our ARMAP framework on ALFWorld using various planning algorithms, including Reflexion and Best-of-N, which we refer to as ARMAP-R and ARMAP-B, respectively. Additionally, we compare our approach with two baseline methods that do not incorporate reward model guidance: Sampling and Greedy. The results are shown below. As shown in Table 4, our model still performs well in this challenging environment, which contains diverse scenes and long-horizon planning tasks.

Models	ALFWorld-std	ALFWorld-dev
Sampling	0.13	0.14
Greedy	0.18	0.30
ARMAP-R	0.22	0.35
ARMAP-B	0.30	0.45

Table 4: Experimental Results on ALFWorld.

We also extended our experiments to ClinicalAgent (Schmidgall et al., 2024), an environment designed for medical decision-making tasks. ClinicalAgent evaluates models on their ability to interpret clinical scenarios and make accurate, high-stakes decisions. Results of ClinicalAgent are provided in Table 5, further supporting the versatility of ARMAP in domains requiring precise reasoning.

Models	AgentClinic-MedQA
Sampling	11.89
Greedy	14.02
ARMAP-B	44.33

Table 5: Experiments Results on AgentClinic.

A.2 ABLATION STUDY.

Dependence on Quality of Synthetic Data from Various LLMs. We choose ScienceWorld and conduct experiments to study the effectiveness of different reward models. As shown in Table 6, the left column represents the results of using LLaMA-8B greedy directly and the Best of N results of LLaMA-8B with the reward model trained by the data generated from LLaMA-70B, LLaMA-8B, Mistral-7B, and Phi-3.8B, respectively. Greedy is our baseline result and it can be observed that using the reward model leads to better experimental outcomes. Among all the results, LLaMA-70B achieves the best performance. Compared to the other three models, LLaMA-70B has the largest scale and is naturally the most capable model. LLaMA-8B and Mistral-7B have a similar number of parameters, and in the ScienceWorld task, Mistral-7B performs better than LLaMA-8B. Phi-3.8B is the smallest of these models, yet it still achieved very good results. Notably, compared to the larger-scale LLaMA-8B and Mistral-7B, Phi-3.8B still scored better. These results indicate that our method exhibits good robustness when faced with LLMs of different scales and capabilities. Even with the smallest model, our method can still achieve good results. From these experimental

Models	SciWorld-seen	SciWorld-unseen
Greedy	29.9	23.8
Llama70B	35.7	28.1
Llama8B	32.2	24.7
Mistral7B	33.7	26.5
Phi3.8B	34.7	26.9

Table 6: Experiments of training data generated from various LLMs.

outcomes, it is clear that our method does not overly rely on the capabilities of language models. In other words, our method is highly efficient and robust.

Reward Modeling Target. To further investigate the optimization target of the reward model, we conduct experiments to compare the performance of pairwise comparison and binary classification as learning methods for the reward model. Specifically, in the classification setting: each input pair is treated as a positive and a negative example. The model is trained to predict a score of 1 for positive examples and 0 for negative examples. The comparative results are shown in Table 7. Across all settings, pairwise comparison consistently outperforms binary classification. This confirms that pairwise comparison captures nuanced preferences more effectively than binary classification, leading to better reward modeling and overall task performance.

Backbone	Algorithms	Classification		Comparative	
		Seen	Unseen	Seen	Unseen
LLaMA-70B	ARMAP-R	57.0	55.4	59.0	56.7
	ARMAP-B	47.2	43.3	57.3	57.0
LLaMA-8B	ARMAP-R	29.0	24.2	31.2	28.0
	ARMAP-B	27.5	22.2	35.7	28.1
Mistral-7B	ARMAP-R	17.8	18.2	21.7	19.7
	ARMAP-B	19.1	17.3	24.5	21.1
Phi-3.8B	ARMAP-R	8.6	4.8	9.6	7.2
	ARMAP-B	17.7	13.7	20.0	17.0

Table 7: Comparison of the Classification target and Comparison target on ScienceWorld.

Computational Efficiency Analysis. We further study the data demands of the reward modelings. We show the performance of using different amounts of training data. In Table 8 and Table 9, we selected ScienceWorld and used ARMAP-B as the experimental setting. In the leftmost column, we listed the different LLMs used in our study. In the first row, we introduced VILA-3B, VILA-13B, and LLaVA-13B, to compare the impact of different sizes and types of reward models on the final outcomes. In the last two columns, we trained the reward models using 1/5 and 1/25 of the original training dataset size, respectively, to assess how varying amounts of training data affect our method. (1) As seen, the effectiveness of our method continues to improve with increasing reward model sizes. However, in the experiments with LLaMA-8B and Phi-3.8B, despite using more potent reward models, there was no improvement in results. We believe that in the processes of planning and reasoning, the capability of the policy model still plays a dominant role. If the policy model is more robust, and concurrently, if we enhance the capability of the reward model, we can continuously achieve better results. (2) We also observe that the performance of LLaVA-13B is not as good as VILA-13B. We attribute this to VILA being an improved version of LLaVA, and it utilizes an interleaved image-text dataset in its training, which better aids the model in perceiving, understanding, and handling multimodal information. Hence, VILA outperforms LLaVA. (3) From the Table 8 and Table 9, it is evident that regardless of whether the data is seen or unseen, increasing the model size improves the final experimental results. If we use the results of the VILA-3B model as a benchmark and compare it with the two settings, 1/5 data and 1/25 data, it is clear that increasing the training data enhances the outcomes. Conversely, even when using extremely limited data amounts like 1/5

or 1/25 of the original dataset, we can still achieve a capable model, and the performance does not dramatically decrease.

These results demonstrate that our method can still yield good results in a low-resource environment. In other words, our approach does not rely on large volumes of data and the strong capability of large models; it is succinct and efficient, capable of performing well in extremely low-resource settings.

Backbone	VILA-3B	VILA-13B	LLaVA-13B	1/5 Data	1/25 Data
LLaMA-70B	57.3	61.2	44.3	52.1	50.6
LLaMA-8B	35.7	34.3	26.0	31.4	29.3
Mistral-7B	24.5	26.0	19.5	22.6	21.7
Phi-3.8B	20.0	19.5	16.7	17.9	13.9

Table 8: Comparison of reward model selection and data demands on ScienceWorld seen set.

Backbone	VILA-3B	VILA-13B	LLaVA-13B	1/5 Data	1/25 Data
LLaMA-70B	57.0	60.7	48.2	50.0	47.7
LLaMA-8B	28.1	27.5	22.2	26.8	24.2
Mistral-7B	21.1	22.9	19.2	21.6	19.7
Phi-3.8B	17.0	15.3	13.7	14.2	11.7

Table 9: Comparison of reward model selection and data demands on ScienceWorld unseen set.

Ablation on Visual Input. We also train a new reward model without visual information. As shown in Table 10, we can see that, in different settings, the reward model with visual information performs better than the model without visual information, which shows the importance of visual context in the Webshop task.

Backbone	Algorithms	w/o visual	w/ visual
LLaMA-70B	ARMAP-R	56.1	56.5
	ARMAP-B	61.6	62.0
Mistral-7B	ARMAP-R	53.6	54.1
	ARMAP-B	51.3	54.4

Table 10: Ablation of the visual input.

Overhead in Data Synthesis. We calculate the tokens we have used for task instruction generation and trajectory exploration. We summarize these overheads in Table 11. To provide a more intuitive comparison, we first calculated the average tokens per sample for these different tasks. We found that although Game of 24 overall consumes the most tokens, the average number of tokens spent per Game of 24 sample is relatively the least. In contrast, Webshop has the fewest total samples but the highest average number of tokens spent per sample. ScienceWorld falls in between these two. The reason Webshop has a higher average number of tokens compared to Game of 24 is that the environment required for Webshop is more complex, involving more diverse elements and possibilities.

Tasks	Samples	Tokens	Tokens per Sample
ScienceWorld	4064	2541255	625
Webshop	2436	6645746	2728
Game of 24	37885	12846182	339

Table 11: Tokens of data generation in three different tasks.

Proprietary Models as Training Data Generators and Policy Models. In the main content, we mainly consider using open-source models to act as training data generators and policy models. In

order to investigate the upper bounds of our proposed method, we also conduct some experiments by using powerful proprietary models. However, to serve as the training data generator, closed-source models have several drawbacks, including high costs, limited commercial access, and lack of reproducibility. In contrast, our approach achieves strong results without relying on closed-source models. Given the expense associated with API-based models like GPT-4o for generating training datasets, we have opted not to pursue this method for now.

For API-based proprietary models serving as policy models, the high cost of GPT-4o and API access rate limitations prompted us to focus our experiments primarily on ALFWorld. Specifically, we used GPT-4o-2024-08-06 to sample five trajectories each on ALFWorld’s Dev and Std sets, then conducted experiments using our automatic reward model. As shown in Table 12, our reward model is able to help the powerful GPT-4o gain better performance, demonstrating the effectiveness of our framework.

GPT-4o	Std	Dev
Sampling	0.74	0.88
Greedy	0.82	0.90
ARMAP-B	0.84	0.95

Table 12: Experiments of using the proprietary model on ALFWorld

A.3 IMPLEMENTATION DETAILS.

Large Pretrain Model Setup. We serve a diverse set of open-source LLM APIs to evaluate the effectiveness of the proposed pipeline. We list all the open-source models and their weights on huggingface in table 13. All these models can be easily setup and reproduced with the VLLM library (Kwon et al., 2023b). We prove the effectiveness of our ARMAP framework across different LLM APIs.

Acronym	Model description and weight on huggingface websites
Llama70B	https://huggingface.co/hugging-quants/Meta-Llama-3.1-70B-Instruct-AWQ-INT4
Llama8B	https://huggingface.co/meta-llama/Meta-Llama-3.1-8B-Instruct
Mistral7B	https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3
Phi3.8B	https://huggingface.co/microsoft/Phi-3.5-mini-instruct
VILA3B	https://huggingface.co/Efficient-Large-Model/VILA1.5-3b

Table 13: Agent models, the reward model, and their associated description on huggingface websites.

Environment Setup. We build our environments based on the environment setup of the previous works (Liu et al., 2023; Song et al., 2024; Yao et al., 2023b; Shridhar et al., 2021; Schmidgall et al., 2024). For Webshop and ALFWorld environment, we start these docker environments from AgentBench (Liu et al., 2023) and implement different planning algorithms, Reflexion, Best-of-N and MCTS on it. Similarly, we build our ScienceWorld, Game of 24 and AgentClinic environments from Song et al. (2024), Yao et al. (2023b) and Schmidgall et al. (2024), respectively.

Planning Algorithm Details. We compare the performance of different planning algorithms by limiting their maximum explored trajectory number. We set the maximum number to be 10 on Webshop and ScienceWorld in consideration of effectiveness and efficiency. We set the maximum number to be 100 on Game of 24 following the setup of Yao et al. (2023b). In Webshop, ScienceWorld, ALFWorld and AgentClinic benchmarks, we only consider the top 10 available actions suggested by the LLM agent at each state to reduce search space. We also set a trajectory’s maximal action number length to 10 for simplicity.

For Reflexion, we set the maximum trial number to be 10 for all tasks. For different tasks and models, we set the threshold of Reflexion separately. During the iteration process, if the reward of the current

972 trail’s trajectory exceeds the threshold, the iteration will stop, and the current trail will be taken as
973 the result. If the maximum number of trials is reached, the last trial will be taken as the result in
974 Webshop and Game of 24, while the first trial will be taken as the result in ScienceWorld.
975

976 **Data Generation.** In total, we generate 2,436, 4,064 and 37,885 pairs of data for Webshop,
977 ScienceWorld and Game of 24, respectively. Sampled synthesized data sample can be seen in
978 Fig. 5, Fig. 6 and Fig. 7. We provide the sampled prompt we use for data generation from Fig. 8 to
979 Fig. 11. In Fig. 8, we show an example how we prompt the LLM to generate language instruction for
980 ScienceWorld. In Fig. 9, we show how we refine the language instruction to refine the instruction goal.
981 In Fig. 10 and Fig. 11, we show the prompt how the LLM agent synthesizes positive and negative
982 trajectories, respectively.
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025

1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079

Training Data Example for Webshop

Task Instruction: I need 2 long-lasting bluetooth radio speaker units for a soundbar with 4ohm impedance and 6w power, and price lower than 50.00 dollars.

Positive Example:
Thought: ...
Action: ...
Current Screenshot: ...
Observation: ...
Thought: The option B09STMXYR5 matches the required characteristics. I should select it.
Action: click[B09STMXYR5]
Current Screenshot:



The screenshot shows a product listing for 'JUSTQIJUN 2pcs 1.5 Inch Bluetooth Radio Speaker Unit'. The product is described as a '4 Ohm 6W Sound Bar Horn 18 Core 45mm Music Portable Loudspeaker Rubber Edge Compatible with Altavoces Bookshelf Speakers (Color : 40mm 4 ohm 6W)'. The price is \$28.36 and the rating is N.A. There are buttons for 'Description', 'Features', 'Reviews', and 'Buy Now'. Navigation buttons 'Back to Search' and '< Prev' are also visible.

Observation: Instruction: [SEP] i need 2 long-lasting bluetooth radio speaker units for a soundbar with 4ohm impedance and 6w power, and price lower than 50.00 dollars [SEP] Back to Search [SEP] < Prev [SEP] JUSTQIJUN 2pcs 1.5 Inch Bluetooth Radio Speaker Unit 4 Ohm 6W Sound Bar Horn 18 Core 45mm Music Portable Loudspeaker Rubber Edge Compatible with Altavoces Bookshelf Speakers (Color : 40mm 4 ohm 6W) [SEP] Price: \$28.36 [SEP] Rating: N.A. [SEP] Description [SEP] Features [SEP] Reviews [SEP] Buy Now

Thought: I should click on the 'Buy Now' button to proceed with the purchase.
Action: click[buy now]

Negative Example:
Thought: ...
Action: ...
Current Screenshot: ...
Observation: ...
Thought: I think I should click on the product I need, which is B09SWKXBY5, since it matches the requirements of 4ohm impedance and 6w power and the price is lower than 50.00 dollars.
Action: click[B09SWKXBY5]
Current Screenshot:



The screenshot shows a product listing for 'JUSTQIJUN 2pcs Full Range 2 Inch Speaker 15W DIY'. The product is described as a 'Soundbar Boombox Unit Portable Radio 10W 20w 4 Ohm Speaker HiFi Bluetooth Speakers 55mm Bookshelf Speakers (Color : 4 ohm 20W)'. The price is \$42.66 and the rating is N.A. There are buttons for 'Description', 'Features', 'Reviews', and 'Buy Now'. Navigation buttons 'Back to Search' and '< Prev' are also visible. Below the product image, there are three radio buttons for color selection: '4 ohm 10w', '4 ohm 15w', and '4 ohm 20w'.

Observation: Instruction: [SEP] i need 2 long-lasting bluetooth radio speaker units for a soundbar with 4ohm impedance and 6w power, and price lower than 50.00 dollars [SEP] Back to Search [SEP] < Prev [SEP] color [SEP] 4 ohm 10w [SEP] 4 ohm 15w [SEP] 4 ohm 20w [SEP] JUSTQIJUN 2pcs Full Range 2 Inch Speaker 15W DIY Soundbar Boombox

1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133

Unit Portable Radio 10W 20w 4 Ohm Speaker HiFi Bluetooth Speakers 55mm Bookshelf Speakers (Color : 4 ohm 20W) [SEP] Price: \$42.66 [SEP] Rating: N.A. [SEP] Description [SEP] Features [SEP] Reviews [SEP] Buy Now
Thought: I think I should click on the 'Buy Now' button to purchase the product.
Action: click[buy now]

Figure 5: Training Data Example for Webshop.

Training Data Example for ScienceWorld

Task Instruction: You are a helpful assistant to do some scientific experiment in an environment. In the environment, there are several rooms: kitchen, foundry, workshop, bathroom, outside, living room, bedroom, greenhouse, art studio, hallway You should explore the environment and find the items you need to complete the experiment. You can teleport to any room in one step. All containers in the environment have already been opened, you can directly get items from the containers.
The available actions are:
open OBJ: open a container
close OBJ: close a container
...
wait1: task no action for a step
Task Description: Your task is to find a non-living thing in the environment. Focus on the jug in the greenhouse. When you are done, move the jug to the green box in the kitchen.

Positive Example:
Action: look around
Observation: This room is called the hallway. In it, you see: the agent, a substance called air, a picture
You also see:
A door to the art studio (that is open), A door to the bedroom (that is open), A door to the greenhouse (that is open), A door to the kitchen (that is open), A door to the living room (that is open), A door to the workshop (that is open)
Action: teleport to the greenhouse
Observation: You teleport to the greenhouse.
Action: look around
Observation: This room is called the greenhouse. In it, you see: the agent, a substance called air, a adult bee, a adult bee, a adult bee, a adult bee, a bee hive. The bee hive door is open. In the bee hive is: a adult bee. a flower pot 2 (containing a peach tree in the adult stage, soil), a flower pot 4 (containing a substance called water), a flower pot 5 (containing a substance called water), a flower pot 6 (containing a banana tree in the adult stage, soil), a flower pot 7 (containing a apple tree in the adult stage, soil), a jug (containing nothing), a shovel, a sink, which is turned off. In the sink is: nothing.
You also see:
A door to the hallway (that is open), A door to the outside (that is open)
Action: focus on the jug
Observation: You focus on the jug.

Negative Example:
Action: look around
Observation: This room is called the hallway. In it, you see: the agent, a substance called air, a picture
You also see:
A door to the art studio (that is open), A door to the bedroom (that is open), A door to the greenhouse (that is open), A door to the kitchen (that is open), A door to the living room (that is open), A door to the workshop (that is open)
Action: teleport to the greenhouse
Observation: You teleport to the greenhouse.

1134
 1135 **Action:** connect painting to kitchen
 1136 **Observation:** terminal 1 on painting is now connected to terminal 1 on kitchen
 1137 **Action:** connect agent to bathtub
 1138 **Observation:** No known action matches that input.
 1139
 1140

Figure 6: Training Data Example for ScienceWorld.

1141
 1142
 1143 Training Data Example for Game of 24
 1144
 1145 **Task Instruction:** You are a start agent and generate data for Game24. Game24 requires
 1146 users to use numbers and basic arithmetic operations (+ - * /) to obtain 24.
 1147 Your task is to generate a new input (4 digital number) for Game 24.
 1148 1. each new input number should be in the range of 1 to 13.
 1149 2. People can use numbers and basic arithmetic operations (+ - * /) to obtain 24. At each step,
 1150 people are only allowed to choose two of the remaining numbers to obtain a new number.
 1151 Here are the few-shot examples.
 1152 3. since there is only four number input and the intermediate steps should only be three.
 1153 Input: 12 10 8 4
 1154
 1155 **Positive Example:**
 1156 **Steps:** $10 - 8 = 2$ (left: 2 4 12)
 1157 $12 / 2 = 6$ (left: 4 6)
 1158 $6 * 4 = 24$ (left: 24)
 1159 **Answer:** $(12 / (10 - 8)) * 4 = 24$
 1160
 1161 **Negative Example:**
 1162 **Steps:** $10 - 12 = -2$ (left: -2 4 8)
 1163 $8 / 4 = 2$ (left: -2 2)
 1164 $-2 * 2 = 4$ (left: 4)
 1165 **Answer:** $((10 - 12) * (8 / 4)) = 4$
 1166
 1167

Figure 7: Training Data Example for Game of 24.

1168 Instruction Generation Prompt for ScienceWorld
 1169
 1170 **Task Instruction:** You are a helpful assistant to do some scientific experiments in an
 1171 environment.
 1172
 1173 In the environment, there are several rooms: kitchen, foundry, workshop, bathroom,
 1174 outside, living room, bedroom, greenhouse, art studio, and hallway.
 1175
 1176 The available actions are:
 1177 open OBJ: open a container
 1178 ...
 1179
 1180 You will be given a dialogue between you (assistant) and a human user. You need
 1181 to generate the task description after understanding the dialogue given to you. In order to
 1182 help you better generate the task description, I will give you an example below.
 1183
 1184 In this example, a human user gives a Task Description at first. Then you work as
 1185 an assistant and talk with user to finish the task step by step using the available actions above.
 1186 Thought and Action are your reasoning and the actions you take at each step. Observation is
 1187 provided by the human user; it is an observation of the surrounding environment after you
 have executed the action. In the end, you need to restore the Task Description based on the

1188 whole dialogue.
 1189
 1190 **Example:**
 1191 **User**
 1192 Task Description
 1193 **Assistant**
 1194 Thought: I need to find materials to create green paint, which are likely to be found in an art
 1195 studio.
 1196 Action: look around
 1197 **User**
 1198 Observation: This room is called the bedroom. In it, you see:
 1199 ...
 1200 **User**
 1201 Observation: Blue and yellow paint mix to produce green paint.
 1202 **Assistant**
 1203
 1204 Thought: The task is to create green paint and then focus on it. Now that the green
 1205 paint is made, I should concentrate on the final product.
 1206 Action: focus on green paint
 1207
 1208 Task Description: Your task is to use chemistry to create green paint. When you
 1209 are done, focus on the green paint.
 1210
 1211 Here is the task you need to do:
 1212 ...
 1213
 1214 You should generate the Task Description following the format like examples below
 1215 and generate it based on the dialogue above without any other words.
 1216 Examples:
 1217 Your task is to boil ice cream. For compounds without a boiling point, combusting the
 1218 substance is also acceptable. First, focus on the substance. Then, take actions that will cause
 1219 it to change its state of matter.
 1220 Your task is to use chemistry to create violet paint. When you are done, focus on the violet
 1221 paint.
 1222 Your task is to find a(n) living thing. First, focus on the thing. Then, move it to the red box in
 1223 the bathroom.
 1224
 1225 **Task Description:**

Figure 8: Instruction Generation Prompt for ScienceWorld.

1227
 1228
 1229
 1230 Instruction Refinement Prompt for ScienceWorld
 1231
 1232 **Task Instruction:** You are a helpful assistant to do some scientific experiments in an
 1233 environment.
 1234 In the environment, there are several rooms: kitchen, foundry, workshop, bathroom, outside,
 1235 living room, bedroom, greenhouse, art studio, and hallway.
 1236 The available actions are:
 1237 open OBJ: open a container
 1238 ...
 1239
 1240 You will be given a task description and a corresponding trajectory. The task de-
 1241 scription concludes what you have done in this trajectory. You need to elaborate this
 description based on this environment by adding more details.

1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295

Example:
Task Description: Your task is to grow an apple. You can find seeds in the kitchen. You should focus on the grown apple.
Corresponding Trajectory:
look around
This room is called the hallway. In it, you see:
...
open door to kitchen
The door is already open.
go to kitchen
You move to the kitchen.
...
Refined Task Description: Your task is to grow an apple. This will require growing several plants, and them being crosspollinated to produce fruit. Seeds can be found in the kitchen. To complete the task, focus on the grown apple.

Here is the task description you need to refine, and the corresponding trajectory is also provided:
...
Refined Task Description:

Figure 9: Instruction Refinement Prompt for ScienceWorld.

Positive Trajectory Synthesis Prompt for ScienceWorld

Task Instruction: You are a helpful assistant to do some scientific experiments in an environment.

In the environment, there are several rooms: kitchen, foundry, workshop, bathroom, outside, living room, bedroom, greenhouse, art studio, and hallway.

The available actions are:
open OBJ: open a container
...

Based on this environment, you need to randomly propose a Task Description, which concludes what you have done in this environment.

Here are some examples:
Your task is to use chemistry to create green paint. When you are done, focus on the green paint.
Your task is to determine whether tall plant height is a dominant or recessive trait in the pea plant. If the trait is dominant, focus on the red box. If the trait is recessive, focus on the green box.
...
Once you obtain the Task Description, you need to navigate through the environment to complete the instruction and generate a trajectory.

Example:
Task Description: Your task is to use chemistry to create green paint. When you are done, focus on the green paint.

1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349

Trajectory:
Thought: I need to find materials to create green paint, which are likely to be found in an art studio.
Action: look around
...
Generated Trajectory:

Figure 10: Positive Trajectories Synthesis Prompt for ScienceWorld.

Negative Trajectory Synthesis Prompt for ScienceWorld

Task Instruction: You are a helpful assistant to do some scientific experiments in an environment.

In the environment, there are several rooms: kitchen, foundry, workshop, bathroom, outside, living room, bedroom, greenhouse, art studio, and hallway.

The available actions are:
open OBJ: open a container
...

You will be given a task description and a corresponding trajectory. Based on them, you need to generate a negative sample that is similar to the correct trajectory but different from it. The generated trajectory should not meet all requirements of the task description. Moreover, the generated trajectory should satisfy all requirements of the environment.

Example:
Task Description: Your task is to focus on the life stages of the apple plant, starting from earliest to latest. The plants are located outside.

Positive Trajectory:
look around
This room is called the hallway. In it, you see:
...
open door to outside
The door is already open
...

Negative Trajectory:
look around
This room is called the hallway. In it, you see:
...
open door to kitchen
The door is already open.

go to kitchen
You move to the kitchen.
...

Here is the task you need to do:
...
Negative Trajectory:

Figure 11: Negative Trajectories Synthesis Prompt for ScienceWorld.

Reward Model Training Details. The detailed hyperparameters we use for reward model during training and inference are shown in Table 14. We employ identical hyperparameters for reward models of different environments. For Webshop, we use checkpoint of 1100 steps in ARMAP-B, and checkpoint of 1200 steps in ARMAP-R and ARMAP-M.

Name	ScienceWorld	Webshop	Game of 24
lora r		64	
lora alpha		16	
lora dropout		0.0	
lora target modules	q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj		
epochs	10	3	10
batch size	8	1	4
batch size per device	1	1	1
gradient accumulation steps	16	4	16
learning rate	1e-5	2e-5	1e-5
warmup ratio	0.2	0.1	0.25
checkpoint steps	160	1100, 1200	1500
temperature	0.0	0.0	0.0

Table 14: Detailed hyperparameters used in reward model.

Implementation Details of Ablation baselines. For SFT, we use all positive examples from the reward model training as the training data. The training objective is to enable the model to predict the output of the LLM in the positive examples.

For using few-shot prompting to guide the LLMs to predict the reward of historical trajectories, we use the following format of the few-shot prompt:

Few-shot Prompt for LLMs Directly Serving as ScienceWorld Reward Model

Task Instruction: You are an autonomous intelligent agent tasked with evaluating the trajectories of the past experience. You will be given the history of a past experience in which you were placed in an environment and given a task to complete. These tasks will be accomplished through the use of specific actions. Now you are trying to evaluate the performance on a past task. You will be given the objective of the task, the history of interaction including the observations you had and the actions you issued, and the status of the task. Your goal is to think about the strategy and provided path to produce a score ranging from 0 to 1 to measure whether the objective of the task has been reached.

Here are 2 examples:

Example1:

Human: You are a helpful assistant to do some scientific experiment in an environment. In the environment, there are several rooms: kitchen, foundry, workshop, bathroom, outside, living room, bedroom, greenhouse, art studio, hallway. You should explore the environment and find the items you need to complete the experiment. You can teleport to any room in one step. All containers in the environment have already been opened, you can directly get items from the containers. The available actions are: open OBJ: open a container, close OBJ: close a container, activate OBJ: activate a device, deactivate OBJ: deactivate a device, connect OBJ to OBJ: connect electrical components, disconnect OBJ: disconnect electrical components, use OBJ [on OBJ]: use a device/item, look around: describe the

1404 current room, examine OBJ: describe an object in detail, look at OBJ: describe a container's
 1405 contents, read OBJ: read a note or book, move OBJ to OBJ: move an object to a container,
 1406 pick up OBJ: move an object to the inventory, pour OBJ into OBJ: pour a liquid into a
 1407 container, mix OBJ: chemically mix a container, teleport to LOC: teleport to a specific room,
 1408 focus on OBJ: signal intent on a task object, wait: task no action for 10 steps, wait1: task no
 1409 action for a step
 1410 **Task Description:** Your task is to find a(n) non-living thing. First, focus on the thing. Then,
 1411 move it to the purple box in the living room. Please only output one Thought and one Action.
 1412 The available actions are: open OBJ: open a container, close OBJ: close a container, activate
 1413 OBJ: activate a device, deactivate OBJ: deactivate a device, connect OBJ to OBJ: connect
 1414 electrical components, disconnect OBJ: disconnect electrical components, use OBJ [on OBJ]:
 1415 use a device/item, look around: describe the current room, examine OBJ: describe an object
 1416 in detail, look at OBJ: describe a container's contents, read OBJ: read a note or book, move
 1417 OBJ to OBJ: move an object to a container, pick up OBJ: move an object to the inventory,
 1418 pour OBJ into OBJ: pour a liquid into a container, mix OBJ: chemically mix a container,
 1419 teleport to LOC: teleport to a specific room, focus on OBJ: signal intent on a task object,
 1420 wait: task no action for 10 steps, wait1: task no action for a step. Please strictly follow the
 1421 format of the actions.
 1422 **Assistant:** Thought: I need to find a non-living thing, which could be an object in any room,
 1423 so I should start by looking around the current room. Action: look around
 1424 **Human:** Observation: This room is called the bathroom. In it, you see: the agent, a substance
 1425 called air, a bathtub, which is turned off. In the bathtub is: nothing, a drawing, a glass cup
 1426 (containing nothing), a sink, which is turned off. In the sink is: nothing, a toilet. In the toilet
 1427 is: A drain, which is open, a substance called water. You also see: A door to the kitchen
 1428 (that is open). Please only output one Thought and one Action. The available actions are:
 1429 open OBJ: open a container, close OBJ: close a container, activate OBJ: activate a device,
 1430 deactivate OBJ: deactivate a device, connect OBJ to OBJ: connect electrical components,
 1431 disconnect OBJ: disconnect electrical components, use OBJ [on OBJ]: use a device/item,
 1432 look around: describe the current room, examine OBJ: describe an object in detail, look at
 1433 OBJ: describe a container's contents, read OBJ: read a note or book, move OBJ to OBJ:
 1434 move an object to a container, pick up OBJ: move an object to the inventory, pour OBJ into
 1435 OBJ: pour a liquid into a container, mix OBJ: chemically mix a container, teleport to LOC:
 1436 teleport to a specific room, focus on OBJ: signal intent on a task object, wait: task no action
 1437 for 10 steps, wait1: task no action for a step. Please strictly follow the format of the actions.
 1438 **Assistant:** Thought: The glass cup in the bathroom is a non-living thing, so I should focus
 1439 on it. Action: focus on glass cup.
 1440 ...
 1441 **Assistant:** Thought: I've found the purple box, and now I need to move the glass cup into it.
 1442 Action: move glass cup to purple box.
 1443 **Human:** Observation: No known action matches that input.
 1444 **Human:** Please evaluate whether you complete the 'Task Description' or not.
 1445 Following your definitions, the task completion score is <0.750>
 1446 **Example2:** ...
 1447 Now is your turn:

1450 Figure 12: Few-shot Prompt for LLMs Directly Serving as ScienceWorld Reward Model.

1453 A.4 MORE QUALITATIVE RESULTS.

1454 We show more qualitative results of our ARMAP in Fig. 13 and Fig. 14. Based on the examples, we
 1455 can see that the automatic reward models in our pipeline can distinguish good trajectories from bad
 1456 ones and guide LLMs to generate better trajectories to finish the tasks.
 1457

1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511

Qualitative Visualization for ScienceWorld

Task Description: Task Description: Your task is to find a(n) non-living thing. First, focus on the thing. Then, move it to the orange box in the living room.

Trajectory A:

Observation: You teleport to the living room.

Thought: Now that I'm in the living room, I should look around to find a non-living thing that I can move to the orange box. **Action:** look around

Observation: This room is called the living room. In it, you see: the agent, a substance called air, a chair. On the chair is: nothing, a couch. On the couch is: a white pillow. a finger painting a orange box (containing nothing) a table. On the table is: nothing.

You also see: A door to the hallway (that is open).

Thought: I see several objects in the living room, but I need to find a non-living thing that I can move to the orange box. **Action:** examine white pillow

ARMAP **Aborted**

Trajectory B:

Observation: This room is called the workshop. In it, you see: the agent, a substance called air, a table. On the table is: a battery, a blue wire, a orange light bulb, which is off, a red light bulb, which is off, a red wire, a switch, which is off, a yellow light bulb, which is off, a yellow wire. a ultra low temperature freezer. The ultra low temperature freezer door is open. In the ultra low temperature freezer is: nothing. You also see:

A door to the hallway (that is open)

Thought: I see several objects in the workshop that could be considered non-living things, such as the battery, blue wire, or light bulbs. I should focus on one of these objects to complete the task. **Action:** focus on battery

Observation: You focus on the battery.

Thought: Now that I've focused on the battery, I need to move it to the orange box in the living room. **Action:** teleport to living room

Observation: You teleport to the living room.

Thought: Now that I'm in the living room, I should look around to find the orange box and move the battery into it. **Action:** look around

Observation: This room is called the living room. In it, you see: the agent, a substance called air, a chair. On the chair is: nothing, a couch. On the couch is: a white pillow. a finger painting a orange box (containing nothing) a table. On the table is: nothing.

You also see: A door to the hallway (that is open)

Thought: I've found the orange box in the living room, and now I can move the battery into it. **Action:** move battery to orange box

ARMAP **Selected**

Figure 13: Qualitative Visualization for ScienceWorld. Based on the comparison of the two trajectories, it is evident that the final action of trajectory B brings the resulting state closer to the task objective. Therefore, our model identifies trajectory B as the superior option.

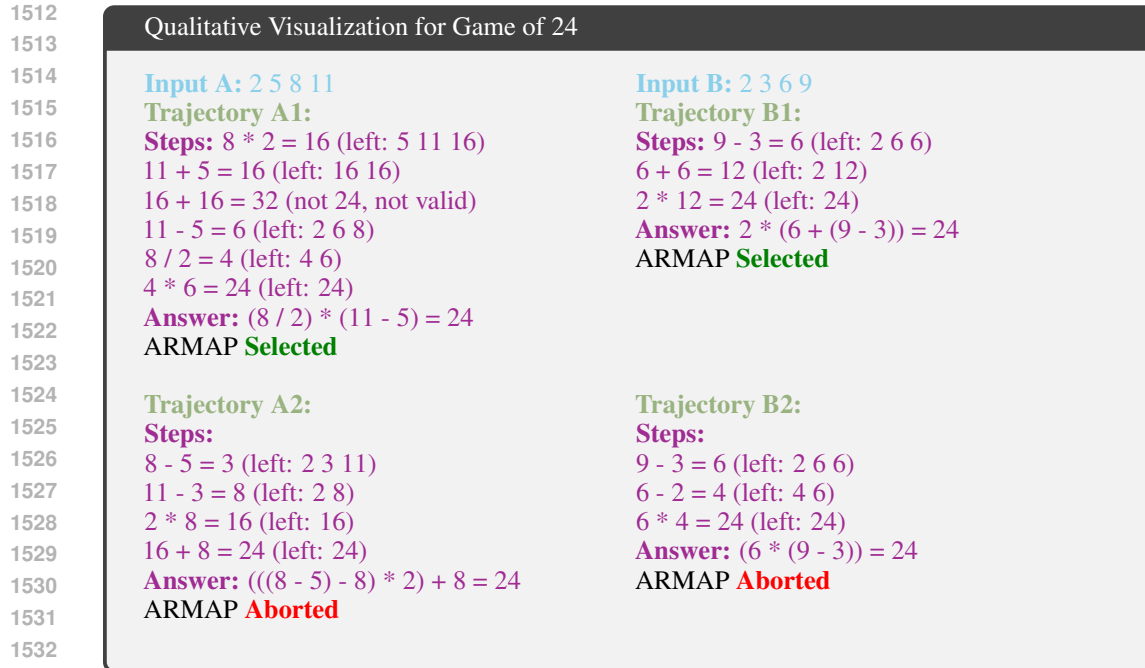


Figure 14: Qualitative Visualization for Game of 24. Trajectory A and Trajectory B correspond to input A and input B respectively. Results show that our ARMAP can accurately pick out the correct trajectory.

A.5 FAILURE CASE ANALYSIS

In this section, we investigate the common failure cases of our framework, aiming to provide data points and insights for future research.

The most common error occurs when there are multiple restrictions in the instruction, the reward model overlooks some of these key conditions. A representative example is illustrated in Fig. 15, the model focuses on price and size but ignores the details about 'Fluoride' hidden in the product description.

Another common failure mode occurs when commonsense knowledge is involved. As demonstrated in Fig. 16, the agent was tasked with buying a blackout shade but failed to choose both the color and the size. While, in everyday life, size is generally more important, the reward model prioritized color instead. In Fig. 17, the reward model cannot assess the lifespan of dragonflies and chipmunks because it lacks the necessary biological knowledge.

Discussion. The analysis of failure modes highlights the significant potential of our framework. To improve its performance, we propose two possible strategies for improvements in reward modeling: (a) Constructing Data with Focus on Complex and Detailed Conditions: enhancing the dataset to include scenarios with higher complexity and more nuanced conditions will help the framework better handle intricate situations and edge cases. (b) Intervening in Reward Scoring with External Knowledge: incorporating external knowledge by combining a prompted Large Language Model with the trained reward model. This approach allows the LLM's generalized knowledge to calibrate the reward scores in a controllable manner, improving the overall accuracy and robustness of the reward model.

1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619

Instruction: I need a **fluoride free** toothpaste for fresh breath. I will need a pack of 4 in 3.5 ounce size, and price lower than 50.00 dollars.

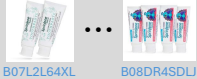



<p>Action1: search[fluoride free toothpaste 3.5 oz pack of 4 price < 50.00]</p>  <p>Action2: click[b07l2l64xl]</p> <p>SprinJene Natural Fluoride Free Toothpaste for Clean and Strong Teeth ✓</p> <p>Action3: click[pack of 4]</p> <p>Action4: click[Buy Now]</p> <p>Price: \$12.69 Size: 3.5 ounce (pack of 2) 3.5 ounce (pack of 4)</p> <p>GT Reward = 1.0  ARMAP Aborted</p>	<p>Action1: search[toothpaste fresh breath fluoride free 4 pack 3.5 oz price < 50.00]</p>  <p>Action2: click[B08DQS3T87]</p> <p>SprinJene Natural Toothpaste with Fluoride for Cavity Protection of Teeth ✗</p> <p>Action3: click[pack of 4]</p> <p>Action4: click[Buy Now]</p> <p>Price: \$23.27 Size: 3.5 ounce (pack of 2) 3.5 ounce (pack of 4)</p> <p>GT Reward = 0.75  ARMAP Selected ☹️</p>
---	--

Figure 15: **Failure Example from Webshop.** The reward model ignores certain key conditions in the task instruction.

Instruction: I want to find **white** blackout shades that are **66 inches in width and 66 inches in height**. they need to be easy to install, and price lower than 90.00 dollars

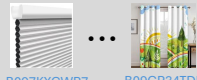



<p>Action1: search[white blackout shades 66 x 66 easy install price < 90.00]</p>  <p>Action2: click[B097KXCWP7]</p> <p>✗ Missing color!</p> <p>Action3: click[66"w x 66"h]</p> <p>Action4: click[Buy Now]</p> <p>Price: \$78.49 Size: 66"w x 66"h ... Color: 3 Black 0 White ...</p> <p>Price: \$78.49 Size: 66"w x 66"h ... Color: 3 Black 0 White ...</p> <p>GT Reward = 0.8  ARMAP Aborted</p>	<p>Action1: search[white blackout shades 66x66 easy install price < 90.00]</p>  <p>Action2: click[B08GFTCMQH]</p> <p>✗ Missing size!</p> <p>Action3: click[white]</p> <p>Action4: click[Buy Now]</p> <p>Price: \$89.99 Color: 0 Black 0 White ...</p> <p>Price: \$89.99 Color: 0 Black 3 White ...</p> <p>GT Reward = 0.4  ARMAP Selected ☹️</p>
--	---

Figure 16: **Failure Example from Webshop.** The reward model misjudged the importance of different conditions, such as the size and color in this case.

1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673

Instruction: Your task is to find the animal with the **longest life span**, then **the shortest life span**. First, focus on the animal with the longest life span. Then, focus on the animal with the shortest life span. The animals are in the 'outside' location.

<p>Action1: look around</p> <p>Observation1: This room is called the art studio. In it, you see: the agent, a substance called air, a large cupboard.</p> <p>Action2: teleport to outside</p> <p>Observation2: You teleport to the outside.</p> <p>Action3: look around</p> <p>Observation3: Here you see: ... an axe, a baby chipmunk, a baby dragonfly ... a parrot egg</p> <p>Action4: focus on parrot egg (Thought: A parrot is a type of bird, and birds are known to live for many years. The life span of a parrot can be quite long, often exceeding 30 years in some species.)</p> <p>Observation4: You focus on the parrot egg. </p> <p>Action5: focus on baby dragonfly (Thought: Dragonflies are known to have a relatively short life span, typically ranging from a few weeks to a few months.)</p> <p>Observation5: You focus on the baby dragonfly.</p>	<p>Action1: look around</p> <p>Observation1: This room is called the art studio. In it, you see: the agent, a substance called air, a large cupboard.</p> <p>Action2: teleport to outside</p> <p>Observation2: You teleport to the outside.</p> <p>Action3: look around</p> <p>Observation3: Here you see: ... an axe, a baby chipmunk, a baby dragonfly ... a parrot egg</p> <p>Action4: focus on parrot egg (Thought: A parrot is a type of bird, and birds are known to live for many years. The life span of a parrot can be quite long, often exceeding 30 years in some species.)</p> <p>Observation4: You focus on the parrot egg. </p> <p>Action5: focus on baby chipmunk (Thought: I need to focus on the animal with the shortest life span, which is the baby chipmunk.)</p> <p>Observation5: You focus on the baby chipmunk .</p>
<p>GT Reward = 1.0 ARMAP Aborted</p>	<p>GT Reward = 0.83 ARMAP Selected </p>

Figure 17: **Failure Example from ScienceWorld**. Reward models have problems when common-sense knowledge is involved, such as the inability to determine the shortest life span.