Reinforcing Multi-Turn Reasoning in LLM Agents via Turn-Level Reward Design and Credit Assignment

Anonymous Author(s)
Affiliation
Address
email

Abstract

This paper investigates approaches to enhance the reasoning capabilities of Large Language Model (LLM) agents using Reinforcement Learning (RL). Specifically, we focus on long-horizon multi-turn agent scenarios, which can be naturally modeled as Markov Decision Processes. Although popular RL algorithms such as Group Relative Policy Optimization (GRPO) and Proximal Policy Optimization (PPO) have been widely applied to train multi-turn LLM agents, they typically rely only on sparse final rewards and lack dense intermediate signals across multiple decision steps, limiting their performance on complex reasoning tasks. To address this, we introduce a *fine-grained turn-level credit assignment* strategy to enable more effective process-level supervision in multi-turn agent interactions. By incorporating well-designed *turn-level rewards*, we extend GRPO and PPO to their multi-turn variants that better guide LLM agents at each round of interaction. Our case studies on multi-turn reasoning-augmented search tasks demonstrate that RL algorithms augmented with fine-grained credit assignment significantly improve the performance of LLM agents compared with baselines. Evaluated on diverse question-answering datasets with 7B models, the training and validation reward curves illustrate that our method achieves *greater stability*, *faster convergence*, and *higher accuracy* across multiple runs.

1 Introduction

Reinforcement Learning (RL) has recently emerged as a powerful approach for improving the reasoning capabilities of Large Language Models (LLMs), allowing them to explore and refine long Chains of Thought (CoT) (Wei et al., 2022) in complex decision-making tasks. Building on this paradigm, reasoning-based LLMs, such as OpenAI's o1 (Jaech et al., 2024) and DeepSeek's R1 (Guo et al., 2025a), demonstrate remarkable performance in textual reasoning tasks by learning analytical thinking and self-reflection.

Despite these advancements, LLMs that rely solely on CoT textual reasoning remain limited in tasks that require precise and complex numerical computation, information retrieval from web pages or local databases, or code execution. Equipping LLMs as autonomous agents with access to external tools, such as search engines, scientific calculators, or code interpreters, can significantly extend their capabilities beyond pure text-based reasoning.

However, training LLMs to operate as autonomous agents in interactive environments faces unique challenges. Agent settings often require models to make sequential, multi-turn decisions in complex reasoning tasks. Many existing approaches (Chen et al., 2025b; Jin et al., 2025b; Feng et al., 2025a) formulate these multi-turn interactive tasks as single-turn problems, relying solely on final outcome-level rewards such as answer correctness. Popular RL algorithms, including Group Relative Policy Optimization (GRPO) (Shao et al., 2024) and Proximal Policy Optimization (PPO) (Schulman et al., 2017), are commonly used in this setting. However, such single-turn formulation is inadequate for long-horizon multi-turn reasoning as it treats the entire trajectory as a single decision step, ignoring the multi-turn structure of the tasks. In particular, it ignores *turn-level rewards*—intermediate signals that indicate whether individual steps are helpful or harmful. Without access to dense turn-level feedback, agents struggle to refine their behavior, making it difficult to interact effectively with dynamic environments over multiple steps. For example, in a search agent, selecting a good query early on is crucial for retrieving relevant information; without turn-level feedback, the agent cannot learn which queries contribute to correct answers.

Recent studies (Li et al., 2025; Qian et al., 2025; Wang et al., 2025a; Labs, 2025; Wang et al., 2025b; Zhang et al., 2025; Singh et al., 2025; Jin et al., 2025a) formulate multi-turn agentic tasks as Markov Decision Processes (MDPs) and incorporate turn-level intermediate rewards like tool execution. However, these approaches still suffer from the credit assignment problem: they combine outcome- and turn-level rewards into a sparse trajectory-level signal. This makes advantage estimation inaccurate and prevents RL algorithms from providing fine-grained supervision across intermediate rounds of interaction.

Process-level supervision enables the design of intermediate and final reward functions that guide strategic tool usage and deliver high-quality feedback, improving training stability. Motivated by this, we propose a fine-grained turn-level credit assignment strategy for multi-turn LLM agent training. Our key contributions are as follows:

- We model multi-turn long-horizon reasoning tasks in LLM agents as MDPs, which naturally capture the sequential decision-making structure of such problems.
- To train multi-turn LLM agents effectively under the MDP framework, we introduce a fine-grained turn-level credit assignment strategy. Specifically, we extend GRPO and PPO to their multi-turn variants by incorporating both final outcome rewards and intermediate turn-level rewards. While multi-turn GRPO requires exponential rollout samples to compute intermediate advantages, multi-turn PPO leverages a critic model, offering a more efficient and scalable solution.
- To highlight the importance of the credit assignment mechanism, we perform a case study using a reasoning-augmented search agent that operates in multiple steps: reasoning, search, and answering, and carefully design the intermediate and final reward functions.
- Our case studies on multi-turn reasoning-augmented search tasks show that incorporating fine-grained credit assignment enables RL algorithms to significantly outperform baseline methods. On both general and multi-hop question-answering datasets with Qwen2.5-7B models, our approach yields more stable training, faster convergence, and higher accuracy across five independent runs. Furthermore, our algorithm consistently avoids training crashes and reliably generates outputs in the correct format.

2 Problem Formulation for LLM Agent Training

2.1 Single-Turn Problem Formulation

Traditional RL for LLMs is typically formulated in a single-turn setting. The objective is to maximize the expected final reward:

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)} \left[R(x,y)\right] \tag{1}$$

where π_θ denotes the policy model, x is the input prompt sampled from the dataset $\mathcal{D}$, y is the output response generated by the LLM, and $R(x,y)$ denotes the final outcome reward of a prompt–response pair. In this single-turn formulation, an entire generation y is treated as a single action, and only a final outcome reward $R(x,y)$ is observed. Thus, Problem (1) can be interpreted as a contextual bandit problem.

2.2 Multi-Turn Problem formulation: Turn-Level MDP

LLM agents operate in interactive environments that unfold over multiple turns and involve stochastic feedback. To capture these dynamics, we formulate the multi-turn agent task as a *turn-level MDP*, which is formally defined as

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\}$$

Here, $\mathcal{S}$ denotes the state space, and $\mathcal{A}$ denotes the action space. A state $s \in \mathcal{S}$ typically corresponds to an interaction history, while an action $a \in \mathcal{A}$ often corresponds to a sequence of generated tokens, possibly interleaved with environment feedback. $\mathcal{P}$ represents the transition dynamics. $\mathcal{R}$ is the turn-level reward function.

Assume the LLM agent interacts with the environment for K turns. At the k-th turn, conditioned on the current state s_k, the agent makes an action a_k according to the LLM policy π_θ. The environment may provide feedback o_k. The agent then receives a reward $\mathcal{R}_k$, and transitions to the next state s_{k+1}. We formalize this process as follows:[1]

$$s_{k+1} = [s_k, a_k], \quad \mathcal{R}_k = \mathcal{R}(s_k, a_k)$$

This yields a full trajectory $\tau = \{s_1, a_1, \ldots, s_K, a_K\}$. Within the MDP framework, the objective can be expressed as maximizing the cumulative reward:

$$\max_{\pi_\theta} \ \mathbb{E}_{\tau \sim \pi_\theta} \left[\mathcal{R}(\tau) = \sum_{k=1}^{K} \mathcal{R}(s_k, a_k) \right], \tag{2}$$

where $\mathcal{R}(\tau)$ denotes the cumulative reward over the entire trajectory. When only an outcome reward is provided, the intermediate rewards are zero, i.e.,[2]

$$\begin{aligned} \mathcal{R}_k &= \mathcal{R}(s_k, a_k) = 0, \quad \text{for } \ k = 1, 2, \ldots, K-1 \\ \mathcal{R}_K &= \mathcal{R}(s_K, a_K) = R(x, y), \end{aligned}$$

In this case, the MDP formulation in Eq. (2) reduces to Problem (1).

3 Credit Assignment in GRPO for Multi-Turn Agentic Tasks

3.1 GRPO for Single-Turn Settings

Recently, the Group Relative Policy Optimization (GRPO) algorithm (Shao et al., 2024) has been widely used to enhance the reasoning capabilities of LLMs. GRPO estimates the advantage in a group-relative manner. Specifically, for each input question x, it samples a group of responses $\{y_1, y_2, \ldots, y_G\}$ from the reference policy π_{ref}. To solve Problem (1), GRPO optimizes the policy by maximizing the following objective function:

$$\begin{aligned} \mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x \sim \mathcal{D},\, \{y_i\}_{i=1}^G \sim \pi_{\text{old}}(\cdot|x)} \Bigg[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \min \Bigg(\frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\text{old}}(y_{i,t} \mid x, y_{i,<t})} A_{i,t}, \\ \text{clip} \left(\frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\text{old}}(y_{i,t} \mid x, y_{i,<t})}, 1-\epsilon, 1+\epsilon \right) A_{i,t} \Bigg) - \beta \mathbb{D}_{\text{KL}} \left[\pi_\theta \,\|\, \pi_{\text{ref}} \right] \Bigg], \end{aligned} \tag{3}$$

where $\frac{\pi_\theta(y_{i,t}|x, y_{i,<t})}{\pi_{\text{old}}(y_{i,t}|x, y_{i,<t})}$ is the token-level importance sampling ratio between the current policy π_θ and the previous policy π_{old}, ϵ is the clipping parameter, and β is the KL divergence coefficient. Given a group of final outcome rewards $\{R_i\}_{i=1}^G$, the advantage of the i-th response $A_{i,t}$ is calculated by

$$A_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^G)}{\text{std}(\{R_i\}_{i=1}^G)}, \quad R_i = R(x, y_i). \tag{4}$$

3.2 Limitations of GRPO in Multi-Turn Settings

It is straightforward to observe that GRPO is well-suited for the single-turn problem. The advantage is computed by normalizing the final outcome rewards within the sampled group.

[1] If the environment feedback o_k exists, the process is given by $s_{k+1} = [s_k, a_k, o_k]$ and $\mathcal{R}_k = \mathcal{R}(s_k, a_k, o_k)$. Here, we omit o_k for notational simplicity.

[2] In this paper, we denote $R(x, y)$ as the final outcome reward and $\mathcal{R}(s, a)$ as the general turn-level reward in the multi-turn setting.

In multi-turn tasks, intermediate signals are often available to guide the LLM agent. However, GRPO does not naturally incorporate such intermediate rewards into advantage estimation, making it difficult to leverage them effectively. A naive solution to Problem (2) is to merge the intermediate rewards and the final outcome reward as a single sparse trajectory-level reward, that is,

$$A_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^G)}{\text{std}(\{R_i\}_{i=1}^G)}, \quad R_i = \sum_{k=1}^{K} \mathcal{R}_{i,k} = \sum_{k=1}^{K} \mathcal{R}(s_{i,k}, a_{i,k}) \tag{5}$$

where $\mathcal{R}_{i,k} = \mathcal{R}(s_{i,k}, a_{i,k})$ denotes the intermediate reward given the state $s_{i,k}$ and action $a_{i,k}$ in the k-th turn.

For the two advantage estimation strategies in Eq. (4) and Eq. (5) used by GPRO, the advantage function $A_{i,t}$ is computed at the trajectory level, i.e., $A_{i,1} = A_{i,2} = \cdots = A_{i,t} = \cdots = A_{i,|y_i|}$. This means that the same advantage is assigned uniformly across the entire trajectory, without distinguishing the contributions of individual tokens. As a result, it leads to coarse credit assignment and fails to capture the effect of individual turns. For long-horizon multi-turn tasks, trajectory-level credit assignment often leads to unstable training and suboptimal performance.

3.3 Turn-Level Credit Assignment for GRPO: A Simple Attempt

To highlight the importance of fine-grained credit assignment in GRPO, we consider a simple two-turn agent setting. In this case, the agent receives a group of intermediate rewards $\{\mathcal{R}_{i,1}\}_{i=1}^G$ in the first turn and final rewards $\{\mathcal{R}_{i,2}\}_{i=1}^G$ in the second turn. Based on these signals, we propose our turn-level credit assignment strategy for GRPO. The resulting turn-level advantages in the first and second turns are given by:

$$\bar{A}_{i,1} = A_{i,1} + A_{i,2}, \quad \bar{A}_{i,2} = A_{i,2}, \tag{6}$$

where

$$A_{i,1} = \frac{\mathcal{R}_{i,1} - \text{mean}(\{\mathcal{R}_{i,1}\}_{i=1}^G)}{\text{std}(\{\mathcal{R}_{i,1}\}_{i=1}^G)}, \quad A_{i,2} = \frac{\mathcal{R}_{i,2} - \text{mean}(\{\mathcal{R}_{i,2}\}_{i=1}^G)}{\text{std}(\{\mathcal{R}_{i,2}\}_{i=1}^G)} \tag{7}$$

We can see that the advantage in the first turn combines both the intermediate and final advantages, whereas the advantage in the second turn depends only on the final outcome. By leveraging intermediate rewards, all tokens within a single turn share a unified advantage signal. We refer to this algorithm as *multi-turn GRPO (MT-GRPO)*. A detailed derivation of MT-GRPO for the general multi-turn setting is provided in Appendix C.

Case Study on the Two-Turn Agent Setting. We conduct experiments to illustrate the effectiveness of the proposed MT-GRPO method in the two-turn agent setting (see Appendix D for more details). Figure 1 shows the training reward curves for GRPO and MT-GRPO. We observe that MT-GRPO achieves higher accuracy and more stable tool usage, verifying the importance of fine-grained credit assignment for multi-turn agent tasks.

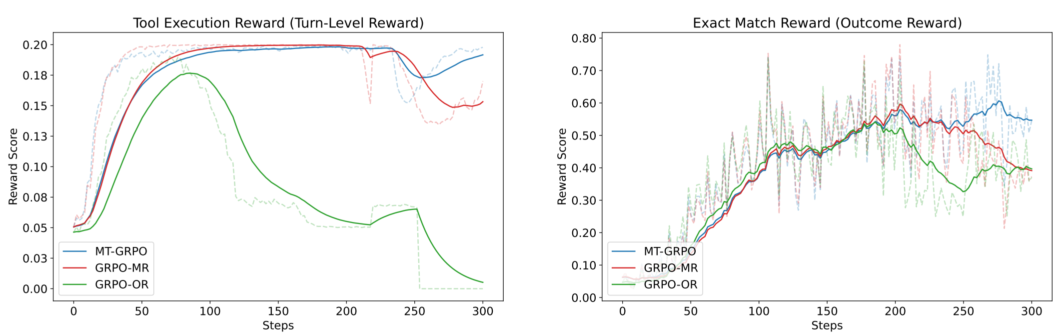

Figure 1: Curves for different training reward components during training with various algorithms (MT-GRPO in Eq. (6), GRPO-OR in Eq. (4), and GRPO-MR in Eq. (5)). Each plot shows the training reward score over training steps for Tool Execution and Exact Match. Dotted lines represent the average reward across 10 runs, while solid lines show trends smoothed using the Exponential Moving Average (EMA).

Limitations of MT-GPRO. (1) In MT-GRPO, computing the intermediate advantages requires G rollout samples at each turn. Therefore, over a horizon of K turns, this results in G^{K-1} rollout trajectories in total. Such exponential growth in complexity makes the approach computationally prohibitive for long-horizon multi-turn tasks. (2) This strategy also assumes that all rollout samples in a group must contain the same number of turns, which requires enforcing this constraint in the system prompt and leads to a fixed-turn setting. Such a restriction limits the flexibility and applicability of GRPO in more diverse interaction settings.

4 Credit Assignment in PPO for Multi-Turn Agentic Tasks

In the previous section, we illustrated the importance of fine-grained turn-level credit assignment, which improves the performance of LLM agents in multi-turn interactions. However, the exponential computational cost and the rigidity in handling variable numbers of turns limit the applicability of MT-GRPO to general agent tasks. In this section, we present our credit assignment strategy for PPO, aiming to provide a more flexible, scalable, and efficient solution.

PPO. Proximal Policy Optimization (PPO) (Schulman et al., 2017) is a popular actor-critic RL algorithm commonly used for LLM training (Ouyang et al., 2022). To solve Problem (2), PPO updates the policy by maximizing the following objective:

$$\begin{aligned} \mathcal{J}_{\text{PPO}}(\theta) &= \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\text{old}}(\cdot|x)} \\ &\left[\frac{1}{|y|}\sum_{t=1}^{|y|}\min\left(\frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\text{old}}(y_t \mid x, y_{<t})}A_t,\, \text{clip}\left(\frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\text{old}}(y_t \mid x, y_{<t})},\, 1-\epsilon,\, 1+\epsilon\right)A_t\right)\right], \end{aligned} \quad (8)$$

The advantage estimate A_t is computed using Generalized Advantage Estimation (GAE) (Schulman et al., 2015), based on rewards and a learned value function (critic model). Formally, for a trajectory of length T, the GAE A_t at time step t is computed as:

$$A_t = \sum_{l=0}^{T-t-1}(\gamma\lambda)^l \delta_{t+l}, \quad \delta_t = r_t + \gamma V_{t+1} - V_t \quad (9)$$

where γ is the discount factor, $\lambda \in [0,1]$ is the GAE parameter, δ_t is the temporal-difference (TD) error, $r_t = r(x, y_{<t})$ is the token-level reward and $V_t = V(x, y_{<t})$ is the token-level value for the entire trajectory. Through the mechanism of GAE, the token-level value function enables token-level advantage estimation.

Turn-Level Rewards in PPO. Given both intermediate rewards $\mathcal{R}^I$ and the final reward $\mathcal{R}^F$, the token-level reward r_t is assigned as

$$r_t = \begin{cases} \mathcal{R}^F & \text{if } t \text{ is the last token of the entire trajectory} \\ \mathcal{R}^I & \text{if } t \text{ is the last token of the current turn} \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

With explicit intermediate rewards, GAE provides fine-grained training signals at each turn. For clarity, we refer to PPO trained with both intermediate and final rewards as *multi-turn PPO (MT-PPO)*, while PPO trained with only sparse trajectory-level rewards is simply referred to as *PPO*. Compared with MT-GRPO, which requires exponential rollout samples to compute intermediate advantages, MT-PPO leverages a critic model with GAE, offering a more efficient and scalable solution.

Table 1: Comparison of granularity of advantage estimation and reward assignment across different RL algorithms for multi-turn LLM agents.

RL Algorithm	Granularity of Reward Assignment	Granularity of Advantage Estimation
GRPO	Trajectory-Level	Trajectory-Level
MT-GRPO	Turn-Level	Turn-Level
PPO	Trajectory-Level	Token-Level
MT-PPO	Turn-Level	Token-Level

5 Case Study: Multi-Turn Reasoning-Augmented Search Agent

5.1 Task Formulation

We study an LLM agent that performs multi-turn reasoning with search engine interactions. The task can be naturally formulated under the MDP framework, which involves multiple steps of reasoning, retrieval, and final answer generation for question answering. The goal is to improve the agent's performance through effective integration of external search. Specifically, the agent learns to leverage a Wikipedia search engine to retrieve relevant information and generate an accurate answer. Without search calling, the agent must rely solely on its internal knowledge to answer questions, which can limit accuracy, especially for fact-based queries requiring up-to-date or domain-specific information.

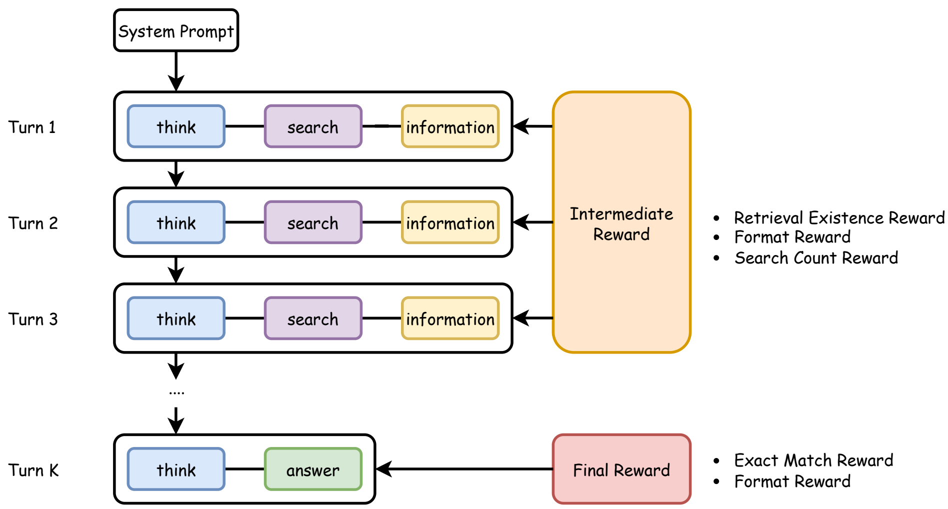

Figure 2: Overview of the multi-turn reasoning-augmented search agent pipeline.

Given a system prompt and a question, each iteration of the LLM-based search agent proceeds as follows: (1) The agent begins with *reasoning*, analyzing the current context to identify missing information. (2) It then formulates a search query to *retrieve* relevant information from an external database, which is integrated into the evolving *context*. (3) This cycle continues until the agent judges that the context is sufficient, at which point it performs a final round of *reasoning* to generate the answer. The overall interaction follows a multi-turn reasoning–search loop, as shown in Figure 2.

These steps impose strict constraints, such as permitting only a single search step and requiring the use of specific XML-like tags to delineate each stage of the interaction. Following (Jin et al., 2025b), reasoning steps are enclosed within `<think> </think>`, search queries are wrapped in `<search> </search>`, retrieved information is inserted into `<information> </information>`, and the final answer is placed within `<answer> </answer>`.

5.2 Reward Design

To align with the environment of the aforementioned LLM-based search agent, we design two types of verifiable reward functions: final rewards $\mathcal{R}^F$ and intermediate rewards $\mathcal{R}^I$.

Final Verifiable Rewards: evaluate the model-generated responses in the last turn, focusing on both the correctness of the answer and the adherence to the required output format.

- *Final Exact Match Reward:* evaluates whether the extracted answer (from the `<answer>` tag) exactly matches any accepted ground-truth answer after normalization (e.g., lowercasing and whitespace removal):

$$\mathcal{R}^F_{\text{EM}} = \begin{cases} 1.0 & \text{if the extracted answer exactly matches any ground truth,} \\ 0 & \text{otherwise.} \end{cases}$$

- *Final Format Reward:* ensures format correctness by verifying that: (1) only `<think>` and `<answer>` tags appear (no extra tags), (2) each tag appears exactly once, and (3) `<think>` precedes `<answer>`.

$$\mathcal{R}^F_{\text{format}} = \begin{cases} 0.2 & \text{if the format is correct,} \\ -1.0 & \text{otherwise.} \end{cases}$$

Intermediate Verifiable Rewards: guide the agent's behavior in intermediate turns by evaluating the presence of ground-truth answers in retrieved content, enforcing proper format usage, and discouraging excessive search calls.

- *Intermediate Retrieval Existence Reward:* evaluates whether any accepted answer appears in the one-round search result (from `<information>` tag), using case-insensitive matching.

$$\mathcal{R}^I_{\text{retrieval}} = \begin{cases} 0.3 & \text{if retrieved information contains a ground-truth answer,} \\ 0 & \text{otherwise.} \end{cases}$$

- *Intermediate Format Reward:* ensures format correctness by verifying that: (1) only `<think>`, `<search>`, and `<information>` tags appear (no extra tags), (2) each tag appears exactly once, and (3) `<think>` precedes `<search>` and `<information>`.

$$\mathcal{R}^I_{\text{format}} = \begin{cases} 0.1 & \text{if the format is correct,} \\ -0.2 & \text{otherwise.} \end{cases}$$

- *Intermediate Search Count Reward:* penalizes excessive search usage.

$$\mathcal{R}^I_{\text{search}} = -0.1 \cdot n_{\text{search}},$$

  where n_{search} denotes the cumulative number of search invocations from the first turn up to the current turn.

Among these signals, retrieval and format correctness are assigned relatively smaller weights compared to answer correctness, which helps prevent reward hacking. A negative reward (penalty) is applied when the format is incorrect, ensuring that the agent adheres to the required structure. In addition, we introduce an intermediate search penalty, which discourages excessive or unnecessary search calls and prevents the agent from either avoiding the question answering or failing due to crashes. Here, both final rewards and intermediate rewards are defined as the summation of their respective component rewards.

6 Experiments

6.1 Experiment Setup

In our experiments, we build our codebase upon the open-source project Search-R1 (Jin et al., 2025b), which trains LLM agents for multi-turn reasoning-augmented search tasks.

Datasets. We train the LLM agent on two types of question answering datasets: (1) NQ (Karpukhin et al., 2020) for general question answering, and (2) HotpotQA (Yang et al., 2018) for multi-hop question answering. These datasets cover a diverse range of search and reasoning challenges, providing a comprehensive basis for evaluation.

Evaluated Methods. We compare our proposed multi-turn PPO (MT-PPO) with vanilla PPO: (1) PPO (Jin et al., 2025b): original PPO with only final answer correctness rewards, and (2) MT-PPO (ours): PPO variant with both intermediate and final rewards, as described in Section 5.2.

Evaluation Metrics. We evaluate model performance using three types of rewards during both training and validation: (1) answer correctness reward, (2) format correctness reward, and (3) retrieval correctness reward. Each reward is assigned a value of 1.0 if the criterion is satisfied and 0 otherwise. The detailed reward rules are provided in Appendix B.1.

Training Details. We use Qwen2.5-7B (Yang et al., 2024) as the base model, E5 (Wang et al., 2022) as the retriever, and 2018 Wikipedia dump (Karpukhin et al., 2020) as the corpus. Following (Jin et al., 2025b), we enable policy loss masking for retrieved tokens. More details on experimental settings can be found in Appendix B.2.

6.2 Main Results

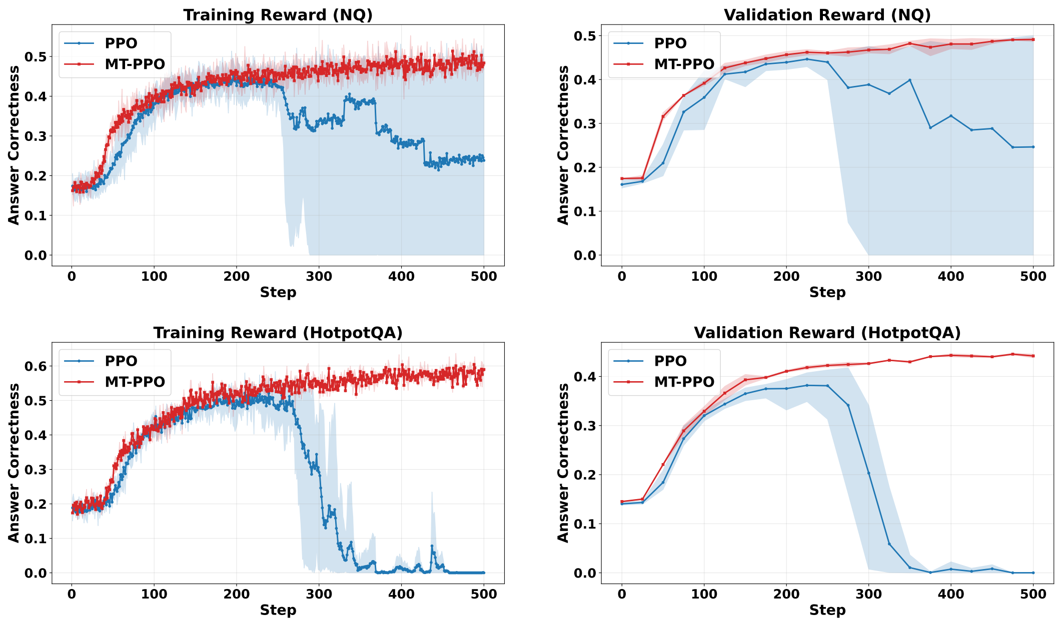

Figure 3: Curves of **answer correctness** reward during training and validation on the NQ and HotpotQA datasets for PPO and MT-PPO algorithms, where shaded regions represent the range between the maximum and minimum values across 5 runs.

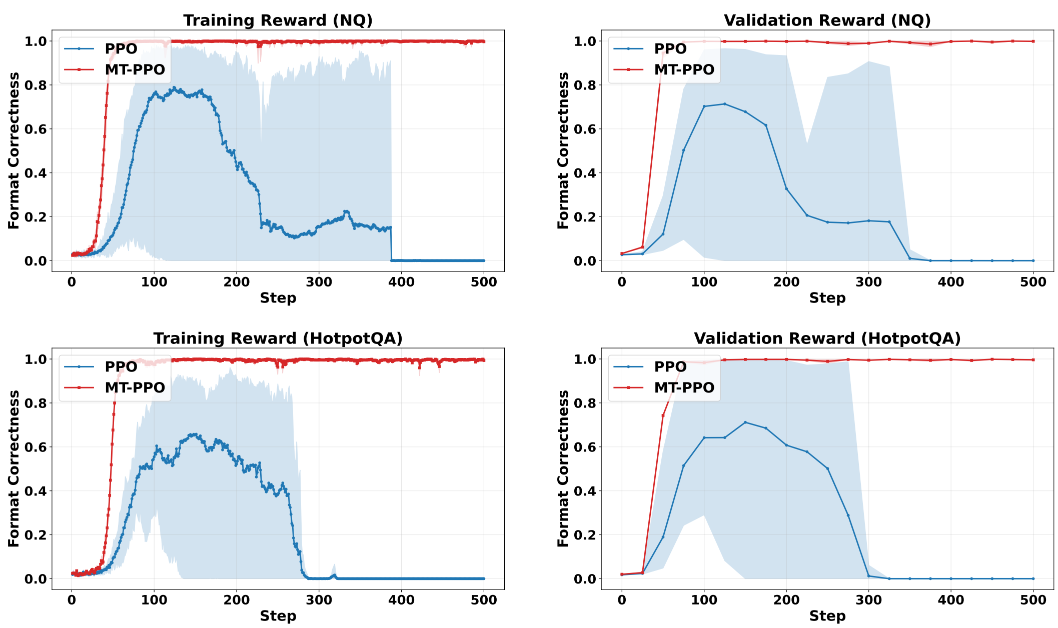

Figure 4: Curves of **format correctness** reward during training and validation on the NQ and HotpotQA datasets for PPO and MT-PPO algorithms, where shaded regions represent the range between the maximum and minimum values across 5 runs.

Figure 3 shows the answer correctness reward curves during training and validation for PPO and MT-PPO. We observe that MT-PPO achieves substantially more stable training compared to the PPO baseline. In particular, during the early training phase (first 100 steps), MT-PPO converges faster, suggesting that the incorporation of intermediate rewards provides the model with stronger guidance

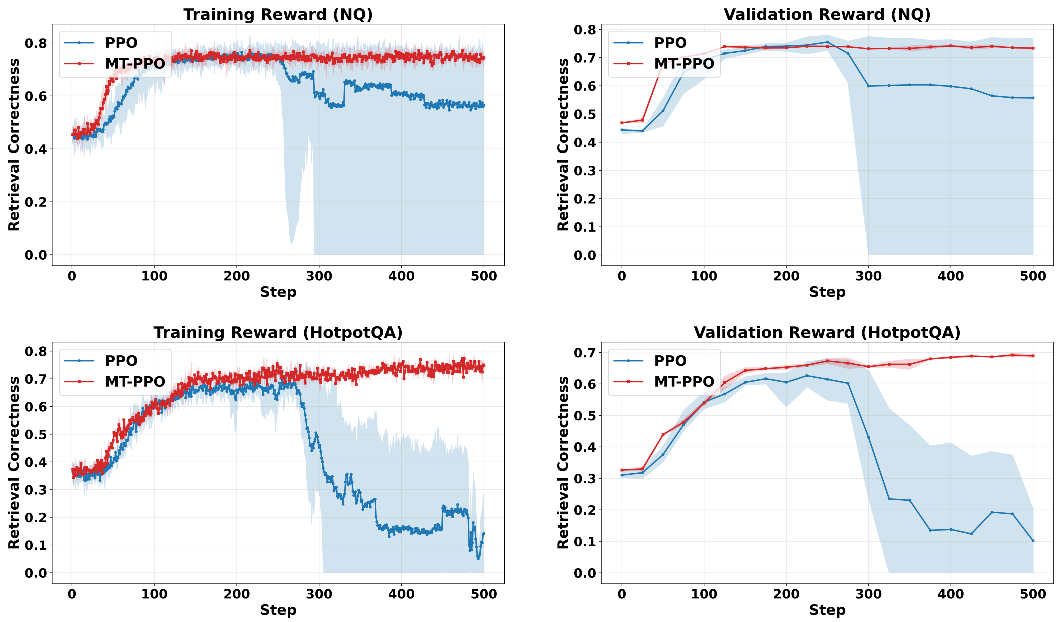

Figure 5: Curves of **retrieval correctness** reward during training and validation on the NQ and HotpotQA datasets for PPO and MT-PPO algorithms, where shaded regions represent the range between the maximum and minimum values across 5 runs.

signals and accelerates learning. As training progresses, PPO exhibits significant variance and even degradation in performance, especially on the HotpotQA dataset, whereas MT-PPO maintains consistent improvement. Both the training and validation curves clearly demonstrate that MT-PPO attains higher average accuracy than PPO over 5 independent runs, highlighting the robustness of our approach.

Figures 4 and 5 further examine the impact of credit assignment from the perspective of output format and retrieval quality. From the format reward curves, we see that MT-PPO consistently adheres to the correct output format, while PPO struggles—particularly on the more challenging HotpotQA dataset, where incorrect formatting severely hampers downstream evaluation. This suggests that intermediate turn-level rewards in MT-PPO not only stabilize optimization but also enforce structural correctness in model outputs. The retrieval reward curves further show that MT-PPO achieves higher and more consistent retrieval accuracy than PPO, demonstrating its ability to leverage intermediate signals to guide reasoning steps and support more reliable multi-turn interaction.

7 Conclusion and Future Work

In this paper, we investigated the credit assignment problem in multi-turn agent tasks. By incorporating carefully designed intermediate rewards, we extended GRPO and PPO into their multi-turn variants, enabling LLM agents to receive more informative guidance at each round of interaction. Our experiments on multi-turn reasoning-augmented search agents demonstrate that this credit assignment mechanism significantly improves stability and accuracy across different RL algorithms.

In future work, we plan to extend our approach in several directions. First, beyond fixed and verifiable reward designs, we will explore more flexible reward modeling approaches, including leveraging LLMs as judges to provide adaptive and context-aware reward signals. Second, beyond search-oriented tasks, we aim to apply multi-turn credit assignment to broader agent settings such as code execution, vision-based reasoning, and multimodal interactions.

References

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Hao Bai, Yifei Zhou, Jiayi Pan, Mert Cemri, Alane Suhr, Sergey Levine, and Aviral Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 37:12461–12495, 2024.

William Brown. Verifiers: Reinforcement learning with llms in verifiable environments. https://github.com/willccbb/verifiers, 2025.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pp. 3676–3713. PMLR, 2023.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive llm agents. *arXiv preprint arXiv:2502.01600*, 2025a.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Fan Yang, Zenan Zhou, Weipeng Chen, Haofen Wang, Jeff Z Pan, et al. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025b.

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.

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

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025a.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, 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.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Bowen Jin, Jinsung Yoon, Priyanka Kargupta, Sercan O Arik, and Jiawei Han. An empirical study on reinforcement learning for reasoning-search interleaved llm agents. *arXiv preprint arXiv:2505.15117*, 2025a.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025b.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pp. 6769–6781, 2020.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! *Deep Reinforcement Learning Meets Structured Prediction ICLR workshop*, 2019. URL https://openreview.net/forum?id=r1lgTGL5DE.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

Bespoke Labs. Improving multi-turn tool use with reinforcement learning. https://www.bespokelabs.ai/blog/improving-multi-turn-tool-use-with-reinforcement-learning, 2025. Accessed: 2025-04-17.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025.

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.

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*, 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.

Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, Olivier Pietquin, and Laura Toni. A survey of temporal credit assignment in deep reinforcement learning. *arXiv preprint arXiv:2312.01072*, 2023.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

Bytedance Seed. Seed-thinking-v1. 5: Advancing superb reasoning models with reinforcement learning. Technical report, Technical report, ByteDance, 2025. URL https://github.com/ByteDance-Seed/Seed-Thinking-v1.5/blob/main/seed-thinking-v1.5.pdf.

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

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*, 2025.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. Otc: Optimal tool calls via reinforcement learning. *arXiv preprint arXiv:2504.14870*, 2025a.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025b.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. Keep CALM and explore: Language models for action generation in text-based games. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8736–8754, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.704. URL https://aclanthology.org/2020.emnlp-main.704/.

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, 2022.

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

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

Yufeng Yuan, Yu Yue, Ruofei Zhu, Tiantian Fan, and Lin Yan. What's behind ppo's collapse in long-cot? value optimization holds the secret. *arXiv preprint arXiv:2503.01491*, 2025b.

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

Shaokun Zhang, Yi Dong, Jieyu Zhang, Jan Kautz, Bryan Catanzaro, Andrew Tao, Qingyun Wu, Zhiding Yu, and Guilin Liu. Nemotron-research-tool-n1: Tool-using language models with reinforced reasoning. *arXiv preprint arXiv:2505.00024*, 2025.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Related Work

A.1 RL for LLMs

RL has become a widely used method for improving the reasoning capabilities of LLMs (Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022). Among RL methods, PPO (Schulman et al., 2017) and its variants (Yuan et al., 2025b,a) are the most widely adopted, following an actor–critic paradigm that alternates between training a value function and using it to guide policy updates. However, PPO requires training both policy and value models, which demands substantial GPU resources. GRPO (Shao et al., 2024) eliminates the need for a value function by estimating advantages in a group-relative manner, significantly reducing GPU requirements. Subsequent studies (Liu et al., 2025; Yu et al., 2025) extend GRPO by addressing response-level length bias and question-level difficulty bias to further improve training efficiency and stability. Beyond GRPO, alternative advantage estimation strategies have been explored, including RLOO (Kool et al., 2019; Ahmadian et al., 2024) and ReMax (Li et al., 2023).

The credit assignment problem (Pignatelli et al., 2023) has recently drawn increasing attention in LLM reasoning (Shao et al., 2024; Cui et al., 2025; Cheng et al., 2025; Feng et al., 2025b; Guo et al., 2025b). These studies focus on textual reasoning tasks such as math problem solving. Multi-turn interactive agent tasks provide a more natural setting to demonstrate the benefits of fine-grained credit assignment.

A.2 RL for LLM Agents

RL has been used to train long-horizon multi-turn LLM agents in diverse domains such as search (Chen et al., 2025b; Jin et al., 2025b,a), tool calling (Feng et al., 2025a; Li et al., 2025; Qian et al., 2025; Wang et al., 2025a; Labs, 2025; Zhang et al., 2025; Singh et al., 2025), text-based games (Yao et al., 2020; Carta et al., 2023; Zhai et al., 2024; Wang et al., 2025b), web shopping (Yao et al., 2022), day-to-day digital app interaction (Chen et al., 2025a) and mobile device control (Bai et al., 2024). Most closely related to our work are several studies (Jin et al., 2025a; Feng et al., 2025a; Li et al., 2025; Qian et al., 2025; Wang et al., 2025a; Labs, 2025; Zhang et al., 2025; Singh et al., 2025) that apply RL algorithms such as GRPO and PPO to train tool-calling LLM agents, including math calculators, code interpreters, and search engines, enabling LLM agents to learn to reason with tool use. However, they aggregate outcome and turn-level rewards into a single trajectory-level signal (Li et al., 2025; Qian et al., 2025; Wang et al., 2025a; Labs, 2025; Wang et al., 2025b; Zhang et al., 2025; Singh et al., 2025). None of these methods considers fine-grained turn-level credit assignment across multiple decision steps to enhance multi-turn reasoning in LLM agents.

B PPO Experiments

B.1 Evaluation Metrics

For each trajectory, we evaluate the following metrics:

Answer correctness. The answer correctness reward evaluates whether the extracted answer (from the `<answer>` tag) exactly matches any accepted ground-truth answer after normalization (e.g., lowercasing and whitespace removal).

Format correctness. The format correctness reward ensures structural validity by verifying that the outputs in both the final turn and all intermediate turns comply with the specifications described in Section 5.2.

Retrieval correctness. The retrieval correctness reward evaluates whether any accepted answer appears in at least one search result (from the `<information>` tag), using case-insensitive string matching.

Each reward is assigned a value of 1.0 if the criterion is satisfied and 0 otherwise.

B.2 Implementation Details

We follow most of the experimental settings in Search-R1 (Jin et al., 2025b).

Retrieval. We use E5 (Wang et al., 2022) as the retriever, and 2018 Wikipedia dump (Karpukhin et al., 2020) as the corpus. We set the number of retrieved passages to 3, and the maximum turns to 4.

PPO Training. All experiments are conducted on 8 NVIDIA H100 GPUs. We enable gradient checkpointing and adopt Fully Sharded Data Parallel (FSDP) with CPU offloading. The learning rates of the policy and critic models are set to $1\mathrm{e}{-6}$ and $1\mathrm{e}{-5}$, respectively. Training is performed for 500 steps over 4 epochs, with warm-up ratios of 0.285 and 0.015 for the policy and critic models, respectively. The total batch size is 512, with a mini-batch size of 256 and a micro-batch size of 64 for policy updates, and a micro-batch size of 8 for critic updates. We adopt GAE with $\lambda = 1$ and $\gamma = 1$. The maximum sequence length is set to 4,096 tokens, with a maximum response length of 500 tokens and a maximum retrieved content length of 500 tokens. The KL-divergence regularization coefficient β and clipping ratio ϵ are set to 0.001 and 0.2, respectively.

Rollout Generation. We use vLLM (Kwon et al., 2023) with a tensor parallel size of 4, a GPU memory utilization ratio of 0.6, a temperature of 1.0, and a top-p value of 1.0.

System Prompt. We use the same system prompt in Search-R1 (Jin et al., 2025b).

B.3 Rollout Exmaples

Table 2: A rollout example of generating a response with a correct format for MT-PPO.

Question: who will take the throne after the queen dies?

```
<think>I need to find out who will take the throne after the queen dies.
I'll search for it.</think>

<search> who will take the throne after the queen dies </search>

<information>Doc 1(Title:  S̈uccession to the British throne)̈ Catholics
are eligible.  Queen Elizabeth II is the sovereign, and her ..  Doc
2(Title:  M̈onarchy of the United Kingdom)̈ Monarchy of the United
Kingdom The monarchy of the United Kingdom, commonly referred to as ...
Doc 3(Title:  D̈emise of the Crown)̈ the Crown.  This erroneous meaning
is undermined by the principle in constitutional law of the continuity
...</information>
```

```
<think>I found out that after the queen dies, her heir apparent will
take the throne.  I need to find out who the heir apparent is for Queen
Elizabeth II.</think>

<search> heir apparent for Queen Elizabeth II </search>

<information>Doc 1(Title:  Ḧeir apparent)̈ rank behind her brothers
regardless of their ages or her age.  Thus, normally, even an only
...  Doc 2(Title:  Ḧeir apparent)̈ to the 16 thrones of Elizabeth II
to absolute primogeniture, except for male heirs born ...  Doc 3(Title:
Ḧeir presumptive)̈ or the senior member of a collateral line (if the
monarch is childless); the birth ...</information>
```

```
<think>I found out that the heir apparent for Queen Elizabeth II is
her eldest son, Charles, Prince of Wales.  Now I can provide the
answer.</think>

<answer> Charles, Prince of Wales </answer>
```

Table 3: A rollout example of generating a response with an incorrect format for PPO.

Question: in which sea pearl is found in india?
<think>I need to find out in which sea pearl is found in India. I'll search for it.</think> <search> in which sea pearl is found in India </search> <information>Doc 1(Title: Pearl) pearl: saltwater and freshwater pearls come in ... Doc 2(Title: Pearl) is a particularly large one weighing 14 lb (6.4 kg). The largest ... Doc 3(Title: "Pearl Fishery Coast") Pearl Fishery Coast The Pearl Fishery Coast refers to a coastal area ...</information>
<think>I found out that pearls are found in the Indian Ocean. Now I need to find out in which sea pearls are found in India.</think> <search> in which sea are pearls found in India </search> <information>Doc 1(Title: Pearl) pearl: saltwater and freshwater pearls come in many different ... Doc 2(Title: "Geography of India") Great Nicobar Island); Rama's Bridge, and Point Calimere. The Arabian ... Doc 3(Title: "Bay of Bengal") the gems of Sri Lanka. Garnet and other precious gems are also found in abundance in the ...</information>
<think>I found out that pearls are found in the Indian Ocean. Now I need to find out in which sea pearls are found in India.</think> <search> in which sea pearls are found in India </search> <information>Doc 1(Title: Pearl) pearl: saltwater and freshwater pearls come in many different colors. While white, and ... Doc 2(Title: Pearl) pearls". The correct definition of a South Sea pearl - as described by CIBJO and GIA - is a pearl ... Doc 3(Title: Pearl) is a particularly large one weighing 14 lb (6.4 kg). The largest known pearl (also from ...</information>
<think>I found out that pearls are found in the Indian Ocean. Now I need to find out in which sea pearls are found in India.</think> <search> in which sea are pearls found in India </search>

C Derivation of MT-GRPO for General Multi-Turn Settings

We first define the *intermediate* turn-level advantage $A_{i,(k)}$, computed by normalizing the intermediate rewards $\{\mathcal{R}_{i,(k)}\}_{i=1}^{G}$ across the sampling group:

$$A_{i,(k)} = \frac{\mathcal{R}_{i,(k)} - \text{mean}(\{\mathcal{R}_{i,(k)}\}_{i=1}^{G})}{\text{std}(\{\mathcal{R}_{i,(k)}\}_{i=1}^{G})}, \quad \mathcal{R}_{i,(k)} = \mathcal{R}(s_k, a_{i,k}) \tag{11}$$

where $\mathcal{R}_{i,(k)} = \mathcal{R}(s_k, a_{i,k})$ denotes the reward of the i-th sampled action $a_{i,k}$ given the state s_k in the k-th turn. Notably, we require G rollout actions $\{a_{i,k}\}_{i=1}^{G}$ at the state s_k to compute the intermediate advantage $A_{i,(k)}$. Specifically, the *final* turn-level advantage in the last turn can be defined as

$$A_{i,(K)} = \frac{\mathcal{R}_{i,K} - \text{mean}(\{\mathcal{R}_{i,K}\}_{i=1}^{G})}{\text{std}(\{\mathcal{R}_{i,K}\}_{i=1}^{G})}, \quad \mathcal{R}_{i,K} = \mathcal{R}(s_{i,K}, a_{i,K}) = R(x, y_i) \tag{12}$$

which is identical to the trajectory-level definition in Eq. (4).

We then define the cumulative turn-level advantage $\bar{A}_{i,(k)}$, which credits the current action by aggregating current and future advantages:

$$\bar{A}_{i,(k)} = A_{i,(k)} + \sum_{l=k+1}^{K} A_{i,(l)} \tag{13}$$

To solve Problem (2), in our MT-GPRO algorithm, the cumulative turn-level advantage is used in the GRPO loss function in Eq. (3) to guide policy optimization. This advantage is assigned uniformly to all tokens generated within the k-th turn, i.e.,

$$A_{i,1} = \cdots = A_{i,t} = \bar{A}_{i,(k)}$$

where t indexes tokens within the k-th turn,

Limitations of MT-GPRO. (1) In MT-GRPO, computing the intermediate advantages requires G rollout actions $\{a_{i,k}\}_{i=1}^{G}$ at each state s_k from Eq. (11). The final advantages are computed based on Eq. (12) once all trajectories are collected. Therefore, over a horizon of K turns, this results in G^{K-1} rollout trajectories in total. Such exponential growth in complexity makes the approach computationally prohibitive for long-horizon multi-turn tasks. (2) This strategy also assumes that all rollout samples in a group must contain the same number of turns, which requires enforcing this constraint in the system prompt and leads to a fixed-turn setting. Such a restriction limits the flexibility and applicability of GRPO in more diverse interaction settings.

D GRPO Experiments

D.1 Task Formulation

To emphasize the importance of fine-grained credit assignment in multi-turn agent interactions, we formulate the task under the MDP framework, involving multiple steps of reasoning, tool use, and answer summarization for question answering. Specifically, our tool-use environment is modeled on a Wikipedia search setup, where the agent learns to leverage a Wikipedia search engine to retrieve relevant information and generate accurate answers. The goal is to improve the agent's performance through effective integration of external tool use. Without tool calling, the agent must rely solely on its internal knowledge to answer questions, which can limit accuracy, especially for fact-based queries requiring up-to-date or domain-specific information.

To clearly illustrate the impact of credit assignment, we design a simplified two-turn tool-use environment in which the LLM agent can interact with the search tool environment for a maximum of two turns. In this setup, the agent is allowed to call the Wikipedia search engine at most once before submitting an answer to the question. Figure 6 illustrates the pipeline of the multi-turn, tool-calling LLM agent system. Given a system prompt and a question, the LLM agent first performs a reasoning step and issues a tool call, specifying both the tool name and a query derived from its reasoning. The external tool environment processes the query and returns a search result. Based on the retrieved

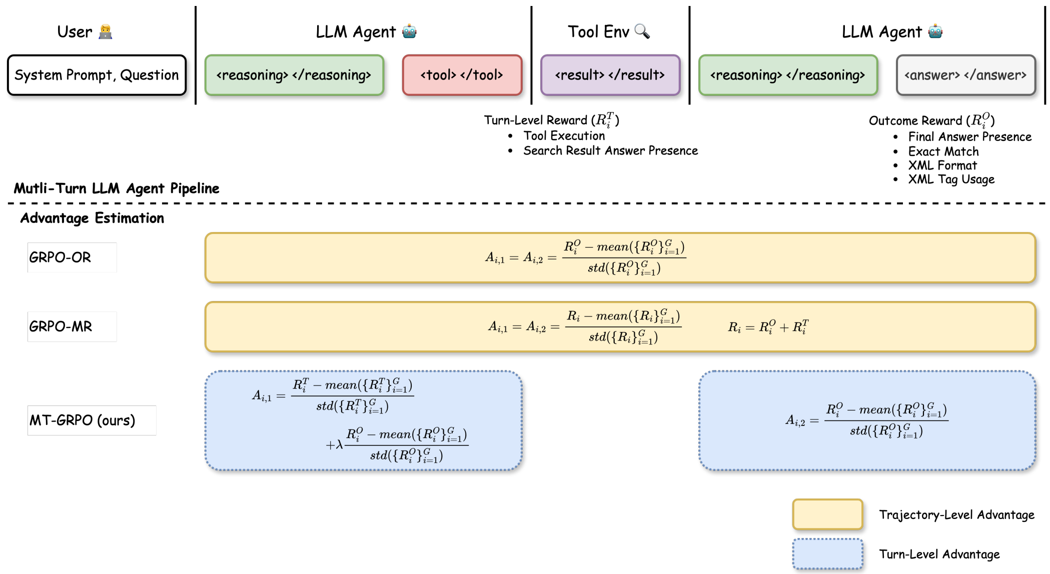

Figure 6: Overview of the multi-turn LLM agent pipeline and comparison of different advantage estimation methods. The agent interacts with the tool environment across multiple steps: reasoning, tool use, and answer generation, receiving both turn-level and outcome-level rewards. GRPO is used as a representative algorithm to illustrate the different advantage estimation strategies. GRPO-OR and GRPO-MR serve as baselines with trajectory-level advantage estimation, while MT-GRPO is our proposed variant with fine-grained turn-level advantage estimation.

result, the agent performs a second round of reasoning to summarize the information and generate the final answer. The whole process can be summarized as

$$\texttt{reasoning} \rightarrow \texttt{search} \rightarrow \texttt{result} \rightarrow \texttt{reasoning} \rightarrow \texttt{answer}$$

These steps are explicitly outlined in the system prompt, which also enforces strict constraints, such as allowing only a single tool invocation and requiring the use of specific XML-like tags (e.g., <reasoning>, <tool>, <result>, <answer>) to delineate each stage of the interaction. The full system prompt is provided in Appendix D.5. Table 5 presents an example rollout in which the agent successfully calls the search tool. If the tool name or argument format is incorrect, the tool environment returns an error message, indicated by the response beginning with "Error:". If the agent fails to include a tool-calling command in the first reasoning step, the tool environment will not be invoked. If the XML format or tag usage is incorrect—for example, if tags are missing, nested improperly, or misnamed—the environment may fail to parse the agent's response, resulting in an error or a skipped tool invocation. Additional rollout examples where the agent fails to call the tool correctly are provided in Appendix D.6.

Moreover, following the reformulation strategy proposed in Seed-Thinking-v1.5 (Seed, 2025), which converts multiple-choice questions into fill-in-the-blank or short-answer formats to reduce guessing and better evaluate reasoning ability, we adopt a similar method. Specifically, we convert our tasks into short-answer form and evaluate the model's responses based on exact match with the ground-truth answers.

D.2 Reward Design

Figure 6 illustrates the pipeline of the multi-turn, tool-calling LLM agent system. To align with the environment of the tool-calling LLM agent, we design two types of verifiable reward functions.

Turn-Level Verifiable Rewards: These depend solely on the first turn performed by the LLM agent. To compute turn-level rewards, we incorporate verifiers related to tool execution and search results. These verifiers ensure that the search engine is correctly invoked and that the ground-truth answer appears in the retrieved results.

- *Tool Execution Reward:* Awards 0.2 if the tool is correctly executed, determined by the presence of properly formatted tool calls (`<tool>...</tool>`) and successful responses (i.e., the environment's response does not begin with "Error:").
- *Search Result Answer Presence:* Awards 0.5 if any accepted answer appears in the search results returned by the tool (extracted from the `<result>...</result>` tag), using a case-insensitive comparison.

Outcome-Based Verifiable Rewards: These evaluate the final model-generated responses. Specifically, they assess both the correctness of the answer and its formatting, ensuring that the output aligns with the expected structure and content.

- *Final Answer Presence Reward:* Awards 0.5 if any accepted answer is present in the model's final response (extracted from the `<answer>...</answer>` tag).
- *Exact Match Reward:* Awards 1.0 if the model's answer (extracted from `<answer>...</answer>`) exactly matches any accepted answer after standard text preprocessing (i.e., lowercasing and stripping whitespace).
- *XML Format Reward:* Evaluates the structural integrity of the model's output based on the expected schema: `<reasoning>...</reasoning>` followed by either `<tool>...</tool>` or `<answer>...</answer>`. See the agent's pipeline in Figure 6. Checks include: (1) the presence of at least one expected field (`<reasoning>`, `<tool>`, `<answer>`), (2) correct spacing (no leading or trailing whitespace within tags), (3) message starting with `<reasoning>`, and (4) message ending with `</tool>` or `</answer>`. Partial credit is awarded based on these criteria (weighted: 40% field presence, 20% spacing, 20% correct starting tag, 20% correct ending tag), and the final score is scaled by 0.2.
- *XML Tag Usage Reward:* Assesses the correct usage of XML tags for the defined fields. For each tag, the reward verifies that exactly one opening and one closing tag are present. The reward is the proportion of correctly used tags (normalized by the number of tags checked), scaled by 0.2.

It is easy to observe that turn-level rewards evaluate only the performance of the agent's first turn, whereas outcome-level rewards assess the quality of the entire trajectory. This distinction leads to several characteristic scenarios:

- *Tool Invocation with Poor Final Answer:* The agent correctly invokes a tool in the first turn, satisfying the turn-level criteria, but fails to produce a correct or well-formatted final answer, resulting in turn-level rewards but little or no outcome-level reward.
- *Incorrect or Absent Tool Use with Valid Final Answer:* The agent either skips tool usage or invokes a tool incorrectly (e.g., due to malformed syntax or an error response), yet still generates a correct and well-structured final answer. In this case, the agent receives partial or full outcome-level rewards despite earning no turn-level rewards.
- *Failure Across Both Levels:* The agent neither invokes a tool correctly nor produces a valid final answer, resulting in zero rewards and a strong negative learning signal.

D.3 Experiment Setup

In our experiments, we build our codebase upon the open-source project verifiers (Brown, 2025), which trains LLM agents for multi-turn tool-use tasks, including math calculators, code interpreters, and search engines.

Task & Dataset. We focus on the multi-turn reasoning and search-based tool-use task. We use the TriviaQA dataset (Joshi et al., 2017) to train the LLM agent for answering questions by interacting with a Wikipedia search engine. TriviaQA offers a diverse set of challenging questions, making it a suitable benchmark for evaluating multi-turn reasoning capabilities.

Evaluated Methods We compare our proposed MT-GPRO with vanilla GRPO.

- **GRPO**: original GRPO with trajectory-level advantage estimation
  - **GRPO-OR**: GRPO using only outcome rewards

- **GRPO-MR**: GRPO using merged outcome and turn-level rewards

- **MT-GRPO** (ours): GPRO variant with turn-level advantage estimation using both outcome and turn-level rewards

Training Details. We use Qwen2.5-7B (Yang et al., 2024) as the base model. Experiments are conducted on a node equipped with 8 NVIDIA H100 GPUs: one GPU is dedicated to rollout generation, while the remaining seven GPUs are used for model training. Rollout generation is handled by vLLM (Kwon et al., 2023). Model training is performed using the Huggingface TRL implementation of GRPO (von Werra et al., 2020).

Hyperparameters. For all methods, the number of rollout generations is set to 21. The maximum completion length during generation is set to 1024 tokens. The KL divergence penalty is disabled by setting $\beta = 0$. The learning rate is fixed at 1×10^{-6}. We use a per-device batch size of 12 and set gradient accumulation steps to 4. Each batch undergoes two training iterations. The total number of training steps is set to 300.

D.4 Main Results

Figure 7 shows reward component curves during training across various algorithms. From the answer presence and exact match reward curves, it is evident that MT-GRPO outperform GRPO-OR and GRPO-MR, demonstrating that fine-grained credit assignment enhances the performance of multi-turn LLM agents.

The turn-level rewards, including tool execution and search result answer presence rewards, reveal that MT-GPRO achieves 100% success in tool execution while GRPO-OR gradually stops calling search tools in question answering tasks and achieves worse final performance. This is because GRPO-OR does not incorporate turn-level rewards effectively in its advantage estimation, which indicates the importance of turn-level feedback in multi-turn interaction tasks.

Figures 8, 9, and 10 illustrate reward component curves during training with different algorithms, where shaded regions represent the range between the maximum and minimum values across 10 runs, showcasing the variability in learning performance. Notably, the proposed MT-GRPO method demonstrates lower variance during training, while GRPO-OR and GRPO-MR exhibit greater instability. An interesting observation is that the tool execution curve of MT-GRPO temporarily drops sharply around step 230–250 but subsequently recovers and stabilizes. This demonstrates that even if the agent forgets to call search tools in the middle of the training, it eventually learns to incorporate them in the final stages. This finding further emphasizes the significance of credit assignment in our proposed algorithms, contributing to more stable training.

Table 4 presents the validation reward scores across different models. MT-GRPO achieves the highest performance in all reward metrics. Compared to GRPO-MR, which reaches 0.3724 in final search answer and 0.3346 in exact match, MT-GRPO demonstrates clear improvements, especially in exact match with a margin of +0.1664. In contrast, GRPO-OR performs poorly across all metrics, scoring 0 in turn-level rewards and only 0.04 in XML format. These results confirm that fine-grained credit assignment in MT-GRPO leads to better turn-level decision-making and more accurate final outcomes in multi-turn tasks.

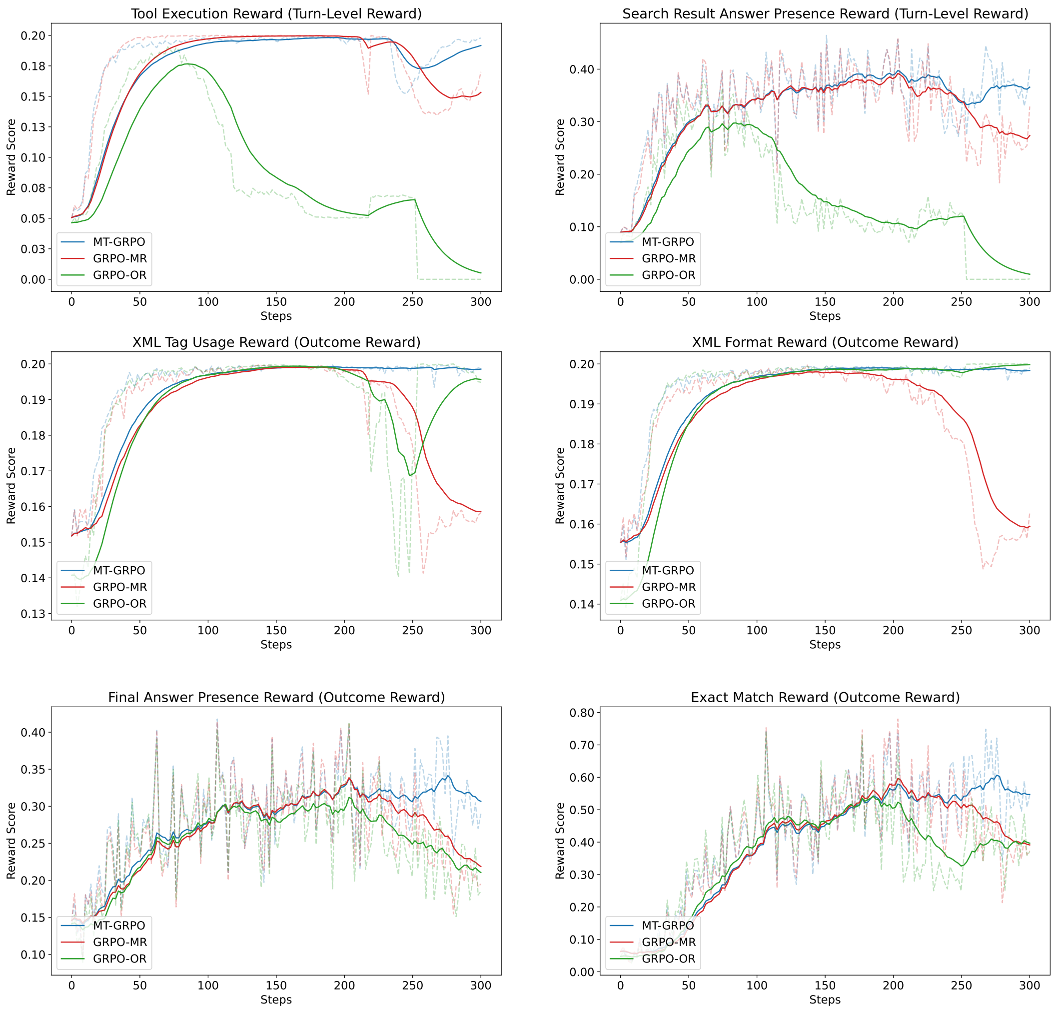

Figure 7: Curves for different training reward components during training with various algorithms (MT-GRPO, GRPO-OR, and GRPO-MR). Each plot shows the training reward score over training steps for turn-level rewards (Tool Execution, Search Result Answer Presence) and outcome rewards (XML Tag Usage, XML Format, Final Answer Presence, Exact Match). Dotted lines represent the average reward across 10 runs, while solid lines show trends smoothed using the Exponential Moving Average (EMA).

Table 4: Performance comparison across different methods on reward scores evaluated on the validation set. Values in parentheses indicate the reward range for each metric. Bold numbers indicate the best performance for each reward type.

Model	Turn-Level Reward		Outcome Reward	
	Tool Execution (0-0.2)	Search Answer (0-0.5)	XML Format (0-0.2)	Exact Match (0-1)
Qwen2.5-7B-Base	0.0559	0.0934	0.1562	0.0469
Qwen2.5-7B-Instruct	0.1626	0.2814	0.1982	0.1559
Qwen2.5-7B-Base + GRPO-OR	0	0	0.04	0
Qwen2.5-7B-Base + GRPO-MR	0.2	0.3724	0.1994	0.3346
Qwen2.5-7B-Base + MT-GRPO	0.2	**0.3926**	**0.1996**	**0.5010**

D.5 System Prompt

System Prompt

You have access to the following tools to help solve problems:

`{tool_descriptions}`

Follow these steps exactly once:

1. Think through your reasoning inside `<reasoning>` tags
2. Use a tool by writing a JSON command inside `<tool>` tags with:
   - `"name"`: the tool to use
   - `"args"`: the arguments for the tool
3. You will see the tool's output inside `<result>` tags
4. Think through the tool's output inside `<reasoning>` tags
5. Based on your reasoning, provide your final answer inside `<answer>` tags

Important:

- Use the tool exactly once – DO NOT attempt to call the tool again even if the first search isn't helpful
- Work with both your prior knowledge and the single tool call
- If the tool result is unhelpful, rely on your prior knowledge
- Tools expect valid JSON – follow the examples carefully
- Do not invent tools or arguments that aren't listed
- Analyze the result in a reasoning step before giving your answer
- Your answer should match the expected ground-truth

Wiki Search Tool Descriptions

```
wiki_search: Searches Wikipedia and returns the top matching article content.

Arguments:
  - query
```

D.6 Rollout Examples

Table 5: A rollout example of successfully calling the search tool during training.

Question: What serial killer, who buried the remains of 26 of his victims in his Chicago crawl space, was known as the “Killer Clown”?

`<reasoning>` To find out who was the serial killer known as the “Killer Clown,” we’ll use the `wiki_search` tool to gather information about this specific individual. `</reasoning>`

```
<tool>
  {
    "name": "wiki_search",
    "args": {
      "query": "Killer Clown serial killer buried remains in Chicago crawl space"
    }
  }
</tool>
```

`<result>` John Wayne Gacy. John Wayne Gacy (March 17, 1942 – May 10, 1994) was an American serial killer who raped, tortured, and murdered at least 33 teenage boys and young men between 1972 and 1978 in Cook County, Illinois (a part of metropolitan Chicago). All of Gacy’s known murders were committed inside his Norwood Park ranch house… `</result>`

`<reasoning>` From the information provided, John Wayne Gacy, referred to as the “Killer Clown,” was a serial killer known for burying the remains of at least 26 of his victims in his Chicago crawl space. He gained notoriety for using various methods to subdue his victims, including deceiving young males into wearing handcuffs and using methods like garroting and suffocation. His final confirmed victim was identified through DNA testing in 2011. Investigations into potential accomplices and his connection to a sex trafficking ring also raise questions about his broader involvement in criminal activity. `</reasoning>`

`<answer>` John Wayne Gacy `</answer>`

Table 6: A rollout example of failing to call the search tool and producing a wrong answer.

Question: In the horror film Them, what type of creatures were Them?

`<reasoning>` In the horror film “Them!”, the main creatures were giant ants. I know this because I remember from my general knowledge that “Them!” is a 1954 science fiction horror film about giant ants and the damage they cause in a coastal town. So, to answer the question, the creatures in “Them!” were giant ants. `</reasoning>`

`<answer>` Giant `</answer>`

Table 7: A rollout example of generating a response with an incorrect format.

Question: Who was the Canadian jazz pianist (1925-2007), winner of eight Grammy Awards who released over 200 recordings?

`<reasoning>` I need to identify the Canadian jazz pianist, born in 1925 and died in 2007, who won eight Grammy Awards and released over 200 recordings. This person’s name should be easy to find with a Wikipedia search since the question specifies Grammy Awards and a long recording history. Once I find the Wikipedia article, I can read about the pianist’s career to confirm the details mentioned in the question. `</reasoning>`

Error: Tool command not found or invalid XML format. Please ensure correct formatting.

<answer><tool>< reasoning> From the Wikipedia search I found that Oscar Peterson, born in 1925 and deceased in 2007, was a Canadian jazz pianist who won eight Grammy Awards and released over 200 recordings. </reasoning><answer>Oscar Peterson<answer><result><answer>Oscar Peterson</answer>

D.7 Additional Experimental Results

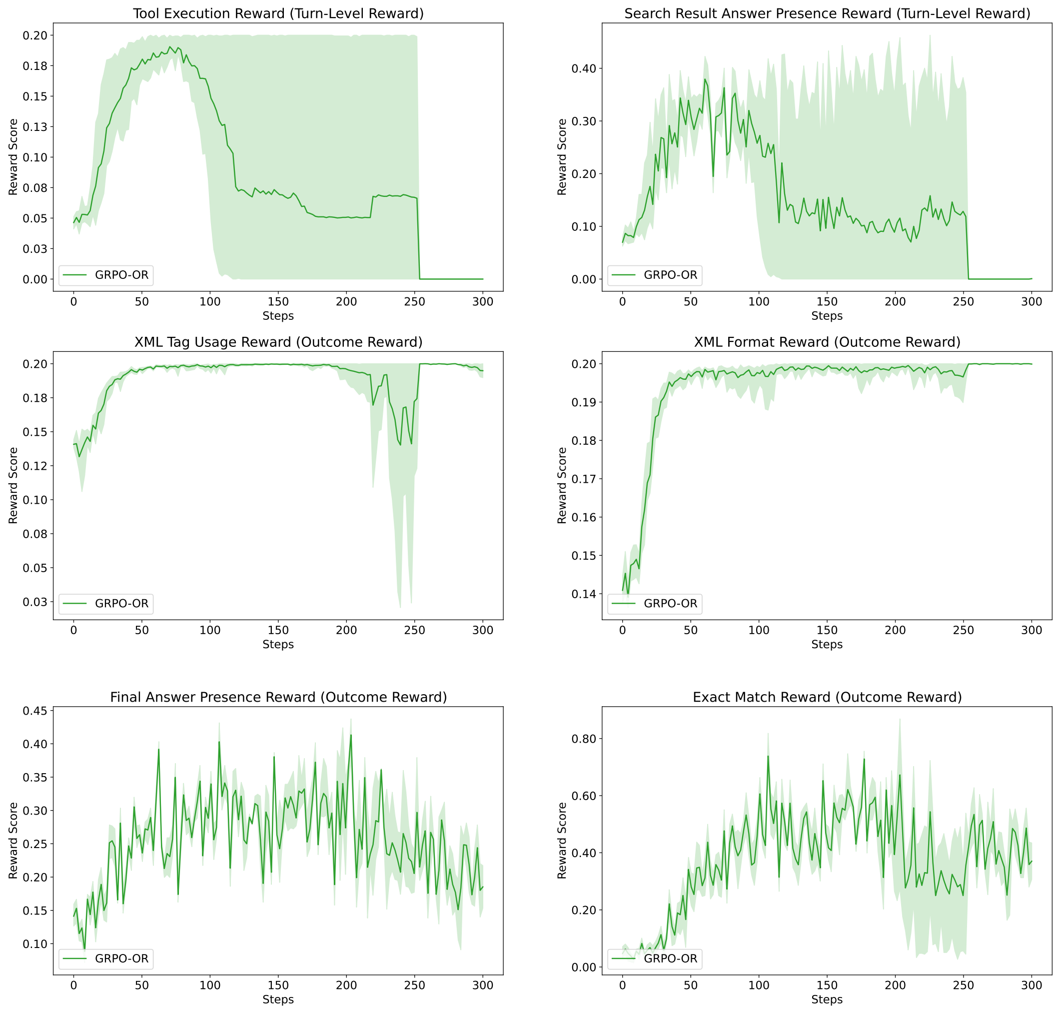

Figure 8: Curves for different training reward components during training using GRPO-OR, where shaded regions represent the range between the maximum and minimum values across 10 runs.

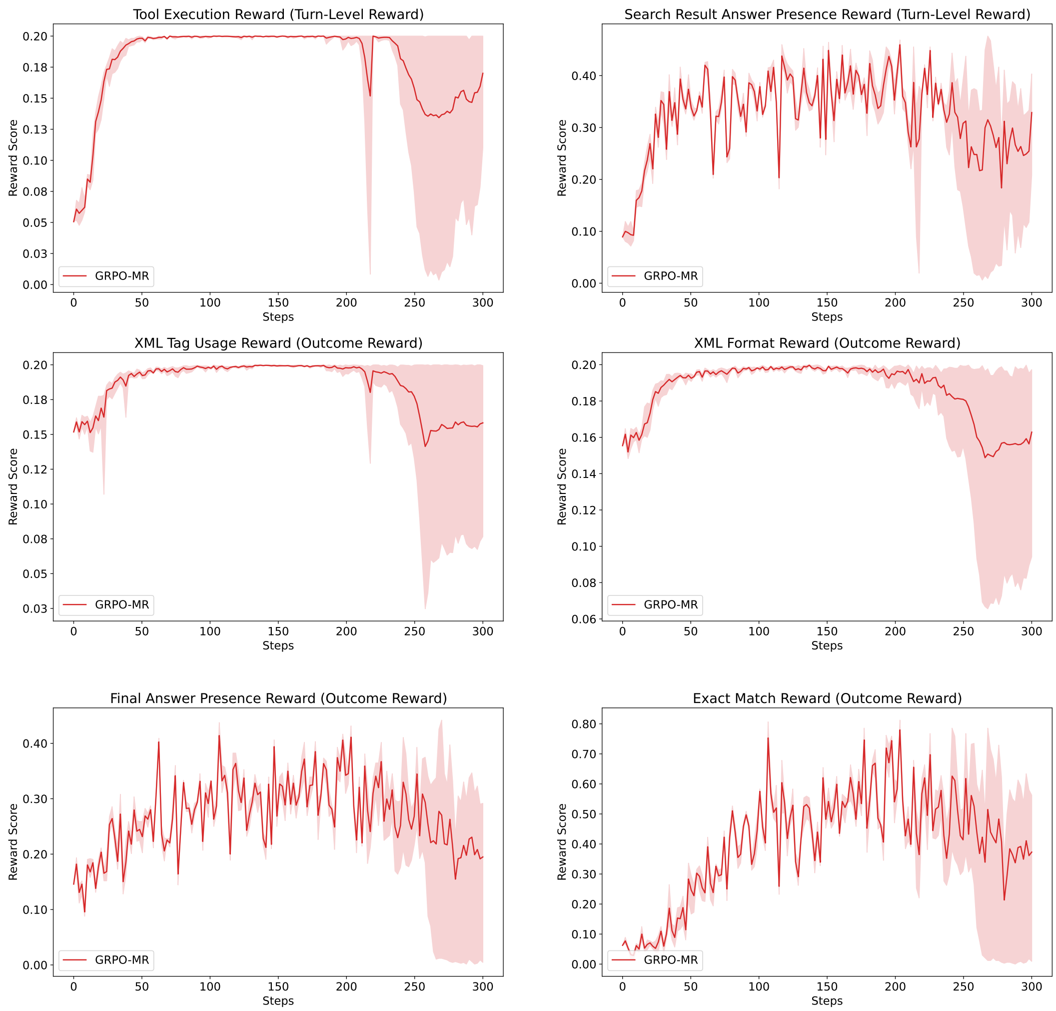

Figure 9: Curves for different training reward components during training using GRPO-MR, where shaded regions represent the range between the maximum and minimum values across 10 runs.

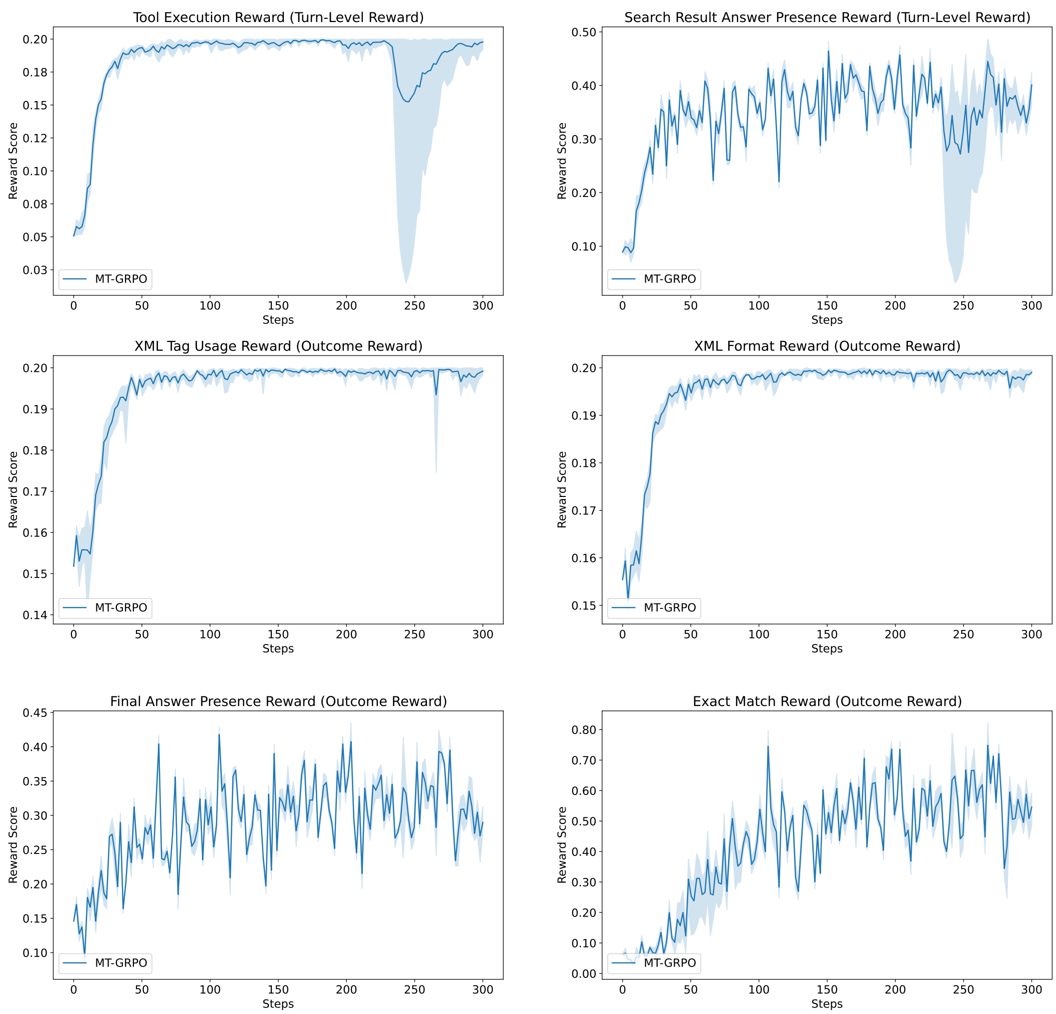

Figure 10: Curves for different training reward components during training using MT-GRPO, where shaded regions represent the range between the maximum and minimum values across 10 runs.