TrajEvolve: Trajectory Evolution for Reinforcement Learning with Hindsight Credit Assignment

Yi Wen[1]*, **Hao Chen**[2]*, **Wanyu Wang**[1], **Maolin Wang**[1], **Pengyue Jia**[1], **Derong Xu**[1], **Huize Tan**[3], **Yingyi Zhang**[1], **Wenlin Zhang**[1], **Weihong Luo**[2], **Xiku Du**[2], **Xiangyu Zhao**[1]†
[1]City University of Hong Kong, [2]Tencent, [3]Nanjing University
yiwen23-c@my.cityu.edu.hk, xianzhao@cityu.edu.hk

Abstract

The Reinforcement Learning (RL) paradigm has achieved remarkable success in enhancing the reasoning capabilities of Large Language Models (LLMs). However, when applied to long-horizon tasks, it faces severe challenges: i) Due to the high task complexity, the model struggles to sample enough successful trajectories, resulting in slow convergence. ii) Relying solely on sparse final rewards causes all steps to be updated indiscriminately, encouraging redundant step generation. When both the quantity and quality of successful trajectories become difficult to guarantee, the performance is severely constrained. This dilemma stems from the limitations: existing RL paradigms rely entirely on the reasoning capability of base models, lacking the ability to actively optimize trajectories. To address the above issues, we propose a novel trajectory evolution paradigm with hindsight credit assignment, termed **`TrajEvolve`**. TrajEvolve not only relies on the trajectory inferred by the model but also generates "evolutionary trajectories" by refining low-quality steps in historical trajectories, thereby exposing the policy to high-quality compositions that lie within the base model's support but are rarely sampled within practical training budgets. Furthermore, TrajEvolve reformulates step-level continuous credit assignment as a binary classification task, whether each step is necessary for task completion. This simplification not only reduces the learning difficulty but also enables step labels to be obtained through counterfactual verification, achieving precise identification of redundant steps. Extensive experiments on two datasets demonstrate the superiority of the proposed method[3].

1 Introduction

Large Language Models (LLMs) [Achiam et al., 2023, Yang et al., 2025] have achieved remarkable breakthroughs in language understanding and generation. Among post-training techniques for LLMs, Reinforcement Learning (RL), which iteratively improves models through exploration and reward-driven optimization, has emerged as one of the most influential paradigms. Group Relative Policy Optimization [Shao et al., 2024] and its variants, which generate multiple rollout trajectories and leverage inter-group reward comparisons for policy optimization, have gained significant attention.

Despite great success achieved, RL paradigms face severe challenges when applied to long-horizon decision-making tasks [Feng et al., 2025, Shridhar et al., 2020]. Such tasks typically require the model to interact with external environments over dozens of steps, with reward signals provided only at the final outcome. The impact of any individual action may not manifest until many steps later, substantially increasing the difficulty of reinforcement learning. They face two typical dilemmas

*Both authors contributed equally to this work.
†Corresponding authors.
[3]https://github.com/Applied-Machine-Learning-Lab/NeurIPS26_TrajEvolve

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

in this setting: i) Due to high task complexity and vast exploration space, the model struggles to sample sufficient successful trajectories, resulting in sparse learning signals and slow convergence. ii) Relying solely on sparse final rewards causes all steps to be updated indiscriminately, encouraging the generation of redundant steps in successful trajectories. When neither the quantity nor the quality of trajectories can be guaranteed, the performance of RL paradigms is severely constrained.

These issues fundamentally stem from a critical limitation of existing RL paradigms: *their performance is bounded by the base model's reasoning capability, lacking the ability to actively optimize trajectories*. Recent studies [Yue et al., 2025a] have pointed out that RL merely improves the sampling efficiency of correct reasoning paths already within the base model's repertoire, rather than eliciting fundamentally new reasoning patterns. This raises a natural question:

> 💡 *Can we proactively construct higher-quality trajectories during training to help models to overcome the sampling barrier that on-policy RL inherently faces in long-horizon tasks?*

The key challenge in addressing this problem [Lu et al., 2025] lies in accurately identifying the valuable steps, which is essentially a step-level credit assignment problem. Existing credit assignment methods can be categorized into two types: i) Heuristic-based signals [Wang et al., 2025c], which rely on indicators such as LLM output confidence that have no direct correlation with task outcomes. Such signals may reinforce confident but incorrect outputs. ii) Outcome-based signals [Feng et al., 2025, Ji et al., 2025], which estimate credit through multiple rollouts on intermediate steps, evaluating credit with final rewards. These methods focus solely on which steps are likely to lead to success, but such biased estimations cannot identify steps that are redundant yet do not affect the final outcome. In summary, existing methods struggle to determine the true contribution of individual steps to the task, let alone identify redundant steps in the success trajectory. Moreover, accurate credit assignment remains extremely challenging given the vast solution space of long-horizon tasks.

To address these challenges, we propose **`TrajEvolve`**, a novel trajectory evolution for reinforcement learning with hindsight credit assignment. TrajEvolve not only relies on the trajectory inferred by the model but also generates "evolutionary trajectories" by refining low-quality steps in historical trajectories, thereby exposing the policy to high-quality compositions that lie within the base model's support but are rarely sampled within practical training budgets. Notably, TrajEvolve reformulates continuous credit assignment as a binary classification of whether each step is necessary for task. This simplification compresses the infinite solution space into a finite subspace and can be verified through counterfactual experiments, thereby reducing learning difficulty and ensuring sufficient labeled data for Discriminator training. Specifically, TrajEvolve obtains step-level redundancy labels through hindsight verification and trains a discriminator model to evolve historical trajectories. The evolved trajectories serve as high-quality samples and are jointly utilized with policy-generated trajectories for reinforcement learning optimization. The main contributions of the paper are as follows:

- We propose a trajectory evolution paradigm that not only trains on the model-inferred trajectory but also generates "Evolutionary Trajectories" by refining low-quality steps in historical trajectories, thereby exposing the policy to high-quality compositions that are rarely sampled within training.
- We reformulate continuous credit assignment as a binary classification of whether each step is necessary for task completion. This simplification compresses the infinite solution space into a verifiable subspace, substantially reducing difficulty and achieving effective credit assignment.
- Experiments on ALFWorld and WebShop validate TrajEvolve's effectiveness.

2 Preliminaries

Problem Setup. Following [Feng et al., 2025], we consider a general multi-step decision-making setting where an LLM agent interacts with an environment based on a task description x. At each time step $t = 1, 2, \ldots, T$, the agent observes a state $s_t = (o_t, \mathcal{A}_t)$, comprising an observation o_t and the set of available actions $\mathcal{A}_t$, then selects an action $a_t \in \mathcal{A}_t$ according to its policy $\pi_\theta(a_t \mid o_t, \mathcal{A}_t)$. The environment subsequently transitions to state s_{t+1}. A complete T-step episode forms a trajectory $\tau = \{(s_1, a_1), (s_2, a_2), \ldots, (s_T, a_T)\}$. Notably, only a sparse binary reward $r \in \{0, 1\}$ is provided at episode termination in our setting, indicating task success or failure.

Group-based RL. Group-based reinforcement learning [Shao et al., 2024] bypasses auxiliary critic networks by computing advantages through within-group normalization. Given a task x, the policy generates N candidate trajectories $\{\tau_1, \ldots, \tau_N\}$ with scalar rewards $R(\tau_i)$. Advantages are computed

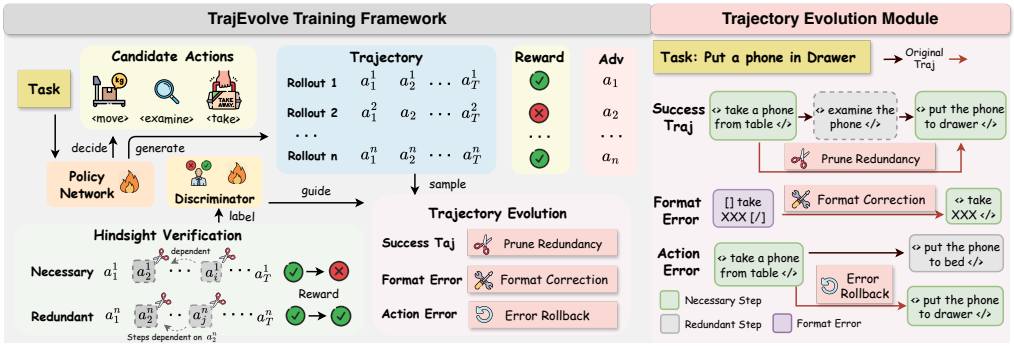


Figure 1: Overview of the TrajEvolve training framework. **Left:** Training pipeline with hindsight verification and trajectory evolution module. **Right:** Three trajectory evolution strategies.

as:$A(\tau_i) = \frac{R(\tau_i)-\mu}{\sigma}$, where μ and σ denote the group mean and standard deviation, respectively. This yields a memory-efficient and scalable framework.

Credit Assignment. Group-based methods assign uniform credit to all steps based solely on terminal rewards, which becomes problematic where steps contribute unequally. Credit assignment addresses this by deriving step-level scores:

$$c_t = g(s_t, a_t, \tau, R(\tau)), \quad t = 1, \ldots, T. \tag{1}$$

where c_t is the credit score for step t, s_t and a_t denote the state and action, τ represents the complete trajectory, and $R(\tau)$ is the terminal reward. However, existing methods attempt to estimate precise credit scores for every step, yet the vast solution space inherent to long-horizon tasks renders exact credit quantification exceedingly difficult.

3 TrajEvolve

In this section, we present **`TrajEvolve`** for step-level credit assignment in long-horizon agent tasks. We begin by reformulating the credit assignment problem from continuous score estimation to binary necessity classification, which substantially reduces the solution space and enables reliable label acquisition (§3.1). We then describe a hindsight verification procedure that obtains ground-truth necessity labels through counterfactual execution in the environment (§3.2). Next, we integrate the obtained labels into policy optimization via redundancy-aware reward shaping and step-level advantage adjustment (§3.3). Finally, we introduce a trajectory evolution mechanism that refines historical trajectories by eliminating redundant steps and correcting errors, enabling the model to learn from improved trajectories beyond its current reasoning capability (§3.4).

3.1 Problem Reformulation: Continuous to Discrete

Existing credit assignment methods either rely on predefined scoring approaches or train implicit process reward models [Cui et al., 2025, Liu et al., 2025a] to estimate step-level contributions. However, these approaches fundamentally attempt to predict precise continuous credit values over an infinite solution space, which becomes intractable for long-horizon tasks. Moreover, since supervision signals are only derived from final results, the feedback is inherently delayed, making it difficult to accurately attribute credit to individual steps.

We observe that for effective policy learning, the critical signal is not the precise credit magnitude, but whether each step contributes positively to task success. This is particularly pronounced in long-horizon tasks, where exploratory interactions with the environment introduce numerous redundant steps that outcome-based methods fail to properly process. This insight motivates us to reformulate credit assignment as a binary *necessary* or *redundant* classification problem. Formally, given a trajectory $\tau = \{(s_t, a_t)\}_{t=1}^{T}$, we define a step-wise discriminator $\mathcal{D}_\phi : \mathcal{S} \times \mathcal{A} \times \mathcal{T} \to \{0, 1\}$:

$$\hat{y}_t = \mathcal{D}_\phi(s_t, a_t, \tau), \tag{2}$$

where $\hat{y}_t = 1$ indicates that step t is predicted as *necessary*, and $\hat{y}_t = 0$ as *redundant*.

We emphasize the superiority of reformulation as follows: i) *it compresses the infinite solution space from continuous values to a finite hypothesis space*, substantially reducing learning complexity and enabling even small models to achieve strong performance. ii) *Unlike continuous credit*

scores whose ground truth is practically unobtainable, binary credit labels can be reliably acquired through hindsight verification, ensuring sufficient supervision for training the discriminator to identify redundant versus necessary steps. We detail the verification procedure in the next subsection.

3.2 Hindsight Verification on Environment-Changing Steps

A key advantage of the reformulation is that credit labels can be reliably acquired through hindsight verification in the environment. In this section, we describe how to obtain ground-truth credit labels.

Action Categorization. We distinguish two types of actions in environment-interactive tasks: *environment-changing actions* that alter the environment state (e.g., *take*, *move*), and *information-gathering actions* that only query the environment without modifying it (e.g., *examine*). Hindsight verification is well-defined only for the former: removing an information-gathering step neither breaks downstream feasibility nor alters task outcomes, so its necessity cannot be determined by environment replay. We therefore apply hindsight verification exclusively to environment-changing steps and assign $y_t = 1$ to all information-gathering steps by default. In ALFWorld and WebShop, environment-changing actions account for the vast majority of steps; the extension to environments with substantial information-gathering steps (e.g., ScienceWorld) is discussed in Appendix C.4.

To determine whether a step is necessary, we must account for its downstream effects.

Definition 3.1 (Dependent Steps)**.** Given a successful trajectory $\tau = \{(s_t, a_t)\}_{t=1}^{T}$ and a step index t, we define the *dependent step set* $\mathcal{D}(t)$ as the set of subsequent steps whose actions become infeasible when step t is removed:

$$\mathcal{D}(t) = \left\{ j : j > t,\ a_j \notin \mathcal{A}(s_j^{-t}) \right\}, \tag{3}$$

where s_j^{-t} denotes the state at step j when step t is skipped, and $\mathcal{A}(s)$ is the set of admissible actions at state s.

Remark 3.2. Accurate computation of the dependent step set $\mathcal{D}(t)$ is critical for subsequent credit labeling. Errors in identifying dependent steps can cause information leakage or incorrect credit labels inheritance via Proposition 3.7, ultimately misleading the entire training process.

Remark 3.3. In environment-interactive scenarios, identifying dependent steps is straightforward. If removing step t causes a subsequent action to become infeasible, that action is dependent on step t. Action feasibility can be directly read from environment feedback—task execution environments such as ALFWorld explicitly return an execution-status signal, making this judgment trivial.

Besides the aforementioned definitions, we also require the following properties to determine steps:

Proposition 3.4 (Necessity)**.** *Step t is **necessary** only if removing step t along with its dependent steps causes task failure in a successful trajectory τ:*

$$\mathcal{V}\left(\tau \setminus (\{t\} \cup \mathcal{D}(t))\right) = 0, \tag{4}$$

where $\mathcal{V}(\cdot) \in \{0, 1\}$ is the task success indicator.

Remark 3.5. The converse of Proposition 3.4 does not hold: task failure upon removal does not imply the step is necessary Consider a trajectory: *go to cabinet* $\rightarrow$ *open cabinet* $\rightarrow$ *go to bathroom* $\rightarrow$ *go to cabinet* $\rightarrow$ *take clothes*. Both *go to bathroom* and the second *go to cabinet* are redundant, yet removing the latter alone still causes failure.

Proposition 3.6 (Sufficiency)**.** *Step t is **redundant** if removing step t along with its dependent steps preserves task success in a successful trajectory τ:*

$$\mathcal{V}\left(\tau \setminus (\{t\} \cup \mathcal{D}(t))\right) = 1. \tag{5}$$

Proposition 3.7 (Transitivity)**.** *The dependent steps of a redundant step are all redundant: if $y_t = 0$, then $y_j = 0$ for all $j \in \mathcal{D}(t)$.*

Remark 3.8. The dependent steps of a necessary step are not necessarily necessary.

Discriminator Training. With the credit labels $\{y_t\}_{t=1}^{T}$ obtained via Algorithm 1, we train the discriminator $\mathcal{D}_\phi$ via supervised fine-tuning on an LLM backbone. Given a prompt $\mathbf{x}_t$ encoding the trajectory context and the target step (s_t, a_t), the model outputs a binary judgment. The training objective minimizes the cross-entropy loss:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(\mathbf{x}_t, y_t) \sim \mathcal{D}_{\text{train}}} \left[\log P_\phi(\hat{y}_t = y_t \mid \mathbf{x}_t) \right], \tag{6}$$

where $\mathcal{D}_{\text{train}}$ denotes the training set constructed from verified trajectories.

Discriminator Pre-training. During early RL training, the policy model struggles to generate sufficient successful trajectories, resulting in sparse credit labels. To address this cold-start problem, we pre-train the discriminator before RL optimization. Specifically, we use Qwen2.5-14B-Instruct to generate approximately 2,000 successful trajectories, which are processed through Algorithm 1 to obtain credit labels.

Algorithm 1: Hindsight Verification for Credit Labeling

Input: Successful Trajectory τ, Environment $\mathcal{E}$
Output: Credit Labels $\{y_t\}_{t=1}^{T}$

Initialize Redundant Set $\mathcal{R} \leftarrow \emptyset$
for $t = 1$ to T **do**
  if $t \in \mathcal{R}$ **then**
    $y_t \leftarrow 0$ ▷ Proposition 3.7
    continue
  end if
  Compute $\mathcal{D}(t) \leftarrow \{j > t : a_j \notin \mathcal{A}(s_j^{-t})\}$
  Execute $\tau \setminus (\{t\} \cup \mathcal{R} \cup \mathcal{D}(t))$ in $\mathcal{E}$
  if task succeeds **then**
    $y_t \leftarrow 0$ ▷ Proposition 3.6
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{t\} \cup \mathcal{D}(t)$ ▷ Proposition 3.7
  else
    $y_t \leftarrow 1$ ▷ Proposition 3.4
  end if
end for

Notably, unlike conventional SFT that requires high-quality trajectories from strong teacher models, our paradigm does not require optimal or concise trajectories. Trajectories with redundant steps provide richer negative samples. This property reduces dependency on powerful LLMs.

3.3 Policy Optimization with Credit Labels

Building upon the credit labels, we integrate step-level credit into the policy optimization.

Redundancy-Aware Reward. Conventional methods often apply uniform penalties based on trajectory length or repeated actions, which may inadvertently discourage longer but correct trajectories and lead to suboptimal performance. To address this limitation, we propose a redundancy-aware reward mechanism. Specifically, for each successful trajectory τ_i, the shaped reward is computed as:

$$\tilde{R}(\tau_i) = R(\tau_i) - \alpha \cdot \rho(\tau_i), \tag{7}$$

where $R(\tau_i)$ is the original task reward, $\rho(\tau_i)$ denotes the redundancy ratio defined as the proportion of redundant steps in the trajectory, and $\alpha \geq 0$ controls the penalty strength.

Instead of penalizing indiscriminately, our formulation applies targeted penalties only to trajectories with genuinely redundant steps, thereby preserving high rewards for efficient executions.

Step-Level Advantage. Given a task x, the policy $\pi_{\theta_{\text{old}}}$ generates a group of N complete trajectories $\{\tau_1, \tau_2, \ldots, \tau_N\}$ under identical initial conditions. Each trajectory τ_i receives the shaped reward $\tilde{R}(\tau_i)$ as described above. To evaluate the relative quality of each trajectory within the group, we compute an episode-level advantage $A^E(\tau_i)$ as follows:

$$A^E(\tau_i) = \frac{\tilde{R}(\tau_i) - \text{mean}(\{\tilde{R}(\tau_j)\}_{j=1}^{N})}{\text{std}(\{\tilde{R}(\tau_j)\}_{j=1}^{N})}. \tag{8}$$

While the episode-level advantage captures trajectory-wide performance, it assigns uniform credit to all steps. To provide finer-grained guidance, we introduce a step-level advantage $A^S(a_t^{(i)})$ derived from the discriminator predictions. The final advantage for each action combines both levels:

$$A(a_t^{(i)}) = A^E(\tau_i) + \lambda \cdot A^S(a_t^{(i)}), \tag{9}$$

where $\lambda > 0$ is a weighting coefficient, and $A^S(a_t^{(i)}) = \mathbb{I}[y_t = 1]$ indicates whether step t is necessary. For failed trajectories, we encourage the correct prefix by applying the bonus to steps preceding the first erroneous step t_{err}, i.e., the first step that is either invalid or predicted as redundant by the discriminator. This hierarchical advantage structure provides both trajectory-wide and step-wise supervision, enabling more effective credit assignment in long-horizon tasks.

Policy Update. We optimize the policy as follows:

$$\mathcal{J}(\theta) = \mathbb{E}\Big[\frac{1}{T}\sum_{t=1}^{T} \min(\rho_\theta(a_t)A(a_t), \text{clip}(\rho_\theta(a_t), 1 \pm \epsilon)A(a_t))\Big], \tag{10}$$

where $\rho_\theta(a_t) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the importance sampling ratio.

3.4 Trajectory Evolution

Existing RL paradigms' performance [Wang et al., 2025d] is inherently bounded by the base model's reasoning capability. In this section, we introduce a trajectory evolution method that actively refines historical trajectories during training.

Evolution Strategies. Let $\mathcal{H} = \{\tau_1, \tau_2, \ldots\}$ denote the historical trajectory sampled during training. Based on the step-level credit labels obtained, we apply different evolution strategies:

- **Redundancy Elimination**: Given a successful trajectory τ with credit labels $\{y_t\}_{t=1}^{T}$, we construct an evolved trajectory τ^* by retaining only the necessary steps:
$$\tau^* = \{(s_t, a_t) \in \tau \mid y_t = 1\}.$$
- **Format Correction**: When a trajectory fails due to an invalid action format at step t_{err}, we map the erroneous action to the nearest valid action in the admissible action space $\mathcal{A}(s_{t_{\text{err}}})$.
- **Error Rollback**: When a trajectory fails due to an incorrect action at step t_{err}, we retain all steps before t_{err} and allow the policy to regenerate, providing opportunities to explore alternative actions.

Training with Evolved Trajectories. The evolved trajectories are re-executed in the environment to obtain updated rewards, and subsequently incorporated into the policy optimization alongside newly sampled trajectories. This augmentation enriches the training distribution with high-quality, non-redundant trajectories that may not be sampled by the current policy, thus exposing the policy to high-quality compositions that are rarely sampled within practical training budgets.

4 Theoretical Analysis

In this section, we provide a theoretical analysis showing that, within a stylized model of group-relative policy optimization, on-policy RL faces a redundancy lower bound in long-horizon tasks, and that trajectory evolution can lift this barrier.

Problem Setup. In this section, we consider the classic GRPO training framework. At each iteration, the policy π_θ generates a group of N trajectories for each task. Each trajectory $\tau = \{(s_t, a_t)\}_{t=1}^{T}$ receives a shaped reward $\tilde{R}(\tau) = R(\tau) - \alpha \cdot \rho(\tau)$, where $\rho(\tau)$ denotes the redundancy ratio. We define a *clean trajectory* as a successful trajectory with zero redundancy, i.e., $R(\tau) = 1, \rho(\tau) = 0$.

Assumption 4.1 (Per-Step Redundancy Model)**.** There exists $\beta \in (0, 1]$ such that, along every rollout, a β-fraction of the states s_t are *redundancy-prone*: under the policy π_{θ_k} at iteration k, the action taken at each such state is redundant independently with probability p_k, i.e.,
$$P_{\pi_{\theta_k}}(r_t = 1 \mid s_t) = p_k, \tag{11}$$
while actions at the remaining states are never redundant, where r_t is the redundancy indicator, $r_t = 1$ if step t is redundant and $r_t = 0$ if necessary. For the initial policy π_{θ_0}, $p_0 > 0$.

Remark 4.2. The condition $p_0 > 0$ is mild since the LLM-based policy outputs a softmax distribution over all admissible actions, where every action retains positive probability, and in long-horizon interactive tasks multiple plausible but unnecessary actions exist at most states; β accounts for the states at which no redundant action is available.

Assumption 4.3 (Gradient Smoothness)**.** Let $m_{\text{clean}}^{(k)}$ denote the number of clean trajectories in the training group at iteration k. There exists a bounded, positive, non-decreasing function $h : \{1, \ldots, N-1\} \to (0, h_{\max}]$, extended by $h(0) = h(N) = 0$, such that, conditional on the group composition, the expected per-iteration reduction of the per-step redundancy probability p_k satisfies
$$E\big[\, p_k - p_{k+1} \mid m_{\text{clean}}^{(k)} = m \,\big] = h(m)\,\eta, \qquad 0 \le m \le N. \tag{12}$$
Remark 4.4. This assumption captures two properties: (i) the total gradient effect is bounded by $h_{\max}\eta$ per iteration, enforced by gradient clipping and learning rates; (ii) the gradient signal for reducing redundancy is governed by the number of clean trajectories, since clean trajectories provide a direct comparison between redundant and non-redundant behavior in the advantage computation.

Theorem 4.5 (Redundancy Lower Bound for On-Policy RL)**.** *Under Assumptions 4.1 and 4.3, for any training iteration $K \le K^*$, the expected redundancy ratio of the trained policy satisfies:*
$$E[\rho(\tau_K)] \ge \frac{\beta p_0}{2}, \tag{13}$$
where $K^* = p_0 / \big(2N h_{\max}\,\eta\,(1 - p_0/2)^{\beta T}\big)$ and T is the length of trajectory τ.

Remark 4.6. As the task horizon T increases, K^* grows exponentially since its denominator $(1 - p_0/2)^{\beta T}$ vanishes, meaning that even reaching the redundancy floor $\beta p_0/2$ requires exponentially more iterations for longer-horizon tasks.

Proposition 4.7. *Suppose at each iteration, trajectory evolution injects $m_{\text{evolve}} \in \{1, \ldots, N-1\}$ clean trajectories into the training group alongside $N - m_{\text{evolve}}$ on-policy rollouts. Under Assumption 4.3, for any target $\varepsilon \in (0, p_0)$, the expected redundancy satisfies*

$$E\big[\rho(\tau_K^{\text{evolve}})\big] \le \beta\varepsilon \quad \text{for all } K \ge \frac{p_0 - \varepsilon}{\delta_\varepsilon}, \qquad \delta_\varepsilon := h(m_{\text{evolve}})\,\eta\left(1 - (1-\varepsilon)^{\beta T(N - m_{\text{evolve}})}\right) > 0. \tag{14}$$

Remark 4.8 (Exponential vs. Polynomial). Within this model, the number of iterations TrajEvolve needs to reach redundancy $\beta\varepsilon$ is at most $(p_0 - \varepsilon)/\delta_\varepsilon$, which is polynomial in $(p_0, 1/\varepsilon, 1/\eta, 1/h(m_{\text{evolve}}))$ and, since δ_ε is non-decreasing in T, bounded independently of the horizon T. In contrast, the barrier K^* of Theorem 4.5 grows exponentially in T.

Due to the space limits, the detailed proof of the above theorem can be found in Appendix A.

5 Experiment

5.1 Experiment Setup

Benchmarks. We evaluate our method on two benchmarks: *ALFWorld* [Shridhar et al., 2020] is a text-based embodied environment where agents receive natural language goals and must accomplish them through multi-turn interaction. It comprises 3,827 task instances across six categories of household activities: Pick & Place (Pick), Examine in Light (Look), Clean & Place (Clean), Heat & Place (Heat), Cool & Place (Cool), and Pick Two & Place (Pick2), requiring compositional reasoning including object identification and state manipulation. *WebShop* [Yao et al., 2022a] simulates realistic online shopping scenarios with 1.18 million real-world products and 12,087 crowd-sourced instructions. Agents must interact with a simulated HTML-based e-commerce website to search, navigate through product pages, and purchase items matching user specifications. This benchmark presents challenges in understanding instructions and strategic exploration across a vast catalog.

Baselines. We compare our approach against three categories of competitive baselines. (1) *Closed-source LLMs*: GPT-4o [Hurst et al., 2024] and Gemini-2.5-Pro [Comanici et al., 2025], which represent frontier models in reasoning capability. (2) *Prompting agents*: ReAct [Yao et al., 2022b] and Reflexion [Shinn et al., 2023], which leverage in-context prompting and self-reflection mechanisms to guide multi-step decision-making. (3) *RL training methods*: PPO [Schulman et al., 2017], which is a classic actor-critic algorithm that relies on a separate value function for step value prediction; RLOO [Ahmadian et al., 2024] and GRPO [Shao et al., 2024] eliminate the critic by computing relative advantages across multiple sampled rollouts; and GiGPO [Feng et al., 2025] introduces a two-level structure combining episode-level advantages with step-level advantages via anchor state grouping for fine-grained credit assignment.

Training details. We use Qwen2.5-1.5B/7B-Instruct as our base models. For fair comparison, all RL training methods share identical hyperparameter configurations, with the rollout group size N set to 8. For the evolution module, we perform trajectory evolution every 5 training epochs. Specifically, for each task, we select 2 trajectories from the rollout samples for subsequent evolution operations, prioritizing trajectories with higher invalid action ratios and longer sequence lengths, as these typically contain more room for improvement. The hyperparameter λ is fixed as 0.5. For the discriminator, we use Qwen2.5-1.5B-Instruct as the backbone and pre-train it for 3 epochs before RL optimization. During RL training, the discriminator is fine-tuned via LoRA and updated alternately with the policy network, training for 1 epoch on credit labels derived from trajectories at each iteration.

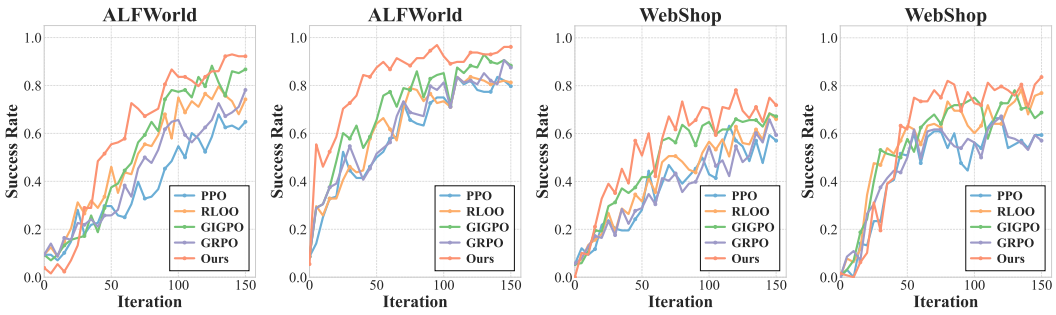
5.2 Overall Evaluation

Table 1 shows the experiment results. We summarize our findings as follows:

- Closed-source LLMs and prompting-based methods exhibit limited performance on multi-step decision-making tasks. Despite their strong general capabilities, GPT-4o and Gemini-2.5-Pro achieve only 45.3% and 57.8% success rates on ALFWorld, respectively. Open-source prompting agents such as ReAct and Reflexion show marginal improvements over vanilla prompting but still underperform significantly, highlighting the inherent difficulty of long-horizon control.
- RL training substantially improves agent performance, with group-based methods demonstrating competitive efficiency. PPO brings notable gains across both benchmarks, achieving 64.8% on

Table 1: Performance Comparison on ALFWorld and Webshop benchmarks. Results are averaged over 3 random seeds. For ALFWorld, we report the success rate (%) for the results. For WebShop, we report both the average score and the success rate (%).

Type	Method	ALFWorld							WebShop	
		Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Closed-Source Model										
Prompting	GPT-4o	81.3	25.0	36.3	38.5	40.7	23.1	45.3	30.8	18.8
Prompting	Gemini-2.5-Pro	83.3	42.9	48.6	54.5	64.7	42.9	57.8	42.8	26.6
Qwen2.5-1.5B-Instruct										
Prompting	Qwen2.5	11.1	0.0	7.1	0.0	0.0	7.7	5.5	22.4	5.5
Prompting	ReAct	17.4	20.5	15.7	6.2	7.7	2.0	13.3	40.1	10.9
Prompting	Reflexion	35.3	22.2	21.7	13.6	19.4	3.7	21.9	55.8	21.9
RL Training	PPO	64.5±2.5	72.7±4.8	73.1±4.7	73.3±5.1	41.2±4.0	64.3±2.7	64.8±3.9	75.9±4.1	57.0±2.3
RL Training	RLOO	90.3±3.9	54.5±6.0	88.5±5.7	86.7±6.8	47.1±5.5	60.7±4.6	74.2±3.1	80.6±5.1	66.7±5.1
RL Training	GRPO	87.1±2.4	81.8±4.6	84.6±4.4	80.0±5.5	70.6±4.7	64.3±3.8	78.1±2.3	76.7±3.2	58.3±4.3
RL Training	GiGPO	94.8±2.5	**95.0±4.4**	85.8±5.1	86.7±6.2	82.4±5.1	82.6±4.4	86.7±2.3	80.9±2.1	67.2±3.9
RL Training	Ours	**95.2±2.2**	87.5±3.2	**95.0±3.3**	**92.9±5.6**	**91.2±3.7**	**91.6±2.9**	**92.2±1.6**	**83.7±1.4**	**71.9±1.6**
Qwen2.5-7B-Instruct										
Prompting	Qwen2.5	31.0	30.3	4.3	10.1	4.4	2.8	14.1	26.0	7.8
Prompting	ReAct	48.5	35.4	34.3	13.2	18.2	17.6	31.2	46.2	19.5
Prompting	Reflexion	62.0	41.6	44.9	30.9	36.3	23.8	43.0	58.1	28.9
RL Training	PPO	90.3±3.4	90.9±3.9	84.6±6.9	73.3±5.0	64.7±5.2	71.4±4.2	79.7±3.9	81.9±2.0	67.4±2.3
RL Training	RLOO	90.3±3.8	90.9±4.5	88.5±6.0	86.7±6.2	70.6±4.3	64.3±3.8	81.5±2.7	83.7±2.5	76.0±3.9
RL Training	GRPO	96.8±2.7	96.4±3.4	84.6±7.5	86.7±6.8	82.4±5.2	78.6±4.1	87.5±3.1	80.8±3.9	65.9±4.6
RL Training	GiGPO	96.8±2.8	96.7±3.0	92.3±6.4	86.7±5.9	88.2±5.1	71.4±3.6	88.3±2.3	82.8±2.5	68.8±3.1
RL Training	Ours	**98.5±1.4**	**98.3±2.9**	**96.1±4.5**	**92.9±4.7**	**89.5±4.4**	**97.8±3.7**	**95.3±1.6**	**90.7±2.2**	**82.0±2.3**



(a) Qwen2.5-1.5B-Instruct (b) Qwen2.5-7B-Instruct (c) Qwen2.5-1.5B-Instruct (d) Qwen2.5-7B-Instruct

Figure 2: Validation success rate during training on ALFWorld and WebShop dataset.

ALFWorld and 57.0% on WebShop with the 1.5B model. GiGPO, with its hierarchical advantage estimation, consistently outperforms other RL baselines, validating the importance of fine-grained credit assignment in multi-step tasks.

- Our method achieves the best overall performance across both model scales and benchmarks. On the 1.5B model, our approach attains 92.2% on ALFWorld and 71.9% on WebShop, surpassing GiGPO by 5.5% and 4.7%, respectively. These results demonstrate that our trajectory evolution mechanism provides complementary benefits to fine-grained credit assignment, enabling more effective policy optimization for LLM agents.

5.3 Training Analysis

Figure 2 presents the validation success rate curves during training, which can show how the training dynamics reflect the effectiveness of our evolution paradigm. We summarize our findings as follows:

- Faster convergence stems from alleviating the sparse signal dilemma. In the early training stages, our method exhibits notably steeper learning curves compared to baselines. This acceleration directly addresses a fundamental limitation of standard RL: the base model struggles to sample sufficient successful trajectories in complex environments, resulting in sparse learning signals. By generating evolutionary trajectories through hindsight refinement, our approach provides additional high-quality training samples that are not bounded by the model's current reasoning capability, thereby enriching the gradient signal and enabling more efficient policy updates.

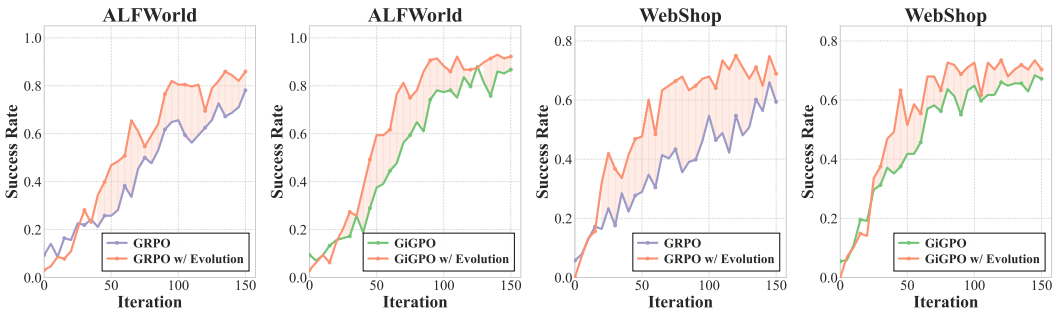


Figure 3: Effect of integrating the trajectory evolution module with different RL baselines. The shaded area indicates the performance gain from adding our evolution module.

- Sustained performance gains validate the trajectory evolution paradigm. Throughout training, our method consistently maintains a performance advantage over GiGPO and other baselines. Compared to existing RL methods can only improve sampling efficiency of reasoning paths already within the base model's repertoire, our trajectory evolution paradigm actively constructs improved trajectories by eliminating redundant steps. This enables the model to learn from trajectories that are rarely sampled within practical training budgets.
- The advantage is more pronounced on WebShop due to higher task complexity. WebShop involves a much larger action space and noisier, lengthier observations, where standard RL methods suffer from indiscriminate updates that encourage redundant steps. Our binary classification approach—determining whether each step is necessary rather than estimating continuous credit scores—compresses the infinite solution space into a verifiable subspace, making credit assignment tractable even in such challenging scenarios. This explains why the performance gap between our method and baselines is particularly evident on WebShop.

5.4 Compatibility of the Evolution Module

Figure 3 demonstrates that our trajectory evolution module serves as a general enhancement applicable to different RL baselines. On both WebShop and ALFWorld, integrating the evolution module yields consistent improvements for both GRPO and GiGPO. Notably, the gain is more substantial when combined with GRPO, which lacks fine-grained credit assignment, and the evolved trajectories compensate for this limitation by directly providing refined training samples. For GiGPO, which already performs step-level credit assignment, the evolution module still brings complementary benefits, indicating that actively constructing higher-quality trajectories offers advantages beyond what credit assignment alone can achieve. The consistent patterns across both benchmarks validate the generalizability of our approach.

5.5 Trajectory Redundancy Analysis

Existing RL methods typically rely solely on sparse final rewards, which update all steps indiscriminately and may inadvertently encourage redundant actions. To investigate whether our method produces more efficient behaviors, we conduct trajectory length analysis in Table 2. Since average trajectory length is confounded by success rate, as failed trajectories may be either prolonged by repeated invalid attempts or cut short by premature termination, we additionally report "Succ. Steps", measuring the mean length of successful trajectories only, to enable fair comparison of execution efficiency. As shown in Table 2, our method achieves the shortest trajectories on both metrics across all benchmarks. This demonstrates that our binary credit assignment effectively identifies and eliminates redundant steps, enabling the model to complete tasks with more concise action sequences.

Table 2: Redundancy analysis on ALFWorld and WebShop. "Avg. Steps" denotes mean steps across all trajectories. "Succ. Steps" measures mean steps for successful trajectories only. Lower values indicate more efficient execution.

Model	ALFWorld		WebShop	
	Avg. Steps	Succ. Steps	Avg. Steps	Succ. Steps
PPO	17.45	10.63	8.94	8.61
RLOO	15.20	10.06	7.98	7.11
GRPO	15.36	11.26	10.97	12.61
GiGPO	11.58	8.76	6.16	5.33
Ours	**9.63**	**7.91**	**5.89**	**5.15**

5.6 Ablation Study

To investigate the individual contribution of each component in our framework, we conduct ablation experiments by systematically removing the step-level credit assignment (w/o Step Credit) and the trajectory evolution module (w/o Evolution). As shown in Figure 4, both components are essential for achieving optimal performance, with their combination yielding the best results across both benchmarks. Notably, removing the evolution module causes a larger performance drop compared to removing step credit, indicating that actively constructing higher-quality trajectories provides more substantial benefits than refining credit assignment alone. Furthermore, the gains from both components are largely additive, confirming that trajectory-level optimization and step-level credit assignment address complementary aspects of the challenge.

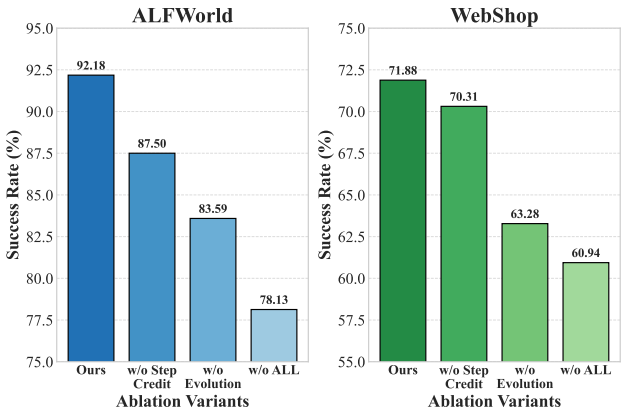


Figure 4: Ablation study on ALFWorld and Webshop benchmarks. Qwen2.5-1.5B-Instruct serves the base models.

Due to the space limits, more experiments about complex scenarios generalization, discriminator accuracy, off-policy analysis, and other ablation studies can be referred in Appendix C and D .

6 Related Work

Reinforcement Learning for LLMs. Reinforcement learning (RL) [Rafailov et al., 2023, Song et al., 2024] has significantly advanced the reasoning capabilities of LLMs. As a representative RL method, PPO [Schulman et al., 2017] achieves desirable performance by enabling dynamic policy updates via environment feedback, yet requires additional critic models for advantage estimation, increasing computational overhead and training complexity. To address this, group-based methods including GRPO [Shao et al., 2024], Dr.GRPO [Liu et al., 2025b], DAPO [Yu et al., 2025], and GSPO [Zheng et al., 2025] estimate advantages from response batches sampled under the same prompt, eliminating critic dependency while maintaining efficiency. Recent work [Yue et al., 2025a] argues that RL primarily improves the sampling efficiency of reasoning paths already within the base model's repertoire, rather than eliciting fundamentally new ones. We take this observation as a starting point rather than a challenge: if the bottleneck of long-horizon RL is sampling efficiency over composed trajectories, then explicitly constructing high-quality compositions from existing rollouts is a principled response. **`TrajEvolve`** operationalizes this view: every action in an evolved trajectory is drawn from the base policy's own outputs, but the composition is one that on-policy sampling would require exponentially many rollouts to discover (Theorem 4.5).

Credit Assignment in RL. Despite the success of group-based RL methods [Ahmadian et al., 2024], they rely solely on outcome rewards without evaluating intermediate steps, motivating credit assignment research. Existing methods fall into two categories: i) Heuristic signals [Wang et al., 2025c, Zhang et al., 2025b, Wang et al., 2025a, Harutyunyan et al., 2019] use LLM confidence or meta-reasoning formats, risking reinforcement of confident but incorrect outputs; ii) Outcome-derived signals [Kazemnejad et al., Ji et al., 2025, Guo et al., 2025b, Feng et al., 2025] estimate step values via Monte Carlo rollouts but only identify success-leading steps, failing to detect redundant ones.

7 Conclusion

We presented **`TrajEvolve`**, a step-level credit assignment framework for long-horizon agent tasks. By reformulating credit assignment as binary necessity classification, **`TrajEvolve`** enables reliable label acquisition through hindsight verification and trains a discriminator to guide both policy optimization and trajectory evolution. Experiments on ALFWorld and WebShop demonstrate that **`TrajEvolve`** consistently outperforms existing RL methods. Future work includes extending our approach to more complex environments and exploring the integration of step-level credit with traditional search question-answer scenarios.

Acknowledgments and Disclosure of Funding

This research was partially supported by National Natural Science Foundation of China (No.62502404), Hong Kong Research Grants Council (Research Impact Fund No.R1015-23, Collaborative Research Fund No.C1043-24GF, RGC Research Fellow Scheme No.RFS2627-1S03, General Research Fund No.11218325, No.11212926), Institute of Digital Medicine of City University of Hong Kong (No.9229503), and Tencent (CCF-Tencent Open Fund, Tencent Rhino-Bird Focused Research Program, Tencent University Cooperation Project).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267, 2024.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Brian Bartoldson, Siddarth Venkatraman, James Diffenderfer, Moksh Jain, Tal Ben-Nun, Seanie Lee, Minsu Kim, Johan Obando-Ceron, Yoshua Bengio, and Bhavya Kailkhura. Trajectory balance with asynchrony: Decoupling exploration and learning for fast, scalable llm post-training. *arXiv preprint arXiv:2503.18929*, 2025.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

Jie Cheng, Ruixi Qiao, Lijun Li, Chao Guo, Junle Wang, Gang Xiong, Yisheng Lv, and Fei-Yue Wang. Stop summation: Min-form credit assignment is all process reward model needs for reasoning. *arXiv preprint arXiv:2504.15275*, 2025.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

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

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025a.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in rl for large language models. *arXiv preprint arXiv:2505.23564*, 2025b.

Bingguang Hao, ZengZhuang Xu, Maolin Wang, Yuntao Wen, Yicheng Chen, Cunyin Peng, Long Chen, Dong Wang, Xiangyu Zhao, Jinjie Gu, et al. Funreason: Enhancing large language models' function calling via self-refinement multiscale loss and automated data refinement. *arXiv preprint arXiv:2505.20192*, 2025a.

Bingguang Hao, Zengzhuang Xu, Maolin Wang, Yuntao Wen, Yicheng Chen, Cunyin Peng, Long Chen, Dong Wang, Xiangyu Zhao, Jinjie Gu, et al. Reasoning through exploration: A reinforcement learning framework for robust function calling. *arXiv preprint arXiv:2508.05118*, 2025b.

Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado P van Hasselt, Gregory Wayne, Satinder Singh, Doina Precup, et al. Hindsight credit assignment. *Advances in neural information processing systems*, 32, 2019.

Hanjiang Hu, Changliu Liu, Na Li, and Yebin Wang. Training task reasoning llm agents for multi-turn task planning via single-turn reinforcement learning. *arXiv preprint arXiv:2509.20616*, 2025.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. *arXiv preprint arXiv:2509.21240*, 2025.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. Vineppo: Refining credit assignment in rl training of llms. In *Forty-second International Conference on Machine Learning*.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv preprint arXiv:2310.10505*, 2023.

Zihan Lin, Xiaohan Wang, Jie Cao, Jiajun Chai, Guojun Yin, Wei Lin, and Ran He. Rest: Reshaping token-level policy gradients for tool-use large language models. *arXiv preprint arXiv:2509.21826*, 2025.

Qirui Liu, Hao Chen, Yi Wen, Weijie Shi, Yao Zhao, Xianhe Chen, Hao Zhao, Xing Tang, Weihong Luo, and Yuyu Luo. Grounded credit assignment in reinforcement learning for multi-turn search agents, 2026. URL https://arxiv.org/abs/2610.23148.

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

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b.

Fanbin Lu, Zhisheng Zhong, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Arpo: End-to-end policy optimization for gui agents with experience replay. *arXiv preprint arXiv:2505.16282*, 2025.

Youssef Mroueh, Nicolas Dupuis, Brian Belgodere, Apoorva Nitsure, Mattia Rigotti, Kristjan Greenewald, Jiri Navratil, Jerret Ross, and Jesus Rios. Revisiting group relative policy optimization: Insights into on-policy and off-policy training. *arXiv preprint arXiv:2505.22257*, 2025.

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

Yun Qu, Yuhang Jiang, Boyuan Wang, Yixiu Mao, Cheems Wang, Chang Liu, and Xiangyang Ji. Latent reward: Llm-empowered credit assignment in episodic reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 20095–20103, 2025.

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

Nicolas Le Roux, Marc G Bellemare, Jonathan Lebensold, Arnaud Bergeron, Joshua Greaves, Alex Fréchette, Carolyne Pelletier, Eric Thibodeau-Laufer, Sándor Toth, and Sam Work. Tapered off-policy reinforce: Stable and efficient reinforcement learning for llms. *arXiv preprint arXiv:2503.14286*, 2025.

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

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

Leyang Shen, Yang Zhang, Chun Kai Ling, Xiaoyan Zhao, and Tat-Seng Chua. Carl: Critical action focused reinforcement learning for multi-step agent. *arXiv preprint arXiv:2512.04949*, 2025.

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

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

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization for llm agents. *arXiv preprint arXiv:2403.02502*, 2024.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon llm agents. *arXiv preprint arXiv:2603.08754*, 2026.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. Information gain-based policy optimization: A simple and effective approach for multi-turn llm agents. *arXiv preprint arXiv:2510.14967*, 2025a.

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. Spa-rl: Reinforcing llm agents via stepwise progress attribution. *arXiv preprint arXiv:2505.20732*, 2025b.

Jiawei Wang, Jiacai Liu, Yuqian Fu, Yingru Li, Xintao Wang, Yuan Lin, Yu Yue, Lin Zhang, Yang Wang, and Ke Wang. Harnessing uncertainty: Entropy-modulated policy gradients for long-horizon llm agents. *arXiv preprint arXiv:2509.09265*, 2025c.

Maolin Wang, Yingyi Zhang, Bowen Yu, Bingguang Hao, Cunyin Peng, Yicheng Chen, Wei Zhou, Jinjie Gu, Chenyi Zhuang, Ruocheng Guo, et al. Function calling in large language models: Industrial practices, challenges, and future directions. *ACM Computing Surveys*, 58(9):1–37, 2026.

Peisong Wang, Ruotian Ma, Bang Zhang, Xingyu Chen, Zhiwei He, Kang Luo, Qingsong Lv, Qingxuan Jiang, Zheng Xie, Shanyi Wang, et al. Rlver: Reinforcement learning with verifiable emotion rewards for empathetic agents. *arXiv preprint arXiv:2507.03112*, 2025d.

Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. *arXiv preprint arXiv:2505.15107*, 2025e.

Feijie Wu, Weiwu Zhu, Yuxiang Zhang, Soumya Chatterjee, Jiarong Zhu, Fan Mo, Rodin Luo, and Jing Gao. Portool: Tool-use llm training with rewarded tree. *arXiv preprint arXiv:2510.26020*, 2025.

Yuxuan Wu, Miaopei Lin, Qiufang Ying, Lin Dan, Hao Zhao, Qirui Liu, Yuteng Shen, Shaohui Wu, Weihong Luo, Xiku Du, Xing Tang, and Hao Chen. Finhopper: A progressive multi-hop benchmark for complex financial agents with step-wise evaluation, 2026. URL https://arxiv.org/abs/2610.26036.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, et al. Agentgym: Evaluating and training large language model-based agents across diverse environments. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27914–27961, 2025.

Derong Xu, Xinhang Li, Ziheng Zhang, Zhenxi Lin, Zhihong Zhu, Zhi Zheng, Xian Wu, Xiangyu Zhao, Tong Xu, and Enhong Chen. Harnessing large language models for knowledge graph question answering via adaptive multi-aspect retrieval-augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25570–25578, 2025.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*, 2025.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

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

Junxi Yin, Haisen Luo, Zhenyu Li, Yihua Liu, Dan Liu, Zequn Li, and Xiaohang Xu. Pinpointing crucial steps: Attribution-based credit assignment for verifiable reinforcement learning. *arXiv preprint arXiv:2510.08899*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Jiazhen Yuan, Wei Zhao, and Zhengbiao Bai. Graph-enhanced policy optimization in llm agent training. *arXiv preprint arXiv:2510.26270*, 2025.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. *arXiv preprint arXiv:2412.01981*, 2024a.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024b.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025a.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025b.

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F Wong, and Yu Cheng. Exgrpo: Learning to reason from experience. *arXiv preprint arXiv:2510.02245*, 2025.

Sheng Zhang, Junyi Li, Wenlin Zhang, Xiaowei Qian, Yichao Wang, Yingyi Zhang, Maolin Wang, Yong Liu, and Xiangyu Zhao. R^2-searcher: Calibrating retrieval and reasoning boundaries for agentic search, 2026. URL https://arxiv.org/abs/2606.28566.

Weinan Zhang, Xiangyu Zhao, Li Zhao, Dawei Yin, Grace Hui Yang, and Alex Beutel. Deep reinforcement learning for information retrieval: Fundamentals and advances. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2468–2471, 2020.

Yaocheng Zhang, Haohuan Huang, Zijun Song, Yuanheng Zhu, Qichao Zhang, Zijie Zhao, and Dongbin Zhao. Criticsearch: Fine-grained credit assignment for search agents via a retrospective critic. *arXiv preprint arXiv:2511.12159*, 2025a.

Zijing Zhang, Ziyang Chen, Mingxiao Li, Zhaopeng Tu, and Xiaolong Li. Rlvmr: Reinforcement learning with verifiable meta-reasoning rewards for robust long-horizon agents. *arXiv preprint arXiv:2507.22844*, 2025b.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn rl. *arXiv preprint arXiv:2402.19446*, 2024.

Technical Appendix

Table of Contents

A Proof of Theorem 4.5 **17**

B Experiment Details **18**

B.1 Pseudo Code 18

B.2 Implementation Details 18

C Additional Experiments **19**

C.1 Base Model Ablation 19

C.2 Discriminator Accuracy Experiment 21

C.3 Computation Cost Analysis 21

C.4 TrajEvolve on Complex Scenarios 22

D Method Analysis **23**

D.1 Method Transferability Analysis 23

D.2 Off-Policy Analysis 25

E Case Study **25**

F Extended Related Work **26**

G Prompts **27**

A Proof of Theorem 4.5

Assumption 4.1 (Per-Step Redundancy Model)**.** There exists $\beta \in (0, 1]$ such that, along every rollout, a β-fraction of the states s_t are *redundancy-prone*: under the policy π_{θ_k} at iteration k, the action taken at each such state is redundant independently with probability p_k, i.e.,

$$P_{\pi_{\theta_k}}(r_t = 1 \mid s_t) = p_k, \tag{11}$$

while actions at the remaining states are never redundant, where r_t is the redundancy indicator, $r_t = 1$ if step t is redundant and $r_t = 0$ if necessary. For the initial policy π_{θ_0}, $p_0 > 0$.

Assumption 4.3 (Gradient Smoothness)**.** Let $m^{(k)}_{\text{clean}}$ denote the number of clean trajectories in the training group at iteration k. There exists a bounded, positive, non-decreasing function $h : \{1, \dots, N-1\} \to (0, h_{\max}]$, extended by $h(0) = h(N) = 0$, such that, conditional on the group composition, the expected per-iteration reduction of the per-step redundancy probability p_k satisfies

$$E\big[\, p_k - p_{k+1} \mid m^{(k)}_{\text{clean}} = m \,\big] = h(m)\,\eta, \qquad 0 \le m \le N. \tag{12}$$

Lemma A.1 (Sampling Barrier)**.** *For any policy π_θ satisfying $P_{\pi_\theta}(r_t = 1 \mid s_t) \ge p$ at a β-fraction of the states along every rollout, the probability that a trajectory of length T is clean satisfies:*

$$P_{\pi_\theta}(\tau_{\textit{clean}}) \le (1-p)^{\beta T}. \tag{15}$$

In a group of N independent rollouts, the number of clean trajectories m_{clean} satisfies $E[m_{\text{clean}}] \le N(1-p)^{\beta T}$, $P(m_{\text{clean}} \ge 1) \le N(1-p)^{\beta T}$, and $P(m_{\text{clean}} = N) \le (1-p)^{\beta T N}$.

Proof. A clean trajectory requires, in particular, each of the (at least) βT redundancy-prone steps to be non-redundant. Conditional on the past, each such step is non-redundant with probability at most $1 - p$, so by the chain rule $P(\text{clean}) \le (1-p)^{\beta T}$. Linearity of expectation gives the bound on $E[m_{\text{clean}}]$, Markov's inequality gives $P(m_{\text{clean}} \ge 1) \le E[m_{\text{clean}}]$, and independence across rollouts gives the bound on $P(m_{\text{clean}} = N)$. □

Lemma A.2 (Self-Locking of the Redundancy Floor)**.** *Under Assumptions 4.1 and 4.3, define the critical iteration count:*

$$K^* \triangleq \frac{p_0}{2\,N h_{\max}\,\eta\,(1 - p_0/2)^{\beta T}}. \tag{16}$$

For all $K \le K^*$, the per-step redundancy probability remains above half its initial value throughout training:

$$p_k \ge \frac{p_0}{2}, \quad \forall\, k \le K. \tag{17}$$

Proof. By contradiction. Suppose k^* is the first iteration at which $p_{k^*} < p_0/2$. For all $k < k^*$, we have $p_k \ge p_0/2$, so by Lemma A.1 the probability that the group contains at least one clean trajectory satisfies $P(m^{(k)}_{\text{clean}} \ge 1) \le N(1 - p_0/2)^{\beta T}$. By Assumption 4.3, using $h(0) = 0$ and $h \le h_{\max}$:

$$E[\Delta p_k] = \sum_{m=0}^{N} P\big(m^{(k)}_{\text{clean}} = m\big)\, h(m)\,\eta \le h_{\max}\,\eta\, P\big(m^{(k)}_{\text{clean}} \ge 1\big) \le N h_{\max}\,\eta\,(1 - p_0/2)^{\beta T}. \tag{18}$$

The cumulative reduction over k^* iterations satisfies:

$$p_0 - p_{k^*} \le k^* \cdot N h_{\max}\,\eta\,(1 - p_0/2)^{\beta T}. \tag{19}$$

For $p_{k^*} < p_0/2$, the left-hand side exceeds $p_0/2$, requiring $k^* > K^*$. This contradicts $k^* \le K \le K^*$. □

Remark A.3 (Self-Locking Interpretation). Lemma A.2 reveals a self-reinforcing mechanism: the redundancy probability p_k decreases slowly *because* p_k is large enough to make clean trajectories exponentially rare, which in turn limits the useful gradient signal for reducing p_k. On-policy methods cannot escape this cycle.

Theorem 4.5 (Redundancy Lower Bound for On-Policy RL)**.** *Under Assumptions 4.1 and 4.3, for any training iteration $K \le K^*$, the expected redundancy ratio of the trained policy satisfies:*

$$E[\rho(\tau_K)] \ge \frac{\beta p_0}{2}, \tag{13}$$

where $K^* = p_0 / \big(2N h_{\max}\,\eta\,(1 - p_0/2)^{\beta T}\big)$ and T is the length of trajectory τ.

Proof. By Lemma A.2, $p_K \geq p_0/2$ for all $K \leq K^*$. By Assumption 4.1, each of the βT redundancy-prone steps of τ_K is redundant with probability p_K and the remaining steps are never redundant, so

$$E[\rho(\tau_K)] = \frac{\beta T \cdot p_K}{T} = \beta p_K \geq \frac{\beta p_0}{2}. \tag{20}$$

□

Proposition 4.7. *Suppose at each iteration, trajectory evolution injects $m_{\text{evolve}} \in \{1, \ldots, N-1\}$ clean trajectories into the training group alongside $N - m_{\text{evolve}}$ on-policy rollouts. Under Assumption 4.3, for any target $\varepsilon \in (0, p_0)$, the expected redundancy satisfies*

$$E\left[\rho(\tau_K^{\text{evolve}})\right] \leq \beta\varepsilon \quad \text{for all } K \geq \frac{p_0 - \varepsilon}{\delta_\varepsilon}, \qquad \delta_\varepsilon := h(m_{\text{evolve}})\,\eta\left(1 - (1-\varepsilon)^{\beta T(N - m_{\text{evolve}})}\right) > 0. \tag{14}$$

Proof. Since the group contains $m_{\text{evolve}} \geq 1$ injected clean trajectories, $m_{\text{clean}}^{(k)} \geq m_{\text{evolve}}$ at every iteration, so the case $m_{\text{clean}}^{(k)} = 0$ never occurs; moreover, since $h \geq 0$, p_k is non-increasing in expectation.

Fix $\varepsilon \in (0, p_0)$ and consider an iteration k with $p_k \geq \varepsilon$. By Assumption 4.3, no progress is made only when $m_{\text{clean}}^{(k)} = N$ (as $h(N) = 0$), i.e., only when all $N - m_{\text{evolve}}$ on-policy rollouts are clean simultaneously, which by Lemma A.1 has probability at most $(1-p_k)^{\beta T(N-m_{\text{evolve}})} \leq (1-\varepsilon)^{\beta T(N-m_{\text{evolve}})}$. Otherwise, $m_{\text{evolve}} \leq m_{\text{clean}}^{(k)} \leq N-1$, and the monotonicity of h gives $h(m_{\text{clean}}^{(k)}) \geq h(m_{\text{evolve}})$. Hence

$$\begin{aligned} E[\Delta p_k] &= \sum_{m=0}^{N} P\big(m_{\text{clean}}^{(k)} = m\big)\, h(m)\, \eta \;\geq\; h(m_{\text{evolve}})\, \eta\, P\big(m_{\text{clean}}^{(k)} \leq N-1\big) \\ &\geq h(m_{\text{evolve}})\, \eta\Big(1 - (1-\varepsilon)^{\beta T(N-m_{\text{evolve}})}\Big) = \delta_\varepsilon. \end{aligned} \tag{21}$$

Therefore, as long as $p_j \geq \varepsilon$ for all $j < k$, we have $p_k \leq p_0 - k \cdot \delta_\varepsilon$, so p_k falls to at most ε within $(p_0 - \varepsilon)/\delta_\varepsilon$ iterations and, being non-increasing, stays there thereafter. Finally, by Assumption 4.1, $E[\rho(\tau_K^{\text{evolve}})] = \beta p_K \leq \beta\varepsilon$ for all $K \geq (p_0 - \varepsilon)/\delta_\varepsilon$. □

B Experiment Details

B.1 Pseudo Code

We present **`TrajEvolve`** training procedure in Algorithm 2, where the parts unique to **`TrajEvolve`** are marked in *italics*. Beyond the base GRPO loop, **`TrajEvolve`** introduces three key extensions: (1) *Redundancy-Aware Rewards* that penalize redundant steps in successful trajectories (Equation 7); (2) *Step-Level Advantage* via fine-grained credit assignment (Equation 9); and (3) a periodic *trajectory evolution phase* that actively refines historical trajectories to produce higher-quality training data, enabling the model to exceed its current inference limitations. The step-level credit labels used throughout the algorithm are obtained via the Hindsight Verification procedure (Algorithm 1). A lightweight step-level discriminator D_ϕ is trained via supervised fine-tuning on these credit labels and is used to identify error steps in failed trajectories during training.

B.2 Implementation Details

Hyperparameters for ALFWorld. We keep the base RL hyperparameters consistent across all methods. The maximum prompt length is 2048 tokens and the maximum response length is 512 tokens. Each episode runs for up to 30 environment steps. The actor learning rate is 1e-6. A rule-based reward scheme assigns 10 for task completion and 0 otherwise, with an additional penalty of -0.1 for invalid actions. All group-based RL methods adopt a group size of 8 with 16 groups sampled per rollout, yielding $16 \times 8 = 128$ parallel environments. The rollout temperature is 1.0 and the validation temperature is 0.4. The mini-batch size is set to 256. For TrajEvolve hyperparameters, the redundancy penalty weight α is set to 1.0, the step-level credit adjustment value λ is set to 0.5,

Algorithm 2 Training LLM Agents with TrajEvolve

Input: Initial policy $\pi_{\theta_{\text{old}}}$, base discriminator D_ϕ, task distribution $p(X)$, clipping parameter ϵ, group size N, *evolution frequency F, credit adjustment value δ, redundancy penalty weight α*

// Discriminator initialization
Fine-tune discriminator D_ϕ on credit labels obtained from Algorithm 1 with SFT
for each training iteration k **do**
  Update the old policy model: $\theta_{\text{old}} \leftarrow \theta$
  // Multi-step rollout phase
  Sample task $x \sim p(X)$ and launch N parallel environment instances
  for $t = 1$ to T **do**
    Generate actions $\{a_t^{(i)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid s_t^{(i)}, x)\}_{i=1}^N$
    Execute actions, collect rewards $\{r_t^{(i)}\}_{i=1}^N$ and next states $\{s_{t+1}^{(i)}\}_{i=1}^N$
  end for
  // Credit analysis and discriminator update phase
  Analyze successful trajectories with Algorithm 1 to obtain credit labels
  Fine-tune D_ϕ on obtained credit labels
  Analyze failed trajectories with D_ϕ to identify first error step
  // Redundancy-aware rewards and Step-level advantage computation
  Compute credit-augmented rewards with Equation 7
  Compute advantages $A(a_t^{(i)})$ with Equation 9
  // Policy update phase
  Update policy θ by maximizing objective $\mathcal{J}(\theta)$ with Equation 10
  if $k \bmod F = 0$ **then**
    // Trajectory evolution phase (Section 3.4)
    Evolve trajectories with evolution strategies and re-execute in $\mathcal{E}$
    Update policy θ on successful evolved trajectories
  end if
end for

and the trajectory evolution phase is triggered every $F = 5$ iterations. The discriminator is initialized from Qwen2.5-1.5B-Instruct and fine-tuned with a learning rate of 1e-5 for 1 epoch per update. All experiments are trained for 150 iterations on 8×H20 GPUs.

Hyperparameters for WebShop. We keep the base RL hyperparameters consistent across all methods. The maximum prompt length is 4096 tokens and the maximum response length is 512 tokens. Each episode runs for up to 20 environment steps. The actor learning rate is 1e-6. A rule-based reward scheme assigns 10 for task completion and 0 otherwise, with an additional penalty of -0.1 for invalid actions. All group-based RL methods adopt a group size of 8 with 16 groups sampled per rollout, yielding $16 \times 8 = 128$ parallel environments. The rollout temperature is 1.0 and the validation temperature is 0.4. The mini-batch size is set to 64. For TrajEvolve hyperparameters, the redundancy penalty weight α is set to 1.0, the step-level credit adjustment value λ is set to 0.5, and the trajectory evolution phase is triggered every 5 iterations. The discriminator is initialized from Qwen2.5-1.5B-Instruct and fine-tuned with a learning rate of 1e-5 for 1 epoch per update. All experiments are trained for 150 iterations on 8×H20 GPUs.

C Additional Experiments

C.1 Base Model Ablation

To verify that TrajEvolve's effectiveness is not contingent on a specific model family, we conduct additional experiments using Llama-3.2-3B-Instruct as the base model on both ALFWorld and WebShop benchmarks. Since the main experiments in Section 5 exclusively use the Qwen2.5 series, demonstrating consistent improvements across different model architectures strengthens the generalizability of our approach.

As shown in Table 3, our method achieves the best performance across nearly all task categories. On ALFWorld, TrajEvolve attains an overall success rate of 92.19%, surpassing the strongest baseline

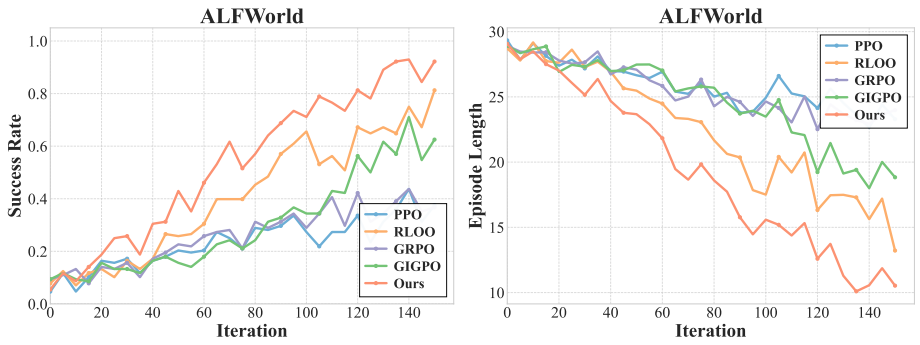


Figure 5: Training curves on ALFWorld with Llama-3.2-3B-Instruct. Left: validation success rate. Right: Average episode length. Our method consistently achieves a higher success rate and shorter, more efficient trajectories than all baselines.

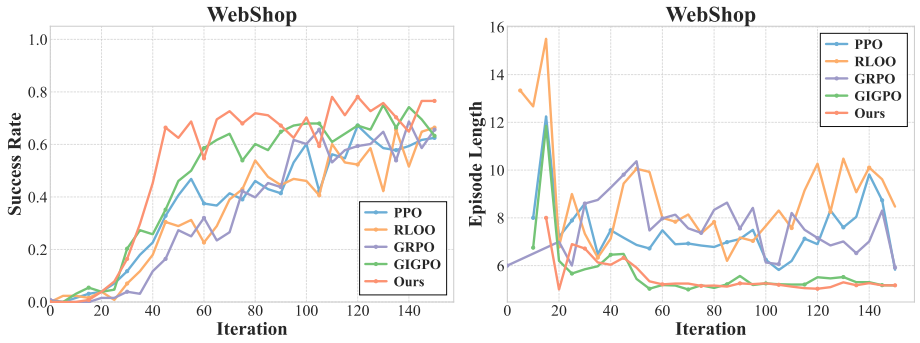


Figure 6: Training curves on Webshop with Llama-3.2-3B-Instruct. Left: validation success rate. Right: Average episode length. Our method consistently achieves a higher success rate and shorter, more efficient trajectories than all baselines.

RLOO at 81.25% by a margin of 10.94%. The improvements are particularly pronounced on task types that require longer interaction horizons, such as Cool and Heat, where our method reaches 89.47% and 88.89% respectively. These categories benefit most from redundancy elimination and trajectory evolution due to the larger room for refinement in longer action sequences. On WebShop, our method reaches 88.13 in score and 76.56% in success rate, outperforming all baselines by a substantial margin. It is worth noting that PPO and GRPO perform considerably worse on ALFWorld with this base model, achieving only 37.50% and 36.72% overall success rates. This suggests that Llama-3.2-3B-Instruct has a weaker initial policy for embodied tasks, which exacerbates the sparse signal problem. Our method effectively alleviates this issue by injecting high-quality evolved trajectories into training.

Figure 5 presents the training dynamics on ALFWorld. Our method exhibits a consistently steeper learning curve starting from around iteration 20 and maintains a clear advantage throughout training, ultimately converging to over 90% success rate. In contrast, PPO and GRPO plateau below 50%, while GIGPO shows unstable convergence behavior. The episode length plot further confirms that our method produces increasingly efficient trajectories, with the average episode length steadily decreasing to approximately 10 steps, the lowest among all methods. This indicates that the binary credit assignment mechanism successfully identifies and eliminates redundant actions even with a different base model.

Figure 6 shows analogous trends on WebShop. Our method achieves both the highest success rate and the shortest episode length. While RLOO exhibits competitive success rates, its episode length remains higher, indicating less efficient action sequences. GRPO shows notably longer episodes that peak above 13 steps in early training, consistent with the redundancy accumulation problem discussed in our theoretical analysis in Theorem 4.5. In contrast, our method rapidly reduces episode length to around 6 steps while simultaneously improving success rate, demonstrating that trajectory evolution and credit assignment work synergistically across model families.

Table 3: Performance comparison on ALFWorld and WebShop with Llama-3.2-3B-Instruct, the best result in each column is highlighted in bold.

Method	ALFWorld							WebShop	
	Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
PPO	57.14	25.00	29.17	27.78	21.05	41.67	37.50	86.25	62.50
GRPO	57.14	25.00	20.83	33.33	21.05	41.67	36.72	82.66	65.63
RLOO	94.29	62.50	95.83	66.67	52.63	87.50	81.25	79.81	66.41
GIGPO	65.71	37.50	66.67	77.78	52.63	58.33	62.50	83.04	63.28
Ours	**97.14**	**62.50**	**100.00**	**88.89**	**89.47**	**91.67**	**92.19**	**88.13**	**76.56**

Overall, these results confirm that TrajEvolve generalizes beyond the Qwen2.5 family and delivers consistent gains in both task success and trajectory efficiency when applied to Llama-3.2-3B-Instruct.

C.2 Discriminator Accuracy Experiment

A natural question is whether the discriminator can reliably identify necessary and redundant steps. To address this concern, we track the discriminator's classification accuracy throughout training on both benchmarks.

For successful trajectories, ground-truth credit labels are readily available through hindsight verification in the environment, so we directly compute accuracy against these labels. For failed trajectories, however, no ground-truth labels can be obtained since counterfactual verification requires a successful trajectory as input. We therefore employ GPT-5.2-2025-12-11 as an approximate judge to evaluate the discriminator's predictions on failed trajectories. To reduce computational cost, we report the accuracy over the first 50 training iterations.

As shown in Figure 7, the discriminator achieves strong overall accuracy on both benchmarks, reaching 90.93% on ALFWorld and 83.17% on WebShop. The accuracy on successful trajectories is notably higher, at 95.06% and 87.59% respectively, while the accuracy on failed trajectories is lower, at 84.95% and 77.26%. The training curves in Figure 7 further show that the discriminator maintains stable and consistent accuracy throughout training, with no significant degradation as the policy evolves.

We emphasize that the discriminator's role in our framework is deliberately limited. For successful trajectories, credit assignment is performed entirely through hindsight counterfactual verification, which yields exact ground-truth labels without relying on the discriminator at all. The discriminator only serves to generalize credit signals from successful trajectories to failed ones and to guide the trajectory evolution module. This design choice explains why even a lightweight 1.5B-parameter discriminator is sufficient to achieve strong overall performance. Furthermore, we note that credit assignment for failed trajectories is an inherently difficult problem, and the GPT-5.2 judge itself may introduce evaluation noise. Therefore, the reported accuracy on failed trajectories should be interpreted as an approximate reference rather than a precise measurement. Nevertheless, the results confirm that the discriminator provides sufficiently reliable signals for trajectory evolution, and our framework's core performance is not bottlenecked by discriminator quality.

C.3 Computation Cost Analysis

To assess the practical overhead introduced by TrajEvolve, we measure the per-step time cost of each training stage on both benchmarks. Understanding the computational budget is essential for evaluating whether our method can scale to larger settings without prohibitive overhead.

We profile all training stages on the same hardware configuration of 8×H20 GPUs with Qwen2.5-1.5B-Instruct as the policy, and report the average per-step wall-clock time in seconds. The results are shown in Figure 8. Beyond the standard RL training pipeline that includes rollout, reference log-probability computation, and policy update, our method introduces two additional modules: hindsight verification and trajectory evolution. Note that in this setting the discriminator has the same scale as the policy, so the relative share of the discriminator-related stages reported here is an upper estimate; with a larger policy, this share shrinks accordingly.

As shown in Figure 8, the rollout stage dominates the total training time on both benchmarks, taking 117.4s on ALFWorld and 63.2s on WebShop per step. In comparison, the overhead introduced by

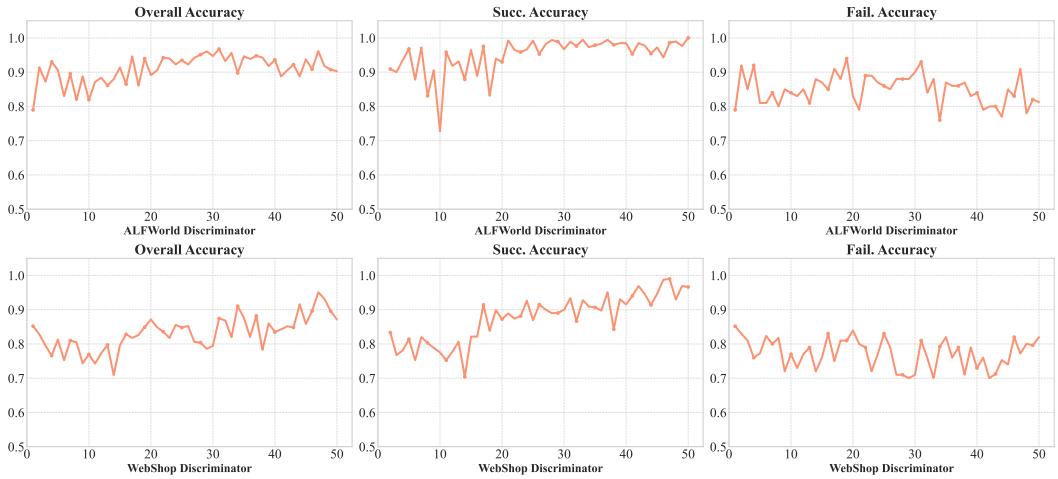


Figure 7: Training accuracy of our discriminator on ALFWorld (top) and WebShop (bottom), reporting overall, success, and failure accuracy across training iterations.

our two additional modules is moderate. Hindsight verification costs 61.2s and 20.3s per step on ALFWorld and WebShop respectively. Although the number of verification calls scales linearly with trajectory length, each call only replays the recorded actions in the environment and checks whether the task still succeeds, without requiring any model inference. The entire process is therefore purely CPU-based and consumes no GPU resources. Moreover, the verification calls are independent of each other, so this stage is fully parallelizable across trajectories and across calls, and its wall-clock time scales down directly with the number of available CPU cores. Since our profiling uses a modest CPU budget, this overhead still has considerable room for reduction.

Trajectory evolution costs 31.6s and 27.8s per step on ALFWorld and WebShop respectively. This stage is an intrinsic, periodic cost of our method rather than a per-step tax: evolution is triggered every 5 iterations with only two trajectories selected per group, so over a 150-iteration run it adds the equivalent of roughly 30 additional steps of work in total, and in some iterations there may not be enough successful trajectories available for evolution. The evolution process is essentially equivalent to re-executing these refined trajectories in the environment, which is comparable to a small fraction of the standard rollout cost. We consider this cost well justified, as the evolved trajectories expose the policy to high-quality compositions outside its current sampling distribution, an effect that on-policy training alone cannot reproduce within a practical budget (Theorem 4.5) and that therefore cannot be substituted by simply allocating additional training steps.

The remaining components, discriminator training and credit labeling, introduce overhead of 18.5s and 27.2s on ALFWorld and 13.3s and 13.1s on WebShop, as the discriminator is a lightweight 1.5B-parameter model trained with LoRA for only 1 epoch per iteration. We emphasize that our method does not critically depend on a strong discriminator: credit labels for successful trajectories come entirely from counterfactual verification in the environment, which yields exact ground-truth labels without any learned estimator and thus provides abundant training signal regardless of discriminator quality. The discriminator can therefore be replaced with an even smaller backbone (e.g., Qwen2.5-0.5B-Instruct) to further reduce this portion of the overhead.

Overall, the total per-step time of TrajEvolve is approximately 1.5$\times$ that of standard GRPO training. A substantial part of this overhead is reducible through more CPU cores and a smaller discriminator, while the remaining periodic cost of evolution is intrinsic to the method. Given the substantial performance gains demonstrated in Section 5, we consider this overhead to be a favorable trade-off between computational cost and agent performance.

C.4 TrajEvolve on Complex Scenarios

To evaluate whether TrajEvolve can generalize to more complex environments beyond ALFWorld and WebShop, we conduct additional experiments on ScienceWorld, a challenging text-based simulation environment that requires agents to perform scientific experiments through multi-step interactions.

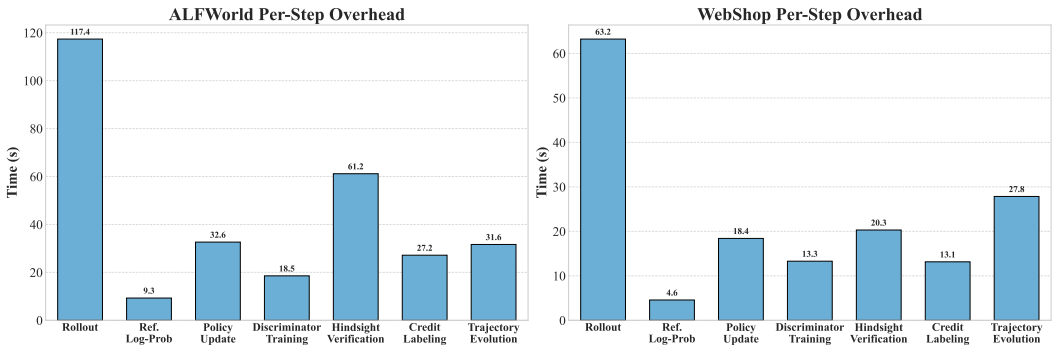


Figure 8: Per-step time overhead of each training stage on ALFWorld (left) and WebShop (right).

Unlike ALFWorld and WebShop where most steps directly alter the environment state, ScienceWorld involves a substantial proportion of information-gathering steps such as looking at objects, reading instructions, and using thermometers to take measurements. These steps do not change the environment or affect task completion, yet they may be essential for the agent to make informed decisions in subsequent steps.

The primary challenge in transferring TrajEvolve to this setting lies in determining the dependent steps during hindsight verification. Since information-gathering steps have no observable effect on the environment state or task outcome, removing them does not cause subsequent actions to become infeasible, nor does it directly lead to task failure. Therefore, the standard counterfactual verification procedure cannot determine whether these steps are truly necessary. To address this, we adopt a simple two-category strategy. We classify all steps into information-gathering steps and environment-changing steps based on the action verb, which provides a reliable and low-cost distinction. We retain all information-gathering steps unconditionally and apply hindsight verification only to environment-changing steps. Under this setting, our method can still effectively identify and eliminate redundant environment-changing actions, although some unnecessary information-gathering steps may be preserved. This simple adaptation allows TrajEvolve to transfer to complex scenarios with minimal modification.

We train all methods using Qwen2.5-7B-Instruct for 150 iterations and evaluate on two settings: IID, which tests on seen task variants and categories, and OOD, which tests on unseen task variants within seen categories. As shown in Table 4, our method achieves the highest performance on both settings, reaching 95.31% on IID and 89.06% on OOD, compared with 92.19% and 86.72% for GRPO and 92.97% and 87.50% for GiGPO. The improvements on the OOD setting are particularly encouraging, as they suggest that the more efficient trajectories learned through our evolution mechanism also benefit generalization to unseen task variants. These results confirm that TrajEvolve can be effectively transferred to more complex environments through lightweight adaptation of the verification procedure, without requiring fundamental changes to the framework.

Table 4: Final success rate (%) on ScienceWorld with Qwen2.5-7B-Instruct after 150 iterations. IID: seen task variants; OOD: unseen variants within seen categories.

Method	IID	OOD
GRPO	92.19	86.72
GiGPO	92.97	87.50
Ours	**95.31**	**89.06**

D Method Analysis

D.1 Method Transferability Analysis

We discuss the applicability of TrajEvolve to different task settings and clarify the conditions under which our method is most effective.

The core requirement of TrajEvolve is that the environment supports trajectory replay, meaning that a sequence of actions can be deterministically re-executed to reproduce the same state transitions. This property is widely satisfied across many practical domains. Task execution environments such as ALFWorld and WebShop are inherently replayable since the environment state is fully determined

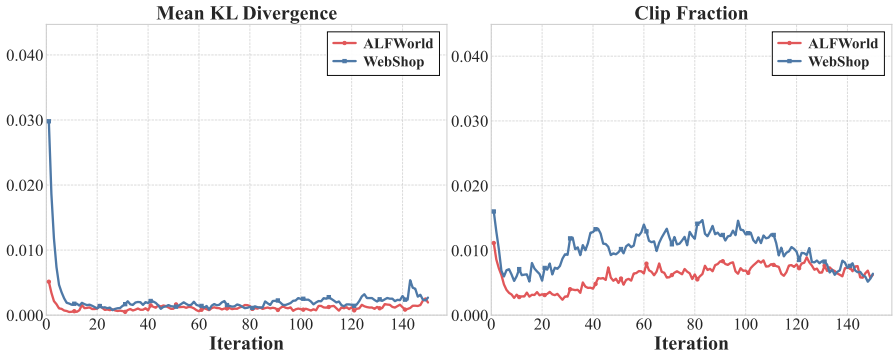


Figure 9: Mean KL divergence (left) and clip fraction (right) during training on ALFWorld and WebShop, indicating stable policy updates.

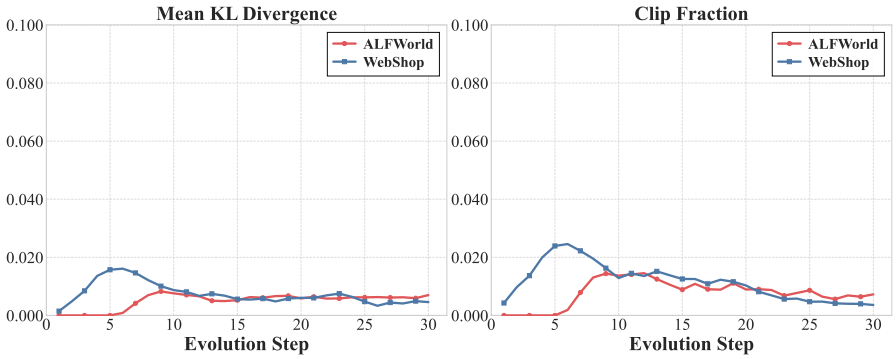


Figure 10: Mean KL divergence (left) and clip fraction (right) during the trajectory evolution stage on ALFWorld and WebShop.

by the action history. Code generation and debugging scenarios also naturally support replay, as programs can be re-executed with the same inputs to verify correctness. Web search and retrieval tasks similarly allow re-issuing queries and re-processing results in a deterministic manner. Despite this broad applicability, existing RL methods for LLM agents have largely overlooked the replay property of these environments. TrajEvolve explicitly leverages this property to construct higher-quality trajectories through hindsight verification and trajectory evolution, providing additional training signals that go beyond what on-policy sampling alone can offer.

Our method is particularly advantageous in environments where environment-changing actions constitute a large proportion of the trajectory. For such actions, counterfactual verification can directly determine whether each step is genuinely necessary for task completion by observing changes in the environment state and task outcome. This yields ground-truth credit labels without relying on any learned estimator, which is a unique strength of our approach. The strong experimental results on ALFWorld, where most steps involve physical state changes, exemplify this advantage.

However, this does not mean that TrajEvolve lacks transferability to more complex scenarios involving substantial information-gathering steps. As demonstrated in Section C.4, a simple two-category adaptation that retains information-gathering steps and applies verification only to environment-changing steps allows our method to achieve consistent improvements on ScienceWorld. Although this adaptation is conservative in that it preserves some potentially redundant information-gathering steps, it maintains the core benefits of trajectory evolution while requiring minimal modification to the framework. Exploring more fine-grained verification strategies for information-gathering steps remains a promising direction for future work.

D.2 Off-Policy Analysis

Since evolved trajectories are generated by refining historical rollouts rather than sampled from the current policy, they are inherently off-policy. A natural concern is whether incorporating such trajectories into PPO-style optimization introduces excessive distributional shift that could destabilize training. To investigate this, we monitor two standard trust-region diagnostics throughout training on both benchmarks.

The first metric is the approximate KL divergence between the updated policy and the reference policy, which measures the magnitude of per-update policy shift. The second metric is the clip fraction, which measures the proportion of tokens whose importance sampling ratio falls outside the clipping range $[1-\epsilon, 1+\epsilon]$. A high clip fraction indicates that many tokens experience large probability changes, suggesting significant distributional mismatch. We note that before each evolution update, we recompute log-probabilities under the current policy as the baseline for the importance sampling ratio. This ensures that the clipping mechanism correctly constrains the policy shift regardless of how the trajectories were originally generated.

Figure 9 presents the two metrics during the standard training phase across 150 iterations. On both benchmarks, the KL divergence drops rapidly within the first 20 iterations and stabilizes at very low values below 0.005 for the remainder of training. The clip fraction on ALFWorld remains consistently low throughout, while on WebShop it stabilizes around 0.01 after an initial transient. These patterns indicate that the overall training process maintains stable policy updates even with the periodic injection of evolved trajectories.

Figure 10 focuses specifically on the trajectory evolution phases, showing the two metrics across the 30 evolution steps that occur once every 5 training iterations. Both KL divergence and clip fraction exhibit a brief peak during the early evolution steps when the policy is changing most rapidly, and then steadily decrease over subsequent phases. By the later stages, both metrics converge below 0.005 and 0.01 respectively, reaching levels comparable to standard on-policy PPO updates. The average KL divergence across all evolution phases is 0.0054 on ALFWorld and 0.0075 on WebShop, while the average clip fraction is 0.0082 and 0.0115 respectively. These results confirm that the distributional shift introduced by trajectory evolution is bounded and small, and that the shift diminishes as training progresses since evolved trajectories become increasingly aligned with the current policy.

We note that leveraging off-policy successful trajectories has emerged as a promising direction in recent RL research. For instance, ExGRPO [Zhan et al., 2025] replays successful rollouts for additional training updates. Our empirical analysis provides evidence that such off-policy utilization can be safely combined with PPO-style optimization through proper importance sampling and clipping. Exploring more efficient use of off-policy data in long-horizon agent training remains a valuable direction for future work.

E Case Study

Table 5 illustrates how step-level credit labels are obtained for a successful ALFWorld trajectory via counterfactual verification. For each step t, we remove it from the trajectory, replay the remaining actions in the environment, and label it **Necessary** if the task can no longer be completed and *Redundant* otherwise. The same replay also yields $\mathrm{Dep}(t)$, the set of subsequent steps whose actions become inexecutable once step t is removed. Three observations are worth highlighting: (i) The agent first explores two unrelated drawers (steps 0–3). Removing step 0 or step 2 leaves the task completable, so both are redundant; their dependents, the corresponding open actions (steps 1 and 3), are thereby marked redundant without additional replays. Hence four perfectly executable actions receive no credit, a distinction that the outcome reward alone cannot make. (ii) Removing any of the remaining six steps causes the task to fail, so all of them are necessary and together form a compact success chain (*locate* $\rightarrow$ *grasp* $\rightarrow$ *clean* $\rightarrow$ *deliver*). (iii) $\mathrm{Dep}(t)$ exposes a dependency structure aligned with the sub-goals of the task: removing the visit to the countertop (step 4) invalidates taking the knife (step 5), cleaning it (step 7), and the final placement (step 9), whereas the sink-basin visit (step 6) only enables the cleaning step. Notably, steps 7 and 9 have empty dependency sets yet are still necessary: removing them leaves every later action executable but the goal unsatisfied. Environment-level success verification and action-level validity checks are therefore complementary. The resulting labels are used directly for credit assignment on successful trajectories, and serve as

Table 5: **Case study of counterfactual credit analysis on a successful ALFWorld trajectory.** The agent is asked to *put a clean knife in sidetable*. Each step t is removed from the trajectory (together with steps already identified as redundant) and the remaining actions are replayed in the environment: t is labeled **Necessary** if the task then fails, and *Redundant* otherwise. $\mathrm{Dep}(t)$ is the set of later steps whose actions become inexecutable (*"Nothing happens."*) in this replay. When t is redundant, all steps in $\mathrm{Dep}(t)$ are marked redundant as well without further replay (†). Steps 0–3 are unsuccessful drawer searches and are all redundant; steps 4–9 form the necessary chain (*locate* $\rightarrow$ *grasp* $\rightarrow$ *clean* $\rightarrow$ *deliver*). These labels supervise the training of our step discriminator.

Step	Action	Observation	Credit	$\mathrm{Dep}(t)$
0	go to drawer 1	The drawer 1 is closed.	*Redundant*	{1}
1	open drawer 1	The drawer 1 is open. In it, you see nothing.	*Redundant*	$\emptyset$
2	go to drawer 2	The drawer 2 is closed.	*Redundant*	{3}
3	open drawer 2	The drawer 2 is open. In it, you see a spoon 1.	*Redundant*	$\emptyset$
4	go to countertop 1	On the countertop 1, you see ... a knife 1 ... (and 18 other objects).	**Necessary**	{5,7,9}
5	take knife 1 from countertop 1	You pick up the knife 1 from the countertop 1.	**Necessary**	{7,9}
6	go to sinkbasin 1	On the sinkbasin 1, you see a potato 2 and a tomato 3.	**Necessary**	{7}
7	clean knife 1 with sinkbasin 1	You clean the knife 1 using the sinkbasin 1.	**Necessary**	$\emptyset$
8	go to sidetable 1	On the sidetable 1, you see a mug 1, a papertowelroll 1, and a saltshaker 1.	**Necessary**	{9}
9	move knife 1 to sidetable 1	You move the knife 1 to the sidetable 1. **[task done]**	**Necessary**	$\emptyset$

supervision for the step discriminator, which is applied to failed trajectories, where counterfactual verification is not applicable, to locate the first erroneous step.

F Extended Related Work

RL fine-tuning of large language models [Shen et al., 2025, Tan et al., 2026, Liu et al., 2026, Wu et al., 2026, Zhang et al., 2026, Xu et al., 2025, Wang et al., 2026] was popularized by RLHF [Cui et al., 2025, Cheng et al., 2025, Guo et al., 2025b, Wang et al., 2025e, Harutyunyan et al., 2019, Zhang et al., 2025b, Lu et al., 2025, Guo et al., 2025b] with PPO [Ouyang et al., 2022, Bai et al., 2022], where on-policy actor-critic updates are used to optimize against a learned reward model. To remove the value network and reduce the memory and compute footprint, GRPO [Shao et al., 2024] estimates advantages from groups of responses sharing the same prompt, and has since become the de facto recipe behind reasoning-oriented LLMs such as DeepSeek-R1 [Guo et al., 2025a], Qwen2.5-Math [Yang et al., 2024], and Kimi-K1.5 [Team et al., 2025]. A growing line of work refines GRPO by addressing its bias and stability issues: Dr. GRPO removes length and difficulty biases in advantage normalization [Liu et al., 2025b], DAPO introduces decoupled clipping and dynamic sampling for long-CoT training [Yu et al., 2025], GSPO performs sequence-level importance weighting to reduce variance [Zheng et al., 2025], and VinePPO and VAPO revisit token-level credit assignment with Monte-Carlo value estimates [Kazemnejad et al., Yue et al., 2025b]. RLOO and ReMax similarly exploit multi-sample baselines to obtain low-variance, value-network-free updates [Ahmadian et al., 2024, Li et al., 2023]. Because each rollout from an LLM is expensive, recent work increasingly emphasizes *reusing off-policy experience*, which can be viewed as the LLM analogue of experience replay. ReST, ReST-EM, and RAFT iteratively collect, filter, and replay self-generated trajectories for reward-weighted updates [Gulcehre et al., 2023, Singh et al., 2023, Dong et al., 2023, Qu et al., 2025, Yuan et al., 2024a], while Self-Rewarding LMs and SPIN bootstrap a replay-style buffer of model completions for iterative DPO-style training [Yuan et al., 2024b, Chen et al., 2024]. Several works push GRPO itself toward the off-policy regime: Tapered Off-Policy REINFORCE and LUFFY incorporate stale or expert trajectories with importance correction and clipping [Roux et al., 2025, Yan et al., 2025], and asynchronous trajectory-balance training decouples data collection from policy updates to better utilize a replay buffer [Bartoldson et al., 2025]. In multi-turn agentic RL [Hu et al., 2025, Ji et al., 2025, Lin et al., 2025, Wang et al., 2025a, Wu et al., 2025, Yuan et al., 2025, Wang et al., 2025b], replay buffers of past trajectories have been shown to stabilize long-horizon credit assignment in environments such as ALFWorld and WebShop [Zhou et al., 2024, Song et al., 2024, Xi et al., 2025]. These developments collectively position experience replay as a key lever for

sample-efficient and scalable RL post-training of LLMs [Wang et al., 2025d, Liu et al., 2025a, Zhang et al., 2025a, Yin et al., 2025, Hao et al., 2025a,b, Zhang et al., 2020].

To improve sample efficiency in online RL, replay-based methods [Zhan et al., 2025, Lu et al., 2025, Mroueh et al., 2025] store and reuse historical rollouts from replay buffers, reducing redundant sampling. These approaches typically prioritize valuable experiences based on correctness or entropy.

G Prompts

We adopt the same prompt design as GiGPO [Feng et al., 2025] for both ALFWorld and WebShop agents. As shown in Figure 11 and Figure 12, each prompt provides the agent with the task description, recent interaction history, current observation, and the set of admissible actions. The agent is instructed to first produce step-by-step reasoning within `<think> </think>` tags, followed by the selected action within `<action> </action>` tags. We set the history length to 5 for both environments.

In addition to the standard prompts, we introduce *think-only* prompt templates (Figure 13 and Figure 14) specifically for the trajectory evolution phase. During evolution, certain steps are assigned predetermined actions that are either retained from the original trajectory or corrected via format mapping, and the model only needs to generate the corresponding reasoning. The think-only prompt replaces the original action instruction block with the predetermined action and asks the model to produce only the reasoning content, without outputting any action tags. This ensures that evolved trajectories maintain coherent thought processes for every step, even when the actions are externally specified rather than generated by the policy.

Prompt Template for ALFWorld

You are an expert agent operating in the ALFRED Embodied Environment. Your task is to: {task_description}. Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}. You are now at step {current_step} and your current observation is: {current_observation}. Your admissible actions of the current situation are: [{admissible_actions}].
Now it's your turn to take an action. You should first reason step-by-step about the current situation. This reasoning process MUST be enclosed within <think> </think> tags. Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

Figure 11: The prompt template of ALFWorld agents.

Prompt Template for WebShop

You are an expert autonomous agent operating in the WebShop e-commerce environment. Your task is to: {task_description}. Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}. You are now at step {current_step} and your current observation is: {current_observation}. Your admissible actions of the current situation are: [{available_actions}].
Now it's your turn to take one action for the current step. You should first reason step-by-step about the current situation, then think carefully which admissible action best advances the shopping goal. This reasoning process MUST be enclosed within <think> </think> tags. Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

Figure 12: The prompt template of WebShop agents.

Think-Only Prompt Template for ALFWorld

You are an expert agent operating in the ALFRED Embodied Environment. Your task is to: {task_description}. Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}. You are now at step {current_step} and your current observation is: {current_observation}. Your admissible actions of the current situation are: [{admissible_actions}].
The action to take is: {evolution_action}. Generate only your reasoning for why this action is appropriate. Do not output any action tags, just the reasoning content.

Figure 13: The think-only prompt template used during the trajectory evolution phase in ALFWorld. The predetermined action is provided and the model is asked to generate only the reasoning.

Think-Only Prompt Template for WebShop

You are an expert autonomous agent operating in the WebShop e-commerce environment. Your task is to: {task_description}. Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}. You are now at step {current_step} and your current observation is: {current_observation}. Your admissible actions of the current situation are: [{available_actions}].
The action to take is: {evolution_action}. Generate only your reasoning for why this action is appropriate. Do not output any action tags, just the reasoning content.

Figure 14: The think-only prompt template used during the trajectory evolution phase in WebShop. The predetermined action is provided and the model is asked to generate only the reasoning.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims are stated in the abstract and introduction

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please refer to Section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: We have given the full assumptions and complete proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have included all the information needed to reproduce the results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have released the code. In addition, the datasets used in our work are all public datasets.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Please refer to the Implementation Details Section.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We conduct the two-sided t-test on our results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to Implementation Details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We conform to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the impacts in the Introduction section.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: All datasets used in our work are public datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly cite the dataset and models used in our work.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We have given the descriptions in our paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing experiments are conducted in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No potential risk is found in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLM usage does not impact the core methodology, scientific rigorousness, or originality of the research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.