Hierarchical Reinforcement Learning with Augmented Step-Level Transitions for LLM Agents

Anonymous ACL submission

Abstract

Large language model (LLM) agents have demonstrated strong capabilities in complex interactive decision-making tasks. However, existing reinforcement learning (RL) approaches for LLM agents typically rely on full interaction histories, resulting in high computational cost and limited scalability. In this paper, we propose **STEP-HRL**, a hierarchical reinforcement learning (HRL) framework that enables step-level learning in LLM agents without relying on full interaction histories. STEP-HRL structures tasks hierarchically, using completed subtasks to represent *global progress* of overall task. By introducing a *local progress* module, it also iteratively and selectively summarizes interaction history within each subtask to produce a compact summary of local progress. Together, these components yield augmented step-level transitions for both high-level and low-level policies, enabling effective step-level policy optimization. Experimental results on ScienceWorld and ALFWorld benchmarks consistently demonstrate that STEP-HRL substantially outperforms baselines in terms of performance and generalization while reducing token usage.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities as autonomous agents in sequential decision-making tasks, exhibiting sophisticated reasoning and planning abilities across diverse interactive environments (Wang et al., 2023; Yao et al., 2022; Li et al., 2022). To further enhance the effectiveness of autonomous agents, reinforcement learning (RL) offers a principled mechanism for enhancing agent decision-making capabilities (Xu et al., 2024; Pang et al., 2024; Peiyuan et al., 2024). Unlike supervised approaches that rely solely on fixed demonstrations (Zeng et al., 2024a; Lin et al., 2023), RL enables agents to refine policy through environmental interaction and reward feedback, thereby discovering more effective strategies that generalize beyond training distributions.

Despite this progress, most RL-based LLM agents adopt a *history-conditioned* formulation, where policies are conditioned on increasingly long sequences of past observations and actions rather than a compact representation of the current decision state. This design choice is largely inherited from sequence-modeling perspectives: LLM agents are built on Transformer architectures (Vaswani et al., 2017), and recent RL formulations cast decision-making as trajectory or sequence prediction (Chen et al., 2021; Janner et al., 2021; Ni et al., 2023). While long histories can help infer latent states in partially observable environments, conflating long-horizon decision-making with long-context conditioning introduces fundamental limitations. Attention-based inference scales quadratically with context length, and unfiltered histories accumulate redundant or irrelevant information that can obscure decision-critical signals and degrade reasoning quality (Zhou et al., 2025; Cherepanov et al., 2023). Importantly, long-context conditioning is a modeling choice rather than a necessity of reinforcement learning.

Existing approaches primarily mitigate the symptoms of this formulation without revisiting its core assumption. Prior work compresses interaction histories (Zhou et al., 2025; Luo et al., 2024) or improves long-term credit assignment (Liu et al., 2025; Zhai et al., 2025; Xiong et al., 2024), but policies remain history-conditioned. Hierarchical reinforcement learning (HRL) introduces temporal abstraction and shows promise for LLM agents (Hu et al., 2025), yet current HRL methods still condition both high-level and low-level policies on accumulated interaction histories, inheriting the same long-context dependence they seek to alleviate.

To address the challenges discussed above, we propose **STEP-HRL** (Augmented **Step**-level **H**ierarchical **R**einforcement **L**earning), which rethinks long-horizon LLM agents from a *progress-*

based perspective. With completed high-level subtasks providing a *global progress* of overall task, STEP-HRL introduces an additional *local progress* module that accumulates subtask-relevant information at each timestep into a compact textual representation with controlled verbosity. The low-level policy conditions exclusively on the current subtask, observation and the distilled local progress, enabling step-level decision making with constant-sized inputs. Meanwhile, the local progress interacts with both the low-level and high-level policies, as well as with its own internal state, facilitating structured information transfer across hierarchical levels. We first perform behavior cloning on expert demonstrations to initialize policies, and then apply step-level offline RL for further optimization.

In summary, our contributions are as follows:

- We propose STEP-HRL, a hierarchical framework that leverages a *local progress* module to enable step-level training for LLM agents, eliminating the need to condition on full interaction histories. To the best of our knowledge, STEP-HRL is the first RL approach that effectively supports step-level training for LLM agents.
- We propose a parameter-efficient two-stage training pipeline, where the high-level, low-level and local progress policies share a unified policy backbone, while being equipped with separate value networks for offline RL. The model is first initialized via behavior cloning and subsequently fine-tuned with step-level offline optimization.
- Extensive experiments on the ScienceWorld and ALFWorld benchmarks demonstrate that our approach significantly improves both performance and generalization, validating the feasibility of step-level RL for LLM agents.

2 Problem Formulation

We formulate the agent operating in an interactive environment as a Partially Observable Markov Decision Process (POMDP), defined by the tuple $\langle \mathcal{C}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R} \rangle$. Here, $\mathcal{C}$ denotes the instruction space specifying task goals, $\mathcal{S}$ is the latent environment state space, $\mathcal{A}$ is the action space, $\mathcal{O}$ is the observation space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ represents the state transition dynamics, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. In the setting of LLM agents, $\mathcal{C}, \mathcal{A}, \mathcal{O}$ are expressed in natural language, while the environment state remains unobserved. Given an instruction $c \in \mathcal{C}$, the agent interacts with the environment with policy π_θ, the policy parameters θ are initialized from a pretrained LLM. The objective is to optimize the policy to maximize the expected discounted return $J(\pi) = \mathbb{E}_\pi \left[\sum_t \gamma^t r_t \right]$.

3 Method

3.1 Step-Level Transitions with Local Progress Modeling

Consider a commonly adopted history-conditioned RL formulation with hierarchical structures. Assume that all previous subtasks have been completed and a new subtask is to be generated. The global interaction history up to time t is denoted as $\mathcal{H}_t = (c, o_0, g_0, a_0, \ldots, a_{t-1}, o_t)$, which concatenates the task instruction with past observations, subtasks, and actions. Conditioned on $\mathcal{H}_t$, the high-level policy generates the next subtask:

$$g_{k+1} \sim \pi_\theta^h(\cdot \mid \mathcal{H}_t). \tag{1}$$

Given the k-th subtask g_k, the low-level policy operates at a finer temporal resolution. We denote the local interaction history as $h_t^k = (o_0^k, a_0^k, \ldots, o_t^k)$, which records observations and actions accumulated during subtask g_k. Conditioned on the g_k and h_t^k, the low-level policy produces primitive actions:

$$a_t^k \sim \pi_\theta^l(\cdot \mid g_k, h_t^k). \tag{2}$$

The local history grows until the subtask terminates, after which the high-level policy is invoked again based on the updated global interaction history.

To enable step-level transitions, a key challenge is compactly representing both local and global interaction histories. Intuitively, the sequence of completed subtasks $G_k = (g_0, g_1, \ldots, g_k)$ already serves as a concise summary of global task progress. Thus, the remaining problem is to compactly summarize the local interaction history within each subtask. To this end, we introduce a *local progress* policy π_θ^p to iteratively achieve this. At the beginning of the g_k, the local progress is initialized as $p_0^k = \varnothing$, reflecting the absence of subtask-local interaction history. The local progress is then updated at each subsequent timestep according to:

$$p_t^k \sim \pi_\theta^p(\cdot \mid g_k, a_{t-1}^k, o_t^k, p_{t-1}^k), \quad t > 0 \tag{3}$$

This design encourages π_θ^p to selectively extract subtask-relevant information from the previous progress p_{t-1}^k and integrate it with the last executed action a_{t-1}^k and its resulting observation o_t^k, yielding an updated local progress p_t^k.

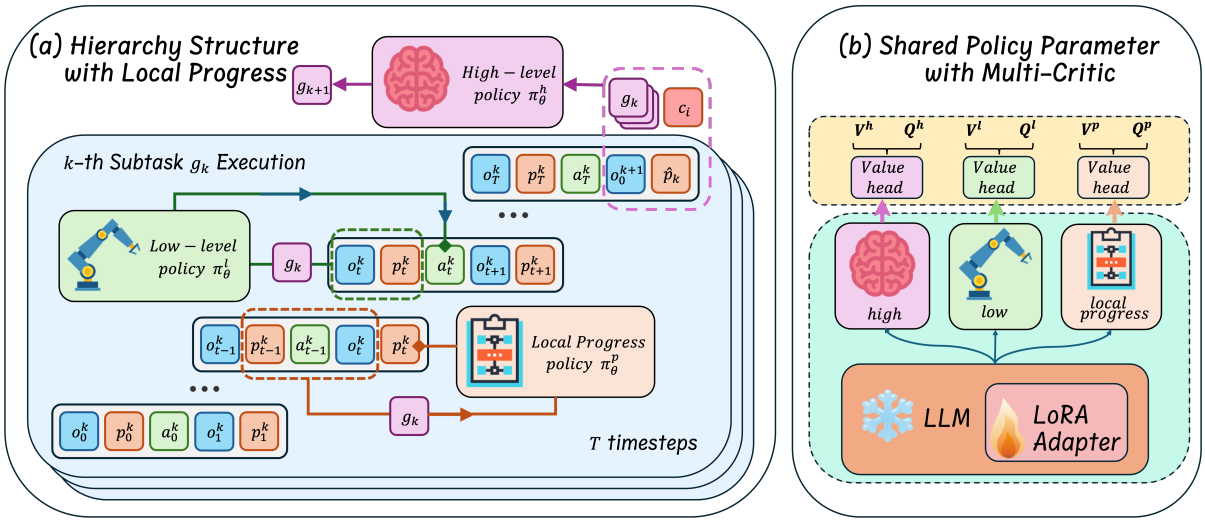

Figure 1: **(a)**: The pipeline of STEP-HRL. Local progress policy is responsible for producing a compact summary of local interaction history within each subtask. Specifically, the local progress policy π_θ^p depends on previous progress p_{t-1}^k, current subtask g_k, executed action a_{t-1}^k and the resulting observation o_t^k to the generate updated local progress p_t^k. The low-level policy π_θ^l combines p_t^k with observation o_t^k and subtask g_k to generate primitive actions. When current subtask g_k terminates, its final local progress $\hat{p}_k$ is forwarded to the high-level policy π_θ^h. Conditioned on the task instruction c_i, completed subtasks G_k, final local progress $\hat{p}_k$ and the initial observation o_0^{k+1} of next subtask, π_θ^h generates the subsequent subtask. **(b)**: The structure of our model. Three different policies share the same parameters, but equipped with different critic network respectively for offline RL training.

With the p_t^k capturing subtask-relevant local interaction information, the low-level policy can make decisions based on the augmented step-level transition $(o_t^k, p_t^k, a_t^k, \hat{r}_t^k, o_{t+1}^k, p_{t+1}^k)$, where $\hat{r}_t^k$ is the intrinsic reward which equals 1 if the current step successfully completes subtask g_k and 0 otherwise. This formulation enables step-level decision making without relying on the full interaction history within each subtask:

$$a_t^k \sim \pi_\theta^l(\cdot \mid g_k, p_t^k, o_t^k). \tag{4}$$

It is worth noting that although p_t^k already encodes information from current observation o_t^k, o_t^k is typically most relevant for current action generation. To prevent π_θ^p from overlooking critical instantaneous information and to strengthen the sensitivity of π_θ^l to the current observation, we still explicitly include o_t^k as an input to the low-level policy.

The local progress p_t^k can also facilitate high-level subtask generation. If we restrict high-level policy π_θ^h to condition solely on the completed subtasks G_k, it does not observe the detailed low-level progress. In this setting, the local progress p_t^k bridges this information gap. For simplicity, we denote the final local progress at the termination of subtask g_k as $\hat{p}_k$. We pass the final progress $\hat{p}_{k-1}$ from the preceding subtask g_{k-1} to the high-level policy. As a result, the step-level transition of high-level can be expressed as $(\hat{p}_{k-1}, o_0^k, g_k, R_k, \hat{p}_k, o_0^{k+1})$, where $R_k = \sum_t r_t^k$ is the accumulated extrinsic environment reward during subtask g_k, and the high-level policy generates the next subtask according to:

$$g_{k+1} \sim \pi_\theta^h(\cdot \mid c, G_k, \hat{p}_k, o_0^{k+1}). \tag{5}$$

We adopt a parameter-efficient design across the three policies, π_θ^h, π_θ^l and π_θ^p share the same parameters. This formulation facilitates efficient knowledge transfer across different decision levels, encourages consistent representations of task semantics and environment dynamics, and reduces the overall training and inference overhead. As a result, the three policies can be jointly optimized while maintaining clear functional specializatio.

3.2 Behavior Cloning

In interactive environments with specialized action and observation spaces, directly training LLM agents with RL often leads to poor sample efficiency. Moreover, since the three policies π_θ^h, π_θ^l and π_θ^p must rapidly internalize their respective roles and output structures, we initialize the agent using expert demonstrations via behavior cloning.

We construct three expert demenstration datasets, $\mathcal{D}^h$, $\mathcal{D}^l$ and $\mathcal{D}^p$ based on the Eqs. (3), (4) and (5). Specifically, We index tasks by $i \in [N]$, subtasks

by $k \in [K_i]$, and within-subtask steps by $t \in [T_{i,k}]$. The datasets are organized as input-target pairs:

$$\mathcal{D}^p = \left\{ \left((g_k, a^k_{t-1}, o^k_t, p^k_{t-1}),\ p^k_t \right) \right\}, \quad (6)$$

$$\mathcal{D}^l = \left\{ \left((g_k, p^k_t, o^k_t),\ a^k_t \right) \right\}, \quad (7)$$

$$\mathcal{D}^h = \left\{ \left((u_i, G_k, \hat{p}_k, o^{k+1}_0),\ g_{k+1} \right) \right\}. \quad (8)$$

For notational convenience, we uniformly denote the policy input by s and the action by u across all three policies. Under this unified notation, behavior cloning optimizes each policy by:

$$\mathcal{L}_{\mathrm{BC}}(\theta) = -\mathbb{E}_{(s,u)\sim\mathcal{D}} \left[\log \pi_\theta(u \mid s) \right], \quad (9)$$

where $\mathcal{D}$ denotes the corresponding expert demonstration dataset for each policy. And the conditional log-likelihood $\log \pi_\theta(u \mid s)$ is computed autoregressively. Let $u = (w^{(1)}, \ldots, w^{(L)})$ denote the tokenization of the target output and $u^{(<\ell)} = (w^{(1)}, \ldots, w^{(\ell-1)})$ denote the preceding tokens. Then:

$$\log \pi_\theta(u \mid s) = \sum_{\ell=1}^{L} \log \pi_\theta\left(u^{(\ell)} \mid s, u^{(<\ell)} \right). \quad (10)$$

This behavior cloning procedure serves as an effective initialization for subsequent RL stages. Empirically, even without further RL, our step-level behavior cloning alone achieves superior performance compared to existing baselines, as demonstrated in Section 4.

3.3 Step-Level Offline RL

To further improve generalization, we collect an additional dataset $\tilde{\mathcal{D}}$ based on the behavior-cloned policies for offline optimization. We then combine the collected data with expert demonstrations to form the offline dataset $\mathcal{D}_{\mathrm{r}} = \mathcal{D} \cup \tilde{\mathcal{D}}$, and optimize the policies on $\mathcal{D}_r$ using an actor-critic framework. We emphasize that the state s corresponds to a *single-step* state defined in Eqs. (6)–(8), instead of the full interaction history, which aligns with our step-level formulation.

Utterance-Level Implicit Value Learning. We implement the critic as an *utterance-level* value estimator based on the hidden state of the last token. Concretely, given the final-token hidden state $H \in \mathbb{R}^{B\times d}$, the critic attaches two lightweight MLP heads that output scalar predictions for the state-value function $V_\psi(s)$ and the action-value function $Q_\phi(s, u)$, respectively.

Following the implicit value learning paradigm introduced in Implicit Q-Learning (IQL) and its language adaptation ILQL (Kostrikov et al., 2021; Snell et al., 2023), we jointly learn the Q_ϕ and the V_ψ using step-level transitions (s, u, r, s') rather than full trajectories. The Q-function is trained by minimizing a TD regression loss bootstrapped from the value function:

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{(s,u,r,s')\sim\mathcal{D}_{\mathrm{r}}} \Big[(r + \gamma V_{\bar{\psi}}(s') - Q_\phi(s, u))^2 \Big], \quad (11)$$

where $V_{\bar{\psi}}$ denotes a softly updated target value network used to stabilize training (Haarnoja et al., 2018). To approximate the constrained Bellman optimality operator without explicitly maximizing over actions, the value function $V_\psi(s)$ is trained using *expectile regression*. Specifically, $V_\psi(s)$ is optimized to regress toward the action-value estimates under an asymmetric squared loss:

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s,u)\sim\mathcal{D}_{\mathrm{r}}} \left[L_2^\tau\big(Q_{\bar{\phi}}(s, u) - V_\psi(s)\big) \right]. \quad (12)$$

where we define $d = Q_{\bar{\phi}}(s, u) - V_\psi(s)$ and $L_2^\tau(d) = |\tau - \mathbf{1}(d < 0)|\, d^2$ is the expectile loss with expectile parameter $\tau \in (0, 1)$ By choosing $\tau > 0.5$, the value function approximates an upper expectile of the maximum. This mechanism implicitly biases learning toward high-value actions, enabling stable offline RL without an explicit policy optimization step.

Implicit Policy Improvement via Advantage-Weighted Regression. The policy is trained to assign higher likelihood to actions with higher estimated value under the learned critic. Concretely, given step-level transitions from the offline dataset $\mathcal{D}_r$, the policy is optimized by regressing toward actions favored by the critic. We employ an advantage-weighted regression objective:

$$\mathcal{L}_A(\theta) = -\mathbb{E}\Big[\exp\Big(\frac{A(s, u)}{\beta}\Big) \log \pi_\theta(u \mid s) \Big], \quad \text{where} \quad A(s, u) = Q_\phi(s, u) - V_\psi(s). \quad (13)$$

The computation of $\log \pi_\theta(u \mid s)$ follows the same autoregressive procedure as in Eq. (10). Following prior offline RL methods (Peng et al., 2019; Kostrikov et al., 2021; Nair et al., 2020), we employ an exponential advantage-weighted objective. The temperature parameter β controls the sharpness of the weighting and balances policy improvement strength and training stability in our setting.

We apply the offline RL procedure to all three policies, π_θ^h, π_θ^l, and π_θ^p, each equipped with a separate critic network while sharing the same policy parameters. This design provides task-specific value supervision at different levels of abstraction, while shared policy parameters facilitate effective knowledge transfer across hierarchical decisions. Consequently, the model captures complementary decision patterns across levels and achieves more consistent and sample-efficient learning across tasks.

4 Experiments

4.1 Experimental Settings

Benchmarks and Datasets. We evaluate our approach on two challenging benchmarks:

- **ScienceWorld** (Wang et al., 2022a) is a text-based interactive benchmark with 30 science task families (e.g., physics, chemistry, biology), each containing many parameterized variants, yielding hundreds to thousands of tasks that require multi-step reasoning and experimentation.
- **ALFWorld** (Shridhar et al., 2020) is a household task benchmark aligned with ALFRED, comprising134 language-conditioned tasks across 6 task types (e.g., pick-and-place, cleaning, heating), focusing on long-horizon action planning.

For dataset construction, we generate hierarchical annotations, including subtasks and local progress signals from expert trajectories using DEEPSEEK. The prompts used for subtask decomposition and progress generation are provided in the Appendix A. For offline RL, we collect additional trajectories using policies initialized via behavior cloning. Following the experimental setup of GLIDER (Hu et al., 2025), we adopt a trajectory mixture ratio of $1:2$, which was identified as the most effective setting in their study.

Models and Baselines. We evaluate STEP-HRL on three outstanding open source models: **Mistral-7B** (Jiang et al., 2023), **Gemma-7B** (Team et al., 2024) and **Llama3-8B** (Meta AI, 2024).

We compare against the following baselines: 1) **ReAct** (Yao et al., 2022), a prompting framework that interleaves reasoning traces and environment actions in a Thought–Action–Observation loop. 2) **Reflexion** (Shinn et al., 2023), which improves subsequent trials by storing self-reflective feedback in an episodic memory. 3) **SwiftSage** (Lin et al., 2023), a dual-process agent that combines a behavior-cloned action model with an LLM-based planner for interactive tasks. 4) **ETO** (Song et al., 2024), which iteratively collects contrastive (failure/success) trajectories and optimizes the policy via DPO (Rafailov et al., 2023). 5) **WKM** (Qiao et al., 2024), which augments planning with a parametric world knowledge model that provides task priors and dynamic state knowledge. 6) **GLIDER** (Hu et al., 2025), an offline HRL framework that decomposes complex tasks and learns high-level and low-level policies for decision making. We also report results of **ChatGPT** (`gpt-3.5-turbo-0125`) and **GPT-4** (`gpt-4-32k-0613`) for comparison by referencing previously published results (Qiao et al., 2024).

Training Details. All fine-tuning baselines and our method are fine-tuned using LoRA (Hu et al., 2022). For behavior cloning, we train the policies for 5 epochs with a learning rate of 1×10^{-4} and a batch size of 128. During the offline RL stage, we train for 3 epochs, using learning rates of 1×10^{-5} and 1×10^{-4} for the actor and critic, respectively, with a batch size of 256. All models are optimized using AdamW (Loshchilov and Hutter, 2017) optimizer. All experiments are conducted on 8 NVIDIA A100 80G GPUs. Detailed hyperparameters and additional experimental settings are provided in Appendix B.

4.2 Results

Main results. Table 4.2 reports the evaluation results of STEP-HRL across three backbone models on the ScienceWorld and ALFWorld benchmarks. Across all settings, STEP-HRL consistently outperforms strong prior baselines on both seen and unseen tasks. On ALFWorld, STEP-HRL achieves near-saturated performance, with success rates exceeding 90% across different backbone models. On ScienceWorld, STEP-HRL also yields consistent and substantial improvements over existing methods, demonstrating its effectiveness in more challenging and diverse environments. Notably, the performance gap between backbones is substantially reduced, indicating strong robustness and scalability of our proposed framework.

Performance across Different Model Scales. Table 4.2 summarizes the performance of STEP-HRL across different model scales. Overall, performance improves steadily as model capacity increases, with larger backbones achieving better performance. Notably, even smaller models such as Llama-1B and Llama-3B demonstrate competi-

Table 1: **Main Results.** Performance comparison across three backbone models on ScienceWorld and AlfWorld benchmarks. indicates prompt-based methods without model parameter update, while represents fine-tuning approaches using LoRA. ↑ denotes the performance improvement of STEP-HRL compared to the best results among the baselines.

Backbone	Method	ScienceWorld		ALFWorld	
		Seen	Unseen	Seen	Unseen
GPT-3.5-Turbo	REACT	8.57	5.97	15.41	13.99
GPT-4	REACT	44.29	38.05	67.32	65.09
Mistral-7B	ReAct	20.72	17.65	7.86	5.22
	Reflexion	21.07	18.11	11.56	6.00
	SwitchSage	48.40	45.25	30.29	26.52
	ETO	58.17	51.85	66.84	71.43
	WKM	62.12	53.62	73.57	76.87
	GLIDER	67.31	65.14	70.02	74.83
	STEP-HRL	**80.28** (↑ 19.27%)	**75.21** (↑ 15.46%)	**96.43** (↑ 31.07%)	**97.01** (↑ 26.20%)
Gemma-7B	ReAct	3.58	3.51	6.43	2.24
	Reflexion	4.94	3.93	7.14	2.99
	SwitchSage	33.43	30.90	8.23	5.72
	ETO	50.44	47.84	66.43	68.66
	WKM	53.68	49.24	70.71	70.40
	GLIDER	63.67	58.50	72.12	70.88
	STEP-HRL	**78.89** (↑ 24.02%)	**74.08** (↑ 26.63%)	**97.86** (↑ 35.69%)	**97.76** (↑ 37.92%)
Llama-3-8B	ReAct	24.76	22.66	2.86	3.73
	Reflexion	27.23	25.41	4.29	4.48
	SwitchSage	42.22	40.58	20.39	10.78
	ETO	57.90	52.33	64.29	64.18
	WKM	60.12	54.75	68.57	65.93
	GLIDER	77.43	68.34	71.56	75.38
	STEP-HRL	**81.57** (↑ 5.35%)	**77.81** (↑ 13.86%)	**97.14** (↑ 35.75%)	**97.76** (↑ 29.69%)

Table 2: Performance of STEP-HRL on ScienceWorld and ALFWorld across model scales.

Model	ScienceWorld		ALFWorld	
	Seen	Unseen	Seen	Unseen
Llama-1B	51.88	49.78	89.86	89.60
Llama-3B	65.31	61.79	94.29	94.00
Llama-8B	81.57	77.81	96.43	97.76

tive performance, particularly on ALFWorld. This trend suggests that STEP-HRL is exceptionally effective even across a wide range of model scales, while additional model capacity further enhances robustness and generalization, especially on more challenging ScienceWorld tasks.

4.3 Ablation Studies

As shown in Figure 2, we examine the effectiveness of key components in STEP-HRL. We consider three variants: 1) `w/o LP`, which removes the local progress policy, forcing the low-level and high-level policies to condition on (g_k, o_t^k) and (c_i, G_k, o_0^{k+1}) respectively; 2) `w/o Hier`, which eliminates the hierarchical structure and directly relies on the local progress module to summarize the global interaction history; 3) `w/o RL`, which omits the offline RL stage and reduces training to behavior cloning only. All variants are trained using the same step-level data for a fair comparison

Across all settings, alternative variants consistently lead to degraded performance. Most notably, the local progress module plays a central role by condensing subtask-relevant interaction information into a compact summary Without this module, many states become indistinguishable, making credit assignment and policy optimization significantly more challenging. The hierarchical structure further contributes by decomposing complex tasks into manageable subtasks, which alleviates the burden on the local progress module and prevents it from being overwhelmed when summarizing long-horizon interactions. Finally, the offline RL stage refines the policies beyond behavior cloning, improving generalization to unseen tasks through value-guided policy updates. Collectively, these results highlight the complementary roles of all components and underscore the importance of the proposed design in STEP-HRL.

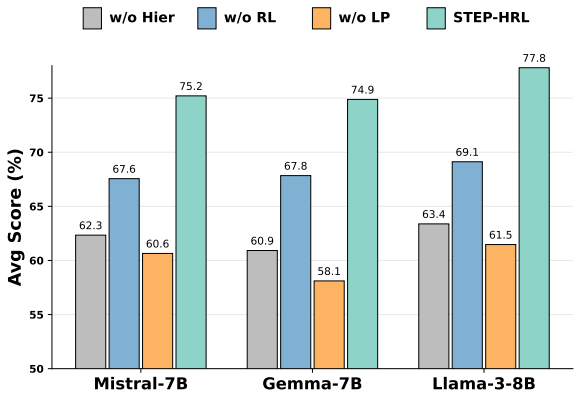

Figure 2: Ablation study of STEP-HRL on unseen ScienceWorld tasks with different backbone models. `w/o LP` denotes removing the local progress policy, `w/o Hier` denotes removing the hierarchical structure, and `w/o RL` denotes removing the offline RL stage, reducing the training procedure to behavior cloning only.

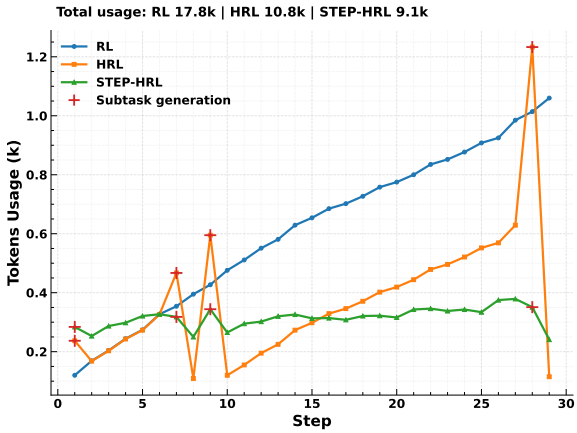

Figure 3: Simulated per-step token usage on ALFWorld `pick_two_obj_and_place` task under identical observation and action sequence across three RL paradigms.

4.4 Analysis on Efficiency

As shown in Figure 3, we analyze the inference-time efficiency of STEP-HRL in comparison with standard RL and HRL on an ALFWorld task. For a fair comparison, all methods are evaluated under the same observation and action sequence. The task is decomposed into four subtasks and requires a total of 29 environment steps to complete.

Standard RL incurs steadily increasing per-step token costs as the interaction progresses, since the policy repeatedly conditions on an ever-growing interaction history. Although HRL reduces the token usage by decomposing the task into subtasks, it exhibits substantial variability, with pronounced spikes at subtask generation steps where long accumulated contexts are processed. Such high variance in input sequence not only increases inference latency, but also leads to inefficient training. In particular, GPU memory allocation must accommodate peak input lengths induced by high-level samples, resulting in underutilization during most steps and reduced overall training efficiency.

In contrast, STEP-HRL maintains an approximately constant per-step token usage. By leveraging compact summaries of both global task progress and local subtask progress, STEP-HRL avoids conditioning on full interaction histories. This design effectively bounds the per-step inference cost, yielding the lowest average token usage with minimal variance. Overall, the result highlights the advantage of STEP-HRL in enabling predictable, efficient and well-balanced inference and training behavior, which is particularly important for long-horizon interactive environments.

4.5 Analysis on Offline RL

Figure 4 presents a sensitivity analysis of the proposed offline RL procedure with respect to key algorithmic and data-related factors. We observe that the advantage temperature β plays a critical role in balancing update aggressiveness and stability: among the tested values, $\beta = 0.95$ consistently yields the highest final performance, while both smaller and larger temperatures lead to inferior convergence. Similarly, varying the expectile parameter τ reveals that a higher expectile (e.g., $\tau = 0.9$) provides more effective value estimation and results in stronger policy improvement compared to lower settings.

We further analyze the impact of data sources used for offline RL. Training on mixed datasets that combine expert demonstrations with BC-collected trajectories consistently outperforms using either expert-only or BC-collected data alone. While BC-collected data contains informative failure trajectories that are valuable for policy improvement, it also introduces lower-quality and noisier samples compared to expert demonstrations. We also study the effect of scaling the amount of training data. Increasing data size improves performance up to a point, with twice the expert data achieving the best overall results. Beyond this regime, additional data yields diminishing returns, as excessively large datasets introduce redundant or low-quality samples that hinder stable learning.

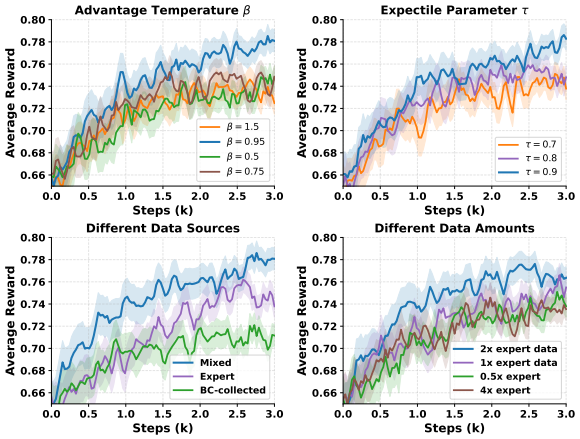

Figure 4: Offline RL sensitivity analysis with respect to advantage temperature β, expectile parameter τ, data sources, and data amounts across training.

5 Related work

LLM Agents. With their strong semantic understanding and emergent reasoning abilities, large language models (LLMs) have been explored as autonomous agents for decision making in complex and interactive environments (Guo et al., 2024; Wang et al., 2024). Early studies primarily adopt prompt-based formulations, where agents generate intermediate reasoning traces to support multi-step decisions, such such as Chain-of-Thought (Wei et al., 2022), ReAct (Yao et al., 2022), Reflexion (Shinn et al., 2023) and their variants (Yao et al., 2023; Wang et al., 2022b). Subsequent work augments LLM agents with additional system components, including tool use (Schick et al., 2023; Qin et al., 2023; Wu et al., 2024; Li et al., 2025), memory mechanisms (Zhang et al., 2025; Sarch et al., 2024; Xu et al., 2025), and multi-agent coordination (Chen et al., 2024a; Bo et al., 2024; Estornell et al., 2024). Beyond architectural augmentation, another line of work focuses on grounding LLM agents through learning from expert demonstration via fine tuning (behavior cloning), demonstrating strong gains (Zeng et al., 2024b; Chen et al., 2023; Yin et al., 2023; Chen et al., 2024b). However, these approaches heavily rely on high-quality expert data and trajectory-level supervision, and their performance degrades in long-horizon decision making and complex interactive tasks due to limited exploration and severe distribution shift..

Reinforcement Learning in LLM Agents. Reinforcement learning (RL) has achieved notable success in aligning and improving large language models (Ouyang et al., 2022; Shao et al., 2024), and has also proven effective for training LLM agents through explicit reward and penalty mechanisms. Most prior work adopts an interaction-driven pipeline, where an LLM agent receives goal-directed feedback from the environment and is fine-tuned using RL algorithms such as PPO (Schulman et al., 2017; Zhai et al., 2024; Szot et al., 2023; Peiyuan et al., 2024). Preference-based methods, such as ETO (Song et al., 2024), further collect contrastive trajectory pairs from environment interactions and update LLM policies via preference optimization objectives like DPO (Rafailov et al., 2023). To achieve fine-grained reinforcement learning signals instead of optimizing the full trajectories, Wen et al. (2024) decompose RL objectives to provide action-level feedback for LLM agents, while GiGPO (Feng et al., 2025) hierarchically estimates step-level advantages to improve training efficiency. Despite their effectiveness, most existing RL-based LLM agents depend on full interaction histories for decision making, where the increasing context length poses significant challenges for credit assignment and computational efficiency.

To mitigate this limitation, several works have explored HRL frameworks such as EPO (Zhao et al., 2024) and GLIDER (Hu et al., 2025) decompose complex tasks into subtasks and learn coordinated high-level and low-level policies. However, even docompsing or fine-grained optimization, these methods still rely on history-conditioned policies, resulting in inefficient credit assignment and high computational overhead.

6 Conclusion

In this paper, we proposed STEP-HRL, a innovative framework that enables efficient step-level learning for LLM agents without relying on full interaction histories. STEP-HRL decomposes tasks into a hierarchical structure and introduces a local progress module to summarize subtask-relevant information, allowing both high-level and low-level policies to operate on compact, step-level state representations. Empirical results on ScienceWorld and ALFWorld demonstrate that STEP-HRL consistently improves performance and generalization. Overall, STEP-HRL provides a practical and scalable approach for training LLM agents. We believe that step-level abstraction with structured progress summaries offers a promising direction for improving both efficiency and robustness in future LLM agent research.

Limitations

Despite the effectiveness of STEP-HRL, it still has several limitations worth noting:

- STEP-HRL highly relies on high-quality expert demonstrations. In particular, the construction of step-level data requires carefully designed subtask and local progress. Designing and curating them can be non-trivial in practice, especially for complex environments with ambiguous subtask boundaries or poorly defined progress signals. This reliance may limit the applicability of STEP-HRL in domains where expert data or structured supervision is scarce.
- In our implementation, subtask termination is predicted jointly with primitive actions, such that each low-level output includes both an action and a termination indicator. This design may result in inaccurate termination decisions, including premature termination or delayed subtask completion. Such errors can degrade the quality of collected transitions and introduce bias into critic value estimation, which in turn may cause misalignment between high-level planning and low-level execution during inference.

References

Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. 2024. Reflective multi-agent collaboration based on large language models. *Advances in Neural Information Processing Systems*, 37:138595–138631.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, and 1 others. 2024a. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *ICLR*.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. 2024b. Agent-flan: Designing data and methods of effective agent tuning for large language models. *arXiv preprint arXiv:2403.12881*.

Egor Cherepanov, Alexey Staroverov, Dmitry Yudin, Alexey K Kovalev, and Aleksandr I Panov. 2023. Recurrent action transformer with memory. *arXiv preprint arXiv:2306.09459*.

Andrew Estornell, Jean-François Ton, Yuanshun Yao, and Yang Liu. 2024. Acc-collab: An actor-critic approach to multi-agent llm collaboration. *arXiv preprint arXiv:2411.00053*.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Zican Hu, Wei Liu, Xiaoye Qu, Xiangyu Yue, Chunlin Chen, Zhi Wang, and Yu Cheng. 2025. Divide and conquer: Grounding llms as efficient decision-making agents via offline hierarchical reinforcement learning. *arXiv preprint arXiv:2505.19761*.

Michael Janner, Qiyang Li, and Sergey Levine. 2021. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286.

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *ArXiv*, abs/2310.06825.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2021. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*.

Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, and 1 others. 2022. Pre-trained language models for interactive decision-making. *Advances in Neural Information Processing Systems*, 35:31199–31212.

Wenjun Li, Dexun Li, Kuicai Dong, Cong Zhang, Hao Zhang, Weiwen Liu, Yasheng Wang, Ruiming Tang,

and Yong Liu. 2025. Adaptive tool use in large language models with meta-cognition trigger. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13346–13370, Vienna, Austria. Association for Computational Linguistics.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. 2023. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. *Advances in Neural Information Processing Systems*, 36:23813–23825.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Junge Zhang, and Jianbin Jiao. 2025. Agentic reinforcement learning with implicit step rewards. *arXiv preprint arXiv:2509.19199*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Fan-Ming Luo, Zuolin Tu, Zefang Huang, and Yang Yu. 2024. Efficient recurrent off-policy rl requires a context-encoder-specific learning rate. *Advances in Neural Information Processing Systems*, 37:48484–48518.

Meta AI. 2024. Introducing meta llama 3: The most capable openly available llm to date. https://ai.meta.com/blog/meta-llama-3/. Accessed: 2024-03.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. 2020. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*.

Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. 2023. When do transformers shine in rl? decoupling memory from credit assignment. *Advances in Neural Information Processing Systems*, 36:50429–50452.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Jing-Cheng Pang, Si-Hang Yang, Kaiyuan Li, Jiaji Zhang, Xiong-Hui Chen, Nan Tang, and Yang Yu. 2024. Kalm: Knowledgeable agents by offline reinforcement learning from large language model rollouts. *Advances in Neural Information Processing Systems*, 37:126620–126652.

Feng Peiyuan, Yichen He, Guanhua Huang, Yuan Lin, Hanchong Zhang, Yuchen Zhang, and Hang Li. 2024. Agile: A novel reinforcement learning framework of llm agents. *Advances in Neural Information Processing Systems*, 37:5244–5284.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. 2019. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*.

Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024. Agent planning with world knowledge model. *Advances in Neural Information Processing Systems*, 37:114843–114871.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Gabriel Sarch, Lawrence Jang, Michael Tarr, William W Cohen, Kenneth Marino, and Katerina Fragkiadaki. 2024. Vlm agents generate their own memories: Distilling experience into embodied programs of thought. *Advances in Neural Information Processing Systems*, 37:75942–75985.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Charlie Victor Snell, Ilya Kostrikov, Yi Su, Sherry Yang, and Sergey Levine. 2023. Offline rl for natural language generation with implicit language q learning. In *The Eleventh International Conference on Learning Representations*.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7584–7600, Bangkok, Thailand. Association for Computational Linguistics.

Andrew Szot, Max Schwarzer, Harsh Agrawal, Bogdan Mazoure, Rin Metcalf, Walter Talbott, Natalie Mackraz, R Devon Hjelm, and Alexander T Toshev. 2023. Large language models as generalizable policies for embodied tasks. In *The Twelfth International Conference on Learning Representations*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, and 1 others. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022a. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Muning Wen, Ziyu Wan, Jun Wang, Weinan Zhang, and Ying Wen. 2024. Reinforcing llm agents via policy optimization with action decomposition. *Advances in Neural Information Processing Systems*, 37:103774–103805.

Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis Ioannidis, Karthik Subbian, Jure Leskovec, and James Y Zou. 2024. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. *Advances in Neural Information Processing Systems*, 37:25981–26010.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. 2024. Watch every step! LLM agent learning via iterative step-level process refinement. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1556–1572, Miami, Florida, USA. Association for Computational Linguistics.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

Zelai Xu, Chao Yu, Fei Fang, Yu Wang, and Yi Wu. 2024. Language agents with reinforcement learning for strategic play in the werewolf game. In *Proceedings of the 41st International Conference on Machine Learning*, pages 55434–55464.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

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

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2023. Lumos: Learning agents with unified data, modular design, and open-source llms. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024a. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3053–3077, Bangkok, Thailand. Association for Computational Linguistics.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024b. Agenttuning: Enabling generalized agent abilities for llms. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3053–3077.

Simon Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Peter Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, and 1 others. 2024. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. *Advances in neural information processing systems*, 37:110935–110971.

Yuanzhao Zhai, Tingkai Yang, Kele Xu, Dawei Feng, Cheng Yang, Bo Ding, and Huaimin Wang. 2025. Enhancing decision-making for llm agents via step-level q-value models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 27161–27169.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47.

Qi Zhao, Haotian Fu, Chen Sun, and George Konidaris. 2024. EPO: Hierarchical LLM agents with environment preference optimization. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6401–6415, Miami, Florida, USA. Association for Computational Linguistics.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. 2025. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*.

A Benchmarks and Datasets

A.1 Benchmarks

We evaluate our approach on two widely used language-based interactive decision-making benchmarks: ScienceWorld and ALFWorld.

- **ScienceWorld** (Wang et al., 2022a) is a text-based environment designed for science experimentation. It consists of 30 tasks spanning 10 categories, where agents are required to demonstrate scientific reasoning through interactive exploration. The environment provides dense rewards at each step, with values ranging from 0 to 1, reflecting incremental task progress.
- **ALFWorld** (Shridhar et al., 2020) simulates household environments that involve navigation and object manipulation. In contrast to ScienceWorld, ALFWorld adopts a sparse reward setting, where an agent receives a reward of 1 only upon successful task completion and 0 otherwise.

Both ScienceWorld and ALFWorld are evaluated under two settings: *Seen* and *Unseen*. The Seen split contains in-distribution tasks that follow the similar task and variations as those observed during training, and is used to evaluate in-distribution performance. In contrast, the Unseen split consists of out-of-distribution task variations with novel mechanism or object, and is used to assess the generalization ability. Dataset statistics for all splits are summarized in Table 3.

Table 3: Dataset statistics.

Dataset	Train	Seen	Unseen
ScienceWorld	1,483	194	211
ALFWorld	3,211	140	134

A.2 Datasets

Expert Dataset. For subtask and local-progress generation, we employ a combination of rule-based heuristics and the DEEPSEEK model. For ScienceWorld, due to the substantial structural diversity across tasks, we adopt task-specific prompts and subtask decomposition strategies tailored to each task category. In contrast, for ALFWorld, we design a unified prompt that guides DEEPSEEK to generate both subtask and local-progress fields in a consistent manner. Figure 8 presents the prompt used for generating local progress.

After generating subtask and local progress fields, we construct the SFT (BC) datasets for all three policies. The data structure of these datasets are shown below, and the prompts used during training and inference are provided in Figure 6.

Training Dataset Structure

High-Level SFT Data:
Input: { high prompt, task description, current observation, completed subtasks, previous local progress }
Target: next subtask

Low-Level SFT Data:
Input: {low prompt, subtask, current observation, local progress}
Target: action

Local-Progress SFT Data:
Input: {local progress prompt, subtask, executed action, resulting observation, previous local progress}
Target: updated local progress

The expert datasets will be released upon paper acceptance.

Offline RL Dataset. We construct the RL dataset by combining expert demonstrations with trajectories collected from behavior-cloned policies.

During data collection, we adopt different sampling temperatures for different policy components to balance exploration and action validity. Specifically, for the high-level policy and the local-progress policy, we set the sampling temperature to **1.0** to encourage diverse subtask sequences and reasoning paths. In contrast, the low-level policy generates primitive actions that must strictly conform to the environment's action format and input constraints. To avoid producing invalid or malformed actions, we therefore set the sampling temperature of the low-level policy to **0**, ensuring deterministic and well-formed action generation.

The final offline RL dataset consists of both expert data and policy-collected trajectories, with an approximate ratio of **1:2**. Among the collected trajectories, around **25%** correspond to unsuccessful episodes. Including such unsuccessful data enables the model to observe negative outcomes and learn to penalize suboptimal actions, which is beneficial for stable offline policy learning.

Local Progress Prompt (DEEPSEEK-ALFWorld)

You are an AI agent responsible for updating *local progress*, a short cumulative summary of what has been achieved within the current subtask.

Inputs:

- current subtask
- previous local progress
- current action
- observation

Global Rules:

1. **Exact token matching:** All object and location names **MUST EXACTLY** match strings in the subtask or observation. Do **NOT** rephrase or normalize names.
2. **No invented facts:** Do **NOT** infer properties, conditions, or failures unless explicitly stated.
3. **Task type:** Subtasks starting with `Locate and pick up` or `Locate and use` are **Locate** tasks; all others are **Non-Locate** tasks.

Locate Tasks
Output: `<progress sentence> || [Checked: loc1, loc2, . . . ]`

1. **Checked update:** Retain all previous locations. Add the current location **ONLY IF** a search action is taken and the target is confirmed **NOT** present.
2. **Termination:** If the target is found or picked up, the search ends and must not continue.
3. **New + Except:** For `new <OBJ> except <OBJ> N`, any different-numbered `<OBJ>` completes the subtask immediately.
4. **Unsuccessful search:** The sentence must include `unchecked` and imply door states (opened / closed), without mentioning specific unchecked locations.
5. **Language constraints:** Do **NOT** mention checked locations, reuse fixed templates, or repeat more than **3 consecutive words** from the previous progress.

Non-Locate Tasks (Place / Clean / Heat / Cool / Use)
Output: `<progress sentence>`

1. Do **NOT** include `Checked` or any tags. Mention the object **ONLY IF** it appears in the current action.
2. **No existence or location statements:** Do **NOT** state or imply object presence, absence, containment, or discovery.
3. **No failure reasoning:** Do **NOT** explain progress via unmet conditions or missing objects.
4. **Assumed availability:** Treat the target object as available by definition of the subtask.
5. **Allowed content only:** Describe only the executed operation or its direct state change.

Figure 5: Full prompt specification for generating local progress on the ALFWorld expert dataset (DEEPSEEK).

Prompt in Training and Inference

High-Level Prompt:
You are a high-level planner. Based on the state (task description, historical subtasks, last subtask progress and current observation), please generate a clear and simple subtask.

Low-Level Prompt:
You are a low-level action executor. Based on the current subtask, observation and local progress, please generate a executable action and determine if the subtask is completed (True/False).

Local-Progress Prompt:
You are an AI agent responsible for updating local progress within a subtask. Based on the current subtask, the previous local progress, the current action, and the resulting observation, update the local progress.

Figure 6: The prompt used in training and inference stages.

B Training Details

Models. We conduct our main experiments using the following instruction-tuned large language models:

- `mistralai/Mistral-7B-Instruct-v0.2`
- `google/gemma-1.1-7b-it`
- `meta-llama/Meta-Llama-3-8B-Instruct`

For scalability experiments across different model sizes, we additionally evaluate:

- `meta-llama/Llama-3.2-1B-Instruct`
- `meta-llama/Llama-3.2-3B-Instruct`

Hyperparameters. The details of all hyperparameters are summarized in Table 4. We adopt LoRA for parameter-efficient fine-tuning across all models. During the behavior cloning (BC) stage, models are trained for 5 epochs using the AdamW optimizer with a learning rate of 1×10^{-4}. Training is performed with a total batch size of 128, achieved via a per-device batch size of 8 and 2 gradient accumulation steps, which balances computational efficiency and training stability.

For the offline reinforcement learning stage, we train the model for 3 epochs, using separate learning rates for the actor (1×10^{-5}) and critic (1×10^{-4}) . The target critic is updated via soft updates with coefficient $\tau_1 = 0.2$, and the discount factor is set to $\gamma = 0.99$. To further stabilize training, we first warm up the critic for 100 steps before jointly optimizing the actor and critic. We employ an advantage-weighted objective with weighting factor $\beta = 0.95$, and adopt expectile regression with parameter $\tau_2 = 0.95$ for value learning.

The training dataset consists of expert demonstrations and BC-collected trajectories mixed at a ratio of 1:2, enabling the model to learn from both high-quality expert and more diverse policy-generated experiences. During trajectory sampling, we use a sampling temperature of 0.7 for the high-level and local-progress policies to encourage diverse reasoning paths, while setting the temperature to 0 for the low-level policy to ensure deterministic and valid primitive action generation. We further constrain text generation by allowing a maximum of 32 tokens for the high-level and low-level policies, and 150 tokens for the local-progress policy.

For evaluation, we impose a maximum of 50 environment steps per episode for both ALFWorld and ScienceWorld tasks, ensuring a consistent evaluation budget across benchmarks.

Table 4: STEP-HRL hyperparameters.

Hyperparameter	**Value**
Optimization	
batch size	128
batch size per device	8
gradient accumulation steps	2
optimizer	AdamW
actor learning rate	1×10^{-5}
critic learning rate	1×10^{-4}
sft learning rate	1×10^{-4}
Training schedule	
sft epochs	5
orl epochs	3
orl warmup steps	100
RL-specific	
discount factor γ	0.99
advantage weighted factor β	0.95
soft update τ_1	0.2
expectile parameter τ_2	0.95
Generation	
sampling temperature	0.7
max new tokens (π_θ^h, π_θ^l)	32
max new tokens (π_θ^p)	150
LoRA	
lora r	16
lora alpha	32
lora dropout	0.05
lora target modules	`q_proj, k_proj, v_proj, o_proj`
Data	
data mixture ratio	1:2
env limit steps	50

C Evaluation Details

We further report model performance on each individual task family in ScienceWorld and ALFWorld. Since the result distributions are similar across different backbone models, we only present results for Llama-3-8B in Tables 5 and 6.

Table 5: Evaluation details on ALFWorld unseen task.

Task ID	Task Name	#Variants	Success Rate (%)	Avg. Steps (Succ.)	Avg. Steps (All)
1	Pick&Place	24	100.0	12.7	12.7
2	Examine in Light	18	100.0	13.3	13.3
3	Clean&Place	31	100.0	10.6	10.6
4	Heat&Place	23	100.0	17.1	17.1
5	Cool&Place	21	100.0	15.7	15.7
6	Pick Two&Place	17	82.4	21.8	26.6
Total	–	**134**	**97.8**	**14.7**	**15.3**

Table 6: Evaluation details on ScienceWorld unseen task.

Task ID	Task Name	#Variants	Avg Score	Avg. Steps (Succ.)	Avg. Steps (All)
0	boil	9	68.9	45.0	48.3
1	change-the-state-of-matter-of	9	62.2	34.0	46.4
2	chemistry-mix	8	67.8	20.8	25.5
3	chemistry-mix-paint-secondary-color	9	88.9	9.5	9.4
4	chemistry-mix-paint-tertiary-color	9	54.4	16.8	13.7
5	find-animal	10	100.0	11.6	11.6
6	find-living-thing	10	100.0	11.6	11.6
7	find-non-living-thing	10	100.0	5.8	5.8
8	find-plant	10	97.5	10.0	14.0
9	freeze	9	55.0	29.0	42.6
10	grow-fruit	10	43.4	–	46.4
11	grow-plant	10	98.8	34.8	35.4
12	identify-life-stages-1	5	77.0	25.0	27.8
13	identify-life-stages-2	4	25.0	5.0	6.2
17	lifespan-longest-lived	10	100.0	4.0	4.0
18	lifespan-longest-lived-then-shortest-lived	10	100.0	5.0	5.0
19	lifespan-shortest-lived	10	100.0	4.0	4.0
20	measure-melting-point-known-substance	10	39.1	–	28.5
22	melt	9	61.9	34.7	39.8
25	power-component	5	100.0	11.8	11.8
26	power-component-renewable-vs-nonrenewable-energy	5	21.2	–	37.8
27	test-conductivity	10	78.3	16.4	19.7
28	test-conductivity-of-unknown-substances	10	70.0	17.1	18.7
29	use-thermometer	10	100.0	17.6	17.6
Total	–	**211**	**77.8**	**15.45**	**22.0**

D Case study

Trajectory Comparison

Normal

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

Observation: This room is called the kitchen. In it, you see the agent, a substance called air, a chair, a counter, a cupboard, a freezer, a fridge, a glass jar, a lighter, an oven, a painting, a sink, soap, a stopwatch, a stove, a table with a glass cup, and a thermometer. You also see: a door to the bathroom (closed), a door to the hallway (closed), and a door to the outside (closed).
Action: open door to bathroom
Reward: 0.08
Observation: The door is now open.
Action: go to bathroom
Reward: 0.17
Observation: This room is called the bathroom. In it, you see a bathtub, a drawing, a glass cup, a purple box, a sink, and a toilet.
Action: focus on cup
Reward: 0.50
Observation: You focus on the glass cup.
Action: move cup to purple box
Reward: 0.25
Observation: You move the glass cup to the purple box.
done.

STEP-HRL

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

Observation: This room is called the kitchen. In it, you see the agent, a substance called air, a chair, a counter, a cupboard, a freezer, a fridge, a glass jar, a lighter, an oven, a painting, a sink, soap, a stopwatch, a stove, a table with a glass cup, and a thermometer. You also see: a door to the bathroom (closed), a door to the hallway (closed), and a door to the outside (closed).
Subtask 1: Navigate to the bathroom and focus on any non-living object.
Action: open door to bathroom
Reward: 0.08
Observation: The door is now open.
Local Progress: Agent is in the kitchen and the door to the bathroom is open, not yet in the bathroom. `[Route: kitchen]`
Action: go to bathroom
Reward: 0.17
Observation: This room is called the bathroom. In it, you see a bathtub, a drawing, a glass cup, a purple box, a sink, and a toilet.
Local Progress: Agent is now in the bathroom, ready to search for a non-living thing. `[Route: kitchen → bathroom]`
Action: focus on cup
Reward: 0.50
Observation: You focus on the glass cup.
Local Progress: Agent has found and focused on a glass cup, a non-living thing, in the bathroom. `[Route: kitchen → bathroom]`

Subtask 2: Move the non-living thing to the purple box in the bathroom.
Action: move cup to purple box
Reward: 0.25
Observation: You move the glass cup to the purple box.
done.

Figure 7: Case study comparing a standard ReAct-style agent and STEP-HRL on a ScienceWorld task. Normal paradigms make decisions based on the accumulated interaction trajectory, which tends to collapse in long-horizon complex tasks. In contrast, STEP-HRL makes decisions based solely on the current observation, augmented with high-level subtask and step-wise local progress, enabling step-level decision making.

STEP-HRL in Long-Horizon Complex Task

Task Description: Your task is to boil water. For compounds without a boiling point, combusting the substance is also acceptable. First, focus on the substance. Then, take actions that will cause it to change its state of matter.

Observation: You are in art studio. You also see: A door to the hallway (that is closed).

Subtask 1: Navigate to kitchen

Action: open door to hallway **Reward:** 0.00 **Observation:** The door is now open. **Local Progress:** Navigating to kitchen and the door to hallway is now open. **Action:** go to hallway **Reward:** 0.00 **Observation:** You move to the hallway. **Local Progress:** Moving from art studio to hallway `[Route: art studio -> hallway]` **Action:** open door to kitchen **Reward:** 0.00 **Observation:** The door is now open. **Local Progress:** The kitchen door is open and ready to go. `[Route: art studio -> hallway]` **Action:** go to kitchen **Reward:** 0.00 **Observation:** You move to the kitchen. **Local Progress:** Arrived in the kitchen. `[Route: art studio -> hallway -> kitchen]`

Subtask 2: Prepare tools for measuring temperature and boiling

Action: pick up thermometer **Reward:** 0.00 **Observation:** You move the thermometer to the inventory. **Local Progress:** Thermometer obtained. **Action:** open cupboard **Reward:** 0.00 **Observation:** The cupboard is now open. **Local Progress:** Thermometer obtained; cupboard opened, revealing a metal pot and other items. **Action:** pick up metal pot **Reward:** 0.00 **Observation:** You move the metal pot to the inventory. **Local Progress:** Metal pot obtained; thermometer already in inventory.

Subtask 3: Fill metal pot with water

Action: move metal pot to sink **Reward:** 0.00 **Observation:** You move the metal pot to the sink. **Local Progress:** The metal pot is now in the sink, ready to be filled. **Action:** activate sink **Reward:** 0.03 **Observation:** The sink is now activated. **Local Progress:** The metal pot is now filled with water in the sink. **Action:** deactivate sink **Reward:** 0.00 **Observation:** The sink is now deactivated. **Local Progress:** The metal pot remains filled with water in the sink, which is now turned off. **Action:** pick up metal pot **Reward:** 0.00 **Observation:** You move the metal pot to the inventory. **Local Progress:** The metal pot containing water is now in the agent's inventory.

Subtask 4: Focus on substance

Action: focus on substance in metal pot **Reward:** 0.67 **Observation:** You focus on the water. **Local Progress:** You focus on the substance called water in the metal pot.

Subtask 5: Heat water with stove in kitchen

Action: move metal pot to stove **Reward:** 0.02 **Observation:** You move the metal pot to the stove. **Local Progress:** Metal pot containing water is now on the stove. **Action:** activate stove **Reward:** 0.00 **Observation:** The stove appears broken, and can't be activated or deactivated. **Local Progress:** The stove is broken and cannot be activated; the metal pot with water remains on it.

Subtask 6: Navigate to foundry to heat water because stove is broken

Action: pick up metal pot **Reward:** 0.00 **Observation:** You move the metal pot to the inventory. **Local Progress:** Agent has picked up the metal pot containing water. **Action:** open door to outside **Reward:** 0.00 **Observation:** The door is now open. **Local Progress:** Agent is still in the kitchen with the pot of water, and the door to outside is open. `[Route: kitchen]` **Action:** go to outside **Reward:** 0.00 **Observation:** You move to the outside. **Local Progress:** Agent is now outside, heading toward the foundry. `[Route: kitchen -> outside]` **Action:** open door

to foundry **Reward:** 0.00 **Observation:** The door is now open. **Local Progress:** Agent has opened the foundry door but has not yet entered. `[Route: kitchen -> outside]` **Action:** go to foundry **Reward:** 0.00 **Observation:** You move to the foundry. **Local Progress:** Agent has entered the foundry. `[Route: kitchen -> outside -> foundry]`

Subtask 7: Heat water with blast furnace
Action: open blast furnace **Reward:** 0.00 **Observation:** The blast furnace is now open. **Local Progress:** The blast furnace is open and ready to be used. **Action:** move metal pot to blast furnace **Reward:** 0.01 **Observation:** You move the metal pot to the blast furnace. **Local Progress:** The metal pot containing water is now inside the open blast furnace. **Action:** activate blast furnace **Reward:** 0.02 **Observation:** The blast furnace is now activated. **Local Progress:** The blast furnace is activated and heating the metal pot containing water.

Subtask 8: Monitor water temperature
Action: examine substance in metal pot **Reward:** 0.02 **Observation:** a substance called water **Local Progress:** Examined water in the pot; temperature not yet measured. **Action:** use thermometer in inventory on substance in metal pot **Reward:** 0.00 **Observation:** the thermometer measures a temperature of 50 degrees celsius **Local Progress:** Water temperature measured at 50°C. **Action:** examine substance in metal pot **Reward:** 0.00 **Observation:** a substance called water **Local Progress:** Water temperature measured at 50°C. **Action:** wait1 **Reward:** 0.23 **Observation:** You decide to wait for 1 iterations.

done.

Figure 8: The trajectory of STEP-HRL on ScienceWorld `boil` task.