Align as Act: Innovations-Based Reward Decomposition for LLM Agents

Anonymous ACL submission

Abstract

In this paper, we propose a new step reward design for large language model (LLM) agents based on innovations: an information-theoretically optimal reward decomposition. Our method defines an intrinsic reward as the log energy of the innovation signal, measuring the amount of net new information provided by each step relative to the agents past interaction history. This intrinsic reward can be computed directly from embedding vectors in the semantic vector space via Gram-Schmidt orthogonalization, without training any additional reward or world models. The proposed design enables the agent to suppress redundant steps by definition and concentrate on progress-aligned transitions. Our method can be seamlessly integrated into standard on-policy objectives (e.g., PPO, GRPO) by augmenting rewards during training. Experiments on agentic benchmarks of ALFWorld and WebShop demonstrate that our method consistently improves performance and sample efficiency over strong RL baselines, suggesting that innovations-driven intrinsic rewards provide a practical principle for credit assignment in LLM agent RL post-training.

1 Introduction

Large language models (LLMs) are evolving into autonomous agents that interact with environments to perform human-like tasks such as household management, scientific discovery, and gaming (Yao et al., 2022b). Among various research directions for building effective LLM agents, we focus on reinforcement learning (RL)-based post-training, which has become a dominant approach following recent advances in policy optimization such as PPO (Schulman et al., 2017a), GRPO (Shao et al., 2024).

For RL training to succeed, good reward signals are essential. From an RL perspective, an agent aligns its internal belief about a task based on rewards received at each interaction step (Wang et al., 2025c). However, most verifiable agentic tasks provide rewards only at the end of the process, indicating success or failure. A simple remedy is to assign the final reward to every intermediate step, but this fails to distinguish important steps from redundant ones; some actions contribute more to task completion while others less. To address this sparse reward problem, recent work has focused on process reward models (PRMs) that evaluate each step individually. Several methods have been proposed to this end (Wang et al., 2024; Xi et al., 2025; Wang et al., 2025a; Feng et al., 2025), but they rely on extra Monte-Carlo simulations, additional model learning or hash-based in-data statistics matching.

In this paper, we approach the step reward design problem from a fundamentally different perspective grounded in information theory and the method of *innovations* (Wiener, 1958). Our development is based on our new geometric interpretation of the amount of information provided by observations, stating that the logarithm of the innovations norm captures the net new information provided by an observation. Our intrinsic reward accelerates task alignment by promoting task-aligned information-bearing steps and suppressing redundant ones. Our method is simple to implement and compatible with standard on-policy optimizers (PPO, GRPO), without requiring any additional model training or in-data statistics search; it relies solely on Gram-Schmidt orthogonalization of extracted embedding vectors. Experiments on representative agentic tasks of ALFWorld (Shridhar et al., 2020) and WebShop (Yao et al., 2022a) demonstrate its superior performance.

2 Preliminaries

Problem Setup In this paper, we consider a partially-observable sequential decision-making problem of an LLM agent. The agent is given a task $\mathcal{T}$. To accomplish $\mathcal{T}$, at each timestep t,

the agent interacts with the environment, performing an action a_t and receiving an observation o_t induced by the action a_t. Here, a_t and o_t are described in natural language. The agent's policy $\pi_\theta(a_t \mid h_t)$ is conditioned on the interaction history $h_t = (\mathcal{T}, a_1, o_1, ..., o_{t-1})$. The reward is sparse so that the agent receives a task success or failure verifier at the end of the interaction, and the goal is to maximize the task success rate.

Innovations Processes The method of innovations was introduced for time series by Wiener (Wiener, 1958) and has been used widely in signal processing from Kalman filtering and to signal detection (Kalman, 1960; Kailath, 1970; Sung et al., 2006, 2008). It can be explained readily under the linear-estimation framework. Suppose that we have a sequence of observation vectors $y_1, y_2, \cdots, y_n$ in some Hilbert space $\mathcal{H}$. The innovation e_t provided by y_t at time step t is defined as

$$e_t := y_t - \hat{y}_{t|t-1} \quad \text{s.t.} \quad e_t \perp e_k, \quad \forall k < t, \qquad (1)$$

where $\hat{y}_{t|t-1}$ is the optimal (in L_2-error sense) prediction of y_t based on all previous observations $y_1, \cdots, y_{t-1}$. Hence, the innovation e_t is the solely new information contained in y_t unpredictable from the previous observations. There exists a transform between the sequence $(y_1, \cdots, y_t)$ and the sequence $(e_1, \cdots, e_t)$ that is causal and causally invertible for all t (Kailath et al., 2000). Thus, the collection of innovations contains all the information of the original observation vectors without any loss but the innovations form a sequence of orthogonal vectors: Obtaining the innovations process can be viewed as the familiar *Gram-Schmidt orthogonalization*, as shown in Fig. 1.

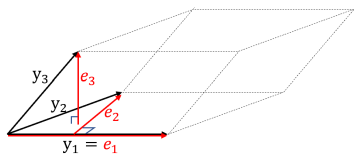


Figure 1: Innovations and observation parallelotope

3 An Information-Theoretic Approach

The main question of this paper is how to quantify the contribution of each step t to task success, when task success or failure verifier is given at the end of each episode. To answer this question, consider a mathematical model for the LLM agent solving the given task $\mathcal{T}$ through an interactive process with the environment:

$$\mathcal{T} \xrightarrow{\text{LLM}} a_1 \xrightarrow{\text{Env.}} o_1 \xrightarrow{\text{LLM}} a_2 \xrightarrow{\text{Env.}} o_2 \cdots \xrightarrow{\text{Env.}} o_T,$$

where T is the final step of the episode, which is either the earliest time step of task success or the maximum allowed time steps in case of failure.

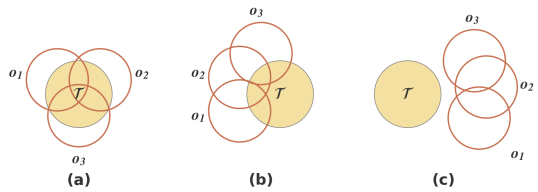


Figure 2: Task information and observations

From an information-theoretic perspective, one can view solving the task $\mathcal{T}$ requires relevant information $H(\mathcal{T})$ about the task $\mathcal{T}$. (We will use the term "information" and the notation "$H(\cdot)$" to denote both information amount and information itself for simplicity.) This necessary information is gathered by making sequential observations $o_1, o_2, \cdots$. When the full information $H(\mathcal{T})$ is obtained from $o_1, o_2, \cdots, o_T$, the task is solved. Otherwise, the task is failed. In this perspective, the actions are intermediate steps to acquire information-bearing observations. A good action a_t is an action that generates an observation o_t with large mutual information $I(\mathcal{T}; o_t)$. The relationship between the task information and the information of observations can be classified into three cases, as shown in Fig. 2. In the first case, as shown in Fig. 2 (a), the total gathered information $H(o_1, \cdots, o_T)$ from observations covers the task information $H(\mathcal{T})$, i.e., $I(\mathcal{T}; o_1, \cdots, o_T) = H(\mathcal{T})$, and the task is solved based on sufficient information from observations. The second and third are the cases in which the total observation information $H(o_1, \cdots, o_T)$ does not fully cover the task information $H(\mathcal{T})$, i.e., $I(\mathcal{T}; o_1, \cdots, o_T) < H(\mathcal{T})$, as shown in Figs. 2 (b) and (c), and the task failure occurs.

For a successful episode, a natural step-wise contribution is the task-relevant information newly provided by each observation, so-called information gain:

$$I(\mathcal{T}; o_t \mid o_1, \ldots, o_{t-1}).$$

However, computing this quantity requires an explicit random variable $\mathcal{T}$ that captures the structure of a complex agentic task, which is generally intractable beyond restricted settings such as search

or question answering (Bae et al., 2025; Auzina et al., 2026; Wang et al., 2026; Liang et al., 2026).

Our key insight to circumvent this difficulty and extend to general sparse agentic tasks is as follows: When the task is successful, we have $H(o_1, o_2, \cdots, o_T) \supseteq H(\mathcal{T})$ so that $I(\mathcal{T}; o_1, \cdots, o_T) = H(\mathcal{T})$, and the interaction stops at the earliest time step in case of task success, yielding minimal T for this inclusion. Then, we can approximate $H(\mathcal{T}) \approx H(o_1, o_2, \cdots, o_T)$, although there exists some non-overlapping region $H(o_1, o_2, \cdots, o_T) \backslash H(\mathcal{T})$, as illustrated in Fig. 2(a). Under this assumption, the information gain can be approximated as $I(\mathcal{T}; o_t | o_1, \cdots, o_{t-1}) \approx H(o_t | o_1, \cdots, o_{t-1})$. This enables us to circumvent the difficulty in setting an explicit variable $\mathcal{T}$ for complicated agentic tasks and to solely rely on the observations and success verifier. Note that the net contribution $H(o_t | o_1, \cdots, o_{t-1})$ of o_t is simply the information amount provided by **innovation** at time t, described in Sec. 2.

This suggests that rewards in a successful multi-turn trajectory can be decomposed by the new information revealed at each observation. We now make this idea computable by representing language observations as vectors and measuring their innovations.

4 Information Gain via Innovations

We first view observation variables as L^2 vectors, so that their covariance matrix is the Gram matrix of these vectors and Gram-Schmidt orthogonalization yields innovations.

Information Decomposition Let $y_1, \ldots, y_n$ be zero-mean, square-integrable observation random variables on a common probability space. With correlation $\langle y_i, y_j \rangle = \mathbb{E}[y_i y_j]$ as inner product, they can be viewed as elements of an L^2 inner-product space (Grimmett and Stirzaker, 2020), which instantiates the Hilbert-space in Sec. 2. Then, applying Eq. (1) to $y_1, \cdots, y_n$ yields innovations $e_1, \ldots, e_n$ such that $\langle e_t, e_k \rangle = 0$ for all $k < t$ (Kailath et al., 2000).

Now consider the amount of information that $y_1, \cdots, y_n$ provide. For continuous observation variables with covariance matrix Σ, the maximum (self-)information, i.e., entropy is given by (Cover and Thomas, 2006)

$$H(y_1, \cdots, y_n) = \tfrac{1}{2} \log \left((2\pi e)^n \det(\Sigma) \right), \quad (2)$$

which is achieved in the Gaussian case. The key property of innovations is as follows: Due to LDL^T decomposition of Σ (Golub and Van Loan, 1996), the total entropy (2) can be decomposed as (Kailath et al., 2000; Sung et al., 2006)

$$H(y_1, \cdots, y_n) = \frac{1}{2} \sum_{t=1}^{n} \log(2\pi e R_{e,t}), \quad (3)$$

where $R_{e,t} = \mathbb{E}[e_t^2]$ is the variance of innovation e_t. In other words, we have

$$H(y_t | y_1, \cdots, y_{t-1}) = \tfrac{1}{2} \log(2\pi e R_{e,t}). \quad (4)$$

This yields a variance-based proxy for the new information provided by each innovation, which motivates our step-wise reward signal in Sec. 5.

Geometric Interpretation Consider the parallelotope generated by edge vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^n$. Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$. Then, the volume of the parallelotope is given by $|\det(\mathbf{X})|$ (Strang, 2006). The Gram matrix $\mathbf{G}$ of vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n$ is defined as $\mathbf{G}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$, i.e., $\mathbf{G} = \mathbf{X}^\top \mathbf{X}$. Then, $\det(\mathbf{G}) = (\det(\mathbf{X}))^2$ (Horn and Johnson, 2012); hence $\sqrt{\det(\mathbf{G})}$ equals the parallelotope volume. Now, consider zero-mean random variables $y_1, \cdots, y_n$ and consider the *parallelotope* constructed by random variables $y_1, \cdots, y_n$ as vectors embedded in the higher-dimensional vector space of random variables with correlation as inner product. The Gram matrix of $y_1, \ldots, y_n$ is then the covariance matrix Σ with $\Sigma_{ij} = \mathbb{E}[y_i y_j]$. Hence, *the entropy formula (2) can be interpreted as the logarithm of the volume of the parallelotope constructed by observation vectors* $y_1, \cdots, y_n$.

The innovations-based information decomposition eq. (3) is now clear under this geometric interpretation. The volume of the parallelotope constructed by vectors $y_1, \cdots, y_n$, i.e., $\sqrt{\det(\Sigma)}$ is nothing but the product of the lengths of orthogonal vectors $e_1, \cdots, e_n$ obtained from Gram-Schmidt orthogonalization of $y_1, \cdots, y_n$, i.e., $\sqrt{\det(\Sigma)} = \prod_{t=1}^{n} ||e_t|| = \prod_{t=1}^{n} \sqrt{R_{e,t}}$, as shown in Fig. 1, yielding the geometric equivalence of (2) and (3).

5 Proposed Method: Align as Act

Motivated by this, we present Align as Act (AAA), an intrinsic-reward framework for training LLM agents that decompose sparse, trajectory-level task returns into step-wise innovation signals. At a high level, AAA operates as follows: At each step, the agent embeds the current observation in form of

language into a semantic vector, computes the innovations, and assigns an intrinsic reward based on the innovations. Hence, this reward encourages actions that reveal new, non-redundant information about the environment.

5.1 Semantic Observation Vectors

AAA computes innovations over language observations by mapping each observation to a continuous semantic vector. For a rollout $\tau = (\mathcal{T}; a_1, o_1, \ldots, a_T, o_T)$ sampled from $\pi_{\theta_{\text{old}}}$, we extract the LLM hidden vector $\mathbf{h}_t$ at the user-to-assistant boundary after the serialized observation o_t and before the next thought/action is generated. This uses the agent's own pretrained Transformer as the semantic encoder; extraction details are given in Appendix B.4.

For efficiency, we project the high-dimensional hidden vector to

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t \in \mathbb{R}^d,$$

using a fixed random projection $\mathbf{P}$, which approximately preserves pairwise geometry by the Johnson–Lindenstrauss property (Dasgupta and Gupta, 2003). We then apply Gram–Schmidt orthogonalization to $\{\mathbf{u}_t\}_{t=1}^T$ to obtain innovation vectors $\{\mathbf{e}_t\}_{t=1}^T$, which define the step-wise intrinsic reward. Details of the innovation computation are provided in Algorithm 1 and Appendix B.6.

5.2 Rewards and Advantages

Eq. (4) states that the net new information is proportional to one half of the logarithm of innovation norm square. However, the innovation norm can be smaller than 1 and this makes the logarithm negative, which is not desirable. So, we set the intrinsic reward at step t as follows:

$$r_t^{\text{innov}} = \tfrac{1}{2}\log(1 + \|\mathbf{e}_t\|^2), \quad t = 2, \cdots, T. \quad (5)$$

Then, r_t^{innov} is always nonnegative and preserves the monotone increasing property with logarithm scale. When $t = 1$, there is no subspace to be projected onto for Gram-Schmidt procedure, and hence r_1^{innov} is large only because it is in the first. This may distort the overall operation, so we replace r_1^{innov} with the mean of the remaining steps.

Our intrinsic reward can be applied to PPO (Schulman et al., 2017b), but this requires an additional critic model learning. Hence, we here focus on the critic-free policy optimization, e.g., GRPO, GiGPO (Shao et al., 2024; Feng et al., 2025).

Algorithm 1 Align as Act

Require: Current policy $\pi_{\theta_{old}}$, projection matrix $\mathbf{P}$, threshold κ

For a given task, run rollouts with $\pi_{\theta_{old}}$.
for each rollout τ^i **do**
  Initialize empty basis $\mathbf{Q} \leftarrow \emptyset$
  Initialize intrinsic rewards $r_t^{\text{innov},i} \leftarrow 0$
  if τ^i is successful **then**
    for $t = 1$ to T_i **do**
      Take action $a_t^i \sim \pi_{\theta_{old}}$, observe o_t^i,
      Extract observation vector $\mathbf{h}_t^i$
      Project $\mathbf{u}_t^i = \mathbf{P}\mathbf{h}_t^i$
      Compute $\mathbf{e}_t^i = \mathbf{u}_t^i - \mathbf{Q}\mathbf{Q}^\top \mathbf{u}_t^i$
      $r_t^{\text{innov},i} \leftarrow \frac{1}{2}\log(1 + \|\mathbf{e}_t^i\|^2)$
      if $\|\mathbf{e}_t^i\| > \kappa$ **then**
        $\mathbf{Q} \leftarrow \left[\mathbf{Q}; \frac{\mathbf{e}_t^i}{\|\mathbf{e}_t^i\|}\right]$
      end if
    end for
    Scale intrinsic rewards by (6)
  end if
end for
Normalize intrinsic rewards by (8)

Consider the design of actual step-wise rewards for GRPO. GRPO runs a set of G rollouts for the same task. We assume that a rollout receives the trajectory return 1 for success and 0 for failure. For a successful rollout, we distribute the total return of 1 across steps, with each step's share being proportional to its contribution as defined in (5). Thus, the step reward at time t of a successful trajectory τ^i is given by

$$r_t^i = \frac{r_t^{\text{innov},i}}{\sum_{k=1}^{T_i} r_k^{\text{innov},i}}, \qquad t = 1, 2, \ldots, T_i. \quad (6)$$

where T_i is the length of the rollout τ^i (now the rollout index i is included in the innovation reward). In this way, every successful rollout has the same total return of one and a shorter successful rollout naturally receives higher step rewards on average.

When a rollout τ^i failed in the task, on the other hand, a step with a large innovation does not necessarily correspond to a step with a large contribution to task success; a large innovation step may correspond to a big step towards an orthogonal direction to task success. Hence, we cannot use (5). The safest way to design step rewards in this case is to simply set them as

$$r_t^i = 0, \ \forall t = 1, 2, \ldots, T_i, \ \forall \text{ failed rollout } i. \quad (7)$$

Based on (6) and (7), we compute the group-relative step rewards as

$$\tilde{r}_t^i = \begin{cases} \max\left(0, \dfrac{r_t^i - \mu_G}{\sigma_G}\right) + \delta, & \text{if } \tau^i \text{ succeeds} \\ \dfrac{r_t^i - \mu_G}{\sigma_G}, & \text{otherwise} \end{cases} \tag{8}$$

where

$$\mu_G = \frac{1}{\sum_{i=1}^{G} T_i} \sum_{i=1}^{G} \sum_{t=1}^{T_i} r_t^i$$

$$\sigma_G = \sqrt{\frac{1}{\sum_{i=1}^{G} T_i} \sum_{i=1}^{G} \sum_{t=1}^{T_i} (r_t^i - \mu_G)^2}.$$

In (8), we ensure the normalized step rewards remain positive by adding a small constant $\delta > 0$ for successful rollouts, which is the only hyperparameter of our reward design. This adjustment addresses the potential imperfections in our approximation $H(\mathcal{T}) \approx H(o_1, o_2, \cdots, o_T)$. Specifically, as illustrated in Fig. 2(a), the set $H(o_1, o_2, \cdots, o_T) \backslash H(\mathcal{T})$ may be non-empty even when a task is successful. Consequently, the intrinsic innovation reward may not perfectly capture each step's contribution. By preventing these rewards from fluctuating below zero, we mitigate the risk of unstable policy learning that may arise from inconsistent or sign-flipping reward signals.

With $\tilde{r}_t^i$ in (8), we compute the advantage $\hat{A}_{i,t} = \sum_{k=t}^{T_i} \gamma^{k-t} \tilde{r}_k^i$ for action a_t^i. Here, we consider discounting because $\tilde{r}_t^i$ is positive for all steps for a successful trajectory and negative for all steps for a failed trajectory. Without proper discounting, early steps receive large advantages only because they are early in the sequence. This is an artifact of the group-relative computation and does not occur when we use the true advantage $Q(s_t, a_t) - V(s_t)$ due to the subtraction by the state value $V(s_t)$. We simply set $\gamma = 0$, considering that r_t^i is already a return share. With advantage $\hat{A}_{i,t}$, we optimize the policy π_θ with the objective of GRPO. Please refer to Appendix B.7 for details and application to GiGPO.

6 Experiments

6.1 Empirical Validation via Task Information Cones

Consider the paralleltope constructed by observation vectors $\mathbf{h}_t,\ t = 1, \cdots, T$ of a successful trajectory τ:

$$\mathcal{P} = \alpha_1 \mathbf{h}_1 + \alpha_2 \mathbf{h}_2 + \cdots + \alpha_T \mathbf{h}_T, \tag{9}$$

where $0 \leq \alpha_k \leq 1,\ \forall k = 1, \cdots, T$. Then, our interpretation of the entropy formula tells us that the volume of parallelotope $\mathcal{P}$ is the amount of information. However, we further posit that this parallelotope within the high dimensional semantic vector space corresponds to the information itself required for task success. We generalize this parallelotope to a convex cone as

$$\mathcal{C} = \alpha_1 \mathbf{h}_1 + \alpha_2 \mathbf{h}_2 + \cdots + \alpha_T \mathbf{h}_T, \tag{10}$$

where $\alpha_k \geq 0,\ \forall k = 1, \cdots, T$. Note that in the (projected) semantic space, the positive and negative direction matters for the same direction. This is evident from the fact that the value vectors representing semantics are linearly combined always with nonnegative (or allowing slight negative) coefficients after softmax, ReLU or GELU of attention coefficients in Transformers. Ideally, when $H(o_1, o_2, \cdots, o_T) \backslash H(\mathcal{T}) = \emptyset$ upon task sucess, this cone $\mathcal{C}$ contains the exact information paralleltope with precise angular alignment from the origin. We define this exact cone as the *task information cone (TIC)* $\mathcal{C}_\mathcal{T}$. In approximated cases with $H(o_1, o_2, \cdots, o_T) \backslash H(\mathcal{T}) \neq \emptyset$ and task sucess, however, we will have $\mathcal{C}_\mathcal{T} \subseteq \mathcal{C}$.

To validate this conjecture, i.e., the information required for a given agentic task $\mathcal{T}$ forms a TIC $\mathcal{C}_\mathcal{T}$ in the semantic vector space, predicted by the theory in Secs. 2 and 3, we designed an experiment with Qwen3-4B-Instruct-2507 (Yang et al., 2025) and ReAct (Yao et al., 2022b) on 133 ALFWorld tasks (Shridhar et al., 2020), using the ETO evaluation codebase (Song et al., 2024). For each and every of the 133 tasks, we ran five rollouts: some of which were successful and others were unsuccessful. We then generated all possible pairs (τ^i, τ^j) with 1) two successful rollouts τ^i and τ^j, 2) one successful rollout τ^i and one unsuccessful rollout τ^j, and 3) two unsuccessful rollouts τ^i and τ^j; took $(\mathbf{h}_1^i, \cdots, \mathbf{h}_{T_i}^i) =: \mathbf{H}^i$ and $(\mathbf{h}_1^j, \cdots, \mathbf{h}_{T_j}^j) =: \mathbf{H}^j$ from τ^i and τ^j; and then tested the overlap between two subspaces $\mathcal{L}(\mathbf{H}^i)$ and $\mathcal{L}(\mathbf{H}^j)$. For this we defined the effective subspace overlap index $\Gamma \in [1, 2]$ based on singular value decomposition, of which value 1 means perfect overlap and 2 means perfect nonoverlap. Please refer to Appendix D.2 for details of Γ.

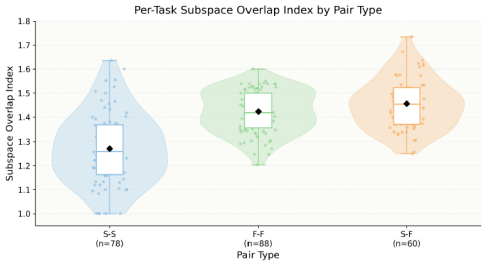


Figure 3: Average subspace overlap index Γ for S-S, F-F, and S-F pairs. Each point is a task-level aggregate for the indicated pair type rather than a raw rollout pair; violins show the resulting distribution across tasks.

Fig. 3 shows the distribution of Γ for three cases–success-success (S-S), success-failure (S-F), and failure-failure (F-F) pairings–based on all pairs obtained from the five rollouts of all 133 tasks. It is seen that Γ is concentrated towards 1 in the S-S case, while Γ's distribution is shifted away from 1 in the S-F and F-F cases. In the S-S case, $\mathbf{H}^i$ and $\mathbf{H}^j$ should cover the same TIC $\mathcal{T}_{\mathcal{T}}$ and hence the overlap should be significant, although it may not be perfect due to non-empty $H(o_1, o_2, \cdots, o_T) \backslash H(\mathcal{T})$. In the S-F and F-F cases, on the other hand, $\mathbf{H}^i$ and $\mathbf{H}^j$ do not share the same TIC $\mathcal{C}_{\mathcal{T}}$, so Γ should be larger. Indeed, Fig. 3 shows the expected tendency, revealing clear separation between S-S pairs and pairs involving failed rollouts: the mean Γ is 1.27 for S-S, compared with 1.42 for F-F and 1.46 for S-F. This result supports our conjecture and approach derived from our new interpretation of the entropy formula.

6.2 Main Experiment and Results

Setup We evaluate AAA on ALFWorld (Shridhar et al., 2020) and WebShop (Yao et al., 2022a). We compare against the baselines reported by Feng et al. (2025): (1) closed-source prompting baselines, GPT-4o and Gemini-2.5-Pro; (2) open-source prompting baselines, including direct prompting, ReAct, and Reflexion (Shinn et al., 2023); and (3) RL training baselines on the same backbone, including PPO, RLOO (Ahmadian et al., 2024), GRPO, and GiGPO. Our AAA variants are implemented on the official GiGPO codebase[1] and evaluated under the matched benchmark protocol.

Training Details We apply AAA to GRPO and GiGPO, yielding GRPO+AAA and GiGPO+AAA, respectively, as described in Sec. 5.2. Our runs use the same backbone models, environment interfaces, rollout group size, maximum episode length, prompt and response lengths, and final evaluation protocol as the corresponding GiGPO benchmark setting. All reported AAA results are averaged over three random seeds. Full hyperparameters and implementation details are provided in Appendix C.

Table 1 reports performance on ALFWorld and WebShop. Across aggregate metrics, AAA improves both base optimizers and both model sizes, showing that innovations-based reward decomposition can be used as a plug-in step-wise reward signal. When added to GRPO, AAA yields large gains: with the 1.5B model, GRPO+AAA improves ALFWorld success from 72.8 to 82.7, WebShop score from 75.8 to 83.9, and WebShop success from 56.8 to 71.3. The same trend holds for the 7B model, where GRPO+AAA improves ALFWorld success by 10.6 pp, WebShop score by 7.2, and WebShop success by 8.8 pp over GRPO.

AAA also improves the stronger GiGPO baseline, which already includes step-level credit assignment. For the 1.5B model, GiGPO+AAA improves over the matched $\text{GiGPO}_{\text{w/o std}}$ baseline from 86.1 to 91.0 on ALFWorld, from 83.5 to 85.8 on WebShop score, and from 67.4 to 70.9 on WebShop success. For the 7B model, the matched GiGPO baseline is already strong, but GiGPO+AAA still gives positive aggregate gains on ALFWorld and WebShop. Overall, these results suggest that AAA not only addresses sparse rewards in practice, but also provides an innovation-based refinement signal for step-wise credit assignment.

6.3 Ablation Study

We conduct component ablations as controlled single-seed studies on Qwen2.5-1.5B-Instruct under the same training budget.

Sensitivity to δ. AAA adds a small positive offset δ to the normalized rewards of successful trajectories. This offset prevents successful steps from receiving negative normalized rewards due to group-relative normalization. We sweep δ on WebShop to examine the sensitivity of AAA to this hyperparameter. The results show that removing the offset weakens performance, while a moderate offset improves success rate. However, too large an offset reduces the step-wise contrast induced by innovation rewards. We therefore use $\delta = 0.1$ as a simple default value across experiments.

[1] https://github.com/langfengq/verl-agent

Type	Method	ALFWorld							WebShop	
		Pick	Look	Clean	Heat	Cool	Pick2	All(Succ.↑)	Score(↑)	Succ.(↑)
Closed-Source Model										
Prompting	GPT-4o	75.3	60.8	31.2	56.7	21.6	49.8	48.0	31.8	23.7
Prompting	Gemini-2.5-Pro	92.8	63.3	62.1	69.0	26.6	58.7	60.3	42.5	35.9
Qwen2.5-1.5B-Instruct										
Prompting	Qwen2.5	5.9	5.5	3.3	9.7	4.2	0.0	4.1	23.1	5.2
Prompting	ReAct	17.4	20.5	15.7	6.2	7.7	2.0	12.8	40.1	11.3
Prompting	Reflexion	35.3	22.2	21.7	13.6	19.4	3.7	21.8	55.8	21.9
RL Training	PPO (with critic)	$64.8_{\pm 3.5}$	$40.5_{\pm 6.9}$	$57.1_{\pm 4.9}$	$60.6_{\pm 6.6}$	$46.4_{\pm 4.0}$	$47.4_{\pm 1.9}$	$54.4_{\pm 3.1}$	$73.8_{\pm 3.0}$	$51.5_{\pm 2.9}$
RL Training	RLOO	$88.3_{\pm 3.0}$	$52.8_{\pm 8.6}$	$71.0_{\pm 5.9}$	$62.8_{\pm 8.7}$	$66.4_{\pm 5.5}$	$56.9_{\pm 4.7}$	$69.7_{\pm 2.5}$	$73.9_{\pm 5.6}$	$52.1_{\pm 6.7}$
RL Training	GRPO	$85.3_{\pm 1.5}$	$53.7_{\pm 8.0}$	$84.5_{\pm 6.8}$	$78.2_{\pm 7.9}$	$59.7_{\pm 5.0}$	$53.5_{\pm 5.6}$	$72.8_{\pm 3.6}$	$75.8_{\pm 3.5}$	$56.8_{\pm 3.8}$
RL Training	$\text{GiGPO}_{\text{w/std}}$	$94.4_{\pm 5.9}$	$67.5_{\pm 4.6}$	$94.8_{\pm 3.8}$	$94.4_{\pm 7.8}$	$79.8_{\pm 5.4}$	$76.4_{\pm 7.3}$	$86.7_{\pm 1.7}$	$83.1_{\pm 1.6}$	$65.0_{\pm 3.2}$
RL Training	$\text{GiGPO}_{\text{w/o std}}$	$96.0_{\pm 1.4}$	$76.5_{\pm 3.9}$	$91.8_{\pm 5.5}$	$91.3_{\pm 6.3}$	$71.7_{\pm 8.4}$	$79.5_{\pm 7.7}$	$86.1_{\pm 4.7}$	$83.5_{\pm 1.8}$	$67.4_{\pm 4.5}$
RL Training	GRPO + AAA	$94.5_{\pm 2.8}$	$79.6_{\pm 6.4}$	$91.7_{\pm 3.3}$	$86.2_{\pm 9.5}$	$82.6_{\pm 6.7}$	$57.9_{\pm 8.8}$	$82.7_{\pm 3.3}$ (+9.9pp)	$83.9_{\pm 1.3}$ (+8.1pp)	$\mathbf{71.3}_{\pm 0.8}$ (+14.5pp)
RL Training	GiGPO + AAA	$98.4_{\pm 1.8}$	$78.1_{\pm 16.3}$	$98.8_{\pm 1.1}$	$97.1_{\pm 5.0}$	$82.0_{\pm 5.8}$	$85.2_{\pm 10.2}$	$\mathbf{91.0}_{\pm 1.5}$ (+4.9pp)	$\mathbf{85.8}_{\pm 1.2}$ (+2.3pp)	$70.9_{\pm 1.1}$ (+3.5pp)
Qwen2.5-7B-Instruct										
Prompting	Qwen2.5	33.4	21.6	19.3	6.9	2.8	3.2	14.8	26.4	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	42.7	58.1	28.8
RL Training	PPO (with critic)	$92.3_{\pm 4.0}$	$64.0_{\pm 8.4}$	$92.5_{\pm 2.4}$	$89.5_{\pm 7.0}$	$80.3_{\pm 2.0}$	$68.8_{\pm 8.3}$	$80.4_{\pm 2.7}$	$81.4_{\pm 3.1}$	$68.7_{\pm 5.1}$
RL Training	RLOO	$87.6_{\pm 4.3}$	$78.2_{\pm 8.3}$	$87.3_{\pm 5.8}$	$81.3_{\pm 7.6}$	$71.9_{\pm 5.2}$	$48.9_{\pm 8.4}$	$75.5_{\pm 4.6}$	$80.3_{\pm 3.2}$	$65.7_{\pm 4.0}$
RL Training	GRPO	$90.8_{\pm 5.1}$	$66.1_{\pm 6.7}$	$89.3_{\pm 5.4}$	$74.7_{\pm 6.9}$	$72.5_{\pm 5.4}$	$64.7_{\pm 7.3}$	$77.6_{\pm 5.2}$	$79.3_{\pm 2.8}$	$66.1_{\pm 3.7}$
RL Training	$\text{GiGPO}_{\text{w/std}}$	$97.7_{\pm 1.6}$	$82.7_{\pm 7.9}$	$98.8_{\pm 1.6}$	$83.7_{\pm 7.2}$	$89.3_{\pm 8.2}$	$79.2_{\pm 6.6}$	$90.8_{\pm 1.3}$	$84.4_{\pm 2.9}$	$72.8_{\pm 3.2}$
RL Training	$\text{GiGPO}_{\text{w/o std}}$	$91.8_{\pm 5.4}$	$88.6_{\pm 6.3}$	$95.9_{\pm 3.2}$	$90.2_{\pm 2.6}$	$86.5_{\pm 5.5}$	$85.2_{\pm 7.5}$	$90.2_{\pm 2.3}$	$86.2_{\pm 2.6}$	$75.2_{\pm 3.8}$
RL Training	GRPO + AAA	$98.7_{\pm 1.4}$	$80.7_{\pm 6.5}$	$95.8_{\pm 2.6}$	$88.7_{\pm 0.3}$	$77.3_{\pm 9.6}$	$81.5_{\pm 5.8}$	$88.2_{\pm 1.1}$ (+10.6pp)	$86.5_{\pm 2.3}$ (+7.2pp)	$74.9_{\pm 3.3}$ (+8.8pp)
RL Training	GiGPO + AAA	$97.9_{\pm 2.9}$	$88.1_{\pm 10.7}$	$95.4_{\pm 6.5}$	$87.7_{\pm 1.8}$	$85.3_{\pm 5.4}$	$86.6_{\pm 7.1}$	$\mathbf{91.0}_{\pm 0.8}$ (+0.8pp)	$\mathbf{87.0}_{\pm 3.3}$ (+0.8pp)	$\mathbf{75.7}_{\pm 3.1}$ (+0.5pp)

Table 1: **Main results on ALFWorld and WebShop.** Tiny colored values in parentheses denote absolute changes from the corresponding non-AAA baseline. For ALFWorld, we annotate only the aggregate success rate. GRPO+AAA is compared against GRPO, and GiGPO+AAA is compared against $\text{GiGPO}_{\text{w/o std}}$, the matched non-AAA variant used for our GiGPO-based setting.

Table 2: **Sensitivity to δ for GRPO+AAA on WebShop.** The $\delta = 0$ row corresponds to the variant without the success-side positive offset. Moderate offsets improve success rate, while overly large offsets can dilute the contrast among innovation rewards.

δ	Score (%)	Success Rate (%)
0	81.1	71.1
0.05	83.2	73.4
0.1	82.9	75.0
0.2	85.0	68.8
0.5	85.7	70.3
1.0	75.5	64.8

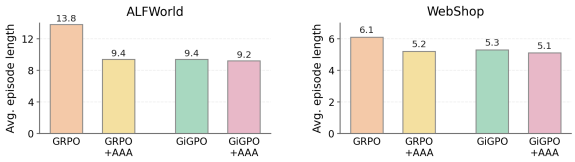


Figure 4: Average episode length of successful rollouts: GRPO, GRPO+AAA, GiGPO and GiGPO+AAA.

7 Analysis

Successful Trajectory Efficiency We further analyze the average length of successful trajectories. Since AAA assigns larger rewards to steps that reveal non-redundant information, it should discourage unnecessary interactions and lead to shorter successful episodes.

Figure 4 shows that AAA generally shortens successful rollouts. On ALFWorld, GRPO+AAA reduces the average successful episode length from 13.8 to 9.4, while GiGPO+AAA yields a smaller reduction from 9.4 to 9.2. On WebShop, the corresponding averages decrease from 6.1 to 5.2 for GRPO and from 5.3 to 5.1 for GiGPO. Thus, the efficiency gain is most pronounced when added to GRPO, whereas GiGPO already produces relatively compact successful trajectories. Detailed sample-efficiency curves are provided in Appendix D.3.

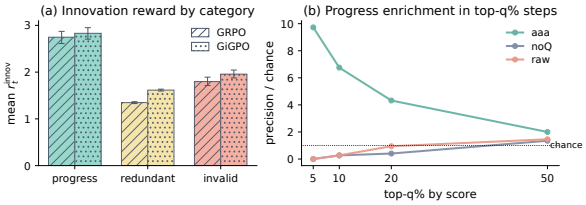


Figure 5: Innovation–progress alignment. Progress steps receive larger innovation rewards than redundant or invalid steps.

Innovation–Progress Alignment We examine whether high-innovation steps correspond to interpretable task progress. Using environment feedback, we annotate each ALFWorld step as *progress*, *redundant*, or *invalid*, and compare the innovation rewards assigned to each category. As shown in Figure 5 (left), progress steps receive substantially larger innovation rewards than redundant or invalid steps, under both GRPO and GiGPO. To test where this signal comes from, we further measure how strongly the top-scoring steps are enriched with goal-relevant predicate changes (Figure 5, right). Selecting steps by the full AAA reward concentrates progress steps far above chance, up to roughly $14\times$ among the top 5%, whereas ablating the Q-residual (NOQ, random projection only) or using the raw hidden-state delta (RAW) yields near-chance enrichment. This indicates that AAA captures task-relevant novelty rather than semantic change alone, and that the Q-residual is what carries the task-aligned signal. Annotation details are provided in Appendix D.4.

Innovation Norm over Time Steps AAA assumes the informativeness of a transition varies across a trajectory. The innovation norm $\|\mathbf{e}_t\|$ indeed fluctuates substantially from step to step rather than staying flat, supporting step-wise credit assignment over uniform spreading of terminal success. Per-rollout visualizations for ALFWorld and WebShop are in Appendix D.1.

8 Related Work

RL for LLM Agents For RL post-training of LLM agents, GRPO (Shao et al., 2024) has emerged as the main approach, eliminating critic models. To address the sparsity of outcome-based rewards in multi-turn reasoning, early works like IPR (Xiong et al., 2024) and STeCa (Wang et al., 2025b) proposed iterative refinement and trajectory calibration. Recent methods focus on automating dense feedback. Agent-PRM (Xi et al., 2025) enhances sample efficiency by employing Monte Carlo rollouts to estimate step-level values, replacing sparse outcome signals. Similarly, SPA-RL (Wang et al., 2025a) addresses credit assignment via Stepwise Progress Attribution, training an estimator to decompose final rewards into cumulative progress contributions for each step. GiGPO (Feng et al., 2025) extended GRPO to enable fine-grained credit assignment with hash-based data matching.

Information-Gain Approaches There exist several works on step reward design for agentic tasks based on information gain (IG), i.e., conditional mutual information between task and observation $I(\mathcal{T}; o_t | o_1, \cdots, o_{t-1})$, because this is a natural step-wise reward from an information-theoretic perspective. Bae et al. (2025) set up an explicit belief about task $\mathcal{T}$ for search tasks. Auzina et al. (2026) demonstrated the effectiveness of IG for the problem of 20 questions. Wang et al. (2026) applied IG for tasks of word solution. However, all the previous methods require some kind of structure on $\mathcal{T}$, which is not readily extendable to general agentic tasks, whereas our method forgoes this.

Additional related work, including Determinantal Point Processes(DPP)-based diversity, curiosity-driven exploration, and memory-augmented agent training, is discussed in Appendix A.

9 Conclusion

We proposed AAA, an innovations-based intrinsic reward model, which sets the net information provided by observation regarding the task as step reward. AAA exploits the fact that semantic observation sentences are represented as finite-dimensional vectors in LLMs and the total information can be quantified as the volume of the parallelotope constructed by these observation embedding vectors. Applying Gram-Schmidt orthogonalization to these observation vectors, AAA readily computes the innovations of the observation at each time step. The proposed process reward model based on innovations precisely quantifies the net information provided by each time step and is effective to emphasize task-relevant steps and suppress redundant steps. Experimental results shows its effectiveness across various tasks with substantial gains.

Limitations

A limitation of AAA is that it is applicable to LLM agentic tasks with multiple action-observation turns. However, reasoning problems such as mathematics form a large portion of LLM reasoning. It is unclear that the innovations approach can be applied to such reasoning problems. In the Chain-of-thought (CoT) formulation, a solution is derived after a sequence of thoughts. It is possible to represent each thought as a vector in the semantic vector space, construct a parallelotope based on these vectors and apply the innovations approach. This remains as a future work. Another challenge lies in the design of step rewards for failed trajectories; currently, AAA assigns zero to all such steps. Since certain actions within a failed trajectory may still be effective, differentiating these steps remains a key objective for subsequent studies.

Broader Impact

AAA offers a simple yet effective step reward design for LLM agentic tasks equipped with terminal verifiers. Given its versatility, it can be broadly integrated into various agentic workflows to enhance performance. The potential applications of agentic AI span diverse domains, as demonstrated in household care (ALFWorld), scientific exploration (SciWorld), and online commerce (WebShop). By successfully automating these complex tasks, AAA contributes to societal progress by streamlining labor-intensive and hazardous operations.

Use of LLMs

We used LLMs to improve and embellish the writing in the paper. We also used LLMs as coding assistants for debugging and refactoring purpose. We make clear that LLMs were not used for idea generation or research design.

References

Joshua Achiam and Shankar Sastry. 2017. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*.

Raja Hafiz Affandi, Emily B Fox, and Ben Taskar. 2013. Approximate inference in continuous determinantal point processes. In *Advances in Neural Information Processing Systems (NIPS)*, volume 26, pages 1430–1438.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*.

Anonymous. 2026. Exploratory memory-augmented LLM agent via hybrid on- and off-policy optimization. In *The Fourteenth International Conference on Learning Representations*.

Ilze Amanda Auzina, Joschka Strüber, Sergio Hernández-Gutiérrez, Shashwat Goel, Ameya Prabhu, and Matthias Bethge. 2026. Intrinsic credit assignment for long horizon interaction. *arXiv preprint arXiv:2602.12342*.

Seohui Bae, Jeonghye Kim, Youngchul Sung, and Woohyung Lim. 2025. Align while search: Belief-guided exploratory inference for world-grounded embodied agents. *arXiv preprint arXiv:2512.24461*.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2018. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*.

Thomas M Cover and Joy A Thomas. 2006. *Elements of information theory*, 2nd edition. John Wiley & Sons, Hoboken, NJ.

Sanjoy Dasgupta and Anupam Gupta. 2003. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65.

Ildefons Magrans de Abril and Ryota Kanai. 2018. Curiosity-driven reinforcement learning with homeostatic regulation. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE.

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

Gene H. Golub and Charles F. Van Loan. 1996. *Matrix Computations*, 3rd edition. Johns Hopkins University Press, Baltimore, MD, USA.

Geoffrey Grimmett and David Stirzaker. 2020. *Probability and Random Processes*, 4th edition. Oxford University Press.

Roger A. Horn and Charles R. Johnson. 2012. *Matrix Analysis*, 2nd edition. Cambridge University Press, Cambridge.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. 2016. VIME: Variational information maximizing exploration. *Advances in Neural Information Processing Systems*, 29.

Thomas Kailath. 1970. The innovations approach to detection and estimation theory. *Proceedings of the IEEE*, 58(5):680–695.

Thomas Kailath, Ali H. Sayed, and Babak Hassibi. 2000. *Linear Estimation*. Prentice Hall, Upper Saddle River, NJ.

Rudolf Emil Kalman. 1960. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.

Alex Kulesza, Ben Taskar, and 1 others. 2012. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2-3):123–286.

Yongqi Liang, Shaohan Wang, Zichao Zhu, Wayne Xin Zhao, and Ji-Rong Wen. 2026. IG-search: Step-level information gain rewards for search-augmented reasoning. *arXiv preprint arXiv:2604.15148*.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding R1-zero-like training: A critical perspective. *Preprint*, arXiv:2503.20783.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR.

Jürgen Schmidhuber. 1991. Curious model-building control systems. In *Proc. international joint conference on neural networks*, pages 1458–1463.

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

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junmei Song, Mingchuan Zhang, Xuan Li, Y. K. Yu, Damai Dai, Daya Guo, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Noah Shinn, Ke Hou, and et al. 2023. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*.

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

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

Gilbert Strang. 2006. *Linear Algebra and Its Applications*, 4th edition. Thomson, Brooks/Cole, Belmont, CA.

Youngchul Sung, Lang Tong, and H. Vincent Poor. 2006. Neyman-Pearson detection of Gauss-Markov signals in noise: Closed-form error exponent and properties. *IEEE Transactions on Information Theory*, 52(4):1354–1365.

Youngchul Sung, Xin Zhang, Lang Tong, and H. Vincent Poor. 2008. Sensor configuration and activation for field detection in large sensor arrays. *IEEE Transactions on Signal Processing*, 56(2):447–463.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. 2026. Information gain-based policy optimization: A simple and effective approach for agentic search. In *The Fourteenth International Conference on Learning Representations (ICLR)*.

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. 2025a. SPA-RL: Reinforcing LLM agents via stepwise progress attribution. *arXiv preprint arXiv:2505.20732*.

Hanlin Wang, Jian Wang, Chak Tou Leong, and Wenjie Li. 2025b. STeCa: Step-level trajectory calibration for LLM agent learning. *arXiv preprint arXiv:2502.14276*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439. Association for Computational Linguistics.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. 2025c. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *Preprint*, arXiv:2504.20073.

Norbert Wiener. 1958. *Nonlinear Problems in Random Theory*. Technology Press Research Monographs. Technology Press of Massachusetts Institute of Technology and John Wiley & Sons, Inc., New York.

Yangjun Wu, Jiaming Ji, Hanyuan Zhang, Jiayi Huang, Chenjun Xiao, Tianqi Du, Dahua Lin, Li Chen, and Xiaotie Deng. 2025. DAPO: Decoupled clip and dynamic sampling policy optimization for post-training large language models. *Preprint*, arXiv:2503.14476.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Yajie Yang, Wenxiang Chen, Zhihao Zhang, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, and 1 others. 2025. AgentPRM: Process reward models for LLM agents via step-wise promise and progress. *arXiv preprint arXiv:2511.08325*.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and

Sujian Li. 2024. Watch every step! LLM agent learning via iterative step-level process refinement. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1556–1572.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

A Additional Related Work

Relation to Determinantal Point Processes The Gaussian entropy formula in Eq. (2)–representing the maximum entropy under a second-moment constraint in information theory (Cover and Thomas, 2006)–shares a commonality with Determinantal Point Processes (DPP) (Kulesza et al., 2012) in that both rely on the matrix determinant. However, DPP focuses on sampling with a probability distribution on the sigma field $\sigma(\Omega)$ of a ground set $\Omega \subset \mathbb{R}^d$, where the density of points $\{x_1, \cdots, x_n\}$ in $\sigma(\Omega)$ according to DPP is given by (Affandi et al., 2013)

$$p(x_1, \cdots, x_n) \propto \det([K(x_i, x_j]_{1 \leq i,j \leq n}) \quad (11)$$

for some positive-semidefinite kernel function $K(\cdot, \cdot)$. In DPP, $\det([K(x_i, x_j]_{1 \leq i,j \leq n})$ is called the "diversity" of $\{x_1, \cdots, x_n\}$. Typically, the kernel function is designed to measure the similarity between x_i and x_j, e.g., Gaussian kernel $\exp(-||x_i - x_j||^2/(2\sigma^2))$, cosine similarity, etc., so that the sampled realization has diversity. In particular, consider the case of cosine similarity as the kernel function:

$$K(x_i, x_j) = \langle \frac{x_i}{||x_i||}, \frac{x_j}{||x_j||} \rangle$$

with unit norm vectors $||x_i|| = ||y_i|| = 1$ like layer-normalized embedding vectors. Then, as we noted in Section 2, the matrix $[K(x_i, x_j]_{1 \leq i,j \leq n}$ becomes the Gram matrix of vectors $x_1, \cdots, x_j \in \mathbb{R}^d$ (Golub and Van Loan, 1996). Hence, the sampling density of $(x_1, \cdots, x_n)$ is proportional to the square of the volume of the parallelotope constructed by $x_1, \cdots, x_n$. When $x_1, \cdots, x_n$ are similar, i.e., highly correlated with small angles between them, the parallelotope volume is small and the joint sampling density is small. When $x_1, \cdots, x_n$ are orthogonal to each other, on the other hand, the parallelotope volume is maximized and the joint sampling density is highest. In this way, DPP yields repulsiveness among similar samples and enables dissimilar sampling with high diversity.

However, DPP does not view $x_1, \cdots, x_n$ as observation vectors. Furthermore, the Hartley entropy function and Gaussian Shannon entropy defines the logarithm of the volume of observations as the amount of information. DPP directly considers the sampling density proportional to the square of the volume of samples. Even so, we can view DPP sampling with the cosine similarity kernel as maximum "entropy" sampling, because the entropy formula and DPP both rely on the Gram matrix of samples $x_1, \cdots, x_n$ in this case and the square root and logarithm are monotone-increasing. That is, we can replace the term "diversity" with "entropy" from an information-theoretic perspective in this special case.

Explolation for Agent Tasks Exploration is crucial for long-horizon tasks where standard RL often struggles. Existing methods like ETO (Song et al., 2024) and improve exploration through trajectory optimization or memory-augmented updates. However, these approaches often rely on external memory or heuristic guidance. In contrast, we propose an information-theoretic approach. We introduce an intrinsic reward based on *innovations* to quantify the novelty of state transitions. This allows the agent to efficiently acquire task-relevant information and accelerates learning in sparse-reward settings.

Memory-Augmented Exploration Standard RL often struggles with long-horizon exploration. While ETO (Song et al., 2024) explored trajectory optimization, EMPO2 (Anonymous, 2026) advances this by integrating external memory with hybrid on- and off-policy updates. The agent retrieves "tips" summarized from past failures to guide exploration. Crucially, EMPO2 uses off-policy distillation to internalize these behaviors directly into the policy, enabling robust performance without reliance on external memory during inference.

Efficient Group-based Optimization Reducing the computational overhead of RL is critical for LLMs. GRPO (Shao et al., 2024) pioneered a critic-free approach, estimating baselines from group statistics of outputs generated from the same input, thereby saving memory and ensuring stability. GiGPO (Feng et al., 2025) extends this logic to address fine-grained credit assignment. By clustering actions sharing identical environmental states into "anchor states," GiGPO computes step-level relative advantages without a value network, retaining GRPOs efficiency while improving precision in multi-turn scenarios.

Curiosity RL In RL with sparse rewards, curiosity-driven intrinsic motivation has been widely adopted to provide intermediate learning signals. Curiosity (Schmidhuber, 1991), often called surprise (Achiam and Sastry, 2017), is defined as the information gain of a prediction

model and encourages exploration toward unexplored states or state-action pairs. Most curiosity-based methods focus on environment dynamics (Houthooft et al., 2016; Pathak et al., 2017; Achiam and Sastry, 2017; de Abril and Kanai, 2018), using either information gain over dynamics parameters or prediction errors of forward models as intrinsic rewards. More recently, Random Network Distillation (Burda et al., 2018) measures curiosity as the discrepancy from a fixed random target network, which can be interpreted as information gain over target network parameters. While our work also employs curiosity-based intrinsic rewards, it differs fundamentally from prior approaches: we define curiosity as the information gain about the task itself given all observations collected thus far, rather than focusing on environmental dynamics or network parameters.

B Algorithm Details

Our algorithm AAA is composed of three steps:

1. Semantic vector extraction - Output: $\mathbf{h}_t^i$
2. Innovation computation - Output: $\mathbf{e}_t^i$
3. Advantage computation - Output: $\hat{A}_{i,t}$

We explain each step in detail here. Please note that all vectors are column vectors throughout this paper .

B.1 Time Indexing in This Paper

First note that our time index in this paper is defined as follows:

$$\mathcal{T} \xrightarrow{\text{LLM}} a_1 \xrightarrow{\text{Env.}} o_1 \xrightarrow{\text{LLM}} a_2 \xrightarrow{\text{Env.}} o_2 \cdots \xrightarrow{\text{Env.}} o_T. \tag{12}$$

In standard RL conventions, the observation resulting from action a_t is typically denoted by o_{t+1}. In this paper, however, we denote it as o_t to align the time indices of cause (action) and effect (observation) for simplicity. Hence, the observation induced by action a_t is o_t, and the action a_t is decided by the policy based on o_{t-1} (and before).

Throughout this paper, o_t denotes the agent-facing textual observation context at step t, rather than only the literal field labeled "current observation" inside the prompt. In implementation, this context is serialized as the full user-side input prompt before the assistant generates the next thought/action.

The overall timing diagram is shown in Fig. 6.

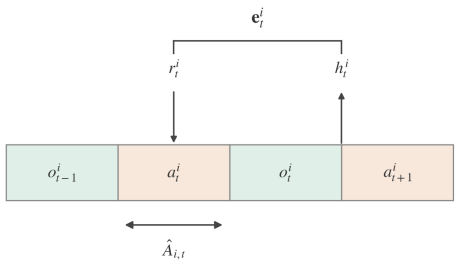


Figure 6: Timing diagram for rollout τ^i

B.2 Semantic Vector Extraction

In the chain of eq. (12) we extract the embedding vector from the last token position between o_t and a_{t+1} at the LLM's final Transformer layer. This ensures the vector captures the context up to and including o_t induced by action a_t. In implementation, the extraction position is the fixed chat-template boundary following the full user-side prompt, specifically the newline after `<|im_end|>` and before `<|im_start|>assistant`. The box below can be used as a schematic example of this boundary.

Schematic extraction boundary

```
<|im_start|>user
Task instruction, prior observation-action
history . . .
Current observation context at step t . . .
Admissible actions . . .
<|im_end|>\n                    ⟵ extract h_t
<|im_start|>assistant\n
Thought: . . .
Action: . . .
```

The core of our extraction method lies in accounting for the operational nature of autoregressive Transformers. Let $(\cdots, \mathbf{t}_{n-1}, \mathbf{t}_n)$ denote the sequence of input tokens, where n is the index of the latest token. These tokens are projected into query, key, and value vectors through linear transformations W_Q, W_K and W_V. For simplicity, considering a single-head attention mechanism at the first layer, we have

$$\mathbf{q}_i = W_Q \mathbf{t}_i \tag{13}$$
$$\mathbf{k}_i = W_K \mathbf{t}_i \tag{14}$$
$$\mathbf{v}_i = W_V \mathbf{t}_i. \tag{15}$$

While Transformer literature typically uses row vectors, we treat all vectors as column vectors for consistency. The attention block transforms the input sequence into a single contextualized summary

vector $\mathbf{s}_n$ at the n-th position:

$$\mathbf{s}_n = \sum_{j \le n} \text{softmax}\left(\frac{\langle \mathbf{q}_n, \mathbf{k}_j \rangle}{\sqrt{d_e}}\right) \mathbf{v}_j, \qquad (16)$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product and d_e is the embedding vector size. Note that the key and value pairs $\{(\mathbf{k}_j, \mathbf{v}_j)\}_{j \le n}$ are cached. The key issue here is that the query vector $\mathbf{q}_n$ of the latest token acts as the anchor for the inner product across all preceding indices to construct $\mathbf{s}_n$, as shown in Eq. (16). Consequently, even if the preceding context remains identical, $\mathbf{s}_n$ can fluctuate significantly based on the last anchor query $\mathbf{q}_n$. For an effective semantic representation, we require a vector that captures the cumulative meaning of the observation, i.e., the information in the value vectors $\mathbf{v}_j$, rather than one dominated by the superficial variance of the final token's query. For instance, changing the final punctuation from '.' to '!' drastically alters $\mathbf{q}_n$, which in turn disproportionately shifts the entire summary vector despite the semantic content of the preceding sentences remaining unchanged.

To prevent such artificial fluctuations and capture the genuine semantic shifts across sentences, we extract hidden states at a fixed chat-template boundary following the serialized observation context o_t. In our implementation, this corresponds to the newline after `<|im_end|>` and before `<|im_start|>assistant`, as illustrated in the schematic prompt above. This yields:

$$\mathbf{s}_n = \sum_{j \le n} \text{softmax}(\langle \mathbf{q}, \mathbf{k}_j \rangle) \mathbf{v}_j, \qquad (17)$$

at every extraction position n. Under this setup, $\mathbf{s}_n$ is determined by the stored key-value pairs $\{(\mathbf{k}_j, \mathbf{v}_j)\}_{j \le n}$. By fixing the $(n+1)$-th token as a constant delimiter for all extraction points, we shift the focus from the Transformer's immediate next-token prediction to the underlying long-term summary. Interestingly, even though the extraction token and its succeeding token remain identical for all observations o_t (where $t = 1, 2, ..., T-1$), the extracted embedding vector $\mathbf{s}_n$ (composing $\mathbf{h}_t$) evolves over time, and an innovation component $\mathbf{e}_t$ exists at each step t, as shown in Fig. 7. This implies that the semantic summary of the input context is captured not by the specific next token itself, but by the distribution over the next token (i.e., the embedding vector prior to the de-embedding operation).

One significant advantage of using the extracted embedding $\mathbf{h}_t$ is its efficiency; it provides a single, compact vector that summarizes the entire preceding context. In contrast, token-based probability techniques require handling a large number of tokens, and it is often unclear how to effectively aggregate these distributions to represent the full semantic meaning of a multi-token sentence o_t.

B.3 Innovation Computation

The computation of innovation vectors via Gram-Schmidt orthogonalization is described in Algorithm 1.

The $\mathbf{h}_t^i$ extraction step in Line 6 is explained in Appendix B.4. For the projection $\mathbf{P}$ in Line 7, we used the following matrix

$$\mathbf{P} = \frac{1}{\sqrt{d_e}}[p_{ij}] \in \mathbb{R}^{d \times d_e}, \qquad (18)$$

where $p_{ij} \overset{i.i.d.}{\sim} N(0,1)$, i.e., zero-mean Gaussian with unit variance. $\mathbf{P}$ is fixed. In Lines 10 to 12, we append a basis vector into $\mathbf{Q}$ only when the innovation vector norm is above a threshold to ignore insignificant components. The threshold κ is set as $\kappa = 0.03||\mathbf{u}_t^i||$.

B.4 Semantic Vector Extraction

AAA represents each language observation by the internal continuous representation of the LLM agent. An autoregressive Transformer maps a serialized interaction history to hidden states at each token position. We use the hidden state at the user-to-assistant boundary after the observation context o_t and before the next thought/action is generated:

$$\mathbf{h}_t = f_{\theta_{\text{old}}}(\mathcal{T}, a_1, o_1, \ldots, a_t, o_t) \in \mathbb{R}^{d_e}.$$

Here, $f_{\theta_{\text{old}}}$ denotes the hidden-state extraction function of the rollout policy before the policy update. We refer to the resulting representation space as the semantic vector space, since it encodes the observation through the LLM's internal continuous representation rather than through discrete output-token probabilities.

This design has two practical advantages. First, it does not require training a separate sentence encoder: the same pretrained Transformer that parameterizes the agent is used to embed observations. Second, it gives a common vector-space representation for free-form language observations, allowing us to apply linear-algebraic operations such as projections, volume computation, and orthogonalization.

B.5 Random Projection

The hidden dimension d_e is typically large, so AAA stores only a low-dimensional projection

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t \in \mathbb{R}^d,$$

where $\mathbf{P} \in \mathbb{R}^{d \times d_e}$ is a fixed random projection. By the Johnson–Lindenstrauss property, a random linear projection can approximately preserve pairwise distances and angles among T vectors when d scales logarithmically with T (Dasgupta and Gupta, 2003). Thus, $\{\mathbf{u}_t\}_{t=1}^T$ provides a compact finite-dimensional approximation to the semantic observation vectors.

In implementation, $\mathbf{P}$ is sampled once and kept fixed during rollout collection and reward computation. We do not backpropagate through $\mathbf{P}$; it is used only to compute the intrinsic reward.

B.6 Innovation Computation

Given projected observation vectors $\{\mathbf{u}_1, \ldots, \mathbf{u}_T\}$, AAA computes innovation vectors by Gram–Schmidt orthogonalization. For each step t, we remove the component of $\mathbf{u}_t$ that is predictable from the span of previous observations:

$$\mathbf{e}_t = \mathbf{u}_t - \sum_{j=1}^{t-1} \frac{\langle \mathbf{u}_t, \mathbf{e}_j \rangle}{\|\mathbf{e}_j\|_2^2 + \epsilon} \mathbf{e}_j,$$

where $\epsilon > 0$ is a small numerical constant. The vector $\mathbf{e}_t$ therefore captures the component of the current observation that is not already explained by previous observations.

The norm of the innovation vector is then used as a step-wise information signal. In the main experiments, we compute an intrinsic reward of the form

$$r_t^{\mathrm{AAA}} = \log\left(1 + \lambda \|\mathbf{e}_t\|_2^2\right),$$

where λ controls the scale of the reward. This reward assigns larger credit to observations that introduce new semantic information relative to the previous interaction history.

B.7 Advantage Computation and Policy Optimization Objectives

GRPO First, consider the application of AAA to GRPO (Shao et al., 2024). This is already explained in Section 5.2. We provide a brief summary here.

For GRPO, we first run G rollouts for a given task. Then, for each successful rollout τ^i, we compute $r_t^{\mathrm{innov},i}$ with Algorithm 1. We replace $r_1^{\mathrm{innov},i}$ of the first step with mean$\{r_t^{\mathrm{innov},i},\ t = 2, \cdots, T_i\}$. Next we set the step rewards as

$$r_t^i = \frac{r_t^{\mathrm{innov},i}}{\sum_{k=1}^{T_i} r_k^{\mathrm{innov},i}}, \qquad t = 1, \ldots, T_i, \tag{20}$$

where T_i is the length of rollout τ^i. For each failed rollout τ^i, we set

$$r_t^i = 0, \ \ \forall t = 1, \cdots, T_i. \tag{21}$$

Then, we compute the group-relative step rewards as

$$\tilde{r}_t^i = \begin{cases} \max\left(0, \dfrac{r_t^i - \mu_G}{\sigma_G}\right) + \delta, & \text{if } \tau^i \text{ succeeds} \\ \dfrac{r_t^i - \mu_G}{\sigma_G}, & \text{otherwise} \end{cases} \tag{22}$$

where

$$\begin{aligned} \mu_G &= \frac{1}{\sum_{i=1}^G T_i} \sum_{i=1}^G \sum_{t=1}^{T_i} r_t^i \\ \sigma_G &= \sqrt{\frac{1}{\sum_{i=1}^G T_i} \sum_{i=1}^G \sum_{t=1}^{T_i} (r_t^i - \mu_G)^2}. \end{aligned}$$

Finally, we set the advantage as

$$\hat{A}_{i,t} = \sum_{k=t}^{T_i} \gamma^{k-t} \tilde{r}_k^i = \tilde{r}_t^i \tag{23}$$

with $\gamma = 0$, considering the fact that $\tilde{r}_t^i$ is already a return share according to each step contribution. This advantage is plugged in the GRPO objective shown in Eq. (19), where $\rho_t^i(\theta)$ is the ratio between π_θ and $\pi_{\theta_{old}}$ for action a_t^i, and we optimize the policy π_θ with this objective.

We did not consider variants of GRPO such as DAPO (Wu et al., 2025) or DrGRPO (Liu et al., 2025) exploiting different length normalization, asymmetric clipping or KLD regularization term, because the main focus of this paper is the step reward design. When AAA is applied to such enhanced versions, we expect improved performance.

The timing of the overall operation is described in Fig. 6. The observation vector induced by action a_t^i is extracted between o_t^i and a_{t+1}^i, because o_t^i is the resulting observation from action a_t^i. The step reward r_t^i for action a_t^i is computed based on the extracted embedding vector $\mathbf{h}_t^i$, and then advantage $\hat{A}_{i,t}$ for acion a_t^i is computed based on this step reward r_t^i. The step advantage $\hat{A}_{i,t}$ is applied to

$$L(\theta) = \mathbb{E}_{\{\tau_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}} \frac{1}{G} \sum_{i=1}^{G} \frac{1}{T_i} \sum_{t=1}^{T_i} \left(\min\left(\rho_t^i(\theta)\, \hat{A}_{i,t},\ \text{clip}\left(\rho_t^i(\theta),\, 1-\epsilon,\, 1+\epsilon \right) \hat{A}_{i,t} \right) - \beta\, \mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}) \right) \tag{19}$$

all tokens corresponding to and a_t^i, as shown in in Fig. 6. In the case of ReAct agent (Yao et al., 2022b), which we used, a_t^i corresponds to thought and action. Note that the tokens corresponding to observations are masked for gradient update.

GiGPO AAA's innovations-based step rewards can be applied to GiGPO (Feng et al., 2025). GiGPO basically adds a hash-based correction term on GRPO's identical step advantage. Let the observation-action sequences of a group of G rollouts be denoted by

$$\{(o_{t-1}^1, a_t^1)\}_{t=1}^{T_1},\ \cdots,\ \{(o_{t-1}^G, a_t^G)\}_{t=1}^{T_G}.$$

Note that due to our time indexing of $(\cdots, o_{t-1} \to a_t \to o_t \to a_{t+1}, \cdots)$ in this paper, the action a_t^i is performed based on o_{t-1}^i. Each rollout τ^i receives the final reward $r_{T_i}^i$, which is 1 for success and 0 for failure, where the intermediate step rewards are all zero in sparse-reward tasks. GiGPO first identifies distinct observations from all $\sum_{i=1}^{G} T_i$ observations $\{o_{t-1}^i,\ \forall i, \forall t\}$ based on hashing. Let the set of identified distinct observations be

$$\mathcal{O}_D = \{\tilde{o}_1, \tilde{o}_2, \cdots, \tilde{o}_N\}. \tag{24}$$

Then, for each $\tilde{o}_k \in \mathcal{O}_D$, GiGPO constructs a set $\mathcal{A}(\tilde{o}_k)$ of all actions taken from $\tilde{o}_k$ captured from all G rollouts:

$$\mathcal{A}(\tilde{o}_k) = \{a_t^i \mid (o_{t-1}^i, a_t^i) \text{ with } o_{t-1}^i = \tilde{o}_k\}. \tag{25}$$

Each element $a_t^i \in \mathcal{A}(\tilde{o}_k)$ is assigned a value as

$$R(a_t^i | o_{t-1}^i = \tilde{o}_k) = \gamma^{T_i - t} r_{T_i}^i, \tag{26}$$

where $r_{T_i}^i = 1$ for success and $r_{T_i}^i = 0$ for failure. Note that on average, an action a_t^i in a shorter successful rollout receives a higher value than that in a longer successful rollout due to this discounting. Then, GiGPO computes a relative advantage for the actions in $\mathcal{A}(\tilde{o}_k)$ based on the values in (26), denoted by $\hat{A}^S(o_{t-1}^i = \tilde{o}_k, a_t^i)$. Finally, this step advantage is added to the GRPO rollout-based advantage, yieldingn the final GiGPO advantage:

$$\hat{A}_{i,t}^{GiGPO} = \hat{A}_{GRPO}(o_{t-1}^i, a_t^i) + c \cdot \hat{A}^S(o_{t-1}^i = \tilde{o}_k, a_t^i), \tag{27}$$

where c is a weighting factor.

The application of AAA to GiGPO is straightforward. First, we do the same hashing and action grouping to construct $\mathcal{O}_D$ and $\mathcal{A}(\tilde{o}_k)$ for each k. However, instead of (26), we use our innovations-based action value (20) and (21) for $R(a_t^i | o_{t-1}^i = \tilde{o}_k)$. With this new action value, we compute the group-relative step advantage $\hat{A}_{AAA}^S(o_{t-1}^i = \tilde{o}_k, a_t^i)$ and add this data-based correction term to the original advantage $\hat{A}_{i,t}$ in (23), yielding the final advantage:

$$\hat{A}_{i,t}^{A3+GiG} = \hat{A}_{i,t} + \lambda_d \cdot \hat{A}_{AAA}^S(o_{t-1}^i = \tilde{o}_k, a_t^i) \tag{28}$$

for action a_t^i, where λ_d is a weighting factor for the correction term. In this way, we have additional data-based correction to innovation-based step rewards to have a better step reward design. With this new advantage, we optimize the policy based on the objective shown in Eq. (19).

C Implementation and Training Details

Codebase and models We implement AAA on top of the released GiGPO codebase and use its ALFWorld and WebShop environment interfaces, rollout format, and default optimization recipe unless otherwise specified. We use Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct as the policy backbones. This setup allows us to compare GRPO, GiGPO, and their AAA-augmented variants under the same interaction budget and environment protocol.

Training and evaluation setup Table 3 summarizes the shared training configuration. For both environments, each training batch contains 16 task instances and each instance is expanded into a rollout group of size 8. We train for 150 epochs and use a validation batch size of 128. ALFWorld uses a maximum episode length of 50, while WebShop uses 15 steps. The prompt contains the current observation together with the recent interaction history following the default GiGPO prompt construction. For each reported main result, we train three independent models with different training seeds and evaluate each checkpoint with three independent final-evaluation seeds; we report the resulting mean

Setting	ALFWorld	WebShop
Training batch size	16	16
Rollouts per task	8	8
Validation batch size	128	128
Training epochs	150	150
Maximum environment steps	50	15
Maximum prompt length	2048	4096
Maximum response length	512	512
PPO mini-batch size	256	64
Learning rate	1×10^{-6}	1×10^{-6}
KL coefficient	0.01	0.01
Invalid-action penalty	0.1	0.1

Table 3: Shared training settings for ALFWorld and WebShop. Memory-only micro-batch sizes are adjusted by model size and hardware, while the interaction budget and optimization hyperparameters are kept fixed.

Hyperparameter	GRPO+AAA	GiGPO+AAA
Projection dim. d_e	512	512
Extraction layer	final	final
GS threshold κ	0.03	0.03
Success offset δ	0.1	0.1
Grouped adv. norm.	–	`mean_norm`
GiGPO weight λ_d	–	0.05 (ALFWorld)
	–	1.0 (WebShop)

Table 4: AAA-specific hyperparameters. The two values for λ_d correspond to ALFWorld and WebShop, respectively.

and standard deviation. All AAA training runs use two H200 GPUs per run. Hardware-dependent runtime settings, such as per-GPU micro-batch sizes, FSDP offloading, and vLLM execution/cache options, are adjusted only to fit the available hardware. We do not change the rollout budget, effective batch size, policy objective, learning rate, KL coefficient, or environment protocol.

AAA hyperparameters Table 4 lists the AAA-specific hyperparameters used in GRPO+AAA and GiGPO+AAA. The projection dimension is fixed to 512. The hidden dimension follows the backbone model: 1536 for Qwen2.5-1.5B-Instruct and 3584 for Qwen2.5-7B-Instruct. We extract hidden states from the final layer at the observation-conditioned assistant boundary described in Appendix B.4. The rollout-local Gram-Schmidt threshold is $\kappa = 0.03$, as defined in Algorithm 1. We use the default success offset $\delta = 0.1$ across ALFWorld and WebShop.

For GiGPO+AAA, we use `mean_norm` for the grouped advantage normalization, which subtracts the group mean without standard-deviation normalization. We use fixed correction weights $\lambda_d = 0.05$ for ALFWorld and $\lambda_d = 1.0$ for WebShop.

D More on Experiments

D.1 Innovations versus Time Step

Fig. 7 visualizes how the innovation norm $\|\mathbf{e}_t\|$ varies over time in selected ALFWorld and WebShop rollouts. Since AAA assigns credit using the innovation extracted from the observation reached after each action, $\|\mathbf{e}_t\|$ can be viewed as a proxy for how much new semantic information the transition reveals relative to the previous trajectory representation. The non-uniform patterns support assigning step-wise credit according to observation-level novelty, rather than spreading the terminal success signal uniformly over the entire trajectory.

ALFWorld trajectory example

Task. Cool some pan and put it in diningtable.

Step 1 `Your current observation is: -= Welcome to TextWorld, ALFRED! =- You are in the middle of a room. Looking quickly around you, you see a cabinet 11, a cabinet 10, a cabinet 9, a cabinet 8, a cabinet 7, a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 2, a countertop 1, a diningtable 1, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.`
`action: go to diningtable 1`

Step 2 `your current observation is: You arrive at diningtable 1. On the diningtable 1, you see nothing.`
`action: go to countertop 1` ($\|\mathbf{e}_t\| = 12.08$)

Step 3 `your current observation is: You arrive at countertop 1. On the countertop 1, you see a bread 2, a butterknife 2, a cup 1, a dishsponge 3, a dishsponge 1, a fork 2, a lettuce 1, a mug 2, a pan 1, a plate 1, a spatula 3, and a spatula 1.`
`action: take pan 1 from countertop 1` ($\|\mathbf{e}_t\| = 12.11$)

Step 4 `your current observation is: You pick up the pan 1 from the countertop 1.`
`action: go to fridge 1` ($\|\mathbf{e}_t\| = 3.75$)

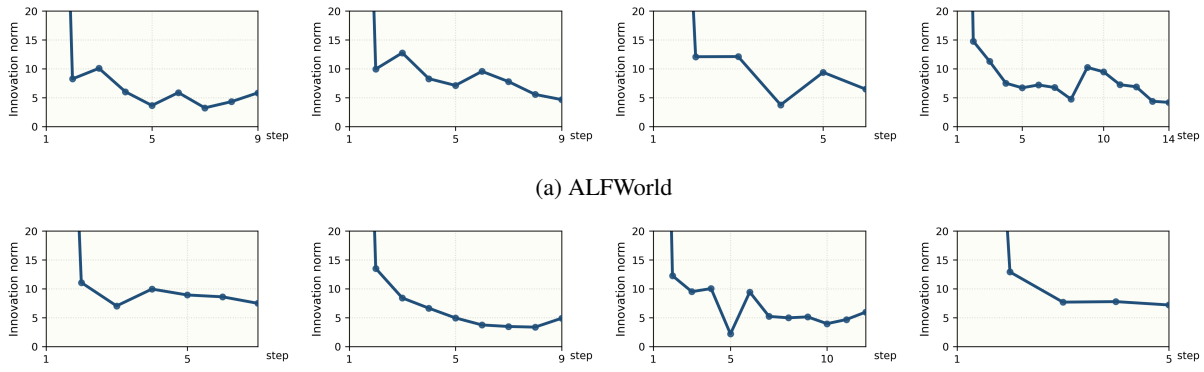
Step 5 `your current observation is: You arrive at fridge 1. The fridge 1 is closed.`
`action: cool pan 1 with fridge 1` ($\|\mathbf{e}_t\| = 9.38$)

Step 6 `your current observation is: You cool the pan 1 using the fridge 1.`
`action: go to diningtable 1` ($\|\mathbf{e}_t\| = 6.46$)

Step 7 `your current observation is: You arrive at diningtable 1. On the diningtable 1, you see nothing.`
`action: move pan 1 to diningtable 1`

ALFWorld trajectory-level interpretation The following ALFWorld example expands one rollout from Fig. 7 in prompt format. Because the agent selects the next action after reading the current observation, we place each $\|\mathbf{e}_t\|$ value next to the observation reached as the result of the immediately preceding action. Thus, the value reflects the residual magnitude of the newly observed state information, rather than the novelty of the action string itself.

This example highlights that the innovation norm does not directly score the importance of an action itself. Instead, it measures how much semantic



(a) ALFWorld

(b) WebShop

Figure 7: Innovation norm $\|\mathbf{e}_t\|$ over time steps for selected rollouts in (a) ALFWorld and (b) WebShop. Each small panel shows a single trajectory. The innovation magnitude varies substantially across steps, supporting step-wise credit assignment based on observation-level novelty. The degenerate first-step value is clipped at 20 for visibility.

residual the observation reached after that action contributes beyond the previous trajectory representation. For example, the low value at Step 5 does not imply that "go to fridge 1" is unimportant; rather, the resulting observation, "You arrive at fridge 1. The fridge 1 is closed," is a relatively predictable state transition given the initial observation and the navigation action. In contrast, the pan-acquisition observation at Step 4 and the cooled-pan observation at Step 6 reveal task-relevant state changes more directly, and therefore receive larger innovation norms.

Overall, this qualitative example illustrates that $\|\mathbf{e}_t\|$ is best understood as a signal for newly acquired observation-level semantic information, rather than as a direct score of which action appears important to a human reader.

D.2 Task Information Cones: Subspace Overlap Index Γ

Diagnostic setup To examine whether the TIC view is reflected in actual agent trajectories, we conduct a diagnostic subspace-overlap analysis on ALFWorld rollouts. This analysis is not part of the AAA training pipeline itself; rather, it uses rollouts and hidden-state dumps generated with the ETO evaluation codebase (Song et al., 2024). Thus, the exact prompt template and tokenization may differ from those used in GiGPO and our training codebase, but the diagnostic targets the same semantic location: the boundary after the agent has observed the current state and before it generates the next action. We generate up to five ReAct rollouts with Qwen3-4B-Instruct-2507 (Yang et al., 2025) for each of 133 ALFWorld tasks and split them into successful and failed trajectories using the environment success label.

Hidden-state extraction For each step, we extract the last-layer hidden state at the chat-template transition boundary immediately before the assistant begins producing its next thought and action. The two codebases instantiate this boundary with slightly different prompt templates. In the ETO-style format used for this diagnostic, the current observation appears in the user block, followed by the user-block terminator and the assistant-start boundary:

ETO-style rollout format

```
<|im_start|>user
Observation: On the desk 2, you see a bowl 1
. . .
<|im_end|>\n        ⟵ extract last-layer hidden state
<|im_start|>assistant\n
Thought: I found a bowl 1 . . .
Action: take bowl 1 from desk 2
<|im_end|>
```

In the GiGPO implementation used for AAA training, the user block contains the task instruction, previous observation-action history, the current observation, and admissible actions. The extraction point is still the assistant-start boundary after this user-side context:

GiGPO / AAA training format

```
<|im_start|>user
You are an expert agent . . .
Prior to this step, you have already taken 14
step(s).
[Observation 13: . . . Action 13: . . . ]
[Observation 14: . . . Action 14: . . . ]
You are now at step 15 and your current
observation is: On the desk 2, you see a bowl
1.
Your admissible actions are: . . .
Now it is your turn to take an action.
<|im_end|>\n        ⟵ extract last-layer hidden state
<|im_start|>assistant\n
Thought: I found a bowl 1 . . .
Action: take bowl 1 from desk 2
```

These examples illustrate that the extracted vector summarizes the observation-conditioned context immediately before the next thought and action are generated. Because the same boundary appears at every step in a multi-step ReAct trajectory, it provides a consistent step-wise semantic representation for comparing rollouts.

Subspace overlap index For each task, we form all available pairs of rollouts within that task and categorize them as success-success (S-S), success-failure (S-F), or failure-failure (F-F). We do not compare rollouts from different tasks. For a pair (τ^i, τ^j), we stack the extracted step hidden states into trajectory matrices

$$\mathbf{H}^i = [\mathbf{h}^i_1, \ldots, \mathbf{h}^i_{T_i}], \qquad \mathbf{H}^j = [\mathbf{h}^j_1, \ldots, \mathbf{h}^j_{T_j}],$$

where $\mathbf{H}^i \in \mathbb{R}^{d \times T_i}$ and $\mathbf{H}^j \in \mathbb{R}^{d \times T_j}$. We then compute the effective rank of $\mathbf{H}^i$, $\mathbf{H}^j$, and the concatenated matrix $[\mathbf{H}^i, \mathbf{H}^j]$ by singular value decomposition. If the singular values of a matrix are $\sigma_1, \ldots, \sigma_T$, its effective rank is the smallest k that explains 90% of the squared singular-value mass:

$$\mathrm{rank}_e := \min \left\{ k : \sum_{\ell=1}^{k} \sigma_\ell^2 \geq 0.9 \sum_{\ell=1}^{T} \sigma_\ell^2 \right\}.$$

The subspace overlap index is then defined as

$$\Gamma = \frac{\mathrm{rank}_e([\mathbf{H}^i, \mathbf{H}^j])}{(\mathrm{rank}_e(\mathbf{H}^i) + \mathrm{rank}_e(\mathbf{H}^j))/2}.$$

When $\mathcal{L}(\mathbf{H}^i)$ and $\mathcal{L}(\mathbf{H}^j)$ largely overlap, the concatenated matrix requires little additional effective rank, so Γ is close to 1. Larger values indicate that the two trajectories require a larger joint effective subspace. Because rank_e is defined by a 90% energy threshold, Γ should be interpreted as a diagnostic index rather than as an exact rank-theoretic quantity.

Task-level aggregation Figure 3 visualizes a distribution over tasks, rather than a distribution over individual rollout pairs. Since the number of successful and failed rollouts can vary across tasks, we first average the pairwise Γ values within each task and pair type, obtaining one task-level overlap index whenever that pair type is available. We then plot the resulting task-level indices for S-S, F-F, and S-F as violin distributions with box and scatter overlays.

Using the extraction boundary described above and final-layer hidden states, the final figure contains 78 S-S task-level indices, 88 F-F indices, and 60 S-F indices, with mean task-level Γ values of 1.271, 1.423, and 1.456, respectively. The resulting separation shows that successful rollouts for the same task occupy more aligned task-relevant semantic subspaces, whereas pairs involving failed rollouts require a larger joint effective subspace. This trend supports the TIC view that task success is associated with shared semantic directions in the model's trajectory representations.

D.3 Sample Efficiency

We additionally compare the learning dynamics of AAA-augmented training against the corresponding base algorithms. While the main paper reports final performance under the same training budget, this analysis examines whether AAA also improves the rate at which the policy learns from sparse terminal rewards.

We compare training rollout performance for GRPO and GiGPO with and without AAA. As shown in Figure 8, across both environments and all three metrics, GRPO+AAA, GiGPO, and GiGPO+AAA surpass GRPO from early in training. On WebShop, GRPO+AAA is particularly notable: it closely tracks GiGPO and GiGPO+AAA on both task score and success rate, and even surpasses GiGPO on the score curveindicating that the innovation-based reward decomposition alone, without the additional step-level grouping of GiGPO, is sufficient to recover much of GiGPO's sample-efficiency gain. On ALFWorld, GiGPO and GiGPO+AAA converge to similar training-time rollout levels, consistent with the saturation effect at this model scale; the gain from AAA is more clearly visible in the validation evaluation (Table 1, +4.9pp). Overall, AAA provides a useful intermediate training signal that accelerates learning under sparse terminal supervision, with the strongest gains in the GRPO+AAA configuration.

To quantify learning speed, we report the area under the learning curve (AUC) and the number of environment interactions required to reach a fixed success threshold. These metrics complement the final-checkpoint results by measuring whether AAA accelerates policy improvement throughout training rather than only improving the final model.

D.4 Innovation–Progress Alignment

Setup. We categorize each interaction step as a progress, redundant, or invalid step using environment feedback. This annotation is used only for analysis and never during training. For ALFWorld, a step is labeled *progress* if it changes a goal-relevant predicateacquiring the target object, opening a required receptacle, applying a required transformation (cleaning, heating, or cooling), or placing the object in the target location. A step is labeled *redundant* if it repeats a previous observation, revisits an already explored state without changing any goal-relevant predicate, or performs an irrelevant action. *Invalid* steps are those for which the environment returns an invalid-action response or no state change. For WebShop, progress steps correspond to milestone transitions such as issuing a relevant search query, clicking the target item, selecting required product options, or purchasing the item; redundant steps include repeated searches, irrelevant item clicks, and option changes that do not move the trajectory closer to the target purchase.

Ablation variants To isolate the source of the task-aligned signal, we compare three scoring variants that share the same hidden representations and per-step labels:

- AAA: the full innovation reward, $\frac{1}{2}\log\big(1 + \|e_t\|^2\big)$ with $e_t = v_t - \mathrm{proj}_Q(v_t)$, combining the random projection and the Q-residual.
- NOQ: $\frac{1}{2}\log\big(1+\|v_t\|^2\big)$, the random projection only, with the Q-residual removed.
- RAW: $\|h_{t+1} - h_t\|$, the raw hidden-state semantic delta.

Result This analysis asks whether high-innovation steps tend to coincide with interpretable goal-progress events. We report two metrics: the average innovation reward for each step category, and the enrichment of progress steps among the top-$q\%$ highest-scoring steps (precision relative to the progress-step base rate). Progress steps consistently receive larger innovation rewards than redundant or invalid steps, and the full AAA reward concentrates progress steps far above chance at small q. Removing the Q-residual (NOQ) or using the raw hidden-state delta (RAW) reduces the enrichment to near chance, confirming that the task-relevant signal is carried by the Q-residual rather than by raw semantic change.

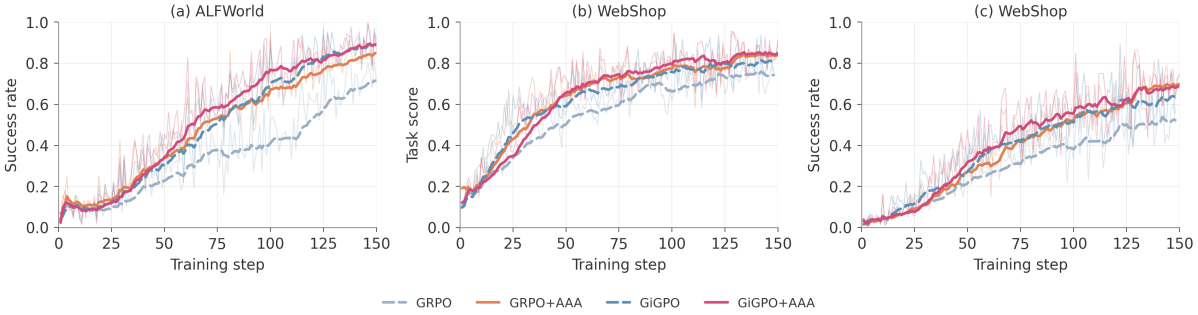


Figure 8: **Sample efficiency of AAA.** EMA-smoothed training rollout curves ($\alpha = 0.1$) on ALFWorld and WebShop comparing GRPO and GiGPO with their AAA-augmented counterparts. Baselines are shown as dashed lines and AAA variants as solid lines. For each method, we plot a single representative training run on Qwen2.5-1.5B-Instruct; the relative ordering at convergence is consistent with the multi-seed validation results in Table 1.