USING EXPERIENCE TWICE: DYNAMIC PROCESS RUBRICS FOR PRIVILEGED TOKEN-LEVEL CREDIT ASSIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent trajectories often contain both critical successful decisions and failed attempts; therefore, optimization with sparse trajectory-level rewards struggles to indicate which intermediate decisions to reinforce or suppress. We observe that meaningful distinctions among on-policy trajectories can be distilled into natural-language process rubrics and used as process supervision signals. Based on this insight, we propose ONLINE RUBRIC SELF-DISTILLATION(ORSD), which leverages self-derived process rubrics for dense token-level supervision, thereby augmenting outcome-centric reinforcement learning. Specifically, at each update, the current policy independently generates multiple rubric bundles from the same group of trajectories, extracting task-specific candidate process criteria from their cross-trajectory behavioral differences. A frozen verifier then evaluates all rubric criteria against each trajectory in the group, and selects one candidate bundle based on bundle-level alignment between rubric satisfaction and task success. ORSD treats the selected rubric bundle as privileged information during training, then compares the log-likelihoods of the original tokens with and without the rubric. The resulting likelihood gap provides token-specific process credit, which is added to the original trajectory-level GRPO advantage through additive advantage shaping. ORSD thus bridges trajectory-level outcome supervision and fine-grained process supervision, enabling token-level credit assignment using process signals. Extensive experiments on ALFWorld and WebShop show that ORSD accelerates policy improvement and boosts final task performance for both Qwen3-1.7B and Qwen3-4B under the same environment-interaction budget.

1 INTRODUCTION

Reinforcement learning (RL) provides a framework for learning decision-making policies from reward feedback obtained through interaction with an environment (Sutton & Barto, 2018). In outcome-based training of LLM agents, each trajectory receives a reward based on its final task outcome. Even a successful trajectory can contain useful steps, redundant actions, and mistakes that the agent later corrects (Yao et al., 2022; Shridhar et al., 2021). When GRPO is applied with trajectory-level rewards, it computes an advantage for each trajectory in the sampled group, and all decision steps within that trajectory inherit the same advantage (Shao et al., 2024). Useful steps and redundant actions thus receive the same advantage, motivating finer-grained credit assignment that distinguishes their roles in completing the task.

Recent studies have refined this coarse-grained credit assignment by introducing natural-language rubrics as a source of fine-grained supervision. Rubric-based RL aggregates explicit criteria into response rewards; Rubrics-to-Tokens learns criterion–token relevance; stepwise rubric evaluation applies criteria to individual reasoning steps (Gunjal et al., 2026; Xu et al., 2026; Xie et al., 2026). In these settings, rubric supervision is commonly prepared in an offline manner, through human-written criteria or generation by stronger models, and thus remains fixed during training and requires additional supervision. This inherent limitation raises an intuitive question: can we dispense with reliance on offline-generated rubrics and instead generate rubrics online during policy training?

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We answer this question with a core insight: meaningful distinctions among on policy trajectories can be leveraged to generate process rubrics. The trajectories in the same group expose different search paths, constraint violations, recoveries and outcomes, allowing the current policy to propose task-specific process rubrics from them. However, these candidate process rubrics exhibit substantial heterogeneity in quality. A dedicated selection mechanism is therefore necessary to identify and retain the most informative rubrics. The filtered rubrics then act as privileged information, and the difference between the policy's likelihood generated with and without this privileged information serves as token-specific process credit.

We introduce ONLINE RUBRIC SELF-DISTILLATION (ORSD) to implement the above idea. As in GRPO, ORSD samples a group of trajectories for the same task from the current policy, obtains their outcome rewards and trajectory-level advantage for each trajectory. ORSD reuses these trajectories to construct process supervision. Conditioned on both task prompt and the group of trajectories, the same policy independently generates multiple rubric bundles, each containing multiple criteria for evaluating intermediate states along the trajectories. To identify the highest-quality bundle, ORSD employs a verifier to assess the satisfaction of each rubric criterion on every trajectory, producing a satisfaction profile for each rubric bundle. Then the alignment between each satisfaction profile and the outcome rewards is measured and the rubric bundle with the highest correlation is selected as privileged information. To obtain token-level credit, ORSD augments the context with the selected privileged information and computes the token likelihoods of the sampled trajectories. The likelihood gap relative to the original and augmented token likelihoods is then used to compute the token-level advantage. As a result, ORSD transforms a group of on-policy trajectories into token-level credit, requiring neither additional rollouts nor additional environment interactions.

Our contributions are summarized as follows:

- We propose ONLINE RUBRIC SELF-DISTILLATION (ORSD), a framework that uses online-generated process rubrics to derive token-level credit. These rubrics are distilled from the policy's own rollouts and serve as privileged information that underpins this fine-grained credit allocation.
- To address the quality variation among generated rubrics, we propose a rubric-reward alignment mechanism that filters out low-value rubrics by measuring the alignment between rubric satisfaction and task success, thereby enhancing the quality of token-level credit.
- Experiments on ALFWorld and WebShop with Qwen3-1.7B and Qwen3-4B show that ORSD improves overall success over GRPO under the same environment-interaction budget. Ablations support the contributions of online rubric generation and rubric–reward alignment.

2 RELATED WORK

RLVR for LLM Agents. Reinforcement learning with verifiable rewards (RLVR) uses automatically verified outcomes to train language models. PPO and GRPO are commonly used for policy optimization, with GRPO estimating relative advantages within groups of sampled responses (Schulman et al., 2017; Shao et al., 2024). For LLM agents, training involves multiple rounds of interaction, during which policy's decisions affect subsequent observations and the final reward. RAGEN examines learning dynamics in multi-turn agent RL, and Agent Lightning provides a framework for training agents from their execution traces (Wang et al., 2025; Luo et al., 2025). The environment interaction also supports additional learning signals. EnvRL uses state transitions for representation learning, while T^2PO adjusts exploration through uncertainty estimates and turn-level resampling (Wang et al., 2026d;a).

Credit Assignment in Agentic RL. Credit assignment assigns different credit to individual steps or tokens in an agent trajectory. GiGPO groups steps across trajectories by state similarity and estimates relative advantages within each group (Feng et al., 2025). HGPO further groups steps according to their interaction contexts (He et al., 2026). ArCHer uses rewards from subsequent interactions to estimate the value of the entire response generated at the current turn, then uses this estimate to update the policy that generates the response (Zhou et al., 2024). POAD further considers the role of individual tokens within an action: it treats action generation as a sequence of token-level decisions and estimates future rewards for each token, providing a finer-grained training signal (Wen

et al., 2024). SC-GRPO conditions on verified solutions and weights token gradients according to the resulting distributional divergence (Shan et al., 2026).

Natural-language Rubrics. A natural-language rubric is a set of written criteria specifying which aspects of a response or reasoning process to evaluate. Prior work uses rubrics to provide feedback at different levels: Rubrics as Rewards assigns a reward to the entire response based on rubric satisfaction (Gunjal et al., 2026); Rubrics to Tokens estimates the relevance of each criterion to individual response tokens, providing token-level feedback (Xu et al., 2026); and Step-wise Rubric Rewards checks whether each reasoning step satisfies the criteria and assigns step-level rewards (Xie et al., 2026). These methods use rubrics to construct fine-grained supervision, with criteria generated offline and held fixed. We further investigate how to generate rubrics online that better match the current policy and use them for token-level credit assignment.

On-policy Distillation with Privileged Context. Knowledge distillation trains a student model using teacher predictions, with supervision applied to individual tokens or complete sequences (Hinton et al., 2015; Kim & Rush, 2016). Training can also use data generated by the learner itself: DAgger collects supervision at states visited by the learner's policy (Ross et al., 2011), while GKD (Agarwal et al., 2024) trains on student-generated sequences with supervision provided by a teacher. OPSD (Zhao et al., 2026) uses the same model for self-distillation: the model first generates a response, then computes its log-probability given additional information to provide a training signal for the original policy. This additional information is called privileged context. Prior work also extracts such information from task execution records: Skill-SD (Wang et al., 2026b) summarizes experience from completed trajectories into natural-language skills, while SDPO (Hübotter et al., 2026) uses environmental feedback and successful attempts for self-distillation.

3 METHOD

In this section, we present the proposed method, Online Rubric Self-Distillation (ORSD), a framework that turns same-task on-policy rollouts into token-level process supervision in detail. The pipeline of ORSD is presented in Figure 1. At each training step, the current policy first samples a group of trajectories for the same task and receives their rewards. It then generates candidate rubric bundles from the task description and these trajectories. The rubric-reward alignment mechanism selects the bundle whose rubric satisfaction scores correlate most strongly with the rewards across the group. Finally, the token-wise log-likelihood difference between cases with and without the selected bundle enables per-token advantage shaping of the original trajectory-level advantage. In this way, ORSD reuses the same on-policy trajectories to derive token-specific credit without introducing additional environment interactions.

3.1 PROBLEM SETUP

Let π_{old} denote the policy used to sample trajectories in the current update. For a task x, ORSD samples N trajectories:

$$\mathcal{T} = \{\tau_i\}_{i=1}^{N}, \qquad \tau_i \sim \pi_{\text{old}}(\cdot \mid x). \tag{1}$$

The reward for trajectory τ_i is $r_i = R_{\text{train}}(\tau_i)$, as defined in Section 4.1. As in GRPO, ORSD normalizes rewards within each task group to obtain a trajectory-level advantage, which serves as the base advantage for subsequent token-level advantage shaping (Shao et al., 2024):

$$A_i^{\text{GRPO}} = \frac{r_i - \mu_r}{\sigma_r + \varepsilon_{\text{num}}}. \tag{2}$$

Here, μ_r and σ_r are the mean and sample standard deviation of the group rewards. ORSD will add a token-level correction to this advantage.

3.2 ONLINE RUBRIC BUNDLE GENERATION AND SELECTION

Building on the same-task rollout group mentioned above, ORSD further exploits the behavioral diversity across trajectories to construct process supervision. For example, in WebShop, one trajectory may verify product attributes before purchasing, whereas another may skip this step and end up

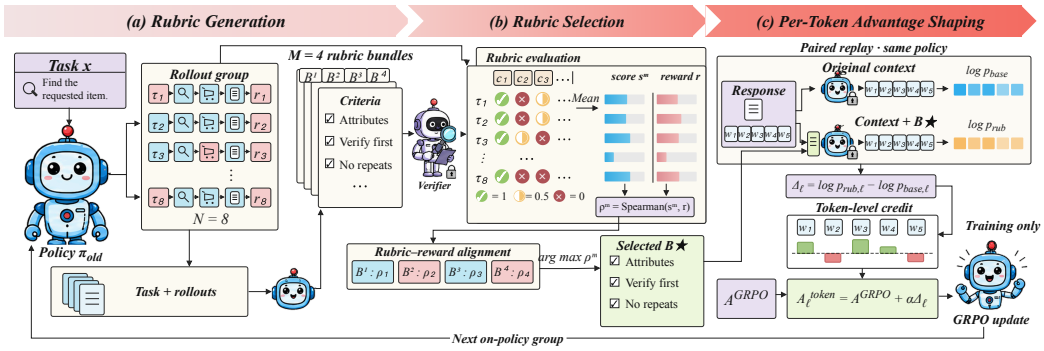

Figure 1: The overview of ORSD. (a) For each task, the policy first samples a group of trajectories. It then takes the task description and these trajectories as inputs to generate multiple rubric bundles. (b) Each Rubric bundle is evaluated on every trajectory, the rubric bundle with the largest rubric-reward alignment score is selected. (c) The selected rubric bundle is used to augment the context and compute the per-token likelihood gap between augmented and non-augmented contexts, enabling per-token advantage shaping of the original trajectory-level advantage.

selecting an item that does not satisfy the user's request. These behavioral differences can be summarized as natural-language process rubrics. At each training step, we provide the current policy with the task description x, the sampled trajectory group $\mathcal{T}$, and a rubric-generation instruction p_r, and independently sample M candidate rubric bundles:

$$\mathcal{B}^{(m)} \sim \pi_{\text{old}}(\cdot \mid x, \mathcal{T}, p_r), \qquad m = 1, \ldots, M. \tag{3}$$

Each bundle contains a set of process criteria, denoted by $\mathcal{B}^{(m)} = \{\kappa_{m,k}\}_{k=1}^{K_m}$, where K_m is the number of criteria in the bundle.

Different from prior works, our method produces candidate rubric bundles online using the policy instead of human-defined criteria or generation by stronger models. This mechanism enables the generated rubrics to capture the dynamics of on-policy trajectories, while the challenge is that candidate rubric bundles exhibit variable quality. To address this, we select one rubric bundle among candidates based on a rubric–reward alignment score. Specifically, a verifier V first scores each trajectory against each criterion $\kappa_{m,k}$:

$$S_{i,m,k} = V(x, \tau_i, \kappa_{m,k}) \in [0, 1]. \tag{4}$$

Averaging the criterion scores in bundle m gives its satisfaction score for trajectory i:

$$s_i^{(m)} = \frac{1}{K_m} \sum_{k=1}^{K_m} S_{i,m,k}. \tag{5}$$

We then compute the Spearman correlation between bundle satisfaction and outcome rewards across all N trajectories for the task:

$$\rho^{(m)} = \text{Spearman}\left(\{s_i^{(m)}\}_{i=1}^N, \{r_i\}_{i=1}^N\right). \tag{6}$$

This correlation measures the agreement between the rankings in $\{s_i^{(m)}\}_{i=1}^N$ and $\{r_i\}_{i=1}^N$. Higher alignment indicates that the bundle better distinguishes higher- and lower-reward trajectories. Note that to prevent the generated rubrics from degenerating into simple rules equivalent to outcome rewards, Equation 3 does not take outcome rewards as input when generating rubric bundles. We finally select the bundle with the highest correlation:

$$m^\star = \arg\max_m \rho^{(m)}, \qquad \mathcal{B}^\star = \mathcal{B}^{(m^\star)}. \tag{7}$$

3.3 Rubric-Based Advantage Shaping

The selected rubric bundle serves as privileged context for computing token-level credit. For each policy-generated token $y_{i,\ell}$ at position ℓ in trajectory τ_i, let $y_{i,<\ell}$ denote the preceding interaction

history, including earlier policy-generated tokens and environment observations. We compare the log-likelihoods assigned by π_{old} to the same token with and without the selected rubric:

$$\Delta_{i,\ell} = \log \pi_{\text{old}}(y_{i,\ell} \mid x, \mathcal{B}^\star, y_{i,<\ell}) - \log \pi_{\text{old}}(y_{i,\ell} \mid x, y_{i,<\ell}). \tag{8}$$

Both passes score the same stored responses without resampling. We compute the correction only for policy-generated tokens and mask environment observations. A positive $\Delta_{i,\ell}$ increases the token's advantage, while a negative value decreases it. The resulting token-level advantage is

$$A_{i,\ell}^{\text{token}} = A_i^{\text{GRPO}} + \alpha \Delta_{i,\ell}. \tag{9}$$

The coefficient $\alpha \geq 0$ controls the correction magnitude. Note that the per-token advantage $A_{i,\ell}^{\text{token}}$ is derived from trajectories generated by the policy itself. For this reason, we categorize ORSD as a self-distillation mechanism that enables fine-grained credit assignment.

3.4 Policy Optimization

We optimize the policy using the token-level advantage $A_{i,\ell}^{\text{token}}$ defined in Eq. equation 9. For each policy-generated token, we define the policy ratio as

$$r_{i,\ell}(\theta) = \frac{\pi_\theta(y_{i,\ell} \mid x, y_{i,<\ell})}{\pi_{\text{old}}(y_{i,\ell} \mid x, y_{i,<\ell})}. \tag{10}$$

The policy is optimized with the clipped objective

$$\begin{aligned} \mathcal{L}_{\text{total}}(\theta) = & -\mathcal{A}_{\text{mb}}\big[\min\big\{r_{i,\ell}(\theta) A_{i,\ell}^{\text{token}}, \text{clip}(r_{i,\ell}(\theta), 1-\varepsilon_c, 1+\varepsilon_c)\, A_{i,\ell}^{\text{token}}\big\}\big] \\ & + \eta \mathcal{R}_{\text{KL}}(\theta). \end{aligned} \tag{11}$$

Here, ε_c is the clipping threshold, $\mathcal{R}_{\text{KL}}$ denotes the KL regularization with respect to the reference policy, and η controls its strength. $\mathcal{A}_{\text{mb}}$ denotes masked averaging over valid policy-generated tokens within each micro-batch.

Algorithm 1 summarizes each training update.

Algorithm 1 Online Rubric Self-Distillation (ORSD)

Given: Policy π_θ, task distribution $\mathcal{D}$, verifier V
Parameters: Trajectories per task N, rubric candidates M, shaping coefficient α, generation instruction p_r
for each training update **do**
  Set $\pi_{\text{old}} \leftarrow \pi_\theta$ and sample a task batch $\mathcal{X}$ from $\mathcal{D}$
  for each task $x \in \mathcal{X}$ **do**
    Sample $\mathcal{T} = \{\tau_i\}_{i=1}^N$, where $\tau_i \sim \pi_{\text{old}}(\cdot \mid x)$
    Collect rewards $\mathbf{r} = (r_1, \ldots, r_N)$ and compute A_i^{GRPO} using Eq. (2)
    Independently sample M bundles $\{\mathcal{B}^{(m)}\}_{m=1}^M$ from $\pi_{\text{old}}(\cdot \mid x, \mathcal{T}, p_r)$
    Compute criterion scores $S_{i,m,k} = V(x, \tau_i, \kappa_{m,k})$ for each trajectory and candidate bundle
    Average over criteria to obtain $s_i^{(m)}$ using Eq. (5)
    Compute $\rho^{(m)}$ between $\{s_i^{(m)}\}_{i=1}^N$ and $\mathbf{r}$ using Eq. (6)
    Select $m^\star = \arg\max_m \rho^{(m)}$ and retain the complete bundle $\mathcal{B}^\star = \mathcal{B}^{(m^\star)}$
    Score the same stored responses under π_{old} with and without $\mathcal{B}^\star$, without resampling
    Compute $\Delta_{i,\ell}$ at policy-generated token positions using Eq. (8)
    Set $A_{i,\ell}^{\text{token}} \leftarrow A_i^{\text{GRPO}} + \alpha \Delta_{i,\ell}$
  end for
  Update θ on the collected batch using Eq. (11), with original contexts and token advantages held constant
end for

4 Experiments

4.1 Experimental Setup

Benchmarks and Metrics. We evaluate ORSD on two long-horizon interactive benchmarks using Qwen3-1.7B and Qwen3-4B (Yang et al., 2025). ALFWorld (Shridhar et al., 2021) requires an agent

to navigate household environments and manipulate objects through text actions. Tasks include locating objects and completing operations such as cleaning, heating, or cooling before placing them at a target location. WebShop (Yao et al., 2022) requires the agent to search for products, inspect their attributes, and complete a purchase that satisfies a user instruction. Both environments require a sequence of decisions conditioned on observations from earlier actions. Our ALFWorld evaluation uses the unseen split; we report success rates for six task types (Pick, Look, Clean, Heat, Cool, and Pick2) and overall success. For WebShop, we report the official task score and success rate.

Prompting Baselines. We include published results for prompting methods that guide interaction without RL training. ReAct (Yao et al., 2023) interleaves reasoning and actions. Reflexion (Shinn et al., 2023) uses reflections on previous attempts to guide later attempts, while ReflAct (Kim et al., 2025) relates the current state to the task goal when choosing actions. ReAct w/ Self-Guidance (SG) adds self-generated guidance during interaction (Wang et al., 2026c).

Training Baselines. We compare ORSD with several baselines. (a) GRPO: its training data, reward definition, thinking mode, rollout count, and the number of training steps are strictly consistent with ORSD. (b) Self-distillation methods, including OPSD (Zhao et al., 2026), GRPO+OPSD, and SDAR (Lu et al., 2026). GRPO+OPSD combines GRPO with an auxiliary distillation loss. The source of numbers and evaluation protocols is specified in the notes to Table 1.

Training Details. We fine-tune all parameters of both backbones with VERL/HybridFlow (Sheng et al., 2025). Each run uses four A100 80GB GPUs. Every update contains eight tasks with eight rollouts per task. We train for 300 training steps at a learning rate of 10^{-6}. WebShop episodes allow for at most 15 assistant actions, while ALFWorld interactions end according to the environment's termination conditions. We generate $M = 4$ candidate rubric bundles per task and use $\alpha = 0.01$, $\varepsilon_c = 0.2$, and $\eta = 10^{-3}$ by default. Appendix C provides the remaining implementation details.

Reward Definition For ALFWorld, the training reward is the binary task success indicator:

$$R_{\text{train}}(\tau) = \mathbb{I}[\text{success}], \tag{12}$$

These rewards also form the outcome vector used for bundle selection. For WebShop, we use

$$\begin{aligned} R_{\text{train}}(\tau) = 0.5R_{\text{env}}(\tau) + 0.5\mathbb{I}[R_{\text{env}}(\tau) = 1] \\ - 0.05\frac{N_{\text{invalid}}(\tau)}{\max(T,1)} - 0.025\frac{N_{\text{loop}}(\tau)}{\max(T,1)}. \end{aligned} \tag{13}$$

Here, $R_{\text{env}}(\tau) \in [0,1]$ is the official WebShop task score, and T counts all attempted environment actions, including invalid ones. N_{invalid} and N_{loop} count invalid actions and repeated actions that make no progress. This training reward is used to compute GRPO advantages and select rubrics on WebShop. Baselines run in this study use the same reward transformation. Evaluation uses the original official metrics.

4.2 Main Results

ORSD improves overall success over GRPO on both benchmarks with both Qwen3 backbones (Table 1). On ALFWorld, the gain is larger for Qwen3-1.7B, reaching 12.12 percentage points; on WebShop, the 4B model continues to benefit, with a 10.90-point increase in success rate. These improvements are obtained with the same environment-interaction budget as GRPO, supporting the use of existing rollout experience to provide additional training supervision.

4.3 Ablation Study

In this section, we use ablation experiments to examine the effectiveness of ORSD's components: online rubric generation, the number of candidate bundles and their selection, and the advantage shaping coefficient α. All ablations use WebShop. We study rubric generation and selection with both Qwen3 backbones and vary α with Qwen3-1.7B.

Table 1: Performance on ALFWorld and WebShop. Bold indicates the best result for each metric.

Model	Method	ALFWorld							WebShop	
		Pick	Look	Clean	Heat	Cool	Pick2	Overall	Score	Succ.
Qwen3 1.7B	ReAct§	24.20	12.50	0.00	0.00	0.00	12.10	10.20	41.60	4.70
	Reflexion§	22.90	15.40	14.80	18.80	16.00	8.30	16.40	39.80	16.00
	ReflAct§	28.60	23.10	7.40	6.30	8.00	0.00	12.90	37.50	4.40
	ReAct w/ SG§	54.20	14.30	18.50	0.00	12.10	19.00	21.10	39.70	5.40
	OPSD†	26.30	33.30	9.10	0.00	4.50	5.30	14.10	47.40	9.30
	GRPO+OPSD†	38.20	50.00	30.80	28.60	30.00	21.10	32.00	70.70	38.30
	SDAR†	73.50	25.00	76.90	33.30	40.00	36.80	53.90	76.80	58.60
	GRPO	68.75	**72.22**	50.81	54.35	36.90	**50.00**	53.92	72.20	55.40
	GiGPO	79.20	27.80	**77.40**	34.80	38.10	41.20	53.00	76.80	60.20
	ORSD (ours)	**81.25**	68.06	69.35	**79.35**	**53.57**	41.18	**66.04**	**78.60**	**63.00**
Qwen3 4B	ReAct§	42.40	16.70	0.00	8.30	9.10	28.00	20.30	31.10	11.30
	Reflexion§	68.60	38.50	37.00	25.00	32.00	4.20	37.10	57.80	24.00
	ReflAct§	74.30	38.50	25.90	25.00	32.00	16.70	38.60	35.10	16.00
	ReAct w/ SG§	87.80	50.00	29.20	58.30	27.30	**80.00**	58.60	42.70	17.60
	GRPO	83.33	**86.11**	73.39	**72.83**	72.62	63.24	74.07	82.69	66.15
	GiGPO	**88.20**	50.00	**96.20**	52.40	65.00	57.90	73.40	81.80	65.60
	ORSD (ours)	83.33	77.78	87.10	70.65	**83.33**	63.24	**77.43**	**89.02**	**77.05**

§: Prompting results from *Co-Evolution of Policy and Internal Reward for Language Agents* (Wang et al., 2026c). †: OPSD, GRPO+OPSD, and SDAR results from *Self-Distilled Agentic Reinforcement Learning* (Lu et al., 2026). Published results retain their source evaluation protocols.

Effectiveness of Online Rubric Generation. To evaluate the effectiveness of online rubric generation, we construct an *Offline Rubric* variant. Before RL training, the initial policy generates one task-specific rubric bundle for each task, and these rubrics remain fixed throughout training. As shown in Table 2, Offline Rubric consistently improves over GRPO on both model scales, confirming that task-specific rubric supervision is already beneficial even when generated only once. However, it still falls short of online ORSD by 4.80 percentage points in success rate for both backbones. This gap highlights the value of refreshing the rubric from the current policy's rollouts, so that the supervision remains aligned with the policy's changing behavior and failure modes during training. Importantly, both variants generate rubrics automatically and require no manually authored criteria.

Effectiveness of Rubric-reward Alignment. ORSD scores candidate rubric bundles by their alignment with outcome rewards and selects the highest-scoring bundle for paired replay. To evaluate this design, we construct an *ORSD w/o Rubric–Reward Alignment* variant, which generates a single rubric bundle and uses it directly, omitting verifier scoring, alignment computation, and candidate selection. As shown in Table 2, full ORSD improves the WebShop score by 1.07 points and success rate by 2.15 percentage points over this variant with Qwen3-1.7B. These results support generating multiple rubric candidates and selecting among them using outcome feedback to improve the process supervision used for policy training.

Sensitivity to α. Figure 2 compares the default ORSD coefficient $\alpha = 0.01$ with $\alpha = 0.001$ and $\alpha = 0.1$ on WebShop using Qwen3-1.7B. We report success rate and use the same GRPO result as the baseline at each setting; GRPO corresponds to removing the correction in Eq. (9). All three coefficients outperform GRPO, with the default $\alpha = 0.01$ achieving the highest success rate of 63.00%. These results indicate that $\alpha = 0.01$ provides the most effective correction strength among the tested values, supporting its use as the default setting.

4.4 Further Analysis

Table 2: Component ablations on WebShop with two policy backbones. Baseline denotes GRPO. ORSD w/o Rubric–Reward Alignment generates one bundle and omits verifier scoring and candidate selection. Rubric-based variants use $\alpha = 0.01$.

	Qwen3-1.7B		Qwen3-4B	
Method	Score	Success (%)	Score	Success (%)
Baseline	72.20	55.40	82.69	66.15
Offline Rubric	74.35	58.20	85.39	72.25
ORSD w/o Rubric–Reward Alignment	77.53	60.85	83.86	67.60
ORSD ($M = 4$)	**78.60**	**63.00**	**89.02**	**77.05**

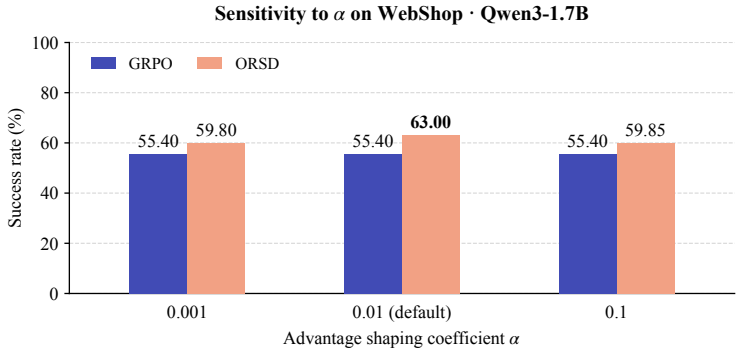

Figure 2: Sensitivity to α on WebShop with Qwen3-1.7B. The default is $\alpha = 0.01$. Each setting uses the same GRPO baseline (55.40% success).

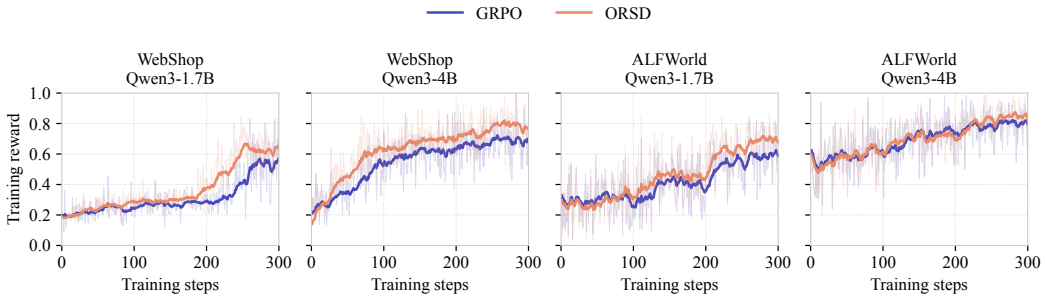

Figure 3: Training reward curves comparing ORSD with GRPO on WebShop and ALFWorld using Qwen3-1.7B and Qwen3-4B. Both methods use the same environment-interaction budget.

Training Dynamics Figure 3 compares the training rewards of ORSD and GRPO on both benchmarks and model scales. On WebShop, ORSD improves earlier with Qwen3-1.7B and maintains higher rewards through most of training with Qwen3-4B. On ALFWorld, the advantage is more pronounced at 1.7B and emerges later at 4B. Across all four settings, ORSD achieves higher mean rewards over the final 300 optimization steps under the same environment-interaction budget. Together with the gains in Table 1, these results support more effective use of sampled trajectories: ORSD reuses rollout experience to derive fine-grained supervision through self-distillation, improving learning without additional environment rollouts.

Analysis of Generated Rubric Bundles Table 3 compares a WebShop rubric saved at step 300 with a rejected proposal, preserving the original wording of both bundles. Both bundles cover product search, variant selection, and purchase checks. The rejected proposal states these checks in general

Table 3: WebShop rubric comparison. The left column contains a rubric saved at step 300; the right column contains a rejected proposal.

Saved rubric	Rejected rubric
WebShop: Buy a women's top with the requested attributes, color, and size for less than $30.	
1. Identify the correct product category and specific product attributes (e.g., slim fit, loose fit, color, size, price) needed to find the desired item. 2. Ensure the search query includes all relevant keywords and filters to narrow down the results effectively. 3. Search using the relevant product attributes and narrow the candidates accordingly. Do not commit to a product based only on a partial match in the title or description. 4. Verify that the selected product meets the price criteria (lower than $30.00) before proceeding to purchase. 5. Confirm that the product is correctly selected and the purchase action is initiated.	1. Identify the appropriate product category and the key attributes specified in the user instruction. 2. Use relevant product keywords and available search or filtering options to narrow down candidate items. 3. Select the appropriate product variant, such as the requested size, color, or other configurable attributes. 4. Verify that the selected product satisfies the important constraints in the instruction, including product attributes and price requirements. 5. Confirm that the final product matches the user's request and complete the purchase successfully.

terms, such as "key attributes" and "price requirements," while the saved rubric names fit attributes and specifies a price below $30.00. The explicit price threshold gives the verifier a concrete condition to check and preserves that task constraint in the context used for paired replay. This is a difference in the specificity of the process guidance; the learning benefit of candidate generation and selection is evaluated by the component ablation above.

5 Discussion and Limitations

Rubric Knowledge is not explicitly accumulated across updates. ORSD generates task-specific rubrics from the current rollout group at each update, but does not maintain a persistent repository of previously discovered criteria. How to retain, refine, and reuse useful rubrics across training stages and related tasks remains an open question. Extending ORSD with a rubric memory could support cumulative process knowledge while preserving adaptation to the evolving policy.

Outcome alignment does not fully resolve credit assignment. ORSD selects rubric bundles by their correlation with trajectory-level rewards. This objective favors rubrics that distinguish stronger trajectories, but does not directly optimize their usefulness for assigning credit within a trajectory. Future work could develop selection objectives that account for how rubric guidance affects individual decisions and subsequent policy improvement.

6 Conclusion

We introduced Online Rubric Self-Distillation (ORSD) to improve credit assignment in LLM agent reinforcement learning. ORSD generates task-specific process rubrics online from differences among on-policy trajectories, without manually authored or fixed offline criteria. Rubric-reward alignment selects the bundle whose satisfaction scores best correlate with outcomes. Through privileged paired replay, the selected rubric converts token-level likelihood gaps into token-level advantage corrections, providing fine-grained supervision without additional environment interactions. Experiments on ALFWorld and WebShop with Qwen3-1.7B and Qwen3-4B show the best overall performance among the compared methods, while ablations validate online rubric generation and candidate selection. These results highlight the value of deriving adaptive process supervision from the policy's own experience for credit assignment in long-horizon agent tasks.

AI USE STATEMENT

Generative AI assisted with manuscript drafting, language editing, literature cross-checking, consistency checks for equations and algorithms, and preparation of editable vector figures. The authors reviewed the resulting text, claims, citations, equations, and figures and take responsibility for the final content. AI-assisted algorithm schematics illustrate the procedure and do not report experimental measurements.

REPRODUCIBILITY STATEMENT

Appendix A gives the update sequence, Appendix B records the prompt contract, and Appendix C lists the implementation and sampling settings. Appendix D derives the form of the token-level correction. We provide anonymized code in the supplementary materials to support reproduction of our experiments.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5be69a584901a26c521c2b51e40a4c20-Abstract-Conference.html.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-Group Policy Optimization for LLM Agent Training. In *Advances in Neural Information Processing Systems*, volume 38, pp. 51797–51830, 2025. doi: 10.52202/085713-1544. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/420c9f777c0b4f78d515e53cf74d58b2-Abstract-Conference.html.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as Rewards: Reinforcement Learning Beyond Verifiable Domains. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/cfd7bee7a651ee9af525098ef67a9e45-Abstract-Conference.html.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-Groups Policy Optimization for Long-Horizon Agentic Tasks. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2602.22817.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. URL https://arxiv.org/abs/1503.02531.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement Learning via Self-Distillation. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. URL https://arxiv.org/abs/2601.20802.

Jeonghye Kim, Sojeong Rhee, Minbeom Kim, Dohyung Kim, Sangmook Lee, Youngchul Sung, and Kyomin Jung. ReflAct: World-grounded decision making in LLM agents via goal-state reflection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33433–33465, 2025. doi: 10.18653/v1/2025.emnlp-main.1697. URL https://aclanthology.org/2025.emnlp-main.1697/.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327. Association for Computational Linguistics, 2016. doi: 10.18653/v1/D16-1139. URL https://aclanthology.org/D16-1139/.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026. URL https://arxiv.org/abs/2605.15155v1.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent Lightning: Train ANY AI Agents with Reinforcement Learning. *arXiv preprint arXiv:2508.03680*, 2025. doi: 10.48550/arXiv.2508.03680. URL https://arxiv.org/abs/2508.03680.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, pp. 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017. doi: 10.48550/arXiv.1707.06347. URL https://arxiv.org/abs/1707.06347.

Yingyu Shan, Yuhang Guo, Zihao Cheng, Zeming Liu, Xiangrong Zhu, Xinyi Wang, Jiashu Yao, Wei Lin, Hongru Wang, and Heyan Huang. Learning from Own Solutions: Self-Conditioned Credit Assignment for Reinforcement Learning with Verifiable Rewards. *arXiv preprint arXiv:2606.18810*, 2026. doi: 10.48550/arXiv.2606.18810. URL https://arxiv.org/abs/2606.18810.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024. doi: 10.48550/arXiv.2402.03300. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A Flexible and Efficient RLHF Framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, 2025. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

Haixin Wang, Hejie Cui, Chenwei Zhang, Xin Liu, Shuowei Jin, Shijie Geng, Xinyang Zhang, Nasser Zalmout, Zhenyu Shi, and Yizhou Sun. T^2PO: Uncertainty-guided exploration control for stable multi-turn agentic reinforcement learning. In *International Conference on Machine Learning*, 2026a. URL https://arxiv.org/abs/2605.02178.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, and Honggang Qi. Skill-SD: Skill-Conditioned Self-Distillation for Multi-turn LLM Agents. *arXiv preprint arXiv:2604.10674*, 2026b. doi: 10.48550/arXiv.2604.10674. URL https://arxiv.org/abs/2604.10674.

Xinyu Wang, Hanwei Wu, Jingwei Song, Shuyuan Zhang, Jiayi Zhang, Fanqi Kong, Tung Sum Thomas Kwok, Xiao-Wen Chang, Yuyu Luo, Chenglin Wu, and Bang Liu. Co-evolution of policy and internal reward for language agents. *arXiv preprint arXiv:2604.03098*, 2026c. URL https://arxiv.org/abs/2604.03098v1.

Zhitong Wang, Songze Li, Hao Peng, Shuzheng Si, Yi Wang, Maosong Sun, and Juanzi Li. EnvRL: Learn from environment dynamics in agentic reinforcement learning. *arXiv preprint arXiv:2606.17680*, 2026d. doi: 10.48550/arXiv.2606.17680. URL https://arxiv.org/abs/2606.17680.

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. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025. URL https://arxiv.org/abs/2504.20073.

Muning Wen, Ziyu Wan, Jun Wang, Weinan Zhang, and Ying Wen. Reinforcing LLM agents via policy optimization with action decomposition. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://papers.neurips.cc/paper_files/paper/2024/hash/bc09efb501c801ed92e181e26a885c2d-Abstract-Conference.html.

Weichu Xie, Haozhe Zhao, Wenpu Liu, Yongfu Zhu, Liang Chen, Minghao Ye, Zirong Chen, Yuqi Xu, Shuai Dong, Ziyue Wang, Xinbo Xu, Kean Shi, Ruoyu Wu, Xiaoying Zhang, Wenqi Shao, Baobao Chang, Nan Duan, and Jiaqi Wang. Step-wise Rubric Rewards for LLM Reasoning. *arXiv preprint arXiv:2605.17291*, 2026. doi: 10.48550/arXiv.2605.17291. URL https://arxiv.org/abs/2605.17291.

Tianze Xu, Yanzhao Zheng, Pengrui Lu, Lyumanshan Ye, Yong Wu, Zhentao Zhang, Yuanqiang Yu, Chao Ma, Jihuai Zhu, Pengfei Liu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, and Gang Yu. Rubrics to Tokens: Bridging Response-Level Rubrics and Token-Level Rewards in Instruction Following Tasks. In *Conference on Language Modeling*, 2026. URL https://arxiv.org/abs/2604.02795.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757, 2022. doi: 10.52202/068431-1508. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-Distilled Reasoner: On-Policy Self-Distillation for Large Language Models. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2601.18734.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 62178–62209, 2024. URL https://proceedings.mlr.press/v235/zhou24t.html.

A ALGORITHM

1. Let π_{old} denote the policy used for trajectory sampling and paired replay in the current update.
2. For every task x, sample a group of $N = 8$ trajectories $\mathcal{T}$ and compute A_i^{GRPO}.
3. Aggregate the task and all trajectories in the group into one rubric-generation prompt, then independently sample four complete rubric bundles with 5–8 criteria each using the current policy.
4. Use the frozen verifier to score every criterion in each complete bundle. Average all criterion scores within each trajectory, then compute bundle-level Spearman correlation with group outcomes and select the complete bundle with the highest score.
5. Replay each stored action turn with and without $\mathcal{B}^\star$. Check response masks and token alignment, compute the raw $\Delta_{i,\ell}$, and add the valid-token correction in Eq. (9).
6. Accumulate the policy loss and reference-policy KL regularizer in Eq. (11); update the shared policy once.

B PROMPT TEMPLATES

One aggregated prompt supplies the current policy with the task and all action–observation transcripts in the rollout group. With the group-hint instruction, the prompt requests a complete process rubric containing 5-8 observable criteria. The policy samples four independent complete proposals from the same prompt, and each proposal remains intact throughout selection.

For verification, the frozen verifier receives the task, the rollout transcripts, and one candidate bundle. Each criterion is scored independently from visible evidence and receives a satisfaction score in $[0, 1]$. All criterion scores in the bundle are averaged within each trajectory, then correlated with group outcomes across trajectories using Spearman correlation.

The following templates specify rubric generation and policy interaction in the two environments. Blue placeholders denote fields populated for each task. For rubric generation, $k = 8$ and the previous-attempts field contains the same-task rollout group.

Rubric Generation Prompt

[SYSTEM]

You are a task coach who writes process rubrics for an interactive environment.

You will see one task and several previous attempts at the same task.
Use them to write useful process criteria for a NEW agent that has never seen those attempts.

Write 5 to {k} short rubric criteria.

Useful rubric criteria can describe:
- important subgoals needed to complete the task,
- useful ordering of subgoals,
- mistakes or no-progress behavior to avoid,
- what must be true before the task is finished.

Rules:
- Use only information supported by the task or the attempts.
- Do not mention attempts, trajectories, rollouts, rewards, sources, evidence, or IDs.
- Do not copy a complete action sequence.
- Do not invent object locations or hidden state.
- Keep each rubric criterion short, concrete, and useful by itself.
- Do not explain your reasoning.

Output only a numbered process rubric:
1. ...
2. ...

Rubric Generation Prompt (continued)

3. ...
4. ...
5. ...

[USER]

Task:
`{problem}`

Previous attempts:
`{attempts}`

Write a useful process rubric for a new agent solving this same task.

ALFWorld Interaction Prompt

[SYSTEM]

You are acting in ALFWorld. Observe the environment and its valid actions, maintain task state, and reply with exactly 'ACTION: <one currently valid action>'.

[USER]

Goal: `{goal}`

WebShop Interaction Prompt

[SYSTEM]

You are an agent interacting with a virtual text-based web shopping environment to test out your ability. Your job is to follow the Instruction provided and mimic the steps to buy the item that are closest to the Instruction provided.

Action Format:
You should give both the action_name and action_arg like the format 'action_name[action_arg]'. You can execute two types of actions, search and click.
- When the button '[button] Search [button_]' is available in the current observation, you can execute the action 'search[xxx]' (you should type the query you want to search in the square brackets here).
- You can click buttons '[button] xxx [button_]' that is available in the current observation, by execute the action 'click[xxx]'.

Below are some examples of action formats.
- ACTION: search[white shoes]
- ACTION: click[Buy Now]

Notes:
At each step, you should first think then perform action to fulfill the instruction. After thinking, output the action on exactly one line in the format 'ACTION: action_name[action_arg]'.
You should ALWAYS take one action each step.
You should finish the task and buy the item within 15 steps.
DO NOT try to interact with the user at anytime. Finish the task and buy the item by yourself.

[USER]

`{instruction}`

Table 4: Optimization and sampling configuration.

Parameter	Value
Base models	Qwen3-1.7B; Qwen3-4B
Optimization	Full-parameter GRPO
Hardware	$4 \times$ A100 80GB
Prompts / rollouts per prompt	8 / 8
Optimizer steps	300
Rollout group size	8 (ALFWorld and WebShop)
Learning rate	10^{-6}
Clipping parameter ε_c	0.2
KL regularizer / coefficient	low-variance estimator / 10^{-3}
Token-credit coefficient α	0.01
Rubric candidates per task	4 complete bundles
Criteria per bundle	5–8
Bundle selection	highest Spearman correlation with group outcomes
Policy micro-batch size per GPU	ALFWorld: 2; WebShop: 1
Replay eligibility	selected bundle; valid alignment and finite scores
Frozen verifier	Qwen3.5-Plus via DashScope API
Verifier decoding	temperature 0, thinking disabled, JSON only; 1,536-token cap
Policy / privileged-replay thinking	on
Generator thinking / prompt	off / aggregated numbered rubric
Generator temperature / top-p / cap	0.3 / 0.9 / 4,096
Verifier concurrency / retries / cap	8 / 4 / 1,536
Validation sampling	4 trajectories per task, temperature 0.6
Framework	VERL 0.8, FSDP, async vLLM rollout

Table 5: Environment-specific configuration. Full history denotes the complete sequence of previous reasoning responses, actions, and observations.

Parameter	ALFWorld	WebShop
Evaluation outcome	official `won`	official score and score $= 1$
Episode horizon	environment termination	15 assistant actions
Response budget	16,384	30,720
Interaction history	full history	full history
Bundle-selection outcome	official success	behavior-regularized return
Training reward	Eq. (12)	Eq. (13)
Invalid-action coefficient	0	0.05
No-progress coefficient	0	0.025
vLLM memory utilization	0.50	0.72
Agent-loop workers	8	4

C Implementation Details

Evaluation and learning curves. We evaluate every 50 optimizer steps on fixed held-out tasks. Comparison runs in this study use the same evaluation checkpoints and training seeds. Figure 3 plots training rewards against training steps. Each update collects 64 environment trajectories, so step 300 corresponds to 19,200 training trajectories. Verifier calls, replay computation, and wall-clock time are recorded separately.

C.1 Policy Loss Reduction

The main text uses $\mathcal{A}_{\mathrm{mb}}$ to denote the masked micro-batch reduction. Let D be the number of data-parallel workers, G the number of gradient accumulation steps per worker, and $\mathcal{C}_{d,g}$ the trajectories in the corresponding micro-batch, indexed across tasks for this reduction. For any token-level quantity f,

$$\mathcal{A}_{\mathrm{mb}}[f] = \frac{1}{DG} \sum_{d=1}^{D} \sum_{g=1}^{G} \frac{\sum_{i \in \mathcal{C}_{d,g}} \sum_{\ell} b_{i,\ell} f_{i,\ell}}{\sum_{i \in \mathcal{C}_{d,g}} \sum_{\ell} b_{i,\ell}}. \tag{14}$$

Here, $b_{i,\ell}$ is a binary mask for valid response tokens, excluding prompts, environment observations, and padding. This reduction first averages over tokens within each micro-batch, then over micro-batches and workers; it does not separately average over tasks or turns. The term $\varepsilon_{\mathrm{num}}$ ensures numerical stability when computing GRPO advantages.

C.2 Paired Replay Data

Paired replay uses records saved at each generation turn. A record contains the turn index, context messages and token IDs, complete response token IDs, action-only response history, subsequent observation, response mask, and rollout log probabilities. Plain replay packs $c_t + y_t$ independently for each turn. Rubric-conditioned replay inserts the selected complete bundle into c_t and reuses the same target response. The response mask selects the log probabilities used by the correction. The implementation mask $e_{i,\ell}$ equals one when the tokenization round-trip succeeds, response tokens align, and both log-likelihoods are finite. A token with $e_{i,\ell} = 0$ retains its GRPO advantage. The main-text equations omit this implementation mask. The policy ratio continues to use the original context and saved rollout log probabilities.

C.3 Training entry points and run records

Two launchers under `examples/agent_rubric2` define the WebShop group-hint and ALFWorld first-version group-hint configurations. Both aggregate all trajectories in a same-task rollout group, sample four complete bundles with the current policy, select one complete proposal, and apply the raw additive token correction. GRPO uses the same four-GPU, eight-task, eight-rollout, 300-step protocol with rubric generation and token correction disabled.

Rubric generation uses temperature 0.3, top-$p = 0.9$, and a 4,096-token cap. Verification calls `qwen3.5-plus` with concurrency 8, at most four retries, and a 1,536-token output cap. Each run record contains the API identifier and date, resolved configuration, task IDs, seeds, saved trajectories, bundle scores, and checkpoint evaluations. Runs are grouped under `verl_rubric_3`.

Both replay branches preserve the per-turn context and response IDs, masks, chat templates, temperature settings, and rubric insertion rules. Before applying the advantage correction, token scores are written back to their original response positions. The micro-batch loss reduction and reference-policy KL settings match GRPO.

D Mathematical Form of the Token Correction

Effect on trajectory credit. For $n_i = \sum_{\ell} b_{i,\ell} > 0$, the actual mean advantage is

$$\frac{1}{n_i} \sum_{\ell} b_{i,\ell} A_{i,\ell}^{\mathrm{token}} = A_i^{\mathrm{GRPO}} + \frac{\alpha}{n_i} \sum_{\ell} b_{i,\ell} e_{i,\ell} \Delta_{i,\ell}.$$

The raw correction can therefore change both the trajectory mean and individual token signs. Its scale is controlled by α and the replay-alignment mask; no additional normalization or redistribution is applied.

Self-distillation through advantage shaping. Both terms in $\Delta_{i,\ell}$ are computed by the same π_{old}. ORSD applies stop-gradient, denoted by sg below, to the likelihood gap and includes it only in $A_{i,\ell}^{\mathrm{token}}$. At the start of an update, the additional contribution to the policy-loss gradient is

$$-\alpha\, \mathcal{A}_{\mathrm{mb}}[e_{i,\ell}\ \mathrm{sg}[\Delta_{i,\ell}]\, \nabla_\theta \log \pi_\theta(y_{i,\ell} \mid x, y_{i,<\ell})]_{\theta=\theta_{\mathrm{old}}}\,. \tag{15}$$

Here, $\mathcal{A}_{\mathrm{mb}}$ denotes the masked micro-batch reduction in Eq. (14), applied to the enclosed token-level quantity. Privileged context changes the weight assigned to each sampled token in the RL update. ORSD implements privileged self-distillation through advantage shaping; $\mathcal{L}_{\mathrm{total}}$ contains no auxiliary KL or cross-entropy matching objective. The likelihood gap measures the policy's response to the selected process context and does not establish token correctness.