

# AGENTGYM: EVALUATING AND EVOLVING LARGE LANGUAGE MODEL-BASED AGENTS ACROSS DIVERSE ENVIRONMENTS

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

Large language models (LLMs), with their generalized capabilities, are considered as a promising foundation to build generally-capable agents that can handle multi-turn decision-making tasks across various interactive environments. Previous attempts typically gather expert-provided trajectories and have LLM-based agents imitate these trajectories step-by-step. However, this supervised fine-tuning approach depends heavily on human supervision, limiting scalability and restricting the agent’s exploration and learning in the environments. In this paper, we take the first step towards developing generally-capable LLM-based agents that can explore and evolve themselves across diverse environments. To achieve this, we identify a trinity of ingredients: 1) diverse interactive environments for agent exploration, 2) a trajectory set to equip agents with basic capabilities and prior knowledge, and 3) an effective and scalable approach for agent improvement across environments. We propose AGENTGYM, a new interactive framework featuring various real-world scenarios and environments for broad, unified, real-time, and concurrent agent exploration. AGENTGYM also includes a database with expanded instructions, high-quality trajectories, and a benchmark suite. Next, we investigate the potential of agent self-evolution across various environments with a derived exploration-learning method named AGENTEVOL. Experimental results show that the evolved agents can achieve results comparable to SOTA models. We will release the code, dataset, benchmark, and checkpoints.

## 1 INTRODUCTION

Developing agents capable of performing a wide spectrum of tasks across various environments at human-level has been a long-standing goal for AI community (Wooldridge & Jennings, 1995; Silver et al., 2017; 2018; Reed et al., 2022; Xi et al., 2023). Large language models (LLMs) are considered a promising foundation for constructing such generalist agents due to their generalized abilities (OpenAI, 2023; Anthropic, 2024; Anil et al., 2023), and many efforts have been made in this realm to train generally-capable LLM-based agents (Xi et al., 2023; Wang et al., 2024b).

Previous work involves recruiting human experts to interact with various environments to collect high-quality multi-turn trajectories, which are then used to train agents to imitate these trajectories step by step through behavioral cloning (BC)<sup>1</sup>. This method, while effective, requires skilled annotators and significant financial resources, making it hard to scale (Yang et al., 2024). Moreover, such a paradigm may encounter bottlenecks in performance and adaptability due to insufficient exploration of the environment (Aksitov et al., 2023). Another line of research allows LLM-based agents to improve themselves based on environmental feedback (i.e., self-improvement), reducing reliance on human supervision while enriching exploration of the environment (Zhou et al., 2024; Tao et al., 2024; Song et al., 2024). Yet, they typically train agents in isolated environments, and the resulted specialist agents are limited to narrow tasks.

In this paper, we take the initial step to explore the potential of self-evolution in generally-capable LLM-based agents across various environments. We expect to let the agents first perform imitation

<sup>1</sup>In LLM field, the term behavioral cloning can be understood as supervised fine-tuning (SFT).

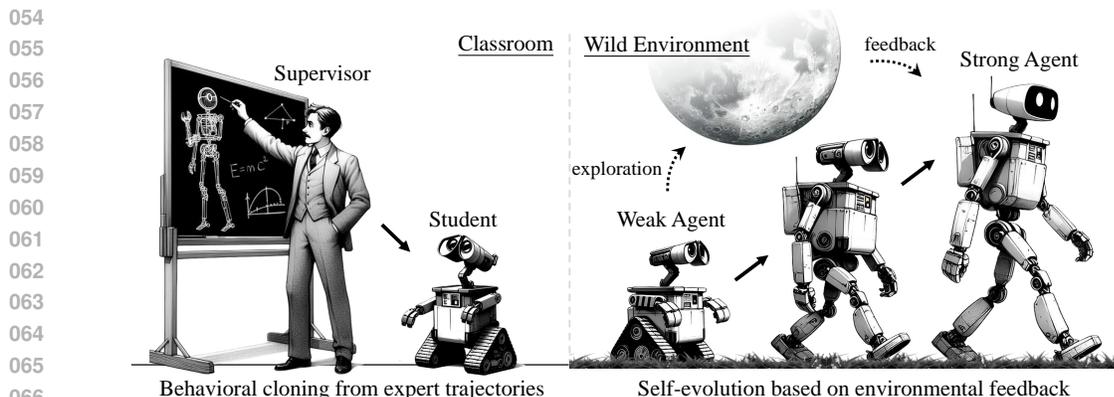


Figure 1: An illustration of self-evolution for generally-capable LLM-based agents in our paper. The agent first imitates human behavior and then performs exploration and learning across various environments and tasks to evolve itself.

and then interact with the environment to evolve themselves just like the way humans learn and develop (Standish, 2003; Taylor et al., 2016; Fan et al., 2022; Team et al., 2024), as shown in Figure 1. To achieve this research goal, we identify three key pillars. First, diverse environments and tasks that allow the agents to explore dynamically and comprehensively, rather than being confined to an isolated world (Standish, 2003; Langdon, 2005; Taylor et al., 2016; Fan et al., 2022). Second, a trajectory set to train a base agent with preliminary abilities and prior knowledge. This facilitates further exploration as in diverse, complex environments, it would be extremely inefficient for an agent to learn everything from scratch through trial and error (Fan et al., 2022; Song et al., 2024). Third, an effective and scalable method for the agents to perform self-improvement across diverse environments and tasks. This involves how the agent interacts with the environment and how it utilizes the feedback to improve itself (Yang et al., 2024; Aksitov et al., 2023).

Considering the three pillars, we present AGENTGYM (see Figure 2), a new framework designed to comprehensively evaluate and develop generally-capable LLM-based agents. Our main contributions are:

**1. An interactive framework that includes diverse scenarios and environments for LLM-based agents.** AGENTGYM offers convenient APIs through HTTP services, standardizing task specifications, environment settings, and the observation/action spaces for agents. Within this framework, we have implemented a unified interface for multi-round interactions and real-time feedback across different environments to support holistic evaluation, trajectory collection, and agent exploration. For comprehensiveness, AGENTGYM identifies 7 real-world scenarios critical for evaluating and developing agents, i.e., web navigating, text games, house-holding tasks, digital games, embodied tasks, tool-using, and programming. Specifically, it encompasses 14 environments and 89 tasks, with high flexibility for further expansion. The diversity of AGENTGYM not only challenges the agents’ multitasking abilities but also enhances their adaptability and practicality in real-world applications, laying the foundation for more robust and versatile LLM-based agents.

**2. Expanded instructions, benchmark suite, and high-quality trajectories across environments.** We collect instructions from various environments and tasks, expanding them through rule-based strategies and AI-based techniques such as self-instruct (Wang et al., 2023b) and instruction evolution (Xu et al., 2023). Subsequently, we leverage several principles to construct a benchmark suite named AGENTVAL to comprehensively evaluate LLM-based agents. Next, we use a gather-and-filter pipeline to get a trajectory set named AGENTTRAJ. The set is used to train a base agent with basic capabilities and prior knowledge, bootstrapping further agent exploration and learning. For a fair comparison, we also collect a larger trajectory set AGENTTRAJ-L with the same pipeline to train an agent that serves as the performance upper bound achievable through SFT/BC. Note that AGENTTRAJ-L is an extension of AGENTTRAJ and we will release it as well.

**3. Investigation of self-evolution for LLM-based agents across various environments with AGENTEVOL method.** Starting from the base agent, we explore its potential of self-improvement across various environments with a proposed method named AGENTEVOL, which is derived based on the classical *RL as Inference* framework (Dayan & Hinton, 1997). We then implement it in an

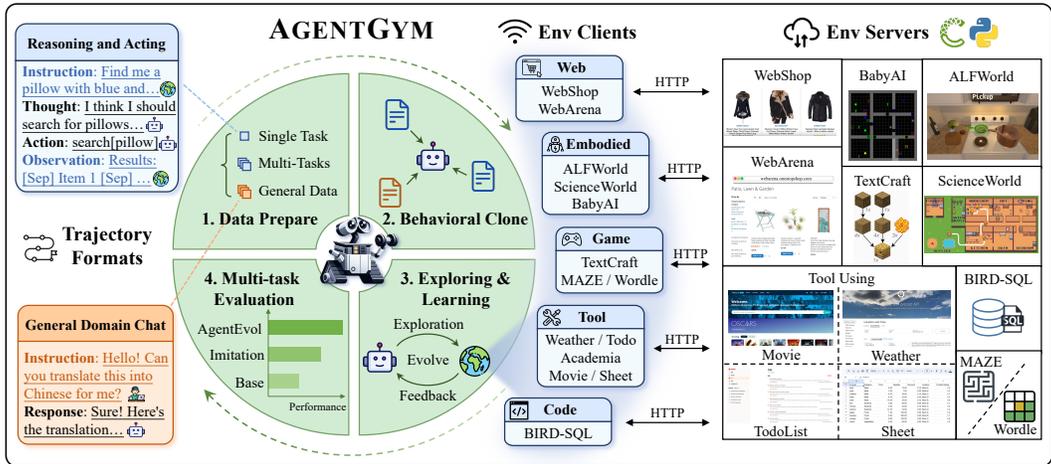


Figure 2: Overview of the AGENTGYM framework. It covers fourteen environments spanning diverse scenarios. It adopts a decoupled client-server architecture for unified and concurrent agent-environment interaction. AGENTGYM also includes expanded instructions, a comprehensive benchmark suite AGENTVAL, and the high quality trajectory set AGENTTRAJ (-L). We also derive and implement an exploration-learning method named AGENTEVOL to explore the agent’s self-evolution across various environments.

exploration-learning paradigm in practice. We focus on whether agents can evolve themselves when facing previously unseen tasks and instructions, where they need to perform exploration and learning. Experimental results demonstrate that the self-evolution process is highly effective, with agents achieving performance on par with or even surpassing SOTA models.

In summary, we propose AGENTGYM (See Figure 2 for an overview), a comprehensive framework that includes an interactive platform covering diverse scenarios, a benchmark suite AGENTVAL, and two trajectory sets AGENTTRAJ and AGENTTRAJ-L. We also derive and implement an exploration-learning method AGENTEVOL to explore self-evolution in LLM-based agents across various environments. We will release the whole suite, implementations, and checkpoints. We hope AGENTGYM will help the community to develop better generalist LLM-based agents.

## 2 AGENTGYM: FRAMEWORK ARCHITECTURE, INSTRUCTION SET, BENCHMARK SUITE, AND TRAJECTORY SET

AGENTGYM is a framework built for the community to facilitate the evaluation, training, and evolution of generally-capable LLM-based agents. It features diverse interactive environments and tasks with ReAct format (Yao et al., 2023). It supports real-time feedback and concurrency, and is easily scalable and extendable. The comparison between AGENTGYM and other LLM-based agent frameworks is demonstrated in Table 1.

Table 1: Comparison of AGENTGYM with other agent frameworks covers several aspects: the number of environments, presence of an interactive platform and its usage, availability of trajectory sets, support for evolution, and the evolution mode.

Frameworks	Env.	Inter. Plat.	Traj.	Evol.
AgentBench (Liu et al., 2023a)	8	Eval	No	No
AgentBoard Ma et al. (2024)	12	Eval	No	No
AgentOhana (Zhang et al., 2024)	10	No	Yes	No
Pangu-Agent Christianos et al. (2023)	6	No	Yes	Single-Env
AGENTGYM (Ours)	14	Eval & Train	Yes	Multi-Env

### 2.1 DIVERSE TARGETED ENVIRONMENTS AND TASKS FOR LLM-BASED AGENTS.

To ensure the comprehensiveness of the framework, we identify 7 real-world scenarios essential for evaluating and developing agents’ capabilities, including web navigating, text games, house-holding tasks, digital games, embodied tasks, tool-using, and programming. These scenarios are represented by 14 environments and 89 tasks in our framework, as shown in Table 2. We further elaborate on the motivation for selecting these diverse environments from two perspectives.

Table 2: Statistics of AGENTGYM, including scenarios, count of task types, evaluation metric, instruction set size, evaluation set size, trajectory set size (AGENTTRAJ and AGENTTRAJ-L), and the average interactive rounds of each environment in AGENTTRAJ-L.

Env.	Scenario	Task Num.	Eval. Metric	Inst. Size	Eval. Size	Traj. Size	Traj-L Size	Rounds
WebArena (WA, Zhou et al. 2023a)	Web Navigating	3	Success rate	812	20	0	0	–
WebShop (WS, Yao et al. 2022)	Web Navigating	1	Success rate	6910	200	1000	3930	5.1
MAZE (MZ, Abdulhai et al. 2023b)	Text Game	1	Success rate	240	25	100	215	4.3
Wordle (WD, Abdulhai et al. 2023b)	Text Game	1	Success rate	980	25	500	955	4.3
ALFWorld (ALF, Shridhar et al. 2021)	House-holding	6	Success rate	3827	200	500	2420	13.3
SciWorld (Sci, Wang et al. 2022)	Embodied Tasks	30	Reward	2320	200	1000	2120	19.9
BabyAI (Baby, Chevalier-Boisvert et al. 2019)	Embodied Tasks	40	Reward	900	90	400	810	5.7
TextCraft (TC, Prasad et al. 2023)	Digital Game	1	Success rate	544	100	300	374	8.0
Tool-Weather (WT, Ma et al. 2024)	Tool Use	1	Success rate	331	20	160	311	5.5
Tool-Movie (MV, Ma et al. 2024)	Tool Use	1	Success rate	235	20	100	215	4.0
Tool-Academia (AM, Ma et al. 2024)	Tool Use	1	Success rate	20	20	0	0	–
Tool-Sheet (ST, Ma et al. 2024)	Tool Use	1	Reward	20	20	0	0	–
Tool-TODOList (TL, Ma et al. 2024)	Tool Use	1	Success rate	155	20	70	135	5.6
BIRD (BD, Zheng et al. 2023a)	Programming	1	Success rate	3200	200	2000	3000	1.0
Total	–	89	–	20494	1160	6130	14485	–

**Starting from the definition of an LLM-based agent.** An LLM-based agent refers to an agent with a decision-making core based on a large language model, extending its input and action (Wooldridge & Jennings, 1995; Xi et al., 2023; Wang et al., 2024b). Consequently, the capabilities required for an LLM-based agent include:

- **Input Side.** An effective agent must demonstrate the ability to process diverse textual observations, such as plain text, HTML, code, etc. Thus, the environments and tasks in AGENTGYM are designed to encompass a wide range of input types. For instance, we include textual environments (ALF, TC), web-based environments (WS, WA), and coding environments (BD).
- **Decision-making Side.** At the core of an LLM-based agent lies its reasoning, planning and environmental understanding capabilities. Therefore, the tasks within AGENTGYM are crafted to be challenging and rigorous. For example, in the WS task, when interacting with a web page containing many products, an LLM-based agent needs to guide its next actions through reasoning and information extraction. Similarly, in the Sci task, an agent needs to read manuals, find raw materials, and synthesize items when facing observations from different rooms. These tasks demand not only basic common knowledge but also advanced planning and situational understanding capabilities.
- **Action Side.** An LLM-based agent’s outputs can take various forms, such as plain text, code, API calls and embodied actions. As a result, the environments must be equipped to accommodate and process these different types of actions. For example, WT and MV environments offer a variety of tool-calling APIs for the agent; the DB environment requires the generation of SQL code; Baby and ALF involve embodied actions, while WD focuses on producing plain text output.

**Starting from the expectation for an LLM-based agent.** An LLM-based agent is expected to tackle complex, long-term, multi-turn sequential decision-making tasks by dynamically interacting with the environment (Yao et al., 2023; Sumers et al., 2024; Zhou et al., 2024). In this context, the environments and tasks included in AGENTGYM are designed to embody these characteristics. Specifically, the tasks require multiple rounds of interaction with the environment, and the context length is usually longer than those of LLM’s Q&A or reasoning tasks. To accomplish these tasks, the agent needs to process not only instructions but also inputs derived from historical information and current observations. From this perspective, AGENTGYM is a framework specifically designed for LLM-based agents.

## 2.2 FRAMEWORK ARCHITECTURE AND DATABASE CONSTRUCTION.

We employ a decoupled architecture in AGENTGYM to facilitate standardized and scalable agent-environment interactions. It supports functions like creating environments, retrieving observations and performing actions via HTTP-based communication. See Appendix C for more details.

Regarding database construction, we first gather 20,494 instructions using rule-based generation and self-instruction techniques. Then, we construct a benchmark suite with a size of 1,160 named

AGENTEVAL to holistically evaluate the capabilities of LLM-based agents. As for the trajectory set, we collect and filter 6,130 high-quality trajectories from 11 environments with various strategies, including but not limited to crowdsourcing and annotations by SOTA models (e.g., GPT-4-Turbo). This set, named AGENTTRAJ, is used to train a base agent with preliminary abilities and prior knowledge. For a fair comparison, we also perform the same annotation pipeline on all instructions to get AGENTTRAJ-L, which represents the performance upper bound of BC. More collection and annotating details are presented in Appendix B and D.

### 3 EVOLVING OF LLM-BASED AGENTS ACROSS VARIOUS ENVIRONMENTS

In this section, we explore the potential of agents to evolve themselves across multiple environments. We begin by training a base agent using BC to equip it with basic interaction capabilities. Then, based on the *RL as Inference* theory (Dayan & Hinton, 1997), we derive and implement an exploration-learning method, AGENTEVOL, for multi-turn interaction scenarios, enabling agents to improve themselves across various environments. The method is summarized in Algorithm 1.

#### 3.1 PRELIMINARIES

We define the collection of environments as  $\mathcal{E}$ . For a specific  $e \in \mathcal{E}$ , we formalize the agent task in the environment as a partially observable Markov decision process (POMDP)  $(\mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, r)_e$  with instruction space  $\mathcal{U}$ , state space  $\mathcal{S}$ , action space  $\mathcal{A}$ , observation space  $\mathcal{O}$ , deterministic state transition function  $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ , and reward function  $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ .

Given a task instruction  $u$  in environment  $e$ , the LLM-based agent parameterized by  $\theta$  generates an action  $a_1 \sim \pi_\theta(\cdot|e, u)$  based on its policy  $\pi_\theta$ . Then, the state space is transitioned to  $s_1 \in \mathcal{S}$ , and the agent receives feedback  $o_1 \in \mathcal{O}$ . Subsequently, the agent interacts with the environment until the task ends or exceeds the maximum number of steps. We adopt ReAct (Yao et al., 2023) to model the outputs of agent, where the LLM-based agent generates a reasoning thought before outputting an action. Thus, at time step  $t$ , given the history and current feedback, the agent generates the thought  $h_{t+1} \sim \pi_\theta(\cdot|e, u, h_1, a_1, o_1, \dots, h_t, a_t, o_t)$  first and the subsequent action  $a_{t+1} \sim \pi_\theta(\cdot|e, u, h_1, a_1, o_1, \dots, h_t, a_t, o_t, h_{t+1})$ . Hence, the trajectory can be represented as:

$$\tau = (h_1, a_1, o_1, \dots, o_{T-1}, h_T, a_T) \sim \pi_\theta(\tau|e, u), \quad (1)$$

$$\pi_\theta(\tau|e, u) = \prod_{t=1}^T \pi_\theta(h_t, a_t|e, u, c_{t-1}) = \prod_{t=1}^T \pi_\theta(a_t|e, u, c_{t-1}, h_t) \cdot \pi_\theta(h_t|e, u, c_{t-1}), \quad (2)$$

where  $T$  is the number of interaction rounds, and  $c_{t-1} = (h_1, a_1, o_1, \dots, h_{t-1}, a_{t-1}, o_{t-1})$  represents the interactive history up to  $t-1$ . The final reward  $r(e, u, \tau) \in [0, 1]$  is computed after the interaction ends or the maximum interactive round number is met.

#### 3.2 BEHAVIORAL CLONING WITH COLLECTED TRAJECTORIES

In diverse and complex environments, learning everything from scratch through trial and error is inefficient for LLM-based agents due to the vast action space of language space and the varying prior knowledge across different environments (Fan et al., 2022; Song et al., 2024). Hence, we employ the behavioral cloning (BC) method to train a base agent using AGENTTRAJ, providing it with basic interaction capability and prior knowledge. Specifically, BC fine-tunes LLM-based agents by having them mimic the expert trajectories step-by-step. In practice, we expect the agent to generate both appropriate thought  $h$  and action  $a$ . So we maximize the following objective:

$$\begin{aligned} \mathcal{J}_{BC}(\theta) &= \mathbb{E}_{(e, u, \tau) \sim \mathcal{D}_s} \left[ \log \pi_\theta(\tau|e, u) \right] \\ &= \mathbb{E}_{(e, u, \tau) \sim \mathcal{D}_s} \sum_{t=1}^T \left[ \log \pi_\theta(a_t|e, u, c_{t-1}, h_t) + \log \pi_\theta(h_t|e, u, c_{t-1}) \right]. \end{aligned} \quad (3)$$

Note that we include a general domain dataset  $\mathcal{D}_{general}$  as in Zeng et al. (2023) to maintain the agent’s ability in language understanding and generation. And the resulting agent  $\pi_{\theta_{base}}$  serves as a starting point for later exploration and learning across diverse environments.

### 3.3 EVOLUTION THROUGH EXPLORATION AND LEARNING

To achieve self-evolution, agents need to explore environments, receive feedback, and improve themselves based on the feedback, as humans do (Standish, 2003; Taylor et al., 2016; Fan et al., 2022; Team et al., 2024). More importantly, the agents may face previously unseen tasks and instructions during exploration. Standard reinforcement learning (RL) algorithms, such as Proximal Policy Optimization (PPO) (Schulman et al., 2017), are worth considering due to their excellent performance in alignment of LLMs (Bai et al., 2022). However, in the setting of multi-turn decision-making for LLM-based agents, PPO faces significant challenges, e.g., large action space, long interaction chains and sparse reward signals, leading to high computational complexity and training instability (Huang et al., 2022; Zheng et al., 2023b; Xi et al., 2024).

Hence, we draw inspiration from the well-established connection between RL and probabilistic inference, i.e., *RL as Inference* (Dayan & Hinton, 1997; Levine & Koltun, 2013; Neumann, 2011; Rawlik et al., 2013; Abdolmaleki et al., 2018; Singh et al., 2023), and derive the exploration-learning method called AGENTEVOL for evolution in multi-turn decision-making scenarios, which involves agents alternating between exploration and learning across various environments.

**Theoretical foundation: learning from the estimated optimal policy using the classical *RL as Inference* framework.** We view RL as an inference problem within a specific probabilistic model (Dayan & Hinton, 1997). Differing from traditional RL formulations that focus on identifying a trajectory that maximizes the expected reward, inference-based approaches start with an optimal distribution over trajectories. Following previous work (Toussaint, 2009; Levine & Koltun, 2013; Abdolmaleki et al., 2018), we initially define  $P(O = 1)$  to represent the event of “obtained optimal policy by maximum expected rewards” or “achieving success in the RL task”, which can be calculated by integrating the optimal policy probability at each sampling point. Given the policy agent  $\pi_\theta$ , the optimal policy can be obtained by maximizing:

$$\log P_{\pi_\theta}(O = 1) = \log \int \pi_\theta(\tau) p(O = 1|\tau) d\tau. \quad (4)$$

However, the above optimization process is difficult to proceed directly due to the fact that LLM-based agents require token-wise feedback to perform gradient updates. In this paper, we alternatively construct the variational lower bound of Eq.4 by introducing an estimation function  $q$  on the optimal policy. With Jensen’s inequality, we soon have:

$$\begin{aligned} \log \int \pi_\theta(\tau) p(O = 1|\tau) d\tau &= \log \mathbb{E}_{q(\tau)} \left[ \frac{\pi_\theta(\tau)}{q(\tau)} p(O = 1|\tau) \right] \geq \mathbb{E}_q \left[ \log \frac{\pi_\theta(\tau)}{q(\tau)} p(O = 1|\tau) \right] \\ &= \mathbb{E}_q \left[ \log p(O = 1|\tau) \right] - \text{KL}[q(\tau) || \pi_\theta(\tau)] = \mathcal{J}(q, \pi_\theta), \end{aligned} \quad (5)$$

where  $\pi_\theta$  is the trajectory distribution induced by the agent, and  $q(\tau)$  is a variational distribution.

Due to the monotonicity of the logarithmic function, by maximizing the lower bound  $\mathcal{J}(q, \pi_\theta)$ , we can obtain a policy with an expected return higher than before. Generally, our framework can be divided into two steps of loop iteration. **The first part** of  $\mathcal{J}(q, \pi_\theta)$  can be explained as estimating the optimal policy distribution on the sampled trajectories by maximizing the expected reward over the state space. **The second part** relates to updating the current agent’s parameters  $\theta$  towards the optimal policy  $q$ , thus completing the optimization of one single iteration. In analogy to SGD (Robbins & Monro, 1951), the estimation process introduces noise to the policy optimization due to the presence of unseen decision-making trajectories. This error gradually decreases as the optimization proceeds and converges to zero when the current agent becomes optimal (Dayan & Hinton, 1997).

In AGENTEVOL, we refer to the two steps as **Exploration step** and **Learning step** like Singh et al. (2023). Specifically, with current agent parameters  $\theta^m$  and the variational distribution  $q^m$ , at **exploration step**, the estimation of optimal policy  $q$  is updated by maximizing the expected reward:  $q^{m+1} = \arg \max_q \mathcal{J}(q, \pi_{\theta^m})$ . As  $\max_q \mathcal{J}(q, \pi_{\theta^m}) = \min_q [\text{KL}(q(\tau) || p(O = 1|\tau)\pi_{\theta^m}(\tau))]$ , we have  $q^{m+1} \propto p(O = 1|\tau)\pi_{\theta^m}(\tau)$ . This step is equivalent to evaluating the likelihood that the samples generated from the current agent’s policy achieve best rewards, and observe the returns of  $q$  by empirically estimating on a pre-constructed training set. And at **learning step**, we optimize  $\mathcal{J}(q^{m+1}, \pi_\theta)$  by updating  $\theta$ . This process is similar to learning a new distribution sampled from the optimal policy on the original training data. Since the first term of  $\mathcal{J}(q^{m+1}, \pi_\theta)$  relates only to  $q$  as

**Algorithm 1:** AGENTEVOL

**Input:** Initialized policy LLM-based agent  $\pi_\theta$ , environment set  $\mathcal{E}$ , trajectory subset  $\mathcal{D}_s$ , full instruction set  $\mathcal{Q}$ , reward function  $r$ .

**Procedure** Behavioral cloning:

Maximize objective  $\mathcal{J}_{BC}(\theta) = \mathbb{E}_{(e,u,\tau) \sim \mathcal{D}_s} [\log \pi_\theta(\tau|e,u)]$  to get  $\pi_{\theta_{base}}$ ;

**Procedure** Evolution :

$\pi_{\theta^1} \leftarrow \pi_{\theta_{base}}$ ;

**for** iteration  $m = 1$  to  $M$  **do**

  // Perform Exploration Step

$\mathcal{D}_m = \bigcup_{e \in \mathcal{E}} \mathcal{D}_m^e$ , where  $\mathcal{D}_m^e = \{(e, u^j, \tau^j) \mid u^j \sim \mathcal{Q}_e, \tau^j \sim \pi_{\theta^m}(\tau|e, u^j)\}_{j=1}^{|\mathcal{D}_m^e|}$ ;

  Compute reward for  $\mathcal{D}_m$  with  $r$ ;

$\mathcal{D}_m \leftarrow \mathcal{D}_m \cup \mathcal{D}_s$ ;

  // Perform Learning Step

  Maximize objective  $\mathcal{J}_{Evol}(\theta) = \mathbb{E}_{(e,u,\tau) \sim \mathcal{D}_m} [r(e, u, \tau) \log \pi_\theta(\tau|e, u)]$  to get  $\pi_{\theta^{m+1}}$ ;

**end**

well as  $\tau$ , the training objective is equivalent to measuring the KL divergence between the estimated policy  $q^{m+1}(\cdot)$  and the current policy  $\pi_\theta(\cdot)$  over all training samples. We finally derive:

$$\theta^{m+1} := \arg \min_{\theta} \text{KL}[q^{m+1}(\tau) \parallel \pi_\theta(\tau)] = \arg \min_{\theta} \sum_{\tau} -q^{m+1}(\tau) \log \pi_\theta(\tau). \quad (6)$$

This involves optimizing a weighted negative log-likelihood function based on  $q^{m+1}$ , which adjusts the agent policy to increase the likelihood of generating higher-reward trajectories, thereby improving the agent’s performance.

**Practical implementation for evolving LLM-based agents.** In our LLM-based agent setting, the trajectory is conditioned on the environment  $e$  and instruction  $u$ . Considering our non-negative reward function  $r(e, u, \tau)$ , we can get  $P(O|e, u, \tau) \propto r(e, u, \tau)$  (Singh et al., 2023). Consequently,  $q^{m+1}(\tau|e, u) \propto r(e, u, \tau) \cdot \pi_{\theta^m}(\tau|e, u)$ . Thus, the policy update in the learning is:

$$\begin{aligned} \theta^{m+1} &:= \arg \min_{\theta} \sum_{\tau} -(r(e, u, \tau) \cdot \pi_\theta(\tau|e, u)) \log \pi_\theta(\tau|e, u) \\ &= \arg \max_{\theta} \mathbb{E}_{e \in \mathcal{E}, u \sim \mathcal{Q}_e, \tau \sim \pi_{\theta^m}(\tau|e, u)} [r(e, u, \tau) \log \pi_\theta(\tau|e, u)]. \end{aligned} \quad (7)$$

This can be viewed as a supervised fine-tuning objective weighted by reward. This approach uses the fixed policy agent from the previous iteration to sample data, thereby separating data collection and policy optimization. This decoupled process can improve the training stability (Zelikman et al., 2022; Singh et al., 2023).

Now we describe the two steps of evolution part in AGENTEVOL in practice:

**Exploration Step.** In the  $m$ -th exploring iteration, for each environment  $e$ , we have an instruction set  $\mathcal{Q}_e$  which is larger than that used in the BC phase, allowing us to investigate agents evolving to unseen tasks and instructions. The current policy agent interacts with this environment, generating a collection of interaction trajectories  $\mathcal{D}_m^e = \{(e, u^j, \tau^j) \mid u^j \sim \mathcal{Q}_e, \tau^j \sim \pi_{\theta^m}(\tau|e, u^j)\}_{j=1}^{|\mathcal{D}_m^e|}$ . Subsequently, based on the reward function of the environment, we calculate the reward  $r(e, u, \tau)$  for each trajectory. The generated dataset from each environment is then merged, resulting in  $\mathcal{D}_m = \bigcup_{e \in \mathcal{E}} \mathcal{D}_m^e$ . Note that we also include the original trajectory set in Section 3.2 for the learning step, i.e.,  $\mathcal{D}_m = \mathcal{D}_m \cup \mathcal{D}_s$ .

**Learning Step.** In the  $m$ -th learning iteration, we utilize the dataset  $\mathcal{D}_m$  obtained from the exploration step to fine-tune the agent with the objective  $\mathcal{J}_{Evol}(\theta) = \mathbb{E}_{(e,u,\tau) \sim \mathcal{D}_m} [r(e, u, \tau) \log \pi_\theta(\tau|e, u)]$  to get  $\pi_{\theta^{m+1}}$ . We also include the general domain dataset as in the BC phase. We optimize the initial agent  $\pi_\theta$  at each iteration, aiming to minimize overfitting and prevent drift from the base agent. In this learning step, the agent is improved, similar to previous work done on LLM reasoning Zelikman et al. (2022); Singh et al. (2023); Aksitov et al. (2023).

By alternating between the two steps, empirical results show that the method facilitates the evolution of an LLM-based agent across both seen and unseen tasks and instructions.

Table 3: Evaluating results on diverse tasks.  $BC_{base}$  means the agent trained with AGENTTRAJ, providing a base agent with basic ability and prior knowledge.  $BC_{large}$  means the agent that performs BC on AGENTTRAJ-L, representing the performance upper limit of BC in this paper. Our evolution method, AGENTEVOL, outperforms  $BC_{large}$  on most tasks and environments through exploration and learning. The best performance of each part is highlighted in **bold**.

Method	WS	ALF	TC	Sci	Baby	MZ	WD	WT	MV	TL	BD
<b>Close-sourced Models &amp; Agents</b>											
DeepSeek-Chat	11.00	51.00	23.00	<b>16.80</b>	45.67	4.00	24.00	70.00	70.00	75.00	13.50
Claude-3-Haiku	5.50	0.00	0.00	0.83	1.93	4.00	16.00	55.00	50.00	65.00	13.50
Claude-3-Sonnet	1.50	13.00	38.00	2.78	<b>79.25</b>	0.00	36.00	65.00	80.00	80.00	<b>17.00</b>
GPT-3.5-Turbo	12.50	26.00	47.00	7.64	71.36	4.00	20.00	25.00	70.00	40.00	12.50
GPT-4-Turbo	<b>15.50</b>	<b>67.50</b>	<b>77.00</b>	14.38	72.83	<b>68.00</b>	<b>88.00</b>	<b>80.00</b>	<b>95.00</b>	<b>95.00</b>	16.00
<b>Open-sourced Models &amp; Agents</b>											
Llama2-Chat-7B	0.50	2.00	0.00	0.83	0.23	0.00	0.00	0.00	0.00	0.00	1.50
Llama2-Chat-13B	1.00	3.50	0.00	0.83	0.10	0.00	0.00	0.00	0.00	0.00	1.50
AgentLM-7B	36.50	71.00	<b>4.00</b>	1.63	0.49	<b>12.00</b>	4.00	0.00	<b>5.00</b>	15.00	5.00
AgentLM-13B	39.50	<b>73.00</b>	0.00	2.75	0.45	8.00	0.00	<b>10.00</b>	<b>5.00</b>	5.00	3.00
AgentLM-70B	<b>49.50</b>	67.00	<b>4.00</b>	<b>10.68</b>	<b>0.66</b>	8.00	4.00	0.00	0.00	<b>40.00</b>	<b>7.50</b>
<b>Ours</b>											
$BC_{base}$	66.50	77.50	44.00	26.42	69.31	<b>12.00</b>	12.00	25.00	5.00	45.00	8.00
$BC_{large}$	73.50	83.00	60.00	<b>74.47</b>	74.19	<b>12.00</b>	<b>36.00</b>	<b>45.00</b>	5.00	65.00	8.50
<b>AGENTEVOL</b>	<b>76.50</b>	<b>88.00</b>	<b>64.00</b>	38.00	<b>82.70</b>	<b>12.00</b>	12.00	25.00	<b>60.00</b>	<b>70.00</b>	<b>9.00</b>

## 4 EXPERIMENTS AND DISCUSSION

### 4.1 EXPERIMENTAL SETUP

**Environments and Tasks.** We explore the self-evolution of LLM-based agents with the AGENTGYM framework. Main experiments cover the following environments: WS, ALF, Sci, Baby, TC, BD, MZ, WD, TL, WT, and MV. Note that instructions used in BC are fewer than those in evolution, to study the agent’s ability to generalize when performing exploration.

**Baselines.** We include closed-source models like GPT-3.5-Turbo (Ouyang et al., 2022), GPT-4-Turbo (OpenAI, 2023), Claude 3 (Anthropic, 2024), and DeepSeek-Chat (DeepSeek-AI, 2024). We also include open-source models like Llama-2-Chat (Touvron et al., 2023), and agents trained on expert trajectories, i.e., AgentLM (Zeng et al., 2023). For a fair comparison, we include a baseline that performs BC on AGENTTRAJ-L, serving as the maximum performance achievable through BC.

**Implementation Details.** Experiments are conducted with eight A100-80GB GPUs. Our main backbone model is Llama-2-Chat-7B. Different environment services are deployed on different ports of the same server. We set the iteration number  $M$  to 4. Each instruction is sampled once during the evolution process for efficiency. Note that some environments provide dense rewards  $r \in [0, 1]$ , while others give only binary feedback  $r \in \{0, 1\}$ . For simplicity and consistency, we follow previous work (Singh et al., 2023) and use binary rewards. We set  $r = 0$  for trajectories where  $r < 1$ , while for those with  $r = 1$ , we keep it unchanged. See Appendix E for more implementation details. Detailed prompts for each environment are in Appendix G.

### 4.2 MAIN RESULTS

Experiment results in Table 3 demonstrate that: (1) While closed-source models perform well, even SOTA closed-source models like GPT-4-Turbo fail to achieve satisfactory performance on all tasks, highlighting the need for developing more capable agents. (2) Open-source models, represented by Llama2-Chat, perform poorly on all tasks, highlighting the importance of the initialization step of BC. (3) Models trained on agent trajectories, like AgentLM (Zeng et al., 2023), can perform on par with GPT-4-Turbo on many tasks, particularly the 70B version. However, they do not match performance on tasks like TextCraft (Prasad et al., 2023) or SciWorld (Wang et al., 2022), which can be attributed to the lack of training data. (4) The agent trained on AGENTTRAJ-L, i.e.,  $BC_{large}$ , achieves excellent performance, matching or even surpassing SOTA models, showing that it is a strong baseline. (5) AGENTEVOL, despite having limited trajectories for imitation, surpasses  $BC_{large}$  and SOTA models on many tasks like WebShop (Yao et al., 2022), ALFWorld (Shridhar et al., 2021) and BabyAI (Chevalier-Boisvert et al., 2019), validating the superiority and promise of agent evolution.

432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485

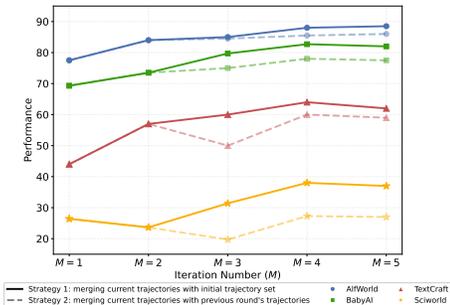


Figure 3: Ablation on data merging strategies and iteration number  $M$ .

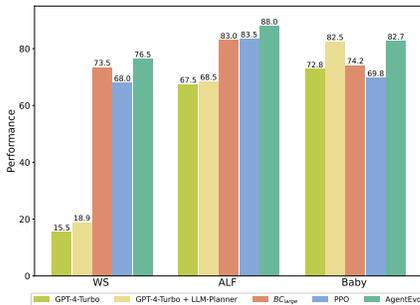


Figure 4: Comparison with other exploration-based methods.

Moreover, we report the number of interactive rounds required by different models to solve the task, in order to demonstrate the efficiency of AGENTEVOL (Appendix F.1). We also conduct experiments to explore evolution in isolated environments (Appendix F.4) and perform case study (Appendix F.6).

### 4.3 DISCUSSION & ANALYSIS

**Ablation on data merging strategies and iteration number  $M$ .** In our experiments, we merge the trajectories sampled during each iteration with the initial trajectories for training, rather than merging it with the trajectories generated in the previous iteration. Here, we conduct an ablation study to show the impact of this merging strategy and the iteration number  $M$ . Experimental results in Figure 3 show that merging with the initial data provides more stable improvements, while merging with the trajectories from the previous iteration leads to performance fluctuations, possibly due to overfitting (Yuan et al., 2023b; Singh et al., 2023). Additionally, as  $M$  increases, performance improves but gradually converges in later iterations. So we choose  $M = 4$  to balance performance and efficiency. More ablation studies can be found in Appendix F.2.

**Comparison with other exploration-based methods.** The AGENTEVOL method explores and learns through interactions with environments. For comparison, we include two additional methods: online RL method PPO (Schulman et al., 2017) and prompt-based method LLM-Planner (Song et al., 2023). PPO employs the same backbone as AGENTEVOL, Llama-2-Chat-7B, whereas LLM-Planner is implemented with GPT-4-Turbo to showcase its effectiveness. We run PPO in the isolated environment, as it tends to experience training instability and show poorer performance when applied across multiple environments. Results in Figure 4 show that (1) LLM-Planner outperforms GPT-4-Turbo but falls short on WS, and overall, it doesn’t surpass AGENTEVOL. (2) PPO performs well but lags behind AGENTEVOL by a significant margin. These findings indicate that AGENTEVOL consistently improves performance for smaller open-source models compared to larger SOTA models.

**Evolution with both successful and failed trajectories.** In the learning step, we only utilize the sampled trajectories with high rewards (success) and do not use failed trajectories. Inspired by previous work (Mitra et al., 2024; Hosseini et al., 2024; Song et al., 2024; Yang et al., 2024; Wang et al., 2023a), we explore whether failed trajectories can be included for better evolution. Specifically, we construct pairs of successful and failed trajectories and optimize the agent using the DPO method (Rafailov et al., 2023), which fits models to the pair-wise dataset (Wang et al., 2023a; Mitra et al., 2024; Lai et al., 2024). Results in Table 4 show that using both types of trajectories can still bring about evolutionary effects, but the performance is not as good as our method, indicating that preference optimization in multi-task setting is more challenging compared to single-task (Mitra et al., 2024; Song et al., 2024). In the future, we hope to explore more advanced algorithms and methods to make full use of all trajectories for comprehensive evolution.

**Effectiveness on different models.** To demonstrate the generalizability of our method across different backbone models, we conduct experiments on Llama-2-13B (Touvron et al., 2023) and DeepSeek-Coder-1.3B (Guo et al., 2024). The experimental results in Table 5 show that our AGENTEVOL maintains its evolutionary capabilities across different backbone models, achieving performance that is comparable to or surpasses BC<sub>Large</sub>.

Table 4: Experiments on evolution with both successful and failed trajectories.

Method	WS	ALF	Baby	TC
BC <sub>base</sub>	66.5	77.5	69.3	44.0
AGENTEVOL	<b>77.0</b>	<b>88.0</b>	<b>82.9</b>	<b>65.0</b>
DPO with failed traj	75.0	86.5	78.3	58.0

Table 5: Effectiveness of AGENTEVOL on different models.

Model	Method	WS	ALF	Baby	TC
DeepSeek-Coder-1.3B	BC <sub>base</sub>	54.0	33.0	68.9	31.0
	BC <sub>large</sub>	65.0	<b>62.5</b>	73.8	37.0
	AGENTEVOL	<b>67.5</b>	54.5	<b>77.3</b>	<b>38.0</b>
Llama2-Chat-13B	BC <sub>base</sub>	65.5	81.5	76.6	59.0
	BC <sub>large</sub>	74.0	85.0	81.1	61.0
	AGENTEVOL	<b>78.5</b>	<b>89.5</b>	<b>86.8</b>	<b>71.0</b>

## 5 RELATED WORK

With the development of LLMs (OpenAI, 2023; Anil et al., 2023), developing agents based on them has become an important research direction (Xi et al., 2023; Wang et al., 2024b). These agents are typically designed to perform multi-turn decision-making tasks (Yao et al., 2023; Aksitov et al., 2023; Chen et al., 2023). To evaluate these agents, researchers have proposed various benchmarks (Yao et al., 2022; Liu et al., 2023a; Ma et al., 2024; Zhou et al., 2023b). Our benchmark AGENTEVOL covers more diverse scenarios and environments, providing a more comprehensive evaluation.

Closed-source LLMs, equipped with prompting methods like ReAct (Yao et al., 2023) and PlanAct (Liu et al., 2023b), can achieve great performance in agent tasks, while agents based on open-source methods perform poorly on these tasks (Liu et al., 2023a; Christianos et al., 2023). To address this challenge, a series of work collects expert trajectories from diverse environments and tasks and trains LLM-based agents through behavioral cloning (Zeng et al., 2023; Chen et al., 2023; 2024; Zhang et al., 2024). However, obtaining these expert trajectories is often costly and they lack sufficient exploration of the environment (Yang et al., 2024; Aksitov et al., 2023).

Another line of work trains LLM-based agents based on environmental feedback, referred to as interactive learning methods (Zhou et al., 2024; Christianos et al., 2023; Song et al., 2024; Abdulhai et al., 2023a). Specifically, they involve training LLMs or agents through exploration and learning. As a representative method, RL has succeeded in LLM alignment (Askell et al., 2021; Bai et al., 2022; Ouyang et al., 2022; Zheng et al., 2023b; Wang et al., 2024a), and has been introduced to reasoning and agent tasks, achieving excellent results (Xi et al., 2024; Luong et al., 2024; Zhou et al., 2024; Christianos et al., 2023). However, in our multi-environment scenarios, reward consistency and training stability can become problematic (Zhou et al., 2024; Song et al., 2024; Cao et al., 2024). Another line of work uses self-evolution/self-improvement, where the model explores the environment to obtain high-reward trajectories and fine-tunes itself based on these trajectories, achieving promising performance in reasoning, coding, and web tasks (Gülçehre et al., 2023; Singh et al., 2023; Zelikman et al., 2022; Yuan et al., 2023a; Aksitov et al., 2023; Yang et al., 2024; Song et al., 2024; Tao et al., 2024; Tian et al., 2024; Lai et al., 2024). However, like RL-based methods, these works only explore within a single environment. With AGENTGYM, our work explores agent evolution using AGENTEVOL method, conducting exploration across multiple environments.

## 6 CONCLUSION

In this work, we present a new framework named AGENTGYM that encompasses 14 interactive environments and 89 tasks, covering 7 key scenarios for agent evaluation and development. It also includes expanded instructions, a comprehensive benchmark named AGENTEVOL, and trajectory sets called AGENTTRAJ and AGENTTRAJ-L. Additionally, we derive and implement an exploration-learning method AGENTEVOL to investigate the self-evolution of LLM-based agents across multiple environments. Empirical results demonstrate the effectiveness of the framework and the method. We also perform sufficient ablation and analysis to investigate how the method works. We hope our work can help the AI community develop more advanced generalist LLM-based agents.

## ETHICS STATEMENT

AGENTGYM and AGENTEVOL facilitate the self-evolution of generally-capable agents, and our focus is on the self-evolution of decision-making capabilities, but it is crucial to consider safety and ethical issues during usage. Agents must not be allowed to violate human values. Therefore, it is

540 essential to strengthen supervision and regulation when eliciting agents’ self-evolution capabilities.  
541 In the future, we hope to improve the framework’s functionality to align agents with human values.  
542

## 543 REPRODUCIBILITY STATEMENT

544 We upload anonymized versions of our data and code in a Zip file with a Readme file. We have  
545 listed our implementation details in Section 4.1 and Appendix E. We will release the code, dataset,  
546 benchmark, and checkpoints.  
547

## 548 REFERENCES

- 549
- 550 Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Rémi Munos, Nicolas Heess, and Mar-  
551 tin A. Riedmiller. Maximum a posteriori policy optimisation. In *6th International Conference*  
552 *on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018,*  
553 *Conference Track Proceedings*. OpenReview.net, 2018. URL [https://openreview.net/](https://openreview.net/forum?id=SLANxQW0b)  
554 [forum?id=SLANxQW0b](https://openreview.net/forum?id=SLANxQW0b).  
555
- 556 Marwa Abdulhai, Isadora White, Charlie Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin  
557 Xu, and Sergey Levine. LMRL gym: Benchmarks for multi-turn reinforcement learning with  
558 language models. *CoRR*, abs/2311.18232, 2023a. doi: 10.48550/ARXIV.2311.18232. URL  
559 <https://doi.org/10.48550/arXiv.2311.18232>.  
560
- 561 Marwa Abdulhai, Isadora White, Charlie Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu,  
562 and Sergey Levine. Lmrl gym: Benchmarks for multi-turn reinforcement learning with language  
563 models, 2023b.
- 564 Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu,  
565 Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, Manzil Zaheer, Felix X. Yu,  
566 and Sanjiv Kumar. Rest meets react: Self-improvement for multi-step reasoning LLM agent.  
567 *CoRR*, abs/2312.10003, 2023. doi: 10.48550/ARXIV.2312.10003. URL [https://doi.org/](https://doi.org/10.48550/arXiv.2312.10003)  
568 [10.48550/arXiv.2312.10003](https://doi.org/10.48550/arXiv.2312.10003).  
569
- 570 Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Jo-  
571 han Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin  
572 Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Tim-  
573 othy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald  
574 Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan  
575 Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha  
576 Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Dani-  
577 helka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati,  
578 Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. Gemini: A fam-  
579 ily of highly capable multimodal models. *CoRR*, abs/2312.11805, 2023. doi: 10.48550/ARXIV.  
580 2312.11805. URL <https://doi.org/10.48550/arXiv.2312.11805>.  
581
- 582 Anthropic. The claude 3 model family: Opus, sonnet, haiku. [https://www-cdn.anthropic.](https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf)  
583 [com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model\\_Card\\_Claude\\_](https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf)  
584 [3.pdf](https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf), 2024.
- 585 Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones,  
586 Nicholas Joseph, Benjamin Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny  
587 Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom B. Brown,  
588 Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. A general language assistant as a  
589 laboratory for alignment. *CoRR*, abs/2112.00861, 2021. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2112.00861)  
590 [2112.00861](https://arxiv.org/abs/2112.00861).
- 591 Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn  
592 Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jack-  
593 son Kernion, Tom Conerly, Sheer El Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Her-  
594 nandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine  
595 Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, Benjamin

- 594 Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning  
595 from human feedback. *CoRR*, abs/2204.05862, 2022. doi: 10.48550/ARXIV.2204.05862. URL  
596 <https://doi.org/10.48550/arXiv.2204.05862>.  
597
- 598 Meng Cao, Lei Shu, Lei Yu, Yun Zhu, Nevan Wichers, Yinxiao Liu, and Lei Meng. DRLC: rein-  
599 forcement learning with dense rewards from LLM critic. *CoRR*, abs/2401.07382, 2024. doi: 10.  
600 48550/ARXIV.2401.07382. URL <https://doi.org/10.48550/arXiv.2401.07382>.
- 601 Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fire-  
602 act: Toward language agent fine-tuning. *CoRR*, abs/2310.05915, 2023. doi: 10.48550/ARXIV.  
603 2310.05915. URL <https://doi.org/10.48550/arXiv.2310.05915>.
- 604 Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and  
605 Feng Zhao. Agent-flan: Designing data and methods of effective agent tuning for large language  
606 models. *CoRR*, abs/2403.12881, 2024. doi: 10.48550/ARXIV.2403.12881. URL <https://doi.org/10.48550/arXiv.2403.12881>.
- 607  
608
- 609 Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia,  
610 Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency  
611 of grounded language learning. In *7th International Conference on Learning Representations,  
612 ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL <https://openreview.net/forum?id=rJeXCo0cYX>.
- 613
- 614 Filippos Christianos, Georgios Papoudakis, Matthieu Zimmer, Thomas Coste, Zhihao Wu, Jingxuan  
615 Chen, Khyati Khandelwal, James Doran, Xidong Feng, Jiacheng Liu, Zheng Xiong, Yicheng  
616 Luo, Jianye Hao, Kun Shao, Haitham Bou-Ammar, and Jun Wang. Pangu-agent: A fine-tunable  
617 generalist agent with structured reasoning. *CoRR*, abs/2312.14878, 2023. doi: 10.48550/ARXIV.  
618 2312.14878. URL <https://doi.org/10.48550/arXiv.2312.14878>.
- 619 Peter Dayan and Geoffrey E Hinton. Using expectation-maximization for reinforcement learning.  
620 *Neural Computation*, 9(2):271–278, 1997.
- 621
- 622 DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language  
623 model, 2024.
- 624 Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, An-  
625 drew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building  
626 open-ended embodied agents with internet-scale knowledge. In Sanmi Koyejo, S. Mo-  
627 hamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural  
628 Information Processing Systems 35: Annual Conference on Neural Information Process-  
629 ing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December  
630 9, 2022*, 2022. URL [http://papers.nips.cc/paper\\_files/paper/2022/  
631 hash/74a67268c5cc5910f64938cac4526a90-Abstract-Datasets\\_and\\_  
632 Benchmarks.html](http://papers.nips.cc/paper_files/paper/2022/hash/74a67268c5cc5910f64938cac4526a90-Abstract-Datasets_and_Benchmarks.html).
- 633 Çağlar Gülçehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek  
634 Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud  
635 Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (rest) for language modeling.  
636 *CoRR*, abs/2308.08998, 2023. doi: 10.48550/ARXIV.2308.08998. URL [https://doi.org/  
637 10.48550/arXiv.2308.08998](https://doi.org/10.48550/arXiv.2308.08998).
- 638 Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao  
639 Bi, Y Wu, YK Li, et al. Deepseek-coder: When the large language model meets programming—the  
640 rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.
- 641
- 642 Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron C. Courville, Alessandro Sordani, and  
643 Rishabh Agarwal. V-star: Training verifiers for self-taught reasoners. *CoRR*, abs/2402.06457,  
644 2024. doi: 10.48550/ARXIV.2402.06457. URL [https://doi.org/10.48550/arXiv.  
645 2402.06457](https://doi.org/10.48550/arXiv.2402.06457).
- 646 Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun  
647 Wang. The 37 implementation details of proximal policy optimization. *The ICLR Blog Track  
2023*, 2022.

- 648 Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen  
649 Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. Autowebglm: Bootstrap and reinforce A  
650 large language model-based web navigating agent. *CoRR*, abs/2404.03648, 2024. doi: 10.48550/  
651 ARXIV.2404.03648. URL <https://doi.org/10.48550/arXiv.2404.03648>.
- 652 WB Langdon. Pfeiffer—a distributed open-ended evolutionary system. In *AISB*, volume 5, pp. 7–13.  
653 Citeseer, 2005.
- 654 Sergey Levine and Vladlen Koltun. Variational policy search via trajectory optimization. *Advances  
655 in neural information processing systems*, 26, 2013.
- 656 Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding,  
657 Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui  
658 Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang.  
659 Agentbench: Evaluating llms as agents. *CoRR*, abs/2308.03688, 2023a. doi: 10.48550/ARXIV.  
660 2308.03688. URL <https://doi.org/10.48550/arXiv.2308.03688>.
- 661 Zhiwei Liu, Weiran Yao, Jianguo Zhang, Le Xue, Shelby Heinecke, Rithesh Murthy, Yihao Feng,  
662 Zeyuan Chen, Juan Carlos Niebles, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming  
663 Xiong, and Silvio Savarese. BOLAA: benchmarking and orchestrating llm-augmented au-  
664 tonomous agents. *CoRR*, abs/2308.05960, 2023b. doi: 10.48550/ARXIV.2308.05960. URL  
665 <https://doi.org/10.48550/arXiv.2308.05960>.
- 666 Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. Reft: Rea-  
667 soning with reinforced fine-tuning, 2024.
- 668 Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan,  
669 Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn  
670 LLM agents. *CoRR*, abs/2401.13178, 2024. doi: 10.48550/ARXIV.2401.13178. URL <https://doi.org/10.48550/arXiv.2401.13178>.
- 671 Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking  
672 the potential of slms in grade school math. *CoRR*, abs/2402.14830, 2024. doi: 10.48550/ARXIV.  
673 2402.14830. URL <https://doi.org/10.48550/arXiv.2402.14830>.
- 674 Gerhard Neumann. Variational inference for policy search in changing situations. In Lise Getoor and  
675 Tobias Scheffer (eds.), *Proceedings of the 28th International Conference on Machine Learning,  
676 ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pp. 817–824. Omnipress, 2011.  
677 URL [https://icml.cc/2011/papers/441\\_icmlpaper.pdf](https://icml.cc/2011/papers/441_icmlpaper.pdf).
- 678 OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774.  
679 URL <https://doi.org/10.48550/arXiv.2303.08774>.
- 680 Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin,  
681 Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton,  
682 Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano,  
683 Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feed-  
684 back. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.),  
685 *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Informa-  
686 tion Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December  
687 9, 2022*, 2022. URL [http://papers.nips.cc/paper\\_files/paper/2022/hash/  
688 bl9fde53be364a73914f58805a001731-Abstract-Conference.html](http://papers.nips.cc/paper_files/paper/2022/hash/bl9fde53be364a73914f58805a001731-Abstract-Conference.html).
- 689 Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational  
690 space control. In Zoubin Ghahramani (ed.), *Machine Learning, Proceedings of the Twenty-Fourth  
691 International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume 227  
692 of *ACM International Conference Proceeding Series*, pp. 745–750. ACM, 2007. doi: 10.1145/  
693 1273496.1273590. URL <https://doi.org/10.1145/1273496.1273590>.
- 694 Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit  
695 Bansal, and Tushar Khot. Adapt: As-needed decomposition and planning with language mod-  
696 els. *CoRR*, abs/2311.05772, 2023. doi: 10.48550/ARXIV.2311.05772. URL [https://doi.  
697 org/10.48550/arXiv.2311.05772](https://doi.org/10.48550/arXiv.2311.05772).

- 702 Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and  
703 Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model.  
704 In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine  
705 (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural*  
706 *Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 -*  
707 *16, 2023*, 2023. URL [http://papers.nips.cc/paper\\_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html](http://papers.nips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html).
- 709 Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and re-  
710 inforcement learning by approximate inference (extended abstract). In Francesca Rossi (ed.),  
711 *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence,*  
712 *Beijing, China, August 3-9, 2013*, pp. 3052–3056. IJCAI/AAAI, 2013. URL <http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6658>.
- 714 Scott E. Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov,  
715 Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom  
716 Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell,  
717 Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *Trans. Mach. Learn.*  
718 *Res.*, 2022, 2022. URL <https://openreview.net/forum?id=likK0kHjvj>.
- 720 Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathemati-*  
721 *cal statistics*, pp. 400–407, 1951.
- 722 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy  
723 optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- 724 Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J.  
725 Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In  
726 *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria,*  
727 *May 3-7, 2021*. OpenReview.net, 2021. URL <https://openreview.net/forum?id=0IOX0YcCdTn>.
- 730 David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez,  
731 Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lilli-  
732 crap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis.  
733 Mastering the game of go without human knowledge. *Nat.*, 550(7676):354–359, 2017. doi:  
734 10.1038/NATURE24270. URL <https://doi.org/10.1038/nature24270>.
- 735 David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez,  
736 Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, et al. A general reinforcement  
737 learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–  
738 1144, 2018.
- 739 Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Pe-  
740 ter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi,  
741 Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin F. Elsayed, Hanie Sedghi,  
742 Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron,  
743 Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L.  
744 Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini  
745 Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human  
746 data: Scaling self-training for problem-solving with language models. *CoRR*, abs/2312.06585,  
747 2023. doi: 10.48550/ARXIV.2312.06585. URL <https://doi.org/10.48550/arXiv.2312.06585>.
- 749 Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su.  
750 Llm-planner: Few-shot grounded planning for embodied agents with large language models. In  
751 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2998–3009,  
752 2023.
- 753 Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error:  
754 Exploration-based trajectory optimization for LLM agents. *CoRR*, abs/2403.02502, 2024. doi: 10.  
755 48550/ARXIV.2403.02502. URL <https://doi.org/10.48550/arXiv.2403.02502>.

- 756 Russell K Standish. Open-ended artificial evolution. *International Journal of Computational Intel-*  
757 *ligence and Applications*, 3(02):167–175, 2003.
- 758
- 759 Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive  
760 architectures for language agents. *Trans. Mach. Learn. Res.*, 2024, 2024. URL <https://openreview.net/forum?id=1i6ZCvflQJ>.
- 761
- 762 Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei  
763 Huang, Dacheng Tao, and Jingren Zhou. A survey on self-evolution of large language models.  
764 *arXiv preprint arXiv:2404.14387*, 2024.
- 765
- 766 Tim Taylor, Mark A. Bedau, Alastair Channon, David H. Ackley, Wolfgang Banzhaf, Guillaume  
767 Beslon, Emily L. Dolson, Tom Froese, Simon J. Hickinbotham, Takashi Ikegami, Barry Mc-  
768 Mullin, Norman H. Packard, Steen Rasmussen, Nathaniel Virgo, Eran Agmon, Edward Clark,  
769 Simon McGregor, Charles Ofria, Glen E. P. Ropella, Lee Spector, Kenneth O. Stanley, Adam  
770 Stanton, Christopher Steven Timperley, Anya E. Vostinar, and Michael J. Wiser. Open-ended  
771 evolution: Perspectives from the OEE workshop in york. *Artif. Life*, 22(3):408–423, 2016. doi:  
772 10.1162/ARTL\A\00210. URL [https://doi.org/10.1162/ARTL\\_a\\_00210](https://doi.org/10.1162/ARTL_a_00210).
- 773
- 774 SIMA Team, Maria Abi Raad, Arun Ahuja, Catarina Barros, Frederic Besse, Andrew Bolt, Adrian  
775 Bolton, Bethanie Brownfield, Gavin Buttimore, Max Cant, Sarah Chakera, Stephanie C. Y. Chan,  
776 Jeff Clune, Adrian Collister, Vikki Copeman, Alex Cullum, Ishita Dasgupta, Dario de Cesare,  
777 Julia Di Trapani, Yani Donchev, Emma Dunleavy, Martin Engelcke, Ryan Faulkner, Frankie Gar-  
778 cia, Charles Gbadamosi, Zhitao Gong, Lucy Gonzalez, Kshitij Gupta, Karol Gregor, Arne Olav  
779 Hallingstad, Tim Harley, Sam Haves, Felix Hill, Ed Hirst, Drew A. Hudson, Jony Hudson,  
780 Steph Hughes-Fitt, Danilo J. Rezende, Mimi Jasarevic, Laura Kampis, Nan Rosemary Ke,  
781 Thomas Keck, Junkyung Kim, Oscar Knagg, Kavya Kopparapu, Andrew K. Lampinen, Shane  
782 Legg, Alexander Lerchner, Marjorie Limont, Yulan Liu, Maria Loks-Thompson, Joseph Marino,  
783 Kathryn Martin Cussons, Loic Matthey, Siobhan McLoughlin, Piermaria Mendolicchio, Hamza  
784 Merzic, Anna Mitenkova, Alexandre Moufarek, Valéria Oliveira, Yanko Gitahy Oliveira, Hannah  
785 Openshaw, Renke Pan, Aneesh Pappu, Alex Platonov, Ollie Purkiss, David P. Reichert, John  
786 Reid, Pierre Harvey Richemond, Tyson Roberts, Giles Ruscoe, Jaume Sanchez Elias, Tasha  
787 Sandars, Daniel P. Sawyer, Tim Scholtes, Guy Simmons, Daniel Slater, Hubert Soyer, Heiko  
788 Strathmann, Peter Stys, Allison C. Tam, Denis Teplyashin, Tayfun Terzi, Davide Vercelli, Bo-  
789 jan Vujatovic, Marcus Wainwright, Jane X. Wang, Zhengdong Wang, Daan Wierstra, Duncan  
790 Williams, Nathaniel Wong, Sarah York, and Nick Young. Scaling instructable agents across  
791 many simulated worlds. *CoRR*, abs/2404.10179, 2024. doi: 10.48550/ARXIV.2404.10179. URL  
792 <https://doi.org/10.48550/arXiv.2404.10179>.
- 793
- 794 Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Haitao Mi, and Dong Yu. To-  
795 ward self-improvement of llms via imagination, searching, and criticizing. *arXiv preprint*  
796 *arXiv:2404.12253*, 2024.
- 797
- 798 Marc Toussaint. Robot trajectory optimization using approximate inference. In Andrea Pohoreckjy  
799 Danyluk, Léon Bottou, and Michael L. Littman (eds.), *Proceedings of the 26th Annual Interna-*  
800 *tional Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18,*  
801 *2009*, volume 382 of *ACM International Conference Proceeding Series*, pp. 1049–1056. ACM,  
802 2009. doi: 10.1145/1553374.1553508. URL <https://doi.org/10.1145/1553374.1553508>.
- 803
- 804 Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Niko-  
805 lay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shrutii Bhosale, et al. Llama 2: Open founda-  
806 tion and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- 807
- 808 Binghai Wang, Rui Zheng, Lu Chen, Yan Liu, Shihan Dou, Caishuang Huang, Wei Shen, Senjie Jin,  
809 Enyu Zhou, Chenyu Shi, Songyang Gao, Nuo Xu, Yuhao Zhou, Xiaoran Fan, Zhiheng Xi, Jun  
810 Zhao, Xiao Wang, Tao Ji, Hang Yan, Lixing Shen, Zhan Chen, Tao Gui, Qi Zhang, Xipeng Qiu,  
811 Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. Secrets of RLHF in large language models  
812 part II: reward modeling. *CoRR*, abs/2401.06080, 2024a. doi: 10.48550/ARXIV.2401.06080.  
813 URL <https://doi.org/10.48550/arXiv.2401.06080>.

- 810 Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen,  
811 Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A  
812 survey on large language model based autonomous agents. *Frontiers Comput. Sci.*, 18(6):  
813 186345, 2024b. doi: 10.1007/S11704-024-40231-1. URL [https://doi.org/10.1007/  
814 s11704-024-40231-1](https://doi.org/10.1007/s11704-024-40231-1).
- 815 Peiyi Wang, Lei Li, Liang Chen, Feifan Song, Binghuai Lin, Yunbo Cao, Tianyu Liu, and Zhi-  
816 fang Sui. Making large language models better reasoners with alignment. *arXiv preprint*  
817 *arXiv:2309.02144*, 2023a.
- 819 Ruoyao Wang, Peter A. Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld:  
820 Is your agent smarter than a 5th grader? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang  
821 (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Process-*  
822 *ing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 11279–11298.  
823 Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.775.  
824 URL <https://doi.org/10.18653/v1/2022.emnlp-main.775>.
- 825 Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and  
826 Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In  
827 Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual*  
828 *Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023,*  
829 *Toronto, Canada, July 9-14, 2023*, pp. 13484–13508. Association for Computational Linguistics,  
830 2023b. doi: 10.18653/V1/2023.ACL-LONG.754. URL [https://doi.org/10.18653/  
831 v1/2023.acl-long.754](https://doi.org/10.18653/v1/2023.acl-long.754).
- 832 Michael J. Wooldridge and Nicholas R. Jennings. Intelligent agents: theory and practice. *Knowl.*  
833 *Eng. Rev.*, 10(2):115–152, 1995. doi: 10.1017/S0269888900008122. URL [https://doi.  
834 org/10.1017/S0269888900008122](https://doi.org/10.1017/S0269888900008122).
- 835 Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Jun-  
836 zhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao  
837 Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan  
838 Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng  
839 Qiu, Xuanjing Huan, and Tao Gui. The rise and potential of large language model based  
840 agents: A survey. *CoRR*, abs/2309.07864, 2023. doi: 10.48550/ARXIV.2309.07864. URL  
841 <https://doi.org/10.48550/arXiv.2309.07864>.
- 842 Zhiheng Xi, Wenxiang Chen, Boyang Hong, Senjie Jin, Rui Zheng, Wei He, Yiwen Ding, Shichun  
843 Liu, Xin Guo, Junzhe Wang, Honglin Guo, Wei Shen, Xiaoran Fan, Yuhao Zhou, Shihan Dou,  
844 Xiao Wang, Xinbo Zhang, Peng Sun, Tao Gui, Qi Zhang, and Xuanjing Huang. Training  
845 large language models for reasoning through reverse curriculum reinforcement learning. *CoRR*,  
846 abs/2402.05808, 2024. doi: 10.48550/ARXIV.2402.05808. URL [https://doi.org/10.  
847 48550/arXiv.2402.05808](https://doi.org/10.48550/arXiv.2402.05808).
- 848 Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and  
849 Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions.  
850 *CoRR*, abs/2304.12244, 2023. doi: 10.48550/ARXIV.2304.12244. URL [https://doi.org/  
851 10.48550/arXiv.2304.12244](https://doi.org/10.48550/arXiv.2304.12244).
- 852 Zonghan Yang, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. React meets actre: When  
853 language agents enjoy training data autonomy. *CoRR*, abs/2403.14589, 2024. doi: 10.48550/  
854 ARXIV.2403.14589. URL <https://doi.org/10.48550/arXiv.2403.14589>.
- 855 Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scal-  
856 able real-world web interaction with grounded language agents. In Sanmi Koyejo, S. Mo-  
857 hamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural*  
858 *Information Processing Systems 35: Annual Conference on Neural Information Process-*  
859 *ing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9,*  
860 *2022*, 2022. URL [http://papers.nips.cc/paper\\_files/paper/2022/hash/  
861 82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html](http://papers.nips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html).

- 864 Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan  
865 Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International  
866 Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenRe-  
867 view.net, 2023. URL [https://openreview.net/pdf?id=WE\\_vluYUL-X](https://openreview.net/pdf?id=WE_vluYUL-X).
- 868 Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou.  
869 Scaling relationship on learning mathematical reasoning with large language models. *CoRR*,  
870 abs/2308.01825, 2023a. doi: 10.48550/ARXIV.2308.01825. URL <https://doi.org/10.48550/arXiv.2308.01825>.
- 871  
872 Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. Scal-  
873 ing relationship on learning mathematical reasoning with large language models. *arXiv preprint  
874 arXiv:2308.01825*, 2023b.
- 875  
876 Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with  
877 reasoning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh  
878 (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural  
879 Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - De-  
880 cember 9, 2022*, 2022. URL [http://papers.nips.cc/paper\\_files/paper/2022/  
881 hash/639a9a172c044fbb64175b5fad42e9a5-Abstract-Conference.html](http://papers.nips.cc/paper_files/paper/2022/hash/639a9a172c044fbb64175b5fad42e9a5-Abstract-Conference.html).
- 882  
883 Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttun-  
884 ing: Enabling generalized agent abilities for llms. *CoRR*, abs/2310.12823, 2023. doi: 10.48550/  
885 ARXIV.2310.12823. URL <https://doi.org/10.48550/arXiv.2310.12823>.
- 886  
887 Jianguo Zhang, Tian Lan, Rithesh Murthy, Zhiwei Liu, Weiran Yao, Juntao Tan, Thai Hoang, Liang-  
888 wei Yang, Yihao Feng, Zuxin Liu, et al. Agentohana: Design unified data and training pipeline  
889 for effective agent learning. *arXiv preprint arXiv:2402.15506*, 2024.
- 890  
891 Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao  
892 Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez,  
893 and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In Alice Oh,  
894 Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.),  
895 *Advances in Neural Information Processing Systems 36: Annual Conference on Neural In-  
896 formation Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10  
897 - 16, 2023*, 2023a. URL [http://papers.nips.cc/paper\\_files/paper/2023/  
898 hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets\\_and\\_](http://papers.nips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html)
- 899  
900 Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie  
901 Jin, Qin Liu, Yuhao Zhou, Limao Xiong, Lu Chen, Zhiheng Xi, Nuo Xu, Wenbin Lai, Minghao  
902 Zhu, Cheng Chang, Zhangyue Yin, Rongxiang Weng, Wensen Cheng, Haoran Huang, Tianxiang  
903 Sun, Hang Yan, Tao Gui, Qi Zhang, Xipeng Qiu, and Xuanjing Huang. Secrets of RLHF in large  
904 language models part I: PPO. *CoRR*, abs/2307.04964, 2023b. doi: 10.48550/ARXIV.2307.04964.  
905 URL <https://doi.org/10.48550/arXiv.2307.04964>.
- 906  
907 Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng,  
908 Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environ-  
909 ment for building autonomous agents. *CoRR*, abs/2307.13854, 2023a. doi: 10.48550/ARXIV.  
910 2307.13854. URL <https://doi.org/10.48550/arXiv.2307.13854>.
- 911  
912 Xuhui Zhou, Hao Zhu, Leena Mathur, Ruohong Zhang, Haofei Yu, Zhengyang Qi, Louis-Philippe  
913 Morency, Yonatan Bisk, Daniel Fried, Graham Neubig, and Maarten Sap. SOTOPIA: interactive  
914 evaluation for social intelligence in language agents. *CoRR*, abs/2310.11667, 2023b. doi: 10.  
915 48550/ARXIV.2310.11667. URL <https://doi.org/10.48550/arXiv.2310.11667>.
- 916  
917 Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language  
918 model agents via hierarchical multi-turn RL. *CoRR*, abs/2402.19446, 2024. doi: 10.48550/  
919 ARXIV.2402.19446. URL <https://doi.org/10.48550/arXiv.2402.19446>.

## 918 A LIMITATIONS

919  
920 This paper proposes a new framework named AGENTGYM. It includes an interactive platform with  
921 diverse environments and tasks, an agent benchmark AGENTVAL, and two collections of expert  
922 trajectories AGENTTRAJ and AGENTTRAJ-L. Additionally, we introduce a novel algorithm, AGENT-  
923 TEVOL, to explore the evolutionary capabilities of generally-capable LLM-based agents. Despite  
924 the contributions and the fact that our method performs well, our work still has some limitations.  
925 **Firstly**, for computational efficiency, we do not perform multiple samplings in each iteration. How-  
926 ever, in the analysis in Section 4.3, we find that more sampling leads to better results, although the  
927 improvement is not significant. In the future, we hope to increase the number of samples  $K$  to a  
928 larger value when sufficient computational resources are available, to explore the upper limits of our  
929 method. **Secondly**, although we validate the effectiveness of our method on three different models  
930 (Llama2-Chat-7B, Llama-2-Chat-13B, and DeepSeek-Coder-1.3B), we hope to verify it on stronger  
931 and larger base models in the future to explore the potential for building more generally-capable  
932 agents.

## 934 B DETAILS OF ENVIRONMENTS IN AGENTGYM

935  
936 **WebShop (WS) (Yao et al., 2022).** WebShop is an interactive web environment for web shopping.  
937 The agents are given instructions, and need to buy a product that matches the specifications. The  
938 agents can click a button on the webpage or search for something by the search engine. WebShop  
939 contains 12k instructions and provides over one million real products from amazon.com. We select  
940 6910 instructions. For AGENTTRAJ, we collect 1000 trajectories with SOTA models (700) and  
941 human annotations (300). For AGENTTRAJ-L, we collect 3930 trajectories with SOTA models  
942 (3430) and human annotations (500). We take the success rate as the evaluation metric and set the  
943 maximum round to 10.<sup>2</sup>

944  
945 **WebArena (WA) (Zhou et al., 2023a).** WebArena is a realistic and reproducible web environ-  
946 ment. It contains four categories: E-commerce platforms, social forum platforms, collaborative  
947 development platforms, and content management systems. It supports 12 different web browsing  
948 actions. The observation space consists of a web page URL, the opened tabs, and the web page  
949 content. Completing tasks in this highly realistic environment requires the agent to possess strong  
950 memory, high-level planning, common sense, and reasoning abilities. The reward from the environ-  
951 ment is consistent with the original paper. We filter 20 evaluating test instances from the original  
952 dataset for three main sub-tasks: Information-seeking, Site Navigation, and Content & configuration  
953 operation. We take the success rate as the evaluation metric and set the maximum round to 25.<sup>3</sup>

954  
955 **MAZE (MZ) (Abdulhai et al., 2023b).** MAZE is a word game. Agents, acting as players, can  
956 know their own position, the goal position, and the directions where there are walls around them.  
957 Agents decide to move one square in one of four directions (up, down, left, or right) each time,  
958 receiving a reward of -1 for every move until they reach the goal position. We use GPT-4-Turbo to  
959 add thoughts to the trajectories sampled by LMRL-Gym and create our dataset. For AGENTTRAJ,  
960 we include 100 trajectories. For AGENTTRAJ-L, we include 215 trajectories. We take the success  
961 rate as the evaluation metric and set the maximum round to 15.<sup>4</sup>

962  
963 **Wordle (WD) (Abdulhai et al., 2023b).** Wordle is a word-guessing game that tests agents' ability  
964 to reason at the level of individual letters. Agents guess the target word from a given vocabulary con-  
965 taining some five-letter words. After each guess, agents are told whether each letter in the guessed  
966 word is in the target word and whether its position is correct and receive a reward of -1 for each step  
967 until they guess the target word or run out of attempts. We take the success rate as the evaluation  
968 metric and set the maximum round to 8. We also use GPT-4-Turbo to add thoughts to the trajectories  
969 sampled by LMRL-Gym. For AGENTTRAJ, we include 500 trajectories. For AGENTTRAJ-L, we  
970 include 955 trajectories.

<sup>2</sup><https://github.com/princeton-nlp/WebShop/blob/master/LICENSE.md>

<sup>3</sup><https://github.com/web-arena-x/webarena/blob/main/LICENSE>

<sup>4</sup><https://github.com/abdulhaim/LMRL-Gym/blob/main/LICENSE>

972 **ALFWorld (ALF) (Shridhar et al., 2021).** ALFWorld is a household environment based on  
 973 TextWorld, where agents need to explore rooms and use common sense reasoning to execute tasks.  
 974 The action space of ALFWorld includes picking up and placing items, observing surroundings, us-  
 975 ing furniture, and more. The environment provides feedback on the execution of actions based on  
 976 predefined logic. We take the success rate as the evaluation metric and set the maximum round  
 977 to 30. ALFWorld has six types of tasks. We get 3827 instructions from the original work. For  
 978 AGENTTRAJ, we collect 500 trajectories with SOTA models(400) and human annotations (100).  
 979 For AGENTTRAJ-L, we collect 2420 trajectories with SOTA models(1920) and human annotations  
 980 (500).<sup>5</sup>

981 **SciWorld (Sci) (Wang et al., 2022).** ScienceWorld is a benchmark for testing agents’ scientific  
 982 reasoning abilities in a new interactive text environment at the standard elementary science curricu-  
 983 lum level. ScienceWorld includes 30 types of tasks, such as using measurement instruments and  
 984 conducting mechanics experiments. Its action space is task-related, with the environment simulator  
 985 providing the effects of actions. Because the ScienceWorld repository provides golden paths and  
 986 existing models cannot achieve high performance, we use GPT-4-Turbo to generate thoughts for  
 987 golden paths of 22 types of interactions that are not too long. For AGENTTRAJ, we include 1000  
 988 trajectories. For AGENTTRAJ-L, we include 2120 trajectories. We take reward as the evaluation  
 989 metric and set the maximum round to 30.<sup>6</sup>

991 **BabyAI (Baby) (Chevalier-Boisvert et al., 2019).** The BabyAI platform is an interactive grid  
 992 world simulator with 40 instruction-following tasks where the agent is asked to interact with ob-  
 993 jects. The agent has a limited 7x7 sight of view and can only operate objects in front. The original  
 994 implementation of BabyAI presents observations in the form of images and low-level actions like  
 995 ”move forward” and ”turn left”. The implementation from AgentBoard converts graphic obser-  
 996 vations into textual instructions and expands the action space with high-level actions like ”pickup  
 997 green key 1” and ”go through blue locked door 2”. The agent receives a non-zero reward discounted  
 998 by the number of steps when reaching the goal, and 0 otherwise. For AGENTTRAJ, we annotate  
 999 400 trajectories of 18 out of all 40 tasks with SOTA models. For AGENTTRAJ-L, we annotate 810  
 1000 trajectories with SOTA models. We take reward as the evaluation metric and set the maximum round  
 1001 to 20.<sup>7</sup>

1002 **TextCraft (TC) (Prasad et al., 2023).** Similar to WordCraft, TextCraft is a text-only environ-  
 1003 ment for crafting Minecraft items. This environment constructs a crafting tree based on Minecraft’s  
 1004 crafting recipes, comprising 544 nodes, each representing a target item. In TextCraft, each task pro-  
 1005 vides a specific target item alongside a list of crafting commands generated by the tree. These tasks  
 1006 are structured compositionally, incorporating crafting recipes of varying complexity ranging from  
 1007 1 to 4 steps. The environment supports three valid actions: craft <item> using <ingredients>,  
 1008 get <item>, and inventory. Each round, the environment checks the agent’s actions and returns  
 1009 the execution state. Apart from craftable items and their ingredients, all other items are obtainable  
 1010 from the environment. Agents can get a reward of 1 only upon successfully crafting the target item.  
 1011 We select 100 tasks for the test set and use the remaining tasks for training. For AGENTTRAJ, we  
 1012 annotate 300 trajectories with SOTA models (254) and human annotation (46), with every action in  
 1013 the trajectories verified by the environment. For AGENTTRAJ-L, we annotate 374 trajectories with  
 1014 SOTA models (299) and human annotation (75). We take the success rate as the evaluation metric  
 1015 and set the maximum round to 20.<sup>8</sup>

1016 **Weather (WT) (Ma et al., 2024).** The Weather Environment allows LLM agents to utilize a  
 1017 weather tool to access data on temperature, precipitation, and air quality for various locations and  
 1018 time periods. It includes 18 different actions that agents can use to achieve weather-related objec-  
 1019 tives. This environment leverages Python code to integrate the Open-Meteo API and implement the  
 1020 necessary functions. If the agent’s final answer matches the reference answer, it receives a reward of  
 1021 1; otherwise, it receives a reward of 0. We expand the original dataset of 20 queries to a total of 331  
 1022

1023 <sup>5</sup><https://github.com/alfworld/alfworld/blob/master/LICENSE>

1024 <sup>6</sup><https://github.com/allenai/ScienceWorld/blob/main/LICENSE>

1025 <sup>7</sup><https://github.com/mila-iqia/babyai/blob/master/LICENSE>

<sup>8</sup><https://github.com/archiki/ADaPT/blob/main/LICENSE>

1026 queries by using GPT-3.5-Turbo and GPT-4-Turbo for augmentation using self-instruct and instruc-  
1027 tion evolution. Finally, we select 20 questions as the evaluating set, leaving the remaining questions  
1028 as the training set. For AGENTTRAJ, we annotate 160 trajectories with SOTA models (140) and  
1029 human annotators (20). We also refine the annotations with human review to ensure accuracy. For  
1030 AGENTTRAJ-L, we annotate 311 trajectories with SOTA models (230) and human annotators (81).  
1031 We take the success rate as the evaluation metric and set the maximum round to 10.<sup>9</sup>

1032  
1033 **Movie (MV) (Ma et al., 2024).** The Movie Environment grants LLM agents to utilize the movie  
1034 tool for accessing cinematic data, including film details, personnel, and production companies. It  
1035 offers 16 distinct actions that agents can use to achieve various movie-related objectives. This tool  
1036 integrates the API and data from The Movie Database, implementing the necessary functions to  
1037 establish its capabilities. If the agent’s final answer matches the reference answer, it receives a  
1038 reward of 1; otherwise, it receives a reward of 0. To enhance the dataset, we expand the original  
1039 20 questions to 235 by using GPT-3.5-Turbo and GPT-4-Turbo for query augmentation. GPT-4-  
1040 Turbo is employed to annotate 100 trajectories in AGENTTRAJ, and the annotations are further  
1041 corrected through human annotations to ensure accuracy. We also use GPT-4-Turbo to annotate 215  
1042 trajectories for AGENTTRAJ-L. We select 20 questions for the evaluating set, with the remaining  
1043 questions designated as the training set. We take the success rate as the evaluation metric and set the  
1044 maximum round to 12.

1045  
1046 **Academia (AM) (Ma et al., 2024).** The Academia Environment equips LLM agents with the aca-  
1047 demic tools to query information related to computer science research, including academic papers  
1048 and author details. It offers 7 different actions for achieving various academic research objectives.  
1049 During its development, it utilizes data from the Citation Network Dataset, crafts the necessary  
1050 functions, and subsequently constructs the Academia tool. If the agent’s final answer matches the  
1051 reference answer, it receives a reward of 1; otherwise, it receives a reward of 0. The original 20  
1052 questions are used as the evaluating set. We take the success rate as the evaluation metric and set the  
1053 maximum round to 12.

1054  
1055 **TODOList (TL) (Ma et al., 2024).** The TodoEnvironment enables LLM agents to query and  
1056 amend personal agenda data through the todo tool, offering 11 different actions. This tool is imple-  
1057 mented based on the TodoList API. If the agent’s final answer or operations matches the reference  
1058 ones, it receives a reward of 1; otherwise, it receives a reward of 0. To enhance the dataset, we ex-  
1059 pand the original 20 questions to 155 using GPT-3.5-Turbo and GPT-4-Turbo for data augmentation.  
1060 For AGENTTRAJ, we annotate 70 trajectories with GPT-4-Turbo. For AGENTTRAJ-L, we annotate  
1061 the queries to get 135 trajectories with GPT-4-Turbo (96) and human annotators (39). The annota-  
1062 tions are further refined by human review to ensure accuracy. Finally, we select 20 questions for the  
1063 evaluating set, with the remaining questions designated as the training set. We take the success rate  
1064 as the evaluation metric and set the maximum round to 15.

1065  
1066 **Sheet (ST) (Ma et al., 2024).** The Sheet Environment allows LLM agents to use the sheet tool to  
1067 access and modify spreadsheet data, providing 20 different actions for operating on an Excel sheet.  
1068 This tool is built upon the Google Sheets API. The reward returned by the environment is based on  
1069 the similarity between the table manipulated by the agent and the reference table, with a value range  
1070 of  $[0, 1]$ . The original 20 questions are used as the evaluating set. We take reward as the evaluation  
1071 metric and set the maximum round to 15.

1072  
1073 **BIRD (BD) (Zheng et al., 2023a).** Code ability is a crucial aspect of capability for LLM-based  
1074 agents. In this environment, we focus on database management ability. We wrap the BIRD-SQL  
1075 dataset and provide a unified API for agents to interact with. BIRD-SQL is a bench for large-  
1076 scale database-grounded text-to-SQL evaluation. It requires the agent to query a database using  
1077 a SELECT statement to get the correct answer. It contains 9428 unique problems with a golden  
1078 answer for training. We select 3200 of them as the instruction set. For AGENTTRAJ, we employ  
1079 GPT-4-Turbo to add thoughts for 2000 of the training set problems. For AGENTTRAJ-L, we employ

---

<sup>9</sup><https://github.com/hkust-nlp/AgentBoard>. The codebase is licensed under an Apache-2.0 License and the dataset is licensed under a GNU General Public License, version 2.

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

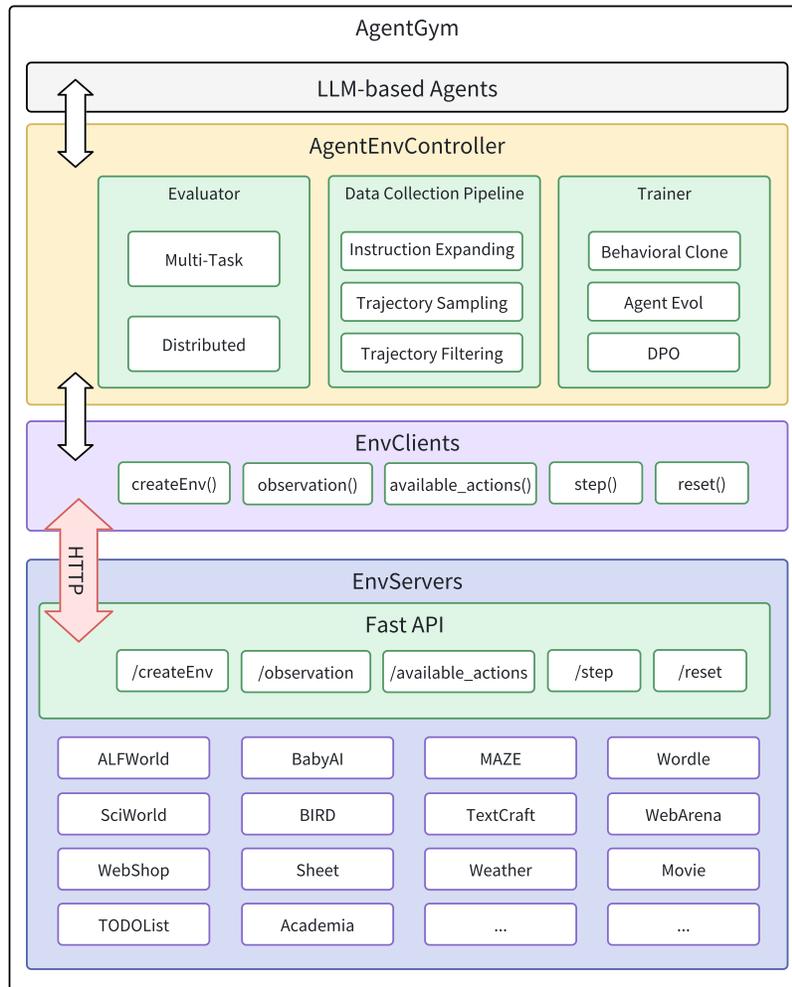


Figure 5: An illustration of the architecture of AGENTGYM.

GPT-4-Turbo to add thoughts for 3000 of the training set problems. We take success rate as the evaluation metric and the maximum round is 1 as BD is a single-round programming task.<sup>10</sup>

## C FRAMEWORK ARCHITECTURE OF AGENTGYM

We adopt a decoupled architecture to construct the agent-environment interaction framework, as illustrated in Figure 5 in Appendix C. Specifically, recognizing the distinct dependencies present in different environments, AGENTGYM deploys separate services for each environment in a user-friendly manner to prevent conflicts. These environments are implemented to offer standardized and parallelizable functions, such as `/createEnv` to create an environment, `/observation` to obtain the current observation from the environment, `/available_actions` to retrieve the currently available actions, `/step` to perform an action, and `/reset` to reset the environment. Clients communicate with the servers via HTTP protocol. At the core of this architecture is the controller, which facilitates interactions between agents and environmental services, providing a unified and encapsulated interface for agents to invoke environmental functions or operations. Additionally, we implement user-friendly components such as the evaluator, trainer, and data collection pipeline to support further development of the community. We have implemented 14 types of environments

<sup>10</sup><https://github.com/AlibabaResearch/DAMO-ConvAI/tree/main/bird>. The bench is under a CC BY-NC 4.0 License.

and 89 tasks, and developers can easily develop new environments and add them to AGENTGYM by encapsulating the aforementioned interfaces.

## D DATABASE CONSTRUCTION OF AGENTTRAJ, AGENTTRAJ-L AND AGENTEVAL

**Instruction collection.** We have gathered 20,494 instructions across the aforementioned environments using appropriate strategies. *(i)* For environments whose original datasets contain sufficient instructions, we use their original instruction sets or subsets (WA, ALF, Sci, AM, ST, BD). *(ii)* For certain environments, we generate instructions using rule-based automated pipelines (TC, WS, MZ, WD, Baby). For example, in TC, we first construct rule trees for forging different items and generate instructions of varying difficulty (levels 1-4) based on these rules. For WS, we generate instructions based on available products by fixing the random seed. For MZ, we randomly select starting points in the maze and construct instructions accordingly. For WD, we fix the seed and generate words for guessing. For Baby, we pass a fixed seed to the generator provided by the environment to generate instructions. *(iii)* In environments where instructions are relatively scarce and difficult to construct through rules, we use self-instruct (Wang et al., 2023b) and instruction evolution (Xu et al., 2023) methods to generate instructions. These methods provide an LLM (GPT-4-Turbo) with available actions and instruction examples, and query it to generate diverse and challenging instructions that might be needed in real-world scenarios (WT, MV, TL). Note that we manually verify the instructions generated by these AI-based techniques to ensure that they can be successfully completed.

**Benchmark construction.** We then construct a benchmark suite with a size of 1160 named AGENTEVAL to holistically evaluate the ability of LLM-based agents in diverse scenarios. Specifically, *(i)* for environments that have different task categories or varying difficulty levels, we either uniformly sample test examples from different subsets or use them all (ALF, Sci, Baby, TC); *(ii)* for remained environments with existing test sets, we use the original test sets or randomly sample from them (WA, AM, ST, BD); *(iii)* for others, we randomly sampled from the collected or augmented instructions (WS, MZ, WD, WT, MV, TL).

**Trajectory collecting and filtering.** In AGENTGYM, the server provides instructions including task description, environment setup, and problem to the agent. Next, as described in Section 3.1, the agent interacts with the environment in ReAct-Style until the task is completed. We collect 6130 trajectories from 11 environments with different strategies. *(i)* For environments with **human annotated** trajectories or where the correct action sequences can be obtained using a rule-based solver, we use GPT-4-Turbo to add thought step by step for each action, thus forming outputs in the ReAct-Style (MZ, WD, Sci, BD). *(ii)* For environments where only instructions are provided and the correct trajectories are neither available nor can be derived through rules, we annotate the correct trajectories with SOTA models (e.g., GPT-4-Turbo) and crowdsourcing. Then, we rigorously filter the trajectories based on rewards and correctness to ensure their quality (WS, ALF, Baby, TC, WT, MV, TL). This set, named AGENTTRAJ, is used to train a base agent with preliminary abilities and prior knowledge. For a fair comparison, we perform annotation and filtering on all instructions using the same pipeline and get AGENTTRAJ-L to represent the performance upper bound of BC.

## E MORE IMPLEMENTATION DETAILS

**Behavior Cloning.** We train the model for 3 epochs with a learning rate of  $1 \times 10^{-5}$ . The batch size is set to 2, and gradient accumulation is performed over 2 steps. We do not employ weight decay or learning rate warmup.

**AGENTEVOL.** First, we train a base agent on the AGENTTRAJ set, running BC for 3 epochs with a learning rate of  $1 \times 10^{-5}$ . Then, we perform the self-evolution phase. In Learning Step, we run 1 epoch per iteration. In Exploration Step, we set the temperature to 0.7 to sample trajectories across environments. We perform a total of  $M=4$  iterations. All other parameters remain the same as in Behavior Cloning. All experiments are conducted on eight A100-80GB GPUs.

**LLM-Planner.** LLM-Planner is a prompt-based baseline. In our experiments, we enhance the initial System Prompt with exploration-based guidance. We directly prompt the agent to generate

a high-level plan in the first turn, mapping the instruction into subgoals, and let it interact with the environment in a ReAct-style during subsequent turns.

**PPO.** We use full parameter fine-tuning instead of LoRA tuning. We load three models: the actor model, the reference model, and the critic model. We do not use a reward model, as our environment automatically assigns rewards to the agent based on interaction results. We follow the implementation from the TRL library<sup>11</sup>, where the actor and critic models share the same backbone. On top of this, we add a trainable value head as the output for the critic model. The learning rate is set to  $5 \times 10^{-7}$ , with a batch size of 1 and gradient accumulation steps of 2. We do not use weight decay or learning rate warmup. We adhere to OpenAI’s implementation of the PPO algorithm Ouyang et al. (2022), where `KL_coef = 0.01`, `gamma = 1.0`, `lambda = 0.95`, and `ppo_epoch = 2`. For all environments, we first perform supervised fine-tuning for 1~2 epochs as a warm-up, followed by PPO training for 5~10 epochs. To alleviate memory constraints, we employ gradient checkpointing and flash-attention 2 techniques.

**DPO.** We also use full parameter fine-tuning. During the data sampling phase, we perform two rounds of sampling on the base model to construct DPO training data pairs. Responses with a reward exceeding the expert threshold are labeled as “chosen responses”. Responses with a reward gap greater than that of the “chosen responses” are labeled as “rejected responses”, and these form the data pairs. In our experiments, the expert threshold is set to 0.9, and the reward gap is set to 0.1. We train for 3 epochs with a learning rate of  $5 \times 10^{-7}$ , a batch size of 2, gradient accumulation steps of 4, weight decay of 0.1, and a warmup ratio of 0.1. Additionally, we include a BC objective to stabilize the training procedure, following previous work (Lai et al., 2024). Both the DPO and BC objectives are assigned equal weights.

**Evaluation.** We set `do_sample = False` during evaluation. When evaluating models that have not been fine-tuned on expert trajectories, we use a few-shot approach; when evaluating models that have been trained on expert trajectories, we use a zero-shot approach.

We upload anonymized versions of our data and code in a Zip file.

## F MORE EXPERIMENTS

### F.1 INTERACTIVE ROUNDS IN MAIN EXPERIMENTS

Interactive rounds reflect the efficiency of an agent in solving tasks. Table 6 shows the interactive rounds of each model/agent across tasks. We also present the evaluation performance in Table 6 for better and clearer illustration. We find that agents trained with AGENTTRAJ-L and AGENTEVOL both demonstrate high efficiency, indicating that they can complete tasks in a small number of rounds. Additionally, we observe a trend: agents that require fewer interactive rounds to complete the same task generally perform better. This may be because underperforming agents often struggle to find the optimal path to achieve the final goal or exceed the maximum number of rounds. For example, in ALFWorld and BabyAI, AGENTEVOL achieves the best performance as well as the fewest interactive rounds.

### F.2 MORE ABLATION STUDIES

**Ablation on sample number  $K$ .** In the exploration step, we perform sampling on each instruction once per iteration. Here, we conduct ablation on sample number  $K$  with four tasks. The results in Table 7 show no significant performance increases with higher  $K$ . So we select  $K = 1$  for computational efficiency.

**Ablation on exploration scope.** In our experiment, we first train a base agent using  $\mathcal{D}_s$  and then let it explore a wider range of instructions and tasks. We conduct an ablation study on four tasks to see how well the agent evolves with limited instructions as in the BC phase. Table 7 shows that even in a limited scope, the base agent’s performance improves, which may be attributed to more

<sup>11</sup><https://github.com/huggingface/trl/blob/main/examples/scripts/ppo/ppo.py>

Table 6: Evaluating performance and interactive rounds on diverse tasks. The first row of each method indicates performance, while the second row of each method shows the number of **interaction rounds** between the model/agent and the environment.

Method	WS	ALF	TC	Sci	Baby	MZ	WD	WT	MV	TL	BD
<b>Close-sourced Models &amp; Agents</b>											
DeepSeek-Chat	11.00	51.00	23.00	<b>16.80</b>	45.67	4.00	24.00	70.00	70.00	75.00	13.50
	6.9	20.4	15.1	20.7	11.7	14.5	5.2	6.1	5.9	4.4	1.0
Claude-3-Haiku	5.50	0.00	0.00	0.83	1.93	4.00	16.00	55.00	50.00	65.00	13.50
	8.0	30.0	20.0	29.8	19.9	14.4	5.7	7.3	6.0	4.0	1.0
Claude-3-Sonnet	1.50	13.00	38.00	2.78	<b>79.25</b>	0.00	36.00	65.00	80.00	80.00	<b>17.00</b>
	9.5	27.9	14.6	28.7	6.6	15.0	5.2	6.9	5.1	4.5	1.0
GPT-3.5-Turbo	12.50	26.00	47.00	7.64	71.36	4.00	20.00	25.00	70.00	40.00	12.50
	4.9	25.2	13.1	16.5	8.4	14.4	5.3	6.6	4.6	3.4	1.0
GPT-4-Turbo	<b>15.50</b>	<b>67.50</b>	<b>77.00</b>	14.38	72.93	<b>68.00</b>	<b>88.00</b>	<b>80.00</b>	<b>95.00</b>	<b>95.00</b>	16.00
	8.2	18.3	9.9	18.1	9.1	9.0	4.0	6.0	4.5	4.0	1.0
<b>Open-sourced Models &amp; Agents</b>											
Llama2-Chat-7B	0.50	2.00	0.00	0.83	0.23	0.00	0.00	0.00	0.00	0.00	1.50
	6.4	22.6	14.5	27.5	9.5	15.0	6.0	9.9	12.0	15.0	1.0
Llama2-Chat-13B	1.00	3.50	0.00	0.83	0.10	0.00	0.00	0.00	0.00	0.00	1.50
	8.1	19.6	16.5	21.3	10.9	13.4	6.0	10.0	12.0	15.0	1.0
AgentLM-7B	36.50	71.00	<b>4.00</b>	1.63	0.49	<b>12.00</b>	4.00	0.00	<b>5.00</b>	15.00	5.00
	4.7	17.7	19.4	28.5	7.5	13.9	2.0	8.3	11.7	10.6	1.0
AgentLM-13B	39.50	<b>73.00</b>	0.00	2.75	0.45	8.00	0.00	<b>10.00</b>	<b>5.00</b>	5.00	3.00
	4.8	17.8	19.4	28.5	7.6	13.9	6.0	6.6	10.7	8.4	1.0
AgentLM-70B	<b>49.50</b>	67.00	<b>4.00</b>	<b>10.68</b>	<b>0.66</b>	8.00	4.00	0.00	0.00	<b>40.00</b>	<b>7.50</b>
	4.9	18.5	18.8	28.2	6.3	13.9	5.2	6.6	11.6	6.7	1.0
<b>Ours</b>											
BC <sub>base</sub>	66.50	77.50	44.00	26.42	69.31	<b>12.00</b>	12.00	25.00	5.00	45.00	8.00
	5.6	16.4	13.7	21.3	6.7	14.3	5.9	6.2	10.8	5.4	1.0
BC <sub>large</sub>	73.50	83.00	60.00	<b>74.47</b>	74.19	<b>12.00</b>	<b>36.00</b>	<b>45.00</b>	5.00	65.00	8.50
	5.5	16.1	14.3	29.3	6.2	14.3	5.7	6.4	10.2	5.0	1.0
<b>AGENTEVOL</b>	<b>76.50</b>	<b>88.00</b>	<b>64.00</b>	38.00	<b>82.70</b>	<b>12.00</b>	12.00	25.00	<b>60.00</b>	<b>70.00</b>	<b>9.00</b>
	5.1	14.0	11.8	18.9	4.3	13.8	5.7	5.9	3.2	5.1	1.0

Table 7: Ablation study on sample number  $K$  and the exploration scope with four tasks.

Method	WS	ALF	Baby	TC
BC <sub>base</sub>	66.5	77.5	69.3	44.0
AGENTEVOL				
-w $K = 1$	77.0	88.0	82.9	65.0
-w $K = 2$	76.0	88.0	83.1	67.0
-w $K = 3$	<b>78.5</b>	<b>89.0</b>	<b>83.6</b>	<b>68.0</b>
-w Limited Scope for Exploration	70.0	80.5	70.7	49.0

diverse trajectories sampled from the agent. However, the improvement is not significant, indicating that effective evolution needs a more extensive environment.

**Ablation on base model selection.** In our experiments, we optimize the initial agent at each iteration rather than continuing training from the last iteration’s agent. To explore this further, we conduct an ablation study to compare these two training strategies. As shown in Figure 6, continuous fine-tuning provides short-term performance gains but often results in performance degradation in later iterations, likely due to overfitting. In contrast, training from the initial agent ensures more consistent and stable performance.

### F.3 EVALUATION OF OUT-OF-DOMAIN TASKS AND ENVIRONMENTS

To explore performance on tasks or environments not seen during the evolution phase (i.e. OOD tasks), we carry out supplementary experiments. In these experiments, the task types and settings in ALF and Baby are unseen by the agent during both the BC and AGENTEVOL phases, and the entire environments of AM and ST are entirely new for the agent. As shown in Table 8, our method

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

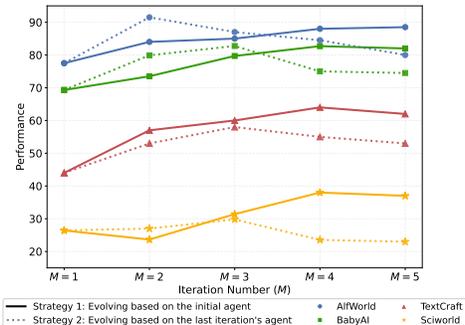


Figure 6: Ablation study regarding the choice of the base model.

demonstrates stronger generalization ability on unseen tasks and environments compared to other methods.

Table 8: Evaluating results on Out-of-domain tasks.

Method	ALF-OOD	Baby-OOD	AM	ST
Llama2-Chat-7B	0.0	2.2	0.0	0.0
AgentLM-7B	57.7	4.4	10.0	14.3
BC <sub>base</sub>	60.8	6.2	20.0	24.3
BC <sub>large</sub>	64.9	6.1	20.0	25.2
AgentEvol	<b>67.5</b>	<b>6.2</b>	<b>25.0</b>	<b>26.2</b>

F.4 ANALYSIS ON THE VARYING PERFORMANCE IN DIFFERENT ENVIRONMENTS

We observe that AGENTEVOL performs well in most environments, but is weaker than BC<sub>large</sub> in some tasks (SC, WD, and WT). We provide two perspectives to analyze the reasons behind the fluctuating performance of our method across different environments.

**Perspective 1: Conflicts between different environments.** We add experiments about the performance of task-specific BC and task-specific AgentEvol. As shown in Table 9, we find that BC on single tasks outperforms BC on multi-tasks, and AGENTEVOL on single tasks outperforms AGENTEVOL on multi-tasks. Additionally, AGENTEVOL on single tasks performs poorly on other tasks. This indicates that there are some conflicts between different environments and tasks, which may lead to a certain degree of performance degradation and cause the varying performance in our method.

**Perspective 2: Lack of exploration capability.** We compare the size of trajectory data used to train the agent between AGENTEVOL and BC methods. The results are presented in Table 10.

Table 9: Experiments of task-specific BC and task-specific AGENTEVOL.

Method	WS	ALF	Baby	WD
BC <sub>base</sub>	66.5	77.5	69.3	12.0
BC <sub>base</sub> (single task)	68.5	79.0	72.6	12.0
BC <sub>large</sub>	73.5	83.0	74.2	<b>36.0</b>
BC <sub>large</sub> (single task)	74.0	84.0	76.9	<b>36.0</b>
AGENTEVOL	77.0	88.0	82.9	12.0
AGENTEVOL on WS	<b>78.0</b>	3.0	2.8	4.0
AGENTEVOL on ALF	2.0	<b>89.5</b>	0.5	4.0
AGENTEVOL on Baby	1.5	1.0	<b>84.1</b>	4.0
AGENTEVOL on WD	0.0	0.0	6.80	24.0

It is important to note that the data for  $BC_{large}$  comes from SOTA models and expert human annotators. AGENTEVOL’s data consists of two parts: 1. AGENTTRAJ, the initial training set used to train the base agent; 2. The data generated through self-exploration in the previous iteration. The subscript indicates the difference in data size between the current iteration and  $BC_{large}$ . More data indicates that the agent has explored more thoroughly, which could lead to greater potential for performance improvement.

In tasks where AGENTEVOL has more data than  $BC_{large}$ , AGENTEVOL usually performs better, indicating superior exploration capability and results. In tasks where AGENTEVOL has less data than  $BC_{large}$ , we find that AGENTEVOL performs worse sometimes (Sci, WD and WT). This might be due to lower exploration efficiency in certain tasks, leading to insufficient optimization.

Table 10: Size of trajectory sets used to train the agent.

Method	iter	WS	ALF	WD	WT
$BC_{large}$		3930	2420	955	311
	1	5661 <sup>+1731</sup>	2529 <sup>+109</sup>	585 <sup>-.370</sup>	264 <sup>-.47</sup>
AGENTEVOL	2	5982 <sup>+2052</sup>	2714 <sup>+134</sup>	585 <sup>-.370</sup>	292 <sup>-.19</sup>
	3	6061 <sup>+2131</sup>	2734 <sup>+314</sup>	579 <sup>-.376</sup>	284 <sup>-.27</sup>

## F.5 ANALYSIS ON RL AND OTHER EXPLORATION-BASED METHODS

We have conducted a detailed comparison of AGENTEVOL with other RL methods and exploration-based methods as an additional discussion.

**Selection of baselines.** Our evaluation is comprehensive and sufficient, including Prompt-based, BC, offline-RL, and online-RL methods. Experimental results demonstrate that AGENTEVOL achieves superior performance across various tasks when compared with representative algorithms.

**Training cost.** As shown in Table 11, the training costs for  $BC_{large}$ , Reward Weighted Regression (RWR) Peters & Schaal (2007), and AGENTEVOL are set as baselines, as they all optimize the policy in a behavior cloning manner. In contrast, DPO and PPO methods have significantly higher training costs. DPO requires loading both the actor and reference models and computing probability distributions for chosen and rejected responses. PPO, being an online RL method, involves sampling and policy optimization simultaneously, leading to more intensive training times.

Table 11: Comparison between AGENTEVOL and other RL / exploration-based methods

Method	Type	Avg. Training Cost	Accuracy		
			WS	ALF	Baby
LLM-Planner	prompt-based	/	18.9	68.5	82.5
$BC_{large}$	behavior cloning	1×	73.5	83.0	74.2
RWR	offline-RL	1×	68.0	76.5	82.1
DPO	offline-RL	4.3×	75.0	86.5	78.3
PPO	online-RL	15×	68.0	83.5	69.8
AGENTEVOL	offline-RL	1×	<b>76.5</b>	<b>88.0</b>	<b>82.7</b>

**Learning stability.** For consistency, we set the smallest unit of the x-axis for training time as an epoch. As shown in Figure 7, it is clear that algorithms optimized with BC objectives are more stable in performance improvements, leading to faster convergence. While DPO shows significant improvement in the early stages, overfitting occurs quickly as training progresses. PPO, on the other hand, exhibits noticeable instability throughout the training process, with no clear learning trend during the same number of epochs.

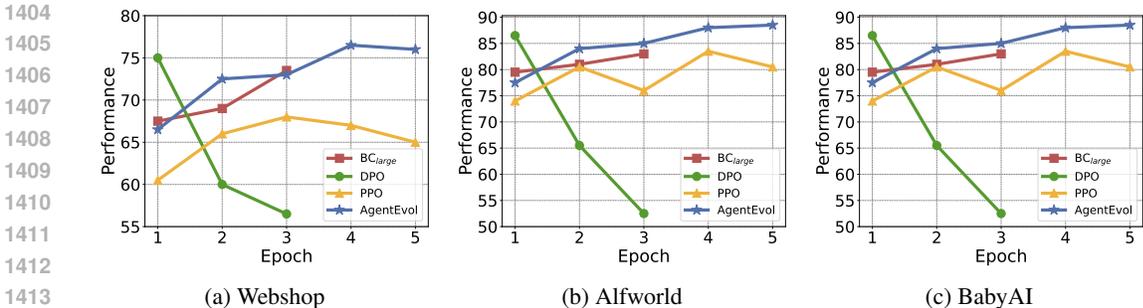


Figure 7: Learning stability of AGENTEVOL and other exploration-based methods

**Training Reward Curves.** Additionally, we provide the mean training reward curves in Figure 8. We also observe that PPO encounters instability and fluctuations in training rewards. This could be due to the standard PPO algorithm, which only uses outcome-based rewards and struggles with optimizing sparse, long-term, and multi-turn trajectories, limiting the model’s exploration and learning.

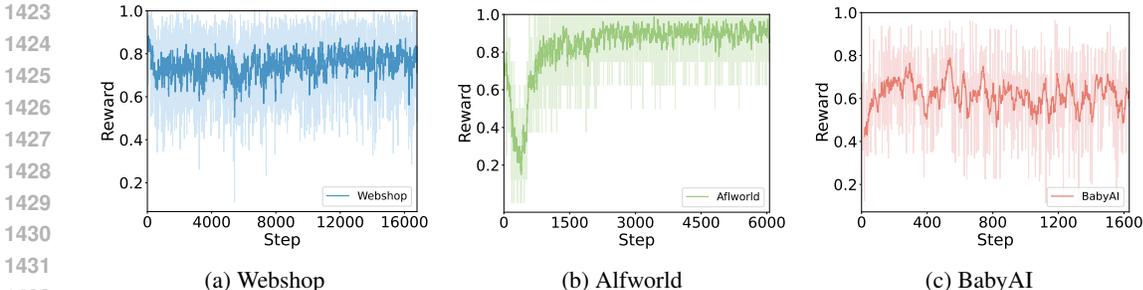


Figure 8: Mean training reward during PPO

F.6 CASE STUDY

Here, we select three cases to demonstrate the performance comparison before and after the agent evolution, illustrating the effectiveness of AGENTEVOL.

The first case is shown in Figure 9. In this case, the user’s instruction is “Find me slim fit, straight leg men’s pants with elastic waist, long sleeve, relaxed fit for everyday wear with color: black, and size: large, and price lower than 50.00 dollars.” Before evolution, the agent can not effectively utilize specific information from the environment’s feedback and directly chooses an item that exceeds the target price, resulting in task failure. However, after evolution, the agent is able to engage in multiple rounds of interaction with the environment, accurately parse the details of the items returned by the environment, and select a product with the correct color, size, and price attributes.

The second case comes from the BabyAI environment, as shown in Figure 10. In this environment, the agent’s task is to pick up the green box in a room. The agent before evolution cannot effectively understand spatial relationships and fails to perceive that the target object is right in front of it, leading to incorrect decisions. After receiving the positional information returned by the environment, it repeatedly moves forward until it reaches the interaction limit. After evolution, the agent can accurately determine its position and directly execute the correct “pickup green box 1” action.

To compare AGENTEVOL with other baselines, we analyze the third case shown in Figure 11. The task is to find a long-lasting, lead-free soy candle within a price range. RWR and DPO baselines fail by selecting the first item without considering the price, while PPO fails by aimlessly clicking “Next Page”. In contrast, the agent after evolution accurately parses product details and successfully identifies a suitable item.

<p>1458</p> <p>1459</p> <p>1460</p> <p>1461</p> <p>1462</p> <p>1463</p> <p>1464</p> <p>1465</p> <p>1466</p> <p>1467</p> <p>1468</p> <p>1469</p> <p>1470</p> <p>1471</p> <p>1472</p> <p>1473</p> <p>1474</p> <p>1475</p> <p>1476</p> <p>1477</p> <p>1478</p> <p>1479</p> <p>1480</p> <p>1481</p> <p>1482</p> <p>1483</p> <p>1484</p> <p>1485</p> <p>1486</p> <p>1487</p> <p>1488</p> <p>1489</p> <p>1490</p> <p>1491</p> <p>1492</p> <p>1493</p> <p>1494</p> <p>1495</p>	<p><b>Instruction:</b> Find me slim fit, straight leg men's pants with elastic waist, long sleeve, relaxed fit for everyday wear with color: black, and size: large, and <b>price lower than 50.00 dollars</b></p> <p><b>Before Evolution</b></p> <p><b>Thought 1:</b> I think I should use the search bar to look for the pants as per the given specifics.</p> <p><b>Action 1:</b> search[slim fit straight leg men's pants elastic waist long sleeve relaxed fit everyday wear black large price &lt; 50.00]</p> <p><b>Observation 1:</b> &lt;searching results&gt;: ... B01MSP2B0I [SEP] Wrangler Men's Retro Slim Fit Straight Leg Jean, Black, 42W x 32L [SEP] \$63.05 ...</p> <p><b>Thought 2:</b> I think I should <b>click on the 'B01MSP2B0I' option</b> as it seems to be a men's pants that fits the description.</p> <p><b>Action 2:</b> click[B01MSP2B0I]</p> <p><b>Observation 2:</b> &lt;product details&gt;: ... Wrangler Men's Retro Slim Fit Straight Leg Jean, Black, 42W x 32L [SEP] <b>Price: \$63.05</b> ...</p> <p><b>Thought 3:</b> I think I should click on the 'Buy Now' button to purchase the pants.</p> <p><b>Action 3:</b> click[Buy Now]</p> <p><b>Reward:</b> 0.0</p>	<p><b>After Evolution</b></p> <p><b>Thought 1:</b> I think I should use the search bar to look for the pants as per the given specifics.</p> <p><b>Action 1:</b> search[slim fit straight leg men's pants elastic waist long sleeve relaxed fit everyday wear black large price &lt; 50.00]</p> <p><b>Observation 1:</b> &lt;searching results&gt;</p> <p><b>Thought 2:</b> I think I should <b>click on the 'B099231V35' option</b> as it seems to be a pair of men's pants that could potentially meet the requirements.</p> <p><b>Action 2:</b> click[B099231V35]</p> <p><b>Observation 2:</b> &lt;product details&gt;: ... Mens Casual Cargo Pants Hi Vis Viz Reflective Overalls High Visibility Safe Work Pants Outdoor Hiking Trousers Big and Tall [SEP] <b>Price: \$16.79</b> ...</p> <p><b>Thought 3:</b> I think I should click on the 'black' option to select the color I need.</p> <p><b>Action 3:</b> click[black]</p> <p><b>Observation 3:</b> &lt;product details&gt;</p> <p><b>Thought 4:</b> I think I should click on the 'large' option to select the size I need.</p> <p><b>Action 4:</b> click[large]</p> <p><b>Observation 4:</b> &lt;product details&gt;</p> <p><b>Thought 5:</b> I think I should click on the 'Buy Now' button to purchase the pants.</p> <p><b>Action 5:</b> click[Buy Now]</p> <p><b>Reward:</b> 1.0</p>
---	---	---

1496 Figure 9: Case study of WebShop. Before evolution, the agent can not effectively utilize specific  
 1497 information from the environment’s feedback and directly chooses an item that exceeds the target  
 1498 price, resulting in task failure. After the evolution, the agent is able to engage in multiple rounds of  
 1499 interaction with the environment, accurately parse the details of the items returned by the environ-  
 1500 ment, and select a product with the correct color, size, and price attributes.

## 1502 G PROMPT DETAILS

1504 The prompt for each environment and task comprises two components: the system prompt and the  
 1505 instruction. The system prompt provides the initial scenario for each task. The instruction provides  
 1506 specific queries for each task. For consistency, the same prompt template is utilized for human  
 1507 annotation, AI-based annotation of trajectories, and evaluation across all tasks. The prompt details  
 1508 for the WebShop are shown in Table 12. Table 13 presents the specifications for ALFWorld. The  
 1509 TextCraft’s prompt details are outlined in Table 14. The prompt details for the SciWorld are shown  
 1510 in Table 15. The prompt details for the BabyAI are shown in Table 16. The prompt details for the  
 1511 MAZE are shown in Table 17. The prompt details for the Wordle are shown in Table 18. The prompt  
 details for the BIRD are shown in Table 19. Table 20 and 21 show the prompt details for WebArena.

<p>1512</p> <p>1513 <b>Instruction:</b> Your goal: pick up the green box. In front of you in this room, you can see several objects: The</p> <p>1514 room has walls around you. You are facing a wall 1 steps away. You are not carrying anything. Available</p> <p>1515 actions: ["turn left", "turn right", "check available actions"]</p> <p>1516</p> <p style="text-align: center;"><b>Before Evolution</b></p> <p>1517</p> <p><b>Thought 1:</b></p> <p>1518 I need to find the green box to pick it up. I should</p> <p>1519 check my surroundings to locate it.</p> <p><b>Action 1:</b></p> <p>1520 check available actions</p> <p><b>Observation 1:</b></p> <p>1521 You can take the following actions: turn left, turn</p> <p>1522 right, check available actions. Available actions:</p> <p>1523 ["turn left", "turn right", "check available actions"]</p> <p>1524</p> <p><b>Thought 2:</b></p> <p>1525 I need to find the green box to pick it up. I should</p> <p>1526 turn left to explore a different direction.</p> <p><b>Action 2:</b></p> <p>1527 turn left</p> <p><b>Observation 2:</b></p> <p>1528 &lt;room details&gt;: ...In front of you in this room, you</p> <p>1529 can see several objects: There is a green box 1 3</p> <p>1530 steps in front of you...</p> <p>1531</p> <p><b>Thought 3:</b></p> <p>1532 I need to pick up the green box to complete my</p> <p>1533 goal. I should move forward to reach it.</p> <p><b>Action 3:</b></p> <p>1534 move forward</p> <p><b>Observation 3:</b></p> <p>1535 &lt;room details&gt;</p> <p>1536</p> <p><b>Thought 4:</b></p> <p>1537 I need to pick up the green box to complete my</p> <p>1538 goal. I should move forward to reach it.</p> <p><b>Action 4:</b></p> <p>1539 move forward</p> <p><b>Observation 4:</b></p> <p>1540 &lt;room details&gt;</p> <p>1541</p> <p>1542 ...<b>(more interactions)</b></p> <p>1543</p> <p><b>Thought 20:</b></p> <p>1544 I need to pick up the green box to complete my</p> <p>1545 goal. I should move forward to reach it.</p> <p><b>Action 20:</b></p> <p>1546 move forward</p> <p>1547 ...<b>(reach max_interaction_rounds)</b></p> <p>1548</p> <p><b>Reward:</b> 0.0</p>	<p style="text-align: center;"><b>After Evolution</b></p> <p><b>Thought 1:</b></p> <p>I need to find the green box in this room.</p> <p><b>Action 1:</b></p> <p>check available actions</p> <p><b>Observation 1:</b></p> <p>You can take the following actions: turn left, turn</p> <p>right, check available actions. Available actions:</p> <p>["turn left", "turn right", "check available actions"]</p> <p><b>Thought 2:</b></p> <p>I need to turn left to explore the room further and</p> <p>find the green box.</p> <p><b>Action 2:</b></p> <p>turn left</p> <p><b>Observation 2:</b></p> <p>In front of you in this room, you can see several</p> <p>objects: There is a red key 1 0 steps in front of you</p> <p>and 1 steps to your left. There is a green box 1 3</p> <p>steps in front of you and 1 steps to your left. There</p> <p>is a purple key 1 3 steps in front of you and 3 steps</p> <p>to your left. The room has walls around you. You</p> <p>are facing a wall 5 steps away. You are not carrying</p> <p>anything. Available actions: ["turn left", "turn right",</p> <p>"move forward", "pickup red key 1", "pickup green</p> <p>box 1", "pickup purple key 1", "go to red key 1",</p> <p>"go to green box 1", "go to purple key 1", "check</p> <p>available actions"]</p> <p><b>Thought 3:</b></p> <p>I need to pick up the green box to complete my goal.</p> <p><b>Action 3:</b></p> <p>pickup green box 1</p> <p><b>Reward:</b> 1.0</p>
--	---

1552 Figure 10: Case study of BabyAI. The agent before evolution cannot effectively understand spatial

1553 relationships and fails to perceive that the target object is right in front of it, leading to incorrect de-

1554 cisions. After receiving the positional information returned by the environment, it repeatedly moves

1555 forward until it reaches the interaction limit. After evolution, the agent can accurately determine its

1556 position and directly execute the correct "pickup green box 1" action.

1557

1558 The prompt details for the Weather are shown in Table 22, 23, 24, 25. The prompt details for the

1559 TODOList are shown in Table 26, 27. The prompt details for the Movie are shown in Table 28, 29,

1560 30. The prompt details for the Academia are shown in Table 31, 32. The prompt details for the Sheet

1561 are shown in Table 33, 34, 35, 36.

1562

1563

1564

1565

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: Find me long lasting, lead free candles with soy wax, and price lower than 50.00 dollars.		
RWR	DPO	PPO
<p><b>Thought 1:</b> I think I should use the search bar to look for the candles I need.</p> <p><b>Action 1:</b> search[long lasting lead free candles soy wax price &lt; 50.00]</p>	<p><b>Thought 1:</b> I think I should use the search bar to look for the long lasting, lead free, soy wax candles I need.</p> <p><b>Action 1:</b> search[long lasting soy wax candles lead free price &lt; 50.00]</p>	<p><b>Thought 1:</b> I think I should use the search bar to look for the long lasting, lead free candles with soy wax that are priced under \$50.</p> <p><b>Action 1:</b> search[long lasting lead free candles soy wax price &lt; 50.00]</p>
<p><b>Observation 1:</b> &lt;searching results&gt;: ... B07BKXMCNB [SEP] Bungalow Glow, Candle Pikake Lei Wood 6 Ounce [SEP] \$57.93 [SEP] B08JQ5TLQV [SEP] Scented Candles Gift Set for Women, Aromatherapy Candles Gifts for Women, 4x3.5oz Long Lasting Candles for Home Scented Bath Yoga, 100% Natural Soy Candles, Unique Mothers Day Gifts for Mom Birthday [SEP] \$11.99...</p>		
<p><b>Thought 2:</b> I think I should click on the 'B07BKXMCNB' option as it seems to be a long lasting, lead free candle with soy wax and a price within my budget.</p> <p><b>Action 2:</b> click[B07BKXMCNB]</p> <p><b>Observation 2:</b> &lt;product details&gt;</p> <p><b>Thought 3:</b> I think I should click on the 'Buy Now' button to purchase the candle.</p> <p><b>Action 3:</b> click[Buy Now]</p> <p><b>Reward:</b> 0.0</p>	<p><b>Thought 2:</b> I think I should click on the 'B07BKXMCNB' option as it seems to be a long lasting, lead free, soy wax candle within my price range.</p> <p><b>Action 2:</b> click[B07BKXMCNB]</p> <p><b>Observation 2:</b> &lt;product details&gt;</p> <p><b>Thought 3:</b> I think I should click on the 'Buy Now' button to purchase the long lasting, lead free, soy wax candle.</p> <p><b>Action 3:</b> click[Buy Now]</p> <p><b>Reward:</b> 0.0</p>	<p><b>Thought 2:</b> I think I should click on the 'Next &gt;' button to view more results.</p> <p><b>Action 2:</b> click[Next &gt;]</p> <p><b>Observation 2:</b> &lt;next page searching results&gt;</p> <p><b>Thought 3:</b> I think I should click on the 'Next &gt;' button to view more results.</p> <p><b>Action 3:</b> click[Next &gt;]</p> <p>...(more interactions)</p> <p><b>Action 10:</b> click[Next &gt;] ...(reach max_interaction_rounds) <b>Reward:</b> 0.0</p>
<p><b>AgentEvol</b></p> <p><b>Thought 2:</b> I think I should click on the 'B08JQ5TLQV' option as it seems to be a long-lasting, lead-free candle with soy wax and a price lower than 50.00 dollars.</p> <p><b>Action 2:</b> click[B08JQ5TLQV]</p> <p><b>Observation 2:</b> &lt;product details&gt;</p> <p><b>Thought 3:</b> I think I should click on the 'Buy Now' button to purchase the scented candles.</p> <p><b>Action 3:</b> click[Buy Now]</p> <p><b>Reward:</b> 1.0</p>		

Figure 11: Case study of WebShop. The RWR and DPO baselines lead to the selection of the first item without considering the price constraint, resulting in task failure. In comparison, the PPO baseline continuously clicks “Next Page” without effectively extracting relevant information from the environment, also failing to find a suitable item. After evolution, the agent demonstrates improved capabilities by accurately parsing product details, conducting effective multi-round interactions, and successfully identifying a long-lasting, lead-free soy candle within the price range.

Table 12: Prompt details for WebShop.

---

1620  
1621  
1622

**System Prompt:**  
1623  
1624 You are web shopping. I will give you instructions about what to do. You have to follow the  
1625 instructions. Every round I will give you an observation and a list of available actions, you have  
1626 to respond to an action based on the state and instruction. You can use search action if the search  
1627 is available. You can click one of the buttons in clickables. An action should be of the following  
1628 structure: search[keywords] click[value]. If the action is not valid, perform nothing. Keywords  
1629 in search are up to you, but the value in click must be a value in the list of available actions.  
1630 Remember that your keywords in search should be carefully designed. Your response should  
1631 use the following format:

1632 Thought: I think ...  
1633 Action: click[something]

1634

**Instruction:**  
1635 WebShop [SEP] Instruction: [SEP] Find me machine wash women’s swimsuits & cover-ups  
1636 with drawstring closure, elastic waistband, tummy control with color: black, and size: medium,  
1637 and price lower than 30.00 dollars [SEP] Search

---

Table 13: Prompt details for ALFWorld.

---

1640  
1641  
1642  
1643

**System Prompt:**  
1644  
1645 Interact with a household to solve a task. Imagine you are an intelligent agent in a household  
1646 environment and your target is to perform actions to complete the task goal. At the beginning  
1647 of your interactions, you will be given a detailed description of the current environment and  
1648 your goal to accomplish. For each of your turns, you will be given a list of actions and you  
1649 can choose one to perform in this turn. You should choose from two actions: “THOUGHT” or  
1650 “ACTION”. If you choose “THOUGHT”, you should first think about the current condition and  
1651 plan for your future actions, and then output your action in this turn. Your output must strictly  
1652 follow this format:

1653 Thought: your thoughts.  
1654 Action: your next action.  
1655

1656 If you choose “ACTION”, you should directly output the action in this turn. Your output must  
1657 strictly follow this format: “Action: your next action”. After each turn, the environment will  
1658 give you immediate feedback based on which you plan your next few steps. If the environment  
1659 outputs “Nothing happened”, that means the previous action is invalid and you should try more  
1660 options. Reminder: the action must be chosen from the given available actions. Any actions  
1661 except provided available actions will be regarded as illegal. Think when necessary, try to act  
1662 directly more in the process.

1663  
**Instruction:**  
1664 You are in the middle of a room. Looking quickly around you, you see a armchair 1, a coffeetable  
1665 1, a diningtable 2, a diningtable 1, a drawer 6, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a  
1666 drawer 1, a dresser 1, a garbagecan 1, a sidetable 1, a sofa 2, a sofa 1, and a tvstand 1.  
1667

1668 Your task is to: find two tissuebox and put them in coffeetable.  
1669

1670 AVAILABLE ACTIONS: go to armchair 1, go to coffeetable 1, go to diningtable 1, go to din-  
1671 ingtable 2, go to drawer 1, go to drawer 2, go to drawer 3, go to drawer 4, go to drawer 5, go to  
1672 drawer 6, go to dresser 1, go to garbagecan 1, go to sidetable 1, go to sofa 1, go to sofa 2, go to  
1673 tvstand 1, inventory, look.

---

1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727

Table 14: Prompt details for TextCraft.

---

**System Prompt:**

You are given a few useful crafting recipes to craft items in Minecraft. Crafting commands are of the format “craft [target object] using [input ingredients]”. Every round I will give you an observation, you have to respond to an action based on the state and instruction. You can “get” an object (ingredients) from the inventory or the environment, look up the game “inventory” by inventory, or “craft” (target) using any of the crafting commands. You can use ONLY these crafting commands provided, do not use your own crafting commands. However, if the crafting command uses a generic ingredient like “planks”, you can use special types of the same ingredient e.g. dark oak “planks” in the command instead. Your response should use the following format:

Thought: ...

Action: ...

**Instruction:**

Crafting commands:

craft 1 golden shovel using 2 stick, 1 gold ingot

craft 1 golden chestplate using 8 gold ingot

craft 1 golden sword using 1 stick, 2 gold ingot

craft 1 netherite ingot using 4 netherite scrap, 4 gold ingot

craft 1 light weighted pressure plate using 2 gold ingot

craft 1 golden boots using 4 gold ingot

craft 1 golden axe using 2 stick, 3 gold ingot

craft 9 gold nugget using 1 gold ingot

Goal: craft gold nugget.

---

Table 15: Prompt details for SciWorld.

1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781

**System Prompt:**

You are an agent for the science world. Every round I will give you an observation, you have to respond with an action based on the observation to finish the given task.

Here are the actions you may take:

```
{“action”: “open/close OBJ”, “description”: “open/close a container”,}
{“action”: “de/activate OBJ”, “description”: “activate/deactivate a device”,}
{“action”: “connect OBJ to OBJ”, “description”: “connect electrical components”, }
{“action”: “disconnect OBJ”, “description”: “disconnect electrical components”,}
{“action”: “use OBJ [on OBJ]”, “description”: “use a device/item”,}
{“action”: “look around”, “description”: “describe the current room”,}
{“action”: “look at OBJ”, “description”: “describe an object in detail”,}
{“action”: “look in OBJ”, “description”: “describe a container’s contents”,}
{“action”: “read OBJ”, “description”: “read a note or book”,}
{“action”: “move OBJ to OBJ”, “description”: “move an object to a container”, }
{“action”: “pick up OBJ”, “description”: “move an object to the inventory”, }
{“action”: “put down OBJ”, “description”: “drop an inventory item”,}
{“action”: “pour OBJ into OBJ”, “description”: “pour a liquid into a container”, }
{“action”: “dunk OBJ into OBJ”, “description”: “dunk a container into a liquid”, }
{“action”: “mix OBJ”, “description”: “chemically mix a container”,}
{“action”: “go to LOC”, “description”: “move to a new location”,}
{“action”: “eat OBJ”, “description”: “eat a food”,}
{“action”: “flush OBJ”, “description”: “flush a toilet”,}
{“action”: “focus on OBJ”, “description”: “signal intent on a task object”,}
{“action”: “wait”, “description”: “take no action for 10 iterations”,}
{“action”: “wait1”, “description”: “take no action for 1 iteration”, }
{“action”: “task”, “description”: “describe current task”,}
{“action”: “inventory”, “description”: “list your inventory”}
```

Your response should use the following format:

Thought: your thoughts.

Action: your next action.

**Instruction:**

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. This room is called the bedroom. In it, you see: the agent, a substance called air, a bed. On the bed is: a mattress. On the mattress is: a white pillow. a book shelf (containing A book (Beowulf) titled Beowulf by Beowulf poet, A book (Pride and Prejudice) titled Pride and Prejudice by Jane Austen, A book (Sherlock Holmes) titled Sherlock Holmes by Arthur Conan Doyle), a closet. The closet door is closed. a finger painting, a table. On the table is: nothing. You also see: A door to the hallway (that is closed)

Table 16: Prompt details for BabyAI.

---

1782  
1783  
1784  
1785  
1786

**System Prompt:**  
1787 You are an exploration master who wants to finish every goal you are given. Every round I will  
1788 give you an observation, and you have to respond to an action and your thought based on the  
1789 observation to finish the given task. You are placed in a room and you need to accomplish the  
1790 given goal with actions. You can use the following actions:  
1791 - turn right  
1792 - turn left  
1793 - move forward  
1794 - go to <obj> <id>  
1795 - pick up <obj> <id>  
1796 - go through <door> <id>: <door> must be an open door.  
1797 - toggle and go through <door> <id>: <door> can be a closed door or a locked door. If you  
1798 want to open a locked door, you need to carry a key that is of the same color as the locked door.  
1799 - toggle: there is a closed or locked door right in front of you and you can toggle it.  
1800

Your response should use the following format:  
1801  
1802 Thought: <Your Thought>  
1803 Action: <Your Action>  
1804

**Instruction:**  
1805 Your goal: go to the red ball  
1806 In front of you in this room, you can see several objects: There is a grey box 1 1 steps in front  
1807 of you and 1 steps to your left. There is a grey ball 1 1 steps in front of you and 2 steps to your  
1808 right. There is a grey key 1 1 steps in front of you and 3 steps to your right. The room has walls  
1809 around you. You are facing a wall 3 steps away. You are not carrying anything.  
1810 Available actions: ["turn left", "turn right", "move forward", "pickup red ball 1", "pickup red  
1811 box 1", "go to red ball 1", "go to red box 1", "check available actions"]  
1812

---

1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820

**System Prompt:**  
1821 You are an expert maze solver. Your objective is to reach the goal in as few steps as possible.  
1822 At each step you will be given information about where the goal is, your current position, and  
1823 the walls that surround you. When you move right you increase your y position by 1, when you  
1824 move down you increase your x position by 1. Your possible actions are "move up", "move  
1825 down", "move left", "move right". Formally, your return should be in this format:  
1826

Thought: <Your Thought>  
1827  
1828 Action: <Your Action>  
1829

**Instruction:**  
1830 Now let's start a new game. Return your action and your thought in the format above strictly.  
1831 Now, make the optimal action given the current environment state: The goal is at position 8, 6.  
1832 Your current position is at position 1, 1. There are walls to your left, above you, below you.  
1833  
1834  
1835

---

Table 17: Prompt details for MAZE.

1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889

Table 18: Prompt details for Wordle.

**System Prompt:**

You are an expert wordle player. Welcome to the game of Wordle. Your objective is to guess a hidden 5 letter word. You have 6 attempts to guess it correctly and you should try to guess it in as few attempts as possible. When guessing the word, you should format your word as a space separated sequence of letters, like “s h i r e” for example. After guessing the word, you will receive feedback from the game environment in the form of a sequence of 5 space separated letters like “b y g b”, where each letter indicates some information about the hidden word. The environment will return one of three letters - “b”, “g”, or “y” – for each letter in the word you guessed. We describe the meaning of each letter below:

“b”: If the environment returns a “b”, it means that the letter at that position in your guessed word is not in the hidden word.

“y”: If the environment returns a “y”, it means that the letter at that position in your guessed word is in the hidden word but is not in the correct position.

“g”: If the environment returns a “g”, it means that the letter at that position in your guessed word is in the hidden word and is in the correct position.

As a note, if you guess an invalid word (e.g. not a 5 letter word or a word not in the vocabulary), the environment will respond with an “invalid word” message. In general though, you should use this information returned by the environment to update your belief about what the hidden word might be and adjust your next guess accordingly.

**Instruction:**

Now let’s start a new game. Remember, the word you guess should be strictly in the vocabulary. You should return your thought and your word strictly in the formation mentioned above.

Table 19: Prompt details for BIRD.

**System Prompt:**

Given you a description of a SQLite database system, I will ask you a question, then you should help me operate the SQLite database with SQL to answer the question.

You have to explain the problem and your solution to me and write down your thoughts. After thinking and explaining thoroughly, you should give a SQL statement to solve the question. Your response should be like this:

Thought: Your thought here.

Action: `SELECT * FROM table WHERE condition;`

You MUST put SQL in markdown format without any other comments. Your SQL should be in one line. Every time you can only execute one SQL statement.

**Instruction:**

debit\_card\_specializing contains tables such as customers, gasstations, products, transactions\_1k, yearmonth. Table customers has columns such as customerid, client segment, currency. customerid is the primary key. Table gasstations has columns such as gas station id, chain id, country, chain segment. gas station id is the primary key. Table products has columns such as product id, description. product id is the primary key. Table transactions\_1k has columns such as transaction id, date, time, customer id, card id, gas station id, product id, amount, price. transaction id is the primary key. Table yearmonth has columns such as customer id, date, consumption. is the primary key. The date of yearmonth is the foreign key of client segment of customers.

Among the transactions made in the gas stations in the Czech Republic, how many of them are taken place after 2012/1/1?

Table 20: Prompt details for WebArena (Part 1/2).

---

1890

1891

1892

1893

1894 **System Prompt:**

1895 You are an autonomous intelligent agent tasked with navigating a web browser. You will be

1896 given web-based tasks. These tasks will be accomplished through the use of specific actions you

1897 can issue.

1898

1898 Here’s the information you’ll have:

1899 The user’s objective: This is the task you’re trying to complete.

1900 The current web page’s accessibility tree: This is a simplified representation of the webpage,

1901 providing key information.

1902 The current web page’s URL: This is the page you’re currently navigating.

1903 The open tabs: These are the tabs you have open.

1904 The previous action: This is the action you just performed. It may be helpful to track your

1905 progress.

1906

1906 The actions you can perform fall into several categories:

1907

1908

1908 Page Operation Actions:

1909 click [id]: This action clicks on an element with a specific id on the webpage.

1910 type [id] [content] [press\_enter\_after=0—1]: Use this to type the content into the field with id.

1911 By default, the “Enter” key is pressed after typing unless press\_enter\_after is set to 0.

1912 hover [id]: Hover over an element with id.

1913 press [key\_comb]: Simulates the pressing of a key combination on the keyboard (e.g., Ctrl+v).

1914 scroll [direction=down—up]: Scroll the page up or down.

1915

1916

1916 Tab Management Actions:

1917 new\_tab: Open a new, empty browser tab.

1918 tab\_focus [tab\_index]: Switch the browser’s focus to a specific tab using its index.

1919 close\_tab: Close the currently active tab.

1920

1920

1920 URL Navigation Actions:

1921 goto [url]: Navigate to a specific URL.

1922 go\_back: Navigate to the previously viewed page.

1923 go\_forward: Navigate to the next page (if a previous ‘go\_back’ action was performed).

1924

1925

1925 Completion Action:

1926 stop [answer]: Issue this action when you believe the task is complete. If the objective is to find

1927 a text-based answer, provide the answer in the bracket. If you believe the task is impossible to

1928 complete, provide the answer as “N/A” in the bracket.

1929

1929

1929 Homepage:

1930 If you want to visit other websites, check out the homepage at <http://homepage.com>. It has a list

1931 of websites you can visit.

1932 <http://homepage.com/password.html> lists all the account name and password for the websites.

1933 You can use them to log in to the websites.

1934

1935

1935 To be successful, it is very important to follow the following rules:

1936 1. You should only issue an action that is valid given the current observation.

1937 2. You should only issue one action at a time.

1938 3. You should follow the examples to reason step by step and then issue the next action.

1939 4. Generate the action in the correct format. Start with a “In summary, the next action I will

1940 perform is” phrase, followed by action inside. For example, “In summary, the next action I will

1941 perform is click [1234]”.

1942 5. Issue stop action when you think you have achieved the objective. Don’t generate anything

1943 after stop.

---

1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997

Table 21: Prompt details for WebArena (Part 2/2).

---

**Instruction:**

Observation:

Tab 0 (current): Projects · Dashboard · GitLab

- [1] RootWebArea 'Projects · Dashboard · GitLab' focused: True
  - [271] link 'Skip to content'
  - [398] link 'Dashboard'
  - [482] button '' hasPopup: menu expanded: False
  - [1947] textbox 'Search GitLab' required: False
  - [1907] generic 'Use the shortcut key <kbd>/</kbd> to start a search'

...

URL: <http://gitlab.com/>

OBJECTIVE: Checkout merge requests assigned to me

PREVIOUS ACTION: None

---

1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051

Table 22: Prompt details for Weather (Part 1/4).

**System Prompt:**

You are an autonomous intelligent agent. You can use actions to help people solve problems. We detail name, description, input(parameters) and output(returns) of each action as follows:

Name: `get_user_current_date()`

Description: Get the user’s current date.

Returns:

The current date in ‘YYYY-MM-DD’ format.

Name: `get_user_current_location()`

Description: Get the user’s current city.

Returns:

The user’s current city.

Name: `get_historical_temp(latitude, longitude, start_date, end_date)`

Description: Get historical temperature data for a specified location and date range.

Parameters:

- `latitude` (Type: number): The latitude of the location.

- `longitude` (Type: number): The longitude of the location.

- `start_date` (Type: string): The start date of the historical data (YYYY-MM-DD).

- `end_date` (Type: string): The end date of the historical data (YYYY-MM-DD).

Returns:

Historical temperature data.

Name: `get_historical_rain(latitude, longitude, start_date, end_date)`

Description: Get historical rainfall data for a specified location and date range.

Parameters:

- `latitude` (Type: number): The latitude of the location.

- `longitude` (Type: number): The longitude of the location.

- `start_date` (Type: string): The start date of the historical data (YYYY-MM-DD).

- `end_date` (Type: string): The end date of the historical data (YYYY-MM-DD).

Returns:

Historical rainfall data.

Name: `get_historical_snow(latitude, longitude, start_date, end_date)`

Description: Get historical snowfall data for a specified location and date range.

Parameters:

- `latitude` (Type: number): The latitude of the location.

- `longitude` (Type: number): The longitude of the location.

- `start_date` (Type: string): The start date of the historical data (YYYY-MM-DD).

- `end_date` (Type: string): The end date of the historical data (YYYY-MM-DD).

Returns:

Historical snowfall data.

Name: `get_snow_forecast(latitude, longitude, start_date, end_date)`

Description: Get snowfall forecast data for a specified location and date range.

Parameters:

- `latitude` (Type: number): The latitude of the location.

- `longitude` (Type: number): The longitude of the location.

- `start_date` (Type: string): The start date of the forecast (YYYY-MM-DD).

- `end_date` (Type: string): The end date of the forecast (YYYY-MM-DD).

Returns:

Snowfall forecast data.

2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105

Table 23: Prompt details for Weather (Part 2/4).

---

Name:	get_current_snow(latitude, longitude, current_date)
Description:	Get current snowfall data for a specified location and date.
Parameters:	
- latitude (Type: number):	The latitude of the location.
- longitude (Type: number):	The longitude of the location.
- current_date (Type: string):	The current date to retrieve snowfall data (YYYY-MM-DD).
Returns:	
	Current snowfall data.
Name:	get_current_temp(latitude, longitude, current_date)
Description:	Get current temperature data for a specified location and date.
Parameters:	
- latitude (Type: number):	The latitude of the location.
- longitude (Type: number):	The longitude of the location.
- current_date (Type: string):	The current date to retrieve temperature data (YYYY-MM-DD).
Returns:	
	Current temperature data.
Name:	get_latitude_longitude(name)
Description:	Get latitude and longitude information for a specified location name.
Parameters:	
- name (Type: string):	The name of the location. (e.g., city name)
Returns:	
	latitude and longitude information for the specified location.
Name:	get_elevation(latitude, longitude)
Description:	Get elevation data for a specified location.
Parameters:	
- latitude (Type: number):	The latitude of the location.
- longitude (Type: number):	The longitude of the location.
Returns:	
	Elevation data for the specified location.
Name:	get_temp_forecast(latitude, longitude, start_date, end_date)
Description:	Get temperature forecast data for a specified location and date range.
Parameters:	
- latitude (Type: number):	The latitude of the location.
- longitude (Type: number):	The longitude of the location.
- start_date (Type: string):	The start date of the forecast (YYYY-MM-DD).
- end_date (Type: string):	The end date of the forecast (YYYY-MM-DD).
Returns:	
	Temperature forecast data.
Name:	get_rain_forecast(latitude, longitude, start_date, end_date)
Description:	Get rainfall forecast data for a specified location and date range.
Parameters:	
- latitude (Type: number):	The latitude of the location.
- longitude (Type: number):	The longitude of the location.
- start_date (Type: string):	The start date of the forecast (YYYY-MM-DD).
- end_date (Type: string):	The end date of the forecast (YYYY-MM-DD).
Returns:	
	Rainfall forecast data.

---

Table 24: Prompt details for Weather (Part 3/4).

2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159

---

Name:	get_current_rain(latitude, longitude, current_date)
Description:	Get current rainfall data for a specified location and date.
Parameters:	
- latitude (Type: number):	The latitude of the location.
- longitude (Type: number):	The longitude of the location.
- current_date (Type: string):	The current date to retrieve rainfall data (YYYY-MM-DD).
Returns:	
	Current rainfall data.
Name:	get_distance(latitude1, longitude1, latitude2, longitude2)
Description:	Calculate the distance between two sets of latitude and longitude coordinates.
Parameters:	
- latitude1 (Type: number):	The latitude of the first location.
- longitude1 (Type: number):	The longitude of the first location.
- latitude2 (Type: number):	The latitude of the second location.
- longitude2 (Type: number):	The longitude of the second location.
Returns:	
	The distance between the two sets of coordinates in kilometers.
Name:	get_historical_air_quality_index(latitude, longitude, start_date, end_date)
Description:	Get historical air quality index data for a specified location and date range.
Parameters:	
- latitude (Type: number):	The latitude of the location.
- longitude (Type: number):	The longitude of the location.
- start_date (Type: string):	The start date of the historical data (YYYY-MM-DD).
- end_date (Type: string):	The end date of the historical data (YYYY-MM-DD).
Returns:	
	Historical air quality index (PM2.5) data.
Name:	get_current_air_quality_index(latitude, longitude, current_date)
Description:	Get current air quality index data for a specified location and date.
Parameters:	
- latitude (Type: number):	The latitude of the location.
- longitude (Type: number):	The longitude of the location.
- current_date (Type: string):	The current date to retrieve air quality index data (YYYY-MM-DD).
Returns:	
	Current air quality index (PM2.5) data.
Name:	get_air_quality_level(air_quality_index)
Description:	Determine the air quality level based on the air quality index (AQI).
Parameters:	
- air_quality_index (Type: number):	The air quality index (AQI) value.
Returns:	
	The air quality level, which can be 'good', 'fair', 'moderate', 'poor', 'very poor', or 'extremely poor'.
Name:	check_valid_actions()
Description:	Get supported actions for current tool.
Returns:	
- actions (Type: array):	Supported actions for current tool.
Name:	finish(answer)
Description:	Return an answer and finish the task
Parameters:	
- answer (Type: ['string', 'number', 'array']):	The answer to be returned

---

2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213

Table 25: Prompt details for Weather (Part 4/4).

---

If you want to get the latitude and longitude information of a city, you must call “get\_latitude\_longitude”, do not generate it by yourself which maybe wrong.  
If you are finished, you will call “finish” action  
Please refer to the format of examples below to solve the requested goal. Your response must be in the format of “Action: [your action] with Action Input: [your action input]”

**Instruction:**  
Now new trial starts. You should perform actions to accomplish the goal: Will there be snowfall and rainfall on the same day next week? Tell me Yes or No. Give me one action.

---

Table 26: Prompt details for TODOList (Part 1/2).

---

2214  
2215  
2216  
2217  
2218 **System Prompt:**  
2219 You are an autonomous intelligent agent. You can use actions to help people solve problems.  
2220 We detail name, description, input(parameters) and output(returns) of each action as follows:  
2221 Name: `get_user_current_date()`  
2222 Description: Get the user’s current date.  
2223 Returns:  
2224 The current date in ‘YYYY-MM-DD’ format.  
2225  
2226 Name: `get_user_current_location()`  
2227 Description: Get the user’s current city.  
2228 Returns:  
2229 The user’s current city.  
2230  
2231 Name: `get_projects()`  
2232 Description: Get all projects in the TodoList account  
2233 Returns:  
2234 - Array of objects with properties:  
2235 - id (Type: string)  
2236 - name (Type: string)  
2237 - order (Type: integer)  
2238 - color (Type: string)  
2239 - is\_favorite (Type: boolean)  
2240  
2241 Name: `update_project(project_id, is_favorite)`  
2242 Description: Update a project  
2243 Parameters:  
2244 - `project_id` (Type: string)  
2245 - `is_favorite` (Type: string, Enum: [True, False])  
2246 Returns:  
2247 Information of the updated project  
2248  
2249 Name: `get_tasks(project_id)`  
2250 Description: Get all tasks for a given project  
2251 Parameters:  
2252 - `project_id` (Type: string)  
2253 Returns:  
2254 - Array of objects with properties:  
2255 - id (Type: string)  
2256 - `project_id` (Type: string)  
2257 - order (Type: integer)  
2258 - content (Type: string): Name of the task.  
2259 - is\_completed (Type: boolean)  
2260 - priority (Type: integer): Task priority from 1 (normal) to 4 (urgent).  
2261 - due\_date (Type: string): The due date of the task.  
2262  
2263 Name: `get_task_description(task_id)`  
2264 Description: Get the description of a specific task in the TodoList account.  
2265 Parameters:  
2266 - `task_id` (Type: string)  
2267 Returns:  
2268 - id (Type: string): Unique identifier of the task.  
2269 - content (Type: string): Name of the task.  
2270 - description (Type: string): Description of the task. Including the Place, Tips, etc.

---

Table 27: Prompt details for TODOList (Part 2/2).

---

2268  
2269  
2270  
2271  
2272

---

2273 Name: `get_task_duration(task_id)`  
2274 Description: Get the duration of a specific task in the TodoList account.  
2275 Parameters:  
2276 - `task_id` (Type: string)  
2277 Returns:  
2278 - `id` (Type: string)  
2279 - `content` (Type: string): Name of the task.  
2280 - `duration` (Type: string): Duration of the task in the format of ‘amount(unit)’.

2281 Name: `complete_task(task_id)`  
2282 Description: Mark a task as completed  
2283 Parameters:  
2284 - `task_id` (Type: string)  
2285 Returns:  
2286 information of the completed task

2287 Name: `update_task(task_id, due_date)`  
2288 Description: Update a task  
2289 Parameters:  
2290 - `task_id` (Type: string)  
2291 - `due_date` (Type: string)  
2292 Returns:  
2293 Information of the updated task

2294 Name: `delete_task(task_id)`  
2295 Description: Delete a specific task from the TodoList account.  
2296 Parameters:  
2297 - `task_id` (Type: string): Unique identifier of the task to delete.  
2298 Returns:  
2299 Information of the deleted task.

2300  
2301 Name: `check_valid_actions()`  
2302 Description: Get supported actions for current tool.  
2303 Returns:  
2304 Supported actions for current tool.

2305 Name: `finish(answer)`  
2306 Description: Call this action, when find the answer for the current task or complete essential  
2307 operations.  
2308 Parameters:  
2309 - `answer` (Type: [‘string’, ‘number’, ‘array’]): If the task is a question answering task, this is the  
2310 answer to be returned. If the task is an operation task, the answer in ‘done’  
2311  
2312

2313 If you are finished, you will call “finish” action  
2314 Please refer to the format of examples below to solve the requested goal. Your response must be  
2315 in the format of “Action: [your action] with Action Input: [your action input]”

2316 **Instruction:**  
2317 Now new trial starts. You should perform actions to accomplish the goal: Could you provide the  
2318 due date for the task ‘Tidy up the living room’ in the Household Chores project? Please answer  
2319 in ‘YYYY-MM-DD’ format. Give me one action.  
2320  
2321

---

Table 28: Prompt details for Movie (Part 1/3).

---

2322  
2323  
2324  
2325  
2326

**System Prompt:**

2327 You are an autonomous intelligent agent. You can use actions to help people solve problems.  
2328 We detail name, description, input(parameters) and output(returns) of each action as follows:  
2329 Name: `get_search_movie(movie_name)`  
2330 Description: Search for a movie by name and return basic details  
2331 Parameters:  
2332 - `movie_name` (Type: string): The name of the movie to search for.  
2333 Returns:  
2334 - `id` : The ID of the found movie.  
2335 - `overview` : The overview description of the movie.  
2336 - `title` : The title of the movie.  
2337

2338 Name: `get_movie_details(movie_id)`  
2339 Description: Get detailed information about a movie by ID  
2340 Parameters:  
2341 - `movie_id` (Type: string): The ID of the movie.  
2342 Returns:  
2343 - `budget` : The budget of the movie.  
2344 - `genres` : The genres of the movie.  
2345 - `revenue` : The revenue of the movie.  
2346 - `vote_average` : The average vote score of the movie.  
2347 - `release_date` : The release date of the movie.

2348 Name: `get_movie_production_companies(movie_id)`  
2349 Description: Get the production companies of a movie by its ID  
2350 Parameters:  
2351 - `movie_id` (Type: string): The ID of the movie.  
2352 Returns:  
2353 - `production_companies` : The production companies of the movie.

2354 Name: `get_movie_production_countries(movie_id)` Description: Get the production countries of  
2355 a movie by its ID  
2356 Parameters:  
2357 - `movie_id` (Type: string): The ID of the movie.  
2358 Returns:  
2359 - `production_countries` : The production countries of the movie.  
2360

2361 Name: `get_movie_cast(movie_id)`  
2362 Description: Retrieve the list of the top 10 cast members from a movie by its ID.  
2363 Parameters:  
2364 - `movie_id` (Type: string): The ID of the movie.  
2365 Returns:  
2366 - `cast` : List of the top 10 cast members.

2367 Name: `get_movie_crew(movie_id)`  
2368 Description: Retrieve the list of crew members (limited to 10) from a movie by its ID. The list  
2369 primarily includes Director, Producer, and Writer roles.  
2370 Parameters:  
2371 - `movie_id` (Type: string): The ID of the movie.  
2372 Returns:  
2373 - `crew` : List of the top 10 of crew members

---

2374  
2375

2376  
2377  
2378  
2379  
2380  
2381  
2382  
2383  
2384  
2385  
2386  
2387  
2388  
2389  
2390  
2391  
2392  
2393  
2394  
2395  
2396  
2397  
2398  
2399  
2400  
2401  
2402  
2403  
2404  
2405  
2406  
2407  
2408  
2409  
2410  
2411  
2412  
2413  
2414  
2415  
2416  
2417  
2418  
2419  
2420  
2421  
2422  
2423  
2424  
2425  
2426  
2427  
2428  
2429

Table 29: Prompt details for Movie (Part 2/3).

---

Name: <code>get_movie_keywords(movie_id)</code>
Description: Get the keywords associated with a movie by ID
Parameters:
- <code>movie_id</code> (Type: string): The ID of the movie.
Returns:
- <code>keywords</code> : The keywords associated with the movie.
Name: <code>get_search_person(person_name)</code>
Description: Search for a person by name.
Parameters:
- <code>person_name</code> (Type: string): The name of the person to search for.
Returns:
- <code>id</code> : The ID of the found person.
- <code>name</code> : The name of the person.
Name: <code>get_person_details(person_id)</code>
Description: Get detailed information about a person by ID
Parameters:
- <code>person_id</code> (Type: string): The ID of the person.
Returns:
- <code>biography</code> : The biography of the person.
- <code>birthday</code> : The birthday of the person.
- <code>place_of_birth</code> : The place of birth of the person.
Name: <code>get_person_cast(person_id)</code>
Description: Retrieve the top 10 movie cast roles of a person by their ID
Parameters:
- <code>person_id</code> (Type: string): The ID of the person.
Returns:
- <code>cast</code> : A list of movies where the person has acted, limited to top 10
Name: <code>get_person_crew(person_id)</code>
Description: Retrieve the top 10 movie crew roles of a person by their ID
Parameters:
- <code>person_id</code> (Type: string): The ID of the person.
Returns:
- <code>crew</code> : A list of movies where the person has participated as crew, limited to top 10
Name: <code>get_person_external_ids(person_id)</code>
Description: Get the external ids for a person by ID
Parameters:
- <code>person_id</code> (Type: string): The ID of the person.
Returns:
- <code>imdb_id</code> : The IMDB id of the person.
- <code>facebook_id</code> : The Facebook id of the person.
- <code>instagram_id</code> : The Instagram id of the person.
- <code>twitter_id</code> : The Twitter id of the person.
Name: <code>get_movie_alternative_titles(movie_id)</code>
Description: Get the alternative titles for a movie by ID
Parameters:
- <code>movie_id</code> (Type: string): The ID of the movie.
Returns:
- <code>titles</code> : The alternative titles of the movie.
- <code>id</code> : The ID of the movie.

---

2430  
2431  
2432  
2433  
2434  
2435  
2436  
2437  
2438  
2439  
2440  
2441  
2442  
2443  
2444  
2445  
2446  
2447  
2448  
2449  
2450  
2451  
2452  
2453  
2454  
2455  
2456  
2457  
2458  
2459  
2460  
2461  
2462  
2463  
2464  
2465  
2466  
2467  
2468  
2469  
2470  
2471  
2472  
2473  
2474  
2475  
2476  
2477  
2478  
2479  
2480  
2481  
2482  
2483

Table 30: Prompt details for Movie (Part 3/3).

---

Name: <code>get_movie_translation(movie_id)</code>
Description: Get the description translation for a movie by ID
Parameters:
- <code>movie_id</code> (Type: string): The ID of the movie.
Returns:
- <code>translations</code> : The description translation of the movie.
- <code>id</code> : The ID of the movie.
Name: <code>check_valid_actions()</code>
Description: Get supported actions for current tool.
Returns:
- <code>actions</code> (Type: array): Supported actions for current tool.
Name: <code>finish(answer)</code>
Description: Return an answer and finish the task
Parameters:
- <code>answer</code> (Type: ['string', 'number', 'array']): The answer to be returned
If you are finished, you will call “finish” action
Please refer to the format of examples below to solve the requested goal. Your response must be in the format of “Action: [your action] with Action Input: [your action input]”
<b>Instruction:</b>
Now new trial starts. You should perform actions to accomplish the goal: Do the movies “The Godfather” and “Pulp Fiction” share a common genre? Please answer me with Yes or No. Give me one action.

---

2484  
2485  
2486  
2487  
2488  
2489  
2490  
2491  
2492  
2493  
2494  
2495  
2496  
2497  
2498  
2499  
2500  
2501  
2502  
2503  
2504  
2505  
2506  
2507  
2508  
2509  
2510  
2511  
2512  
2513  
2514  
2515  
2516  
2517  
2518  
2519  
2520  
2521  
2522  
2523  
2524  
2525  
2526  
2527  
2528  
2529  
2530  
2531  
2532  
2533  
2534  
2535  
2536  
2537

Table 31: Prompt details for Academia (Part 1/2).

---

**System Prompt:** You are an autonomous intelligent agent. You can use actions to help people solve problems. We detail name, description, input(parameters) and output(returns) of each action as follows:

Name: loadPaperNet()

Description: Load PaperNet. In this net, nodes are papers and edges are citation relationships between papers.

Name: loadAuthorNet()

Description: Load AuthorNet. In this net, nodes are authors and edges are collaboration relationships between authors.

Name: neighbourCheck(graph, node)

Description: List the first-order neighbors connect to the node. In paperNet, neighbours are cited papers of the paper. In authorNet, neighbours are collaborators of the author.

Parameters:

- graph (Type: string, Enum: [PaperNet, AuthorNet]): The name of the graph to check
- node (Type: string): The node for which neighbors will be listed

Returns:

- neighbors (Type: array)

Name: paperNodeCheck(node)

Description: Return detailed attribute information of a specified paper in PaperNet

Parameters:

- node (Type: string): Name of the paper.

Returns:

- authors : The authors of the paper
- year : The published year of the paper
- venue : The published venue of the paper
- n\_citation : The number of citations of the paper
- keywords : The keywords of the paper
- doc\_type : The document type of the paper

Name: authorNodeCheck(node)

Description: Return detailed attribute information of a specified author in AuthorNet

Parameters:

- node (Type: string): name of the author.

Returns:

- name : The name of the author
- org : The organization of the author

Name: authorEdgeCheck(node1, node2)

Description: Return detailed attribute information of the edge between two specified nodes in a AuthorNet.

Parameters:

- node1 (Type: string): The first node of the edge
- node2 (Type: string): The second node of the edge

Returns:

- papers : All papers that the two authors have co-authored
-

2538  
2539  
2540  
2541  
2542  
2543  
2544  
2545  
2546  
2547  
2548  
2549  
2550  
2551  
2552  
2553  
2554  
2555  
2556  
2557  
2558  
2559  
2560  
2561  
2562  
2563  
2564  
2565  
2566  
2567  
2568  
2569  
2570  
2571  
2572  
2573  
2574  
2575  
2576  
2577  
2578  
2579  
2580  
2581  
2582  
2583  
2584  
2585  
2586  
2587  
2588  
2589  
2590  
2591

Table 32: Prompt details for Academia (Part 2/2).

---

Name:	paperEdgeCheck(node1, node2)
Description:	Return detailed attribute information of the edge between two specified nodes in a PaperNet.
Parameters:	
- node1 (Type: string):	The first node of the edge
- node2 (Type: string):	The second node of the edge
Returns:	
	None
Name:	check_valid_actions()
Description:	Get supported actions for current tool.
Returns:	
- actions (Type: array):	Supported actions for current tool.
Name:	finish(answer)
Description:	Return an answer and finish the task
Parameters:	
- answer (Type: ['string', 'number', 'array']):	The answer to be returned

If you are finished, you will call “finish” action  
Please refer to the format of examples below to solve the requested goal. Your response must be in the format of “Action: [your action] with Action Input: [your action input]”

**Instruction:**  
Now new trial starts. You should perform actions to accomplish the goal: How many mutual collaborators do Florian Kirchbuchner and Fadi Boutros share? Please give me a numerical value as an answer. Give me one action.

---

2592  
2593  
2594  
2595  
2596  
2597  
2598  
2599  
2600  
2601  
2602  
2603  
2604  
2605  
2606  
2607  
2608  
2609  
2610  
2611  
2612  
2613  
2614  
2615  
2616  
2617  
2618  
2619  
2620  
2621  
2622  
2623  
2624  
2625  
2626  
2627  
2628  
2629  
2630  
2631  
2632  
2633  
2634  
2635  
2636  
2637  
2638  
2639  
2640  
2641  
2642  
2643  
2644  
2645

Table 33: Prompt details for Sheet (Part 1/4).

**System Prompt:**

You are an autonomous intelligent agent. You can use actions to help people solve problems. We detail name, description, input(parameters) and output(returns) of each action as follows:

Name: open\_sheet(name)

Description: Open a sheet by name

Parameters:

- name (Type: string): The name of the sheet to open.

Returns:

- result (Type: object): The opened worksheet object or an error message.

Name: del\_sheet(name)

Description: Deletes the specified sheet.

Parameters:

- name (Type: string): The name of the sheet to be deleted.

Returns:

- result (Type: object): Whether the operation was successful.

Name: freeze\_data(dimension, num)

Description: Freeze rows and/or columns on the worksheet

Parameters:

- dimension (Type: string): The dimension to freeze, either 'rows' or 'columns'

- num (Type: integer): Number of rows/cols to freeze.

Returns:

- result (Type: object): Whether the operation was successful.

Name: get\_A1\_annotation(row, col)

Description: Translate the cell position (row,col) into A1 annotation

Parameters:

- row (Type: integer): Row index.

- col (Type: integer): Column index.

Returns:

- result (Type: string): The A1 notation of the cell or an error message.

Name: insert\_cols(values\_list, col\_idx)

Description: Insert columns into sheet at specified column index

Parameters:

- values\_list (Type: array[array[string]]): A list of lists, each list containing one column's values, which can be expressions

- col\_idx (Type: integer): Start column to update. Defaults to 1.

Returns:

- result (Type: object): The updated worksheet data or an error message.

Name: insert\_rows(values\_list, row\_idx)

Description: Insert rows into sheet at specified row index

Parameters:

- values\_list (Type: array[array[string]]): A list of lists, each list containing one row's values, which can be expressions

- row\_idx (Type: integer): Start row to update. Defaults to 1.

Returns:

- result (Type: object): The updated worksheet data or an error message.

Table 34: Prompt details for Sheet (Part 2/4).

---

2646	
2647	
2648	
2649	Name: delete_batch_data(dimension, index_list)
2650	Description: Delete a batch of data in the sheet
2651	Parameters:
2652	- dimension (Type: string): The dimension to delete, either 'row' or 'col'.
2653	- index_list (Type: array[integer]): List of the indexes of rows/cols for deletion.
2654	Returns:
2655	- result (Type: object): The updated worksheet data or an error message.
2656	
2657	Name: update_cell(position, value)
2658	Description: Update the value of the cell
2659	Parameters:
2660	- position (Type: string): A1 notation of the cell position.
2661	- value: The value to set.
2662	Returns:
2663	- result (Type: object): The updated worksheet data or an error message.
2664	
2665	Name: update_cell_by_formula(start_position, end_position, position_list, result_position, operator)
2666	Description: Update the value of the target cell by applying formulas on some specified cells.
2667	Note: Either specify position_list or start_position and end_position.
2668	Parameters:
2669	- start_position (Type: string): The starting position of the range. Default: 'B1'.
2670	- end_position (Type: string): The ending position of the range. Default: 'D2'.
2671	- position_list (Type: array[string]): A list of cell positions in A1 notation.
2672	- result_position (Type: string): The position of the cell where the result of the formula will be stored in. Default: 'G2'.
2673	- operator (Type: string): The operator to be applied on selected cells. Choose one from ['SUM', 'AVERAGE', 'COUNT', 'MAX', 'MIN', 'MINUS', 'PRODUCT'].
2674	Returns:
2675	- result (Type: object): The updated worksheet data or an error message.
2676	
2677	
2678	Name: update_range(start_position, end_position, values_list)
2679	Description: Update a range of the cells from a list
2680	Parameters:
2681	- start_position (Type: string): A1 notation of the start cell.
2682	- end_position (Type: string): A1 notation of the end cell.
2683	- values_list (Type: array[array[Any]]): List of values to be inserted, which can be expressions
2684	Returns:
2685	- result (Type: object): The updated worksheet data or an error message.
2686	
2687	Name: sort_sheet_by_col(col_num, order)
2688	Description: Sorts the current sheet using given sort orders
2689	Parameters:
2690	- col_num (Type: integer): The index of the sort column.
2691	- order (Type: string): The sort order. Possible values are 'asc' or 'des'.
2692	Returns:
2693	- result (Type: object): The updated worksheet data or an error message.
2694	
2695	Name: merge_cells(start_position, end_position)
2696	Description: Merge cells in sheet
2697	Parameters:
2698	- start_position (Type: string): Starting cell position(top left) in A1 annotation.
2699	- end_position (Type: string): Ending cell position(bottom right) in A1 annotation.
2699	Returns:
2699	- result (Type: object): The updated worksheet data or an error message.

---

Table 35: Prompt details for Sheet (Part 3/4).

2700  
2701  
2702  
2703  
2704  
2705  
2706  
2707  
2708  
2709  
2710  
2711  
2712  
2713  
2714  
2715  
2716  
2717  
2718  
2719  
2720  
2721  
2722  
2723  
2724  
2725  
2726  
2727  
2728  
2729  
2730  
2731  
2732  
2733  
2734  
2735  
2736  
2737  
2738  
2739  
2740  
2741  
2742  
2743  
2744  
2745  
2746  
2747  
2748  
2749  
2750  
2751  
2752  
2753

---

Name: update_note(position, content)
Description: Update a note in a certain cell
Parameters:
- position (Type: string): cell position in A1 annotation.
- content (Type: string): The text note to insert.
Returns:
- result (Type: string): The updated note or an error message.
Name: get_all_values()
Description: Display all cell values in current sheet
Returns:
- result (Type: array[array[Any]]): Return all cell values or an error message.
Name: get_range_values(start_position, end_position)
Description: Returns a list of cell data from a specified range.
Parameters:
- start_position (Type: string): Starting cell position in A1 annotation.
- end_position (Type: string): Ending cell position in A1 annotation.
Returns:
- result (Type: array[array[Any]]): List of cell data from the specified range or an error message.
Name: get_cell_value(position)
Description: Get the value of a specific cell
Parameters:
- position (Type: string): Cell position in A1 annotation.
Returns:
- result : Cell value or an error message.
Name: get_value_by_formula(start_position, end_position, position_list, operator)
Description: Calculate a value applying formulas on specified cells. Note: Either specify position_list or start_position and end_position.
Parameters:
- start_position (Type: string): The starting position of the range. Default: 'B1'.
- end_position (Type: string): The ending position of the range. Default: 'D2'.
- position_list (Type: array[string]): A list of cell positions in A1 notation.
- operator (Type: string): The operator to be applied on selected cells. Choose one from ['SUM', 'AVERAGE', 'COUNT', 'MAX', 'MIN', 'MINUS', 'PRODUCT'].
Returns:
- result (Type: string): Calculated result or an error message.
Name: filter_cells(query, in_row, in_column)
Description: Find all cells matching the query, return all cells' position.
Parameters:
- query (Type: ['string', 're.RegexObject']): A string to match or compiled regular expression.
- in_row (Type: ['integer', 'None']): Row number to scope the search. Default is all rows
- in_column (Type: ['integer', 'None']): Column number to scope the search. Default is all columns
Returns:
- result (Type: array[string]): List of cell addresses that match the query or an error message.

---

2754  
2755  
2756  
2757  
2758  
2759  
2760  
2761  
2762  
2763  
2764  
2765  
2766  
2767  
2768  
2769  
2770  
2771  
2772  
2773  
2774  
2775  
2776  
2777  
2778  
2779  
2780  
2781  
2782  
2783  
2784  
2785  
2786  
2787  
2788  
2789  
2790  
2791  
2792  
2793  
2794  
2795  
2796  
2797  
2798  
2799  
2800  
2801  
2802  
2803  
2804  
2805  
2806  
2807

Table 36: Prompt details for Sheet (Part 4/4).

---

Name:	get_note(position)
Description:	Get the note at the certain cell, or return empty string if the cell does not have a note.
Parameters:	- position (Type: string): Cell position in A1 annotation.
Returns:	- result (Type: string): Note content or an error message.
Name:	finish()
Description:	Return an answer and finish the task
Returns:	- result (Type: array[array[Any]]): Return all cell values or an error message.

---

**Instruction:**  
Now new trial starts. You should perform actions to accomplish the goal: Product Update: The table in “Sheet1” contains the product inventory information, and [[‘Product’, ‘Today Sold’], [‘beef’, ‘5’], [‘pork’, ‘2’], [‘chicken’, ‘8’], [‘lamb’, ‘12’], [‘duck’, ‘3’], [‘fish’, ‘23’], [‘shrimp’, ‘21’], [‘salmon’, ‘12’], [‘apple’, ‘100’], [‘banana’, ‘287’], [‘orange’, ‘234’], [‘carrot’, ‘12’]] is today’s sales data. Please update the product information in “Sheet1” in time and then sort by “Quantity” in descending order. Give me one action.

---