RISED: RUBRICS FOR AGENTIC MULTI-ENVIRONMENT SELECTION AND SELF-DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Training a single LLM agent jointly across diverse interactive environments has attracted increasing attention as a route to generalist agents. Existing curriculum and data-selection strategies often allocate training at the environment level or prioritize local reward-based signals, without explicitly considering relationships between current rollouts across environments for prompt-group selection. Meanwhile, as environments are learned at different rates, all-failure and all-success rollout groups can coexist within a batch, leaving those data without group-relative reward signals. Both challenges highlight limitations of relying solely on scalar rewards in multi-environment RL: they provide limited information about cross-environment relationships and no within-group reward contrast when rewards are identical. This motivates richer textual feedback, such as rubrics describing rollout behaviours, to guide learning. Beyond rubrics' usage in reward construction, we repurpose rubrics to guide both online data selection and policy supervision. An LLM judge tags each rollout using a predefined rubric vocabulary shared across environments. The resulting profiles guide the selection of data that aligns with the overall behavioural composition of the mixed-environment batch while limiting overlap with already-selected data. Available positive rubrics (describing desired behaviours) provide privileged context for an on-policy self-distillation teacher, supplying additional token-level supervision, while negative rubrics (describing undesired behaviours) guide subsequent rollout generation away from recurring failure modes. Together, these components form RISED. Across model backbones, RISED achieves the highest mean pass rate across environments and ranks first or second in every individual environment. Rubric-based analysis of RISED can further characterize the behavioural changes accompanying these gains.

1 INTRODUCTION

Training a single LLM agent across diverse interactive environments offers a promising route toward generalist agents (NVIDIA, 2025a; Ren et al., 2026). Joint training can benefit from transfer across environments and achieve performance comparable to separately trained specialists (Zeng et al., 2024; Zhang et al., 2025). We study joint training on mixed-environment batches, where multiple environments contribute to each policy update (NVIDIA, 2025b; Zhang et al., 2025). Within a limited update budget, an important question is which prompt groups to select from the current rollouts to improve joint training.

Existing approaches address different aspects of training-data allocation. Curriculum methods typically allocate training across domains or environments (Chen et al., 2026; Yang et al., 2026b;a; Zhu et al., 2026b), or adjust sample-level priorities using periodically refreshed difficulty estimates (NVIDIA, 2025b). Online selection instead operates on current rollouts and can adapt to the evolving policy, but commonly prioritizes individual examples using local signals such as reward variability or estimated difficulty (Jiang et al., 2025; Sun et al., 2025; Yu et al., 2025). Neither explicitly accounts for relationships between current rollouts from different environments. A second challenge arises from reward-based supervision. Under GRPO, identical rollout rewards provide no within-group reward contrast to distinguish which behaviours should be reinforced. In multi-environment training, all-failure and all-success groups can occur frequently because environments are learned at different rates. In our experiments, they constitute more than half of the candidate prompt groups, leaving substantial training data without a group-relative reward signal.

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.

Both challenges highlight limitations of relying solely on scalar reward signals in multi-environment RL. Reward-level statistics may be too coarse to characterize relationships relevant to cross-environment transfer, while identical rewards provide no within-group reward contrast. Recent work has explored textual feedback, such as rubric-based guidance, as richer supervision than scalar rewards and demonstrated its effectiveness for training LLMs and agents (Gunjal et al., 2026; Song et al., 2026; Zhang et al., 2026). We find that rubric-based feedback is useful beyond policy supervision: it can serve as an effective online data-selection signal for capturing cross-environment relationships, while the same annotations can supply complementary supervision for groups with identical rewards (Bablani et al., 2026; Wang et al., 2026a).

Concretely, an LLM judge tags each rollout using a shared, predefined rubric vocabulary, producing rubric profiles that make rollout behaviours comparable across environments. We also investigate whether rubric profile similarity is informative about cross-environment transfer: our preliminary analysis finds that it correlates more strongly with cross-environment reward improvements than gradient similarity (Sec. 2.3). Motivated by this finding, we select prompt groups that align with the overall rubric profile of the mixed-environment batch while limiting overlap with already-selected groups (Sec. 2.4). Additionally, the same rubric information is reused during post-training. Available positive rubrics (describing desired behaviours) provide privileged context for an on-policy self-distillation teacher, yielding additional token-level supervision even when rollout rewards are identical, while negative rubrics (describing undesired behaviours) guide subsequent rollout generation away from recurring failure modes (Sec. 3). Together, these components form RISED: RubrIcs for agentic multi-environment Selection and sElf-Distillation. Across model backbones, RISED achieves the highest mean pass rate across environments, while component ablations demonstrate the complementary benefits of selection and self-distillation. Rubric-based analysis further reveals more frequent positive behaviours than in baseline rollouts, providing an interpretable account of the behavioural changes accompanying these gains.

Our contributions are:

- We adopt rollout-level rubric profiles as behavioural descriptions (Sec. 2.2) and empirically show that rubric similarity correlates more strongly than gradient similarity with cross-environment reward improvements (Sec. 2.3).
- We propose RISED, a framework for multi-environment agentic RL that uses the same predefined rubric vocabulary across environments for online prompt-group selection and self-distillation. The selection component uses rubric profiles to select prompt groups across environments (Sec. 2.4). Positive rubrics condition the self-distillation teacher, while negative rubrics guide subsequent rollout generation (Sec. 3).
- We conduct extensive experiments and ablations to evaluate RISED across multiple model backbones (Sec. 4.3) and demonstrate the complementary benefits of selection and self-distillation. We further use rubric profiles to characterize the behavioural changes accompanying these gains (Sec. 5).

2 Rubrics For Data Selection

2.1 Problem Setup

We train a single policy jointly over a set of environments $\mathcal{E}$ using GRPO (Shao et al., 2024). Prompts are scheduled for rollout generation in a round-robin manner across environments, targeting a uniform environment mixture, aligned with previous work setting (Zhang et al., 2025). At each training step t, we construct a candidate batch $\mathcal{B}_t$ of rollout groups. Each group $g = (\tau_{g,i})_{i=1}^{n}$ contains n rollouts sharing a prompt x_g from environment $e(g) \in \mathcal{E}$. A rollout $\tau_{g,i} = (x_g, y_{g,i})$ consists of the prompt and a completion or interaction sequence $y_{g,i}$ generated by the rollout policy. We oversample by a factor of two, forming $|\mathcal{B}_t| = 2k$ candidate groups ($2kn$ rollouts in total), and select a subset $\mathcal{S}_t \subseteq \mathcal{B}_t$ of $|\mathcal{S}_t| = k$ groups (kn rollouts) for the policy update.

Two properties of this setup are worth emphasizing. First, the choice is made *online*, once per step, against the current policy, unlike one-time selection of a fixed training subset, which cannot respond to a policy that keeps changing. Second, selecting from a $2\times$ candidate pool rather than the full task set bounds the cost: in RL, scoring a candidate usually requires generation, so evaluating the

entire pool at every step is not affordable. Generating a surplus and selecting down is a standard compromise in online batch selection (Loshchilov & Hutter, 2015; Mindermann et al., 2022; Zou et al., 2026), and similar patterns are seen in RL post-training (Yu et al., 2025; Zheng et al., 2025).

2.2 Rubric Construction

From rubrics as rewards to rubrics as descriptions. Rubrics decompose rollout quality into individually identifiable behaviours and have primarily been used as reward signals for RL (Huang et al., 2026; Jia et al., 2026; Rezaei et al., 2026; Zhang et al., 2026). We repurpose these behavioural descriptions for sample selection, using shared rubric profiles to compare rollouts across environments. Using them as rewards requires careful calibration: every criterion is eventually folded into a scalar (Tyagi et al., 2026), so a missing or miscalibrated criterion can provide no gradient toward an omitted behaviour, or the wrong one. This requirement is expensive in agentic environments, where a task may admit many successful rollouts and exhaustively specifying in advance the desired good behaviours is difficult. We ask considerably less of the criteria: we identify the prominent behaviours exhibited by each rollout and compare rollouts through these behavioural profiles.[1]

Rubric Vocabulary. Each rubric denotes a fine-grained behavioural pattern, represented by a `snake_case` name. We construct a predefined rubric vocabulary from warm-up model rollouts and augment it with behaviours identified in stronger LLM-generated rollouts. Because warm-up rollouts predominantly end in failure, successful LLM-generated rollouts are introduced to improve coverage of desired behaviours that would otherwise be underrepresented. Together, these sources broaden coverage of desired and undesired behaviours, described by positive and negative rubrics, respectively. We use `Gemini-2.5-Pro` to propose rubric names from warm-up rollout behaviours, consolidate semantically similar names, and organize them into skills. We then augment the vocabulary by extracting additional behaviours from teacher rollouts generated by `Claude-Opus-4.6` and warm-up rollouts again, merging redundant names before incorporating new rubrics. The final vocabulary contains 82 rubrics across 11 skills[2], covering both environment-general and environment-specific behaviours. Each rubric includes a short description for teacher conditioning and training-time rollout guidance (Sec. 3).

Rubric Extraction. Given a rollout and the two-level vocabulary, an LLM judge identifies a set of $(\text{skill}, \text{rubric}, \text{role}, \text{count})$ tuples from the predefined vocabulary. This gives rollouts from different environments a common behavioural representation while allowing the judge to navigate the vocabulary hierarchically rather than through a single flat list. The `role` indicates whether the behaviour is desirable (*demonstrate*) or undesirable (*violate*), while `count` records the number of occurrences of that behaviour identified in the rollout. Rubric extraction requires one judge inference call per rollout, without gradient computation. We serve `Qwen3-32B` locally and tag completed rollouts asynchronously, overlapping rubric annotation with rollout generation. App. C.6 reports the tagging overhead. The complete construction and extraction pipeline, including prompts and example rubric names, is provided in App. A.

2.3 Does Rubric Similarity Track Cross-Environment Transfer?

Setup. We test whether similarity between rubric profiles is associated with cross-environment transfer. We train a policy on one source environment and evaluate it on both the source and held-out environments. We measure reward improvements in each environment to assess the benefits of source-environment training, with improvements on held-out environments indicating cross-environment transfer. A useful similarity signal should assign higher similarity to environments that exhibit larger reward improvements. We present results with WebShop (Yao et al., 2022) as the source environment in the main text and repeat the analysis with DBBench (Liu et al., 2024) as the source in App. C.1. For the WebShop setting, we evaluate on WebShop, DBBench, ALF-

[1] For selection, an incomplete rubric set may make rollouts appear more similar, weakening the signal rather than directly defining an incorrect reward. A more important requirement when using rubrics for selection is consistency across rollouts. The criteria need not exhaustively cover desirable behaviours, but should capture enough variation to distinguish rollouts.

[2] The vocabulary is constructed from warm-up and teacher rollouts across four environments. The main experiments use three of these environments; removing the rubrics and skill specific to the excluded environment yields 59 rubrics grouped into 10 skills.

Table 1: Correlation between similarity signals and reward improvement under WebShop-only training. Spearman (ρ) and Pearson (r) correlations are pooled over checkpoint–environment pairs across steps. Held-out results exclude WebShop. Positive-only, negative-only, and combined profiles are denoted by pos, neg, and all, respectively.

		Rubric-level			Skill-level			Gradient
Val. envs		pos	neg	all	pos	neg	all	full layer.
Held-out	ρ	-0.52	$+0.04$	$\mathbf{+0.60}$	-0.69	$+0.30$	$+0.43$	$+0.18$
	r	-0.49	$+0.07$	$\mathbf{+0.77}$	-0.61	$+0.43$	$+0.52$	$+0.21$
All env	ρ	$+0.32$	$+0.57$	$\mathbf{+0.81}$	$+0.25$	$+0.69$	$+0.75$	$+0.28$
	r	$+0.76$	$+0.84$	$\mathbf{+0.87}$	$+0.69$	$+0.87$	$+0.87$	$+0.27$

World (Shridhar et al., 2021), and AppWorld (Trivedi et al., 2024), covering shopping, database interaction, household object manipulation, and application-based tasks, respectively. Detailed environment descriptions are provided in App. B.1.

Every 25 training steps up to step 500, we freeze the policy and collect 512 fresh WebShop training rollouts (64 prompts $\times$ 8 samples), together with four rollouts per validation prompt from each environment. For this analysis, we use a fixed set of up to 200 validation prompts per environment across all checkpoints, retaining the full set when fewer than 200 are available. Each rollout is annotated with its corresponding rubrics. For every checkpoint–environment pair (t, e), we measure the reward improvement $\Delta_t(e) = \bar{r}_t(e) - \bar{b}_e$, where $\bar{r}_t(e)$ is the mean validation reward in environment e at checkpoint t, and $\bar{b}_e$ is the mean reward at the first measured checkpoint. We exclude the first checkpoint because $\Delta_{t_0}(e) = 0$ by construction. We compare this reward-improvement signal against two alignment signals: the cosine similarity between training and validation rubric profiles and, as a baseline, the cosine similarity between the corresponding training and validation gradients.

Rubric-based signals. Let $\mathcal{R} = \{r_1, \ldots, r_m\}$ denote the rubric vocabulary. For each rollout τ, let $c_r(\tau)$ denote the judge-assigned count of rubric r, with zero assigned when r is absent. For any collection of rollouts $\mathcal{A}$, define its *rubric profile* as the count vector: $\mathbf{q}(\mathcal{A}) = \left(\sum_{\tau\in\mathcal{A}} c_{r_1}(\tau), \ldots, \sum_{\tau\in\mathcal{A}} c_{r_m}(\tau)\right)^\top$. Let $\mathcal{T}_t$ denote the WebShop training rollouts at step t and $\mathcal{V}_t(e)$ the validation rollouts from environment e. We define rubric similarity as $s_t^{\text{rubric}}(e) = \mathrm{sim}(\mathbf{q}(\mathcal{T}_t), \mathbf{q}(\mathcal{V}_t(e)))$, where sim denotes cosine similarity, defined as zero when either profile is zero. Because cosine similarity is invariant to positive scaling, this signal captures the relative composition of tagged behaviours rather than the number of aggregated rollouts. We additionally compute positive-only and negative-only profiles using the same rubric coordinates, including $c_r(\tau)$ only when the corresponding role is *demonstrate* or *violate*, respectively, and assigning zero otherwise. Skill-level profiles aggregate counts over rubrics belonging to the same skill. Tab. 1 reports Spearman and Pearson correlations between each signal and reward improvement, computed across checkpoint–environment pairs. We report results both across all four environments and over the three held-out environments, excluding WebShop.

Results. A stronger correlation indicates that a similarity signal more closely tracks which environments improve during WebShop training. As shown in Tab. 1, the strongest positive correlations are obtained using the combined positive-and-negative rubric profile. Using both roles provides stronger positive associations, particularly on held-out environments, suggesting that similarities in both successful behaviours and failure modes are informative for cross-environment transfer analysis. Aggregating rubrics into broader skills generally weakens this signal. For example, WebShop and DBBench both contain rubrics belonging to the skill `tool_usage_and_formatting`. Skill-level aggregation therefore makes them appear similar even when the underlying fine-grained behaviours differ. Rubric similarity also consistently identifies which environment benefits most from training across all four environments: it identifies the most-improved environment at all 19 evaluated checkpoints, compared with 8 checkpoints for gradient similarity. Gradient signal is less strongly correlated with reward improvements in this analysis. Additionally, we evaluate the stability of the tagging procedure by independently re-tagging the same rollouts. Independent re-tagging yields similar environment-level transfer correlations, with the reported Spearman coefficients changing by at most 0.060 (App. C.2).

2.4 RUBRICS FOR ONLINE DATA SELECTION

The environment-level analysis motivates using fine-grained rubric counts from both positive and negative rubrics as a shared representation. A fixed vocabulary shared across environments makes these profiles directly comparable across environments and training steps. We now apply this representation to online prompt-group selection.

Selection. For each candidate prompt group $g \in \mathcal{B}_t$, we aggregate the rubric counts of its n rollouts into a group profile $\mathbf{q}(g) = (\sum_{i=1}^{n} c_{r_1}(\tau_{g,i}), \ldots, \sum_{i=1}^{n} c_{r_m}(\tau_{g,i}))^\top$. For the candidate pool, we use the shorthand $\mathbf{q}(\mathcal{B}_t) = \sum_{g \in \mathcal{B}_t} \mathbf{q}(g)$, which summarizes all candidate rollouts at step t. Starting from $\mathcal{S} = \varnothing$, we select groups greedily in a round-robin manner across environments. On each environment's turn, we choose the highest-scoring unselected candidate using an MMR-inspired relevance–redundancy score (Goldstein & Carbonell, 1998):

$$\text{score}(g \mid \mathcal{S}) = \underbrace{\text{sim}(\mathbf{q}(g), \mathbf{q}(\mathcal{B}_t))}_{\text{relevance}} - \underbrace{\text{sim}\Big(\mathbf{q}(g), \textstyle\sum_{g' \in \mathcal{S}} \mathbf{q}(g')\Big)}_{\text{redundancy}}. \tag{1}$$

We add the chosen group to $\mathcal{S}$ and continue until $|\mathcal{S}| = k$, yielding $\mathcal{S}_t = \mathcal{S}$. The relevance term favours groups aligned with the overall rubric profile of the mixed-environment batch, while the redundancy term penalizes overlap with the selected aggregate. Under round-robin selection, choosing a group in one environment changes the redundancy scores for subsequent choices in others, potentially favouring complementary behaviours as selection process proceeds, such as environment-specific ones. Fixed quotas preserve the baseline environment mixture. A similar form is common in online batch selection for supervised fine-tuning (Zou et al., 2026) and RL (Lin et al., 2026), though typically over embeddings rather than rubrics.

Rubric selection and reward variance capture different properties. Rubric selection does not implicitly act as a reward-variance filter. Over a 1000-step run on three environments, selected groups have mean within-group reward variance 0.0557, compared with 0.0507 in the candidate pool ($1.10\times$), while a variance-targeted selector reaches approximately $2\times$ in this run. The two signals therefore capture different properties: reward variance indicates whether a group offers within-group reward contrast, whereas rubric overlap describes which behaviours it exercises.

Auxiliary supervision for groups with identical rewards. Across a 1000-step run on three environments, 62.9% of candidate groups have zero within-group reward variance, with a similar fraction among rubric-selected groups. Rubric-based selection therefore retains many groups whose identical rewards provide no within-group contrast to distinguish which behaviours should be reinforced. We then reuse the same rubric annotations for self-distillation to provide complementary token-level supervision on the selected rollouts.

3 RUBRICS FOR DISTILLATION

On-policy self-distillation. Distillation trains a student to match a teacher's output distribution rather than a hard label, providing a dense per-token target in place of a scalar reward. *On-policy* distillation applies this target to the student's own rollouts, correcting the policy on states it actually visits (Agarwal et al., 2024). In *self*-distillation, teacher and student generally share parameters but differ in conditioning: the teacher is the same policy given privileged context c unavailable to the student (Hübotter et al., 2026). Thus, no separate teacher is trained, and supervision requires only an additional forward pass. For a completion y generated from prompt z, self-distillation conditions the teacher on an additional privileged context c while leaving the student's input unchanged. Following Wang et al. (2026a), we use

$$\mathcal{L}_{\text{SD}} = \mathbb{E}[\rho_j k_3(\ell_j)], \qquad \ell_j = \log \pi_\theta(y_j \mid z, y_{<j}) - \log \pi_{\bar{\theta}}(y_j \mid z \oplus c, y_{<j}), \qquad k_3(\ell) = e^{-\ell} - 1 + \ell, \tag{2}$$

where $\bar{\theta}$ denotes the parameter values used for the teacher forward pass, whose log-probabilities are treated as constants when differentiating $\mathcal{L}_{\text{SD}}$. $\rho_j = \frac{\pi_\theta(y_j|z,y_{<j})}{\text{sg}[\pi_{\theta_{\text{old}}}(y_j|z,y_{<j})]}$ is the importance ratio between the current student and the rollout policy, where sg denotes stop gradient. Crucially, this loss provides dense token-level supervision independently of the GRPO-based advantage coefficients. The overall policy update minimizes $\mathcal{L}_{\text{GRPO}} + \lambda \mathcal{L}_{\text{SD}}$, where $\mathcal{L}_{\text{GRPO}}$ follows the AgentRL training

objective (Zhang et al., 2025) and λ is the self-distillation coefficient. Both losses are evaluated on the selected rollout groups $\mathcal{S}$, with token-level terms restricted to generated tokens. Expansion of $\mathcal{L}_{\mathrm{GRPO}}$ and hyperparameters are given in App. B.3.

Positive rubrics distill; negative rubrics steer. The rubrics already tagged for batch selection also provide the privileged context c, requiring no additional effort. For each rollout, we take the deduplicated union of its top-3 (3 highest-count rubrics) positive rubrics and the top-3 positive rubrics of its group, and render their descriptions as a privileged context for the teacher. Groups without positive rubrics may receive no rubric-derived teacher context (App. E.6). Distillation then trains the student to reproduce the desired behaviour without the privileged context c. The choice to include only positive rubrics in the distillation context is motivated by prior work and supported by our ablation. Prior work similarly constructs rubric-conditioned teachers from the best rollouts (Xia et al., 2026a) or compress rollouts into separate contextual lessons (Xia et al., 2026b). Mixing positive and negative material is not consistently beneficial, with its effect depending on the difficulty and comprehensibility of the context (Armandpour et al., 2026). Our ablation in App. C.3 likewise shows that filling context slots with both positive and negative rubrics degrades overall performance.

Nevertheless, negative rubrics contain valuable information, particularly early in training. On ALFWorld, for example, negative-tagged rollouts outnumber positive-tagged rollouts by more than $10\times$ at step 10, but fall below parity around step 330. We use negative rubrics to steer the next step's rollouts rather than to condition the teacher. At the end of each step, we aggregate tags over all candidate rollouts from each environment and compare the numbers containing at least one negative rubric and at least one positive rubric. If violations are more prevalent, we prepend that environment's top-3 negative rubrics to the next step that environment's prompts as an "anti-patterns" block. When this steering is active, z in Eq. 2 denotes the resulting steered prompt. The same z is provided to both student and teacher, while only the positive-rubric context c is privileged to the teacher. Overall, negative rubrics steer how rollouts are sampled, whereas positive rubrics provide the self-distillation target. Negative-rubric injection is used only during training. Validation rollouts use the original task prompts, without injected rubrics or privileged teacher context. Combining selection in Sec. 2.4 and the distillation here, we propose our approach RISED: RubrIcs for agentic multi-environment Selection and sElf-Distillation, as the full algorithm summarized in Alg. 1 in Appendix.

4 Experiments and Results

4.1 Experimental Setting

We build on AgentRL (Zhang et al., 2025), which provides a unified function-call interface for multi-turn interaction. We train jointly on three AgentBench environments (Liu et al., 2024): ALFWorld, WebShop and DBBench. The three differ in action space, reward mechanics and rollout length, which is what makes their interaction non-trivial. An episode is capped at 20 interaction turns and 8192 total tokens, and an unfinished episode receives -0.2. We use GRPO with a batch of 64 prompts and $n = 8$ rollouts per prompt. At each step, the rollout buffer generates 128 prompt groups, from which we select 64 (Sec. 2.4). Across all runs, we follow the AgentRL training protocol, including environment-wise advantage normalization and stale-policy rollout generation.

We evaluate validation pass rate every 10 training steps on fixed held-out sets, sampling 4 rollouts for each validation prompt. For each environment, we report the mean pass rate (binary success indicator) across prompts and rollouts. Overall performance is the unweighted mean of the three environment-level pass rates. To evaluate effectiveness across model backbones, we train `Qwen2.5-3B-Instruct` (use `Qwen2.5-3B` later for simplicity) on four nodes with eight H100 GPUs each and `Qwen3-4B` without thinking on eight B200 GPUs. Methods requiring rubric annotations, including RISED, its component ablations, and distillation baselines, additionally use a dedicated eight-GPU server for `Qwen3-32B` judge inference. We report main results as the mean and standard deviation over multiple runs. Additional ablation experiments use a single seed unless otherwise stated. Full optimization and rollout hyperparameters are given in App. B.

4.2 Baselines

We compare against five baselines. **GRPO-64** uniformly samples 64 prompts and generates 8 rollouts per prompt, using all 512 rollouts per step without selection. **GRPO-128** doubles the prompt

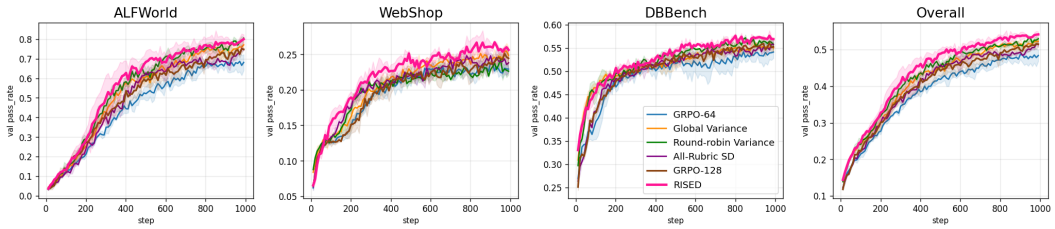

Figure 1: **Validation dynamics for RISED compared with baseline**. Validation pass rate during joint training on ALFWorld, WebShop, and DBBench with `Qwen2.5-3B-Instruct`, shown per environment and overall. RISED achieves higher validation pass rates than the baselines, both overall and in each environment, during most of the middle and later stages of training.

Table 2: **Main results of RISED compared with baselines.** Final-checkpoint pass rates on ALFWorld, WebShop, and DBBench, and overall, for RISED and five baselines on two backbones. **Bold** and underlined values indicate the best and second-best results, respectively, within each backbone. RISED achieves the highest mean overall pass rate and ranks among the top two methods in every individual environment.

Model	Method	ALFWorld	WebShop	DBBench	**Overall**
Qwen2.5-3B	GRPO-64	0.681±0.041	0.229±0.004	0.541±0.013	0.484±0.015
	GRPO-128	0.748±0.000	0.244±0.009	0.552±0.009	0.515±0.006
	Global Variance	0.771±0.020	0.248±0.013	0.550±0.008	0.523±0.006
	Round-robin Variance	**0.805±0.005**	0.226±0.005	0.557±0.011	0.530±0.003
	All-Rubric SD	0.747±0.003	0.238±0.013	0.557±0.007	0.514±0.008
	RISED	0.800±0.023	**0.256±0.003**	**0.569±0.002**	**0.542±0.008**
Qwen3-4B	GRPO-64	0.706±0.115	0.279±0.014	0.563±0.011	0.516±0.046
	GRPO-128	0.862±0.009	0.290±0.000	0.583±0.018	0.579±0.009
	Global Variance	0.821±0.069	0.294±0.008	0.567±0.020	0.561±0.019
	Round-robin Variance	**0.881±0.032**	0.279±0.003	0.557±0.053	0.573±0.030
	All-Rubric SD	0.841±0.008	0.284±0.005	**0.599±0.013**	0.574±0.009
	RISED	0.865±0.011	**0.306±0.012**	0.584±0.003	**0.585±0.001**

batch size to 128 and updates on all 1024 rollouts. Like RISED, **Global Variance** and **Round-Robin Variance** generate 128 candidate groups and select 64 for training based on within-group reward variance. Reward variance provides an inexpensive proxy for a GRPO learning signal: groups with identical rewards have zero variance. For binary rewards, variance induces the same ordering as mean absolute group-standardized advantage. This motivates our variance-based selectors and relates them to DAPO's filtering of zero-variance groups (Yu et al., 2025). Global Variance selects the highest-variance groups across the candidate pool, whereas Round-Robin Variance ranks groups within each environment and interleaves selection across environments, matching RISED's environment scheduling. Finally, as a distillation-oriented baseline, **All-Rubric SD** adapts Bablani et al. (2026) to our agentic RL setting by retaining GRPO and conditioning the teacher on all rubrics. Because our environments do not provide predefined task rubrics, these annotations come from our fixed vocabulary. It uses 64 prompt groups per update, without selection or negative-rubric guidance.

4.3 Main Results

Tab. 2 reports final-checkpoint performance on both backbones, while Fig. 1 shows the full validation dynamics for `Qwen2.5-3B`. **RISED improves overall performance on both backbones.** RISED achieves the highest mean overall pass rate on both `Qwen2.5-3B` and `Qwen3-4B`, improving over GRPO-64 by 5.8 and 6.9 percentage points, respectively. Relative to the strongest baseline on each backbone, the gains are 1.2 and 0.6 points. Across individual environments, RISED ranks first or tied for second in all six backbone–environment comparisons. Fig. 1 further shows that, on `Qwen2.5-3B`, RISED generally outperforms the baselines both overall and in each environment throughout the middle and later stages of training. Its advantage thus extends beyond just the final

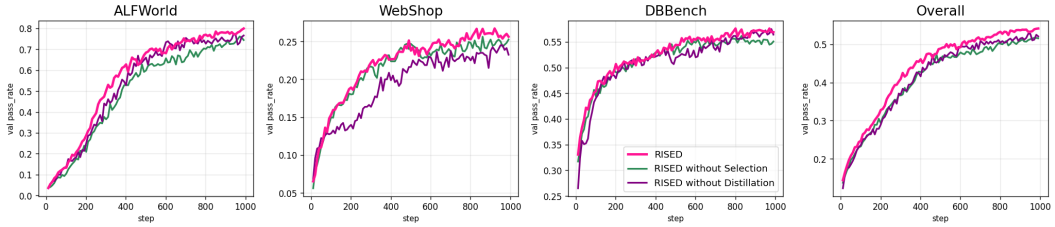

Figure 2: **Component ablations.** Validation pass rate across training steps for RISED (pink), RISED without selection (green), RISED without distillation (purple). Combining selection and distillation yields the strongest overall performance, while their individual contributions vary across environments.

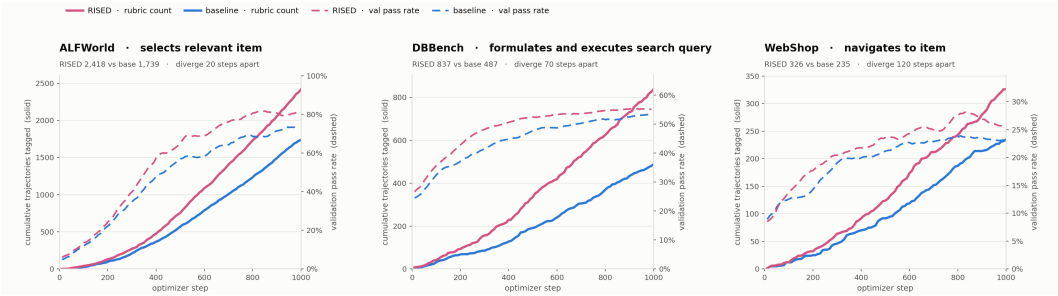

Figure 3: **Positive-rubric accumulation and task performance.** Solid lines show cumulative counts of training rollouts exhibiting the indicated rubric (left axis); dashed lines show validation pass rates (right axis). Pink denotes RISED and blue denotes GRPO-64. RISED exhibits positive behaviours more frequently than GRPO-64.

checkpoint. **RISED reaches baseline performance in fewer optimizer steps.** As shown in Fig. 1, RISED reaches the final overall pass rate of GRPO-64 in approximately half as many optimizer steps. It also reaches the final performance levels of the other baselines earlier in training, although the reductions are smaller. **Selecting training data improves on using all candidates.** RISED outperforms GRPO-128 overall on both backbones, despite updating on only 512 of the 1024 generated rollouts per update. These results suggest that rubric-based selection together with distillation can be more effective than increasing the GRPO update batch alone.

5 Ablations and Further Analysis

Component ablations. Using `Qwen2.5-3B`, we separately remove rubric-guided selection (remove Sec. 2.4 component) and rubric-conditioned distillation (remove Sec. 3 component means also removing negative steering) from RISED (Fig. 2). Removing either component reduces overall performance, supporting their complementary roles: selection prioritizes groups that balance rubric coverage and diversity, while distillation can provide dense supervision even for groups with identical rollout rewards. Their relative contributions vary across environments: distillation contributes more on WebShop, whereas selection provides larger gains on ALFWorld and DBBench.

Qualitative analysis. To examine how RISED changes agent behaviour, Fig. 3 tracks representative positive rubrics in training batches of `Qwen2.5-3B`, with both RISED and GRPO-64 updating on 64 prompt groups per step. RISED accumulates rollouts exhibiting environment-relevant positive behaviours earlier and in greater numbers, including `relevant item selection` in ALFWorld, `search-query formulation and execution` in DBBench, and `item navigation` in WebShop. Because these rollouts enter the policy updates, their earlier accumulation exposes the policy more frequently to these environment-relevant behaviours during training. Together with the corresponding emerging validation-performance gap, the results suggest that prioritizing and reinforcing desirable behaviours contributes to RISED's performance gains. Additional rubric-level analyses reveal distinct stages of behaviour improvement over training (App. C.4).

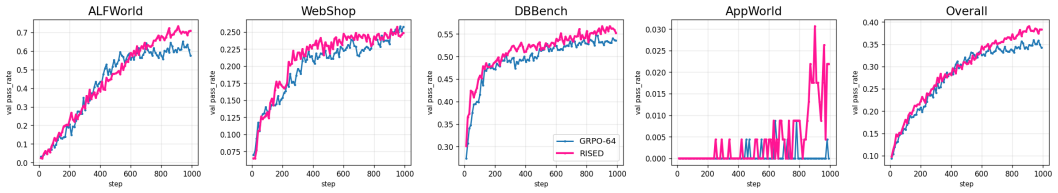

Figure 4: **Joint training with a challenging fourth environment.** Validation pass rates for RISED (pink) and GRPO-64 (blue) when AppWorld is added. RISED achieves higher overall performance and shows emerging progress on AppWorld, while GRPO-64 remains near zero for AppWorld.

Stress test with a challenging fourth environment. Using `Qwen2.5-3B`, we extend joint training to four environments by adding AppWorld. For this experiment, we retain the full four-environment vocabulary described in Sec. 2.2. As shown in Fig. 4, RISED achieves higher overall performance than GRPO-64 and begins to make progress on AppWorld, where the baseline's AppWorld validation pass rate remains near zero. Although AppWorld performance looks low and variable, RISED produces more frequent successes late in training. Environment-specific rubric guidance can expose AppWorld-relevant behaviours, while behaviours shared with the other environments can receive additional reinforcement during joint training. Additional analyses of judge model size and wall-clock costs relative to baselines are in App. C.5 and C.6.

6 Related Work

Traditional online data selection prioritizes training examples using loss (Loshchilov & Hutter, 2015), gradient-based importance estimates (Wang et al., 2024; Li et al., 2025) or estimated reducible loss (Mindermann et al., 2022). RL online selection methods prioritize training examples using reward variability or policy-dependent difficulty. DAPO filters zero-variance groups and replenishes the training batch (Yu et al., 2025), while VCRL combines variance-based filtering with replay (Jiang et al., 2025). To avoid generating rollouts for uninformative prompts, PCL and DOTS predict policy-dependent difficulty and favor intermediate-difficulty examples (Gao et al., 2026; Sun et al., 2025). InSight combines a reward-variance term with difficulty weighting and information gain from historical outcomes (Zhou et al., 2026). For binary rewards, success rates closer to 0.5 correspond to higher reward variance, connecting the intermediate-difficulty preference of PCL and DOTS to VCRL's variance-based filtering and InSight's variance component. These connections motivate our variance baselines as representative implementations of the shared preference for reward variability. We instantiate this principle using observed rollout rewards to select a fixed number of already-generated groups, matching RISED's candidate-generation budget and selection size. CROPI periodically selects training subsets using validation-guided influence estimates computed from off-policy rollouts and projected gradients (Zhu et al., 2026a). Its reliance on validation data for selection places it outside our comparison protocol, which uses neither validation-guided scoring nor per-group gradient computation. RISED instead uses rubric profiles of current rollouts to jointly select prompt groups for coverage of the mixed-environment batch while limiting redundancy across environments. Further discussion of curriculum, rubrics and distillation appears in App. D.

7 Conclusion

We introduced RISED, a rubric-guided framework that combines online data selection with self-distillation for multi-environment agent training. By representing rollouts through a predefined and shared vocabulary across environments, RISED selects groups that balance cross-environment relevance and behavioral diversity, while rubric-conditioned self-distillation provides dense supervision even for groups with zero variance in rewards. Across two model backbones on multiple agent environments, RISED consistently improves over competitive baselines and rubric-level analyses provide an interpretable view of how behaviors evolve during training. A limitation is that rubric tagging and teacher inference add overhead. Future work shall improve efficiency and scale to larger policies and more environments.

AI USE STATEMENT.

In this work, we used Claude to assist with implementing methods in code. We did not use generative AI tools to generate research ideas or propose the methodology. Additionally, we used Claude and GPT to support literature review and transform author-written bullet points and rough drafts into polished manuscript text. We also use Claude and Gemini to generate methodology related artifacts, such as rubrics, descriptions and augmented rollouts. We reviewed all AI-assisted work and take responsibility for the final content, including text, claims, and code produced with AI assistance.

REPRODUCIBILITY STATEMENT.

We provide the prompts in App. A and the pseudocode, datasets, training configuration, and hyperparameters in App. B. The main experiments use multiple random seeds; additional ablation experiments use a single seed, as indicated in the corresponding captions. We will release the code upon acceptance.

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 *Proc. ICLR*, 2024.

Mohammadreza Armandpour, Fatih Ilhan, David Harrison, Ajay Jaiswal, Duc N. M. Hoang, Fartash Faghri, Yizhe Zhang, Minsik Cho, and Mehrdad Farajtabar. Unmasking on-policy distillation: Where it helps, where it hurts, and why. *arXiv preprint arXiv:2605.10889*, 2026.

Deepika Bablani, Ajay Gupta, and Wanming Chen. Rubrics as privileged information for open-ended generation. *arXiv preprint arXiv:2608.02948*, 2026.

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

Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. Self-evolving curriculum for LLM reasoning. In *Proc. COLM*, 2026.

Jorge (Zhoujun) Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, Yuexin Bian, Nilabjo Dey, Yonghao Zhuang, Yuheng Zha, Yi Gu, Kun Zhou, Yuqi Wang, Yuan Li, Richard Fan, Jianshu She, Chengqian Gao, Abulhair Saparov, Taylor W. Killian, Haonan Li, Mikhail Yurochkin, Eric P. Xing, Zhengzhong Liu, and Zhiting Hu. Revisiting reinforcement learning for LLM reasoning from a cross-domain perspective. In *Proc. NeurIPS Datasets and Benchmarks Track*, 2025.

Zhaolin Gao, Joongwon Kim, Wen Sun, Thorsten Joachims, Sid Wang, Richard Yuanzhe Pang, and Liang Tan. Prompt curriculum learning for efficient LLM post-training. In *Proc. ICLR*, 2026.

Jade Goldstein and Jaime Carbonell. Summarization: (1) using MMR for diversity-based reranking and (2) evaluating summaries. In *TIPSTER TEXT PROGRAM PHASE III: Proceedings of a Workshop held at Baltimore, Maryland, October 13-15, 1998*, pp. 181–195, 1998.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean M. Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *Proc. ICLR*, 2026.

Tzu-Heng Huang, Sirajul Salekin, Javier Movellan, Frederic Sala, and Manjot Bilkhu. RubiCap: Rubric-guided reinforcement learning for dense image captioning. *arXiv preprint arXiv:2603.09160*, 2026.

Jonas Hübotter, Frederike Lübeck, Lejs Deen 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 *Proc. ICML*, 2026.

Ruipeng Jia, Yunyi Yang, Wen Wang, Yuxin Wu, Yongbo Gai, Siyuan Tao, Mengyu Zhou, Jianhe Lin, Xiaoxi Jiang, and Guanjun Jiang. Open rubric system: Scaling reinforcement learning with pairwise adaptive rubric. *arXiv preprint arXiv:2602.14069*, 2026.

Guochao Jiang, Wenfeng Feng, Guofeng Quan, Chuzhan Hao, Yuewei Zhang, Guohua Liu, and Hao Wang. VCRL: Variance-based curriculum reinforcement learning for large language models. *arXiv preprint arXiv:2509.19803*, 2025.

Junjie Li, Ziao Wang, Jianghong Ma, and Xiaofeng Zhang. Uncovering intrinsic capabilities: A paradigm for data curation in vision-language models. *arXiv preprint arXiv:2510.00040*, 2025.

Xiaozhe Li, Tianyi Lyu, Yang Li, Yichuan Ma, Peiji Li, Linyang Li, Qipeng Guo, Dahua Lin, and Kai Chen. What and when to distill: Selective hindsight distillation for multi-turn agents. *arXiv preprint arXiv:2605.19447*, 2026a.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan ang Gao, Wenkai Yang, Zhiyuan Liu, and Ning Ding. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026b.

Yuying Li, Leqi Zheng, Yongzi Yu, Wenrui Zhou, Xuchang Zhong, Xing Hu, Jing Jin, Hangjie Yuan, and Tao Feng. Filter, then reweight: Rethinking optimization granularity in on-policy distillation. *arXiv preprint arXiv:2606.02684*, 2026c.

Yiqing Liang, Jielin Qiu, Wenhao Ding, Zuxin Liu, James Tompkin, Mengdi Xu, Mengzhou Xia, Zhengzhong Tu, Laixi Shi, and Jiacheng Zhu. MoDoMoDo: Multi-domain data mixtures for multimodal LLM reinforcement learning. *arXiv preprint arXiv:2505.24871*, 2025.

Xiaoqiang Lin, Arun Verma, Zhongxiang Dai, Daniela Rus, See-Kiong Ng, and Bryan Kian Hsiang Low. ActiveDPO: Active direct preference optimization for sample-efficient alignment. In *Proc. ICLR*, 2026.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *Proc. ICLR*, 2024.

Yang Liu, Jiaqi Li, and Zilong Zheng. RuleReasoner: Reinforced rule-based reasoning via domain-aware dynamic sampling. In *Proc. ICLR*, 2026a.

Zhihan Liu, Lin Guan, Yixin Nie, Kai Zhang, Zhuoqun Hao, Lin Chen, Asli Celikyilmaz, Zhaoran Wang, and Na Zhang. Paying less generalization tax: A cross-domain generalization study of RL training for LLM agents. *arXiv preprint arXiv:2601.18217*, 2026b.

Ilya Loshchilov and Frank Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

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.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *Proc. ICML*, 2022.

NVIDIA. Llama-Nemotron: Efficient reasoning models. *arXiv preprint arXiv:2505.00949*, 2025a.

NVIDIA. Nemotron 3 nano: Open, efficient mixture-of-experts hybrid Mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2512.20848*, 2025b.

Xuhui Ren, Shaokang Dong, Chen Yang, Qing Gao, Yunbin Zhao, Yongsheng Liu, Xinwei Geng, Xiang Li, Demei Yan, Yanqing Li, Chenhao Huang, Dingwei Zhu, Junjie Ye, Boxuan Yue, Yingnan Fu, Mengzhe Lv, Zezeng Feng, Boshen Zhou, Bocheng Wang, Xuanjing Huang, Yu-Gang Jiang, Tao Gui, Qi Zhang, and Yunke Zhang. MagicAgent: Towards generalized agent planning. *arXiv preprint arXiv:2602.19000*, 2026.

MohammadHossein Rezaei, Robert Vacareanu, Zihao Wang, Clinton Wang, Bing Liu, Yunzhong He, and Afra Feyza Akyürek. Online rubrics elicitation from pairwise comparisons. In *Proc. ICML*, 2026.

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.

Leheng Sheng, Wenchang Ma, Ruixin Hong, Xiang Wang, An Zhang, and Tat-Seng Chua. Reinforcing chain-of-thought reasoning with self-evolving rubrics. *arXiv preprint arXiv:2602.10885*, 2026.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *Proc. ICLR*, 2021.

Yuda Song, Lili Chen, Fahim Tajwar, Rémi Munos, Deepak Pathak, Drew Bagnell, Aarti Singh, and Andrea Zanette. Expanding the capabilities of reinforcement learning via text feedback. In *Proc. ICML*, 2026.

Yifan Sun, Jingyan Shen, Yibin Wang, Tianyu Chen, Zhendong Wang, Mingyuan Zhou, and Huan Zhang. Improving data efficiency for LLM reinforcement fine-tuning through difficulty-targeted online data selection and rollout replay. In *Proc. NeurIPS*, 2025.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proc. ACL*, pp. 16022–16076, 2024.

Utkarsh Tyagi, Xingang Guo, MohammadHossein Rezaei, Daniel George, Anas Mahmoud, Jackson Lee, Bing Liu, and Yunzhong He. Not every rubric teaches equally: Policy-aware rubric rewards for RLVR. *arXiv preprint arXiv:2605.20164*, 2026.

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*, 2026a.

Haoqing Wang, Xiang Long, Ziheng Li, Yilong Xu, Tingguang Li, and Yehui Tang. To mix or to merge: Toward multi-domain reinforcement learning for large language models. In *Proc. COLM*, 2026b.

Jiachen T. Wang, Tong Wu, Dawn Song, Prateek Mittal, and Ruoxi Jia. GREATS: Online selection of high-quality data for LLM training in every iteration. In *Proc. NeurIPS*, 2024.

Ruiyi Wang and Prithviraj Ammanabrolu. A practitioner's guide to multi-turn agentic reinforcement learning. In *Proc. First Workshop on Multi-Turn Interactions in Large Language Models*, 2025.

Yuanyi Wang, Su Lu, Yanggan Gu, Pengkai Wang, Yifan Yang, Zhaoyi Yan, Congkai Xie, Jianmin Wu, and Hongxia Yang. Not all disagreement is learnable: Token teachability in on-policy distillation. *arXiv preprint arXiv:2605.26844*, 2026c.

Mingxuan Xia, Yuhang Yang, Chao Ye, Shuai Zhu, Shenzhi Yang, Guangcheng Zhu, Yuhang Zhang, Cheng Peng, Haobo Wang, and Siqing Wang. CriPO: Enhancing rubric-based RL via self-distillation. *arXiv preprint arXiv:2607.18082*, 2026a.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. SkillRL: Evolving agents via recursive skill-augmented reinforcement learning. In *Proc. ICLR 2026 Workshop on Memory for LLM-Based Agentic Systems*, 2026b.

Xingrun Xing, Haoqing Wang, Boyan Gao, Ziheng Li, and Yehui Tang. Trust region on-policy distillation. *arXiv preprint arXiv:2606.01249*, 2026.

Yuanda Xu, Hejian Sang, Zhengze Zhou, Ran He, Zhipeng Wang, and Alborz Geramifard. TIP: Token importance in on-policy distillation. *arXiv preprint arXiv:2604.14084*, 2026.

Yongjin Yang, Jiarui Liu, Yinghui He, Lechen Zhang, Bernhard Schölkopf, and Zhijing Jin. Transferability for general reasoning: An automated curriculum for multi-domain LLM RL. In *Proc. ICML 2026 Workshop on RL from World Feedback*, 2026a.

Zhuolin Yang, Zihan Liu, Yang Chen, Wenliang Dai, Boxin Wang, Sheng-Chieh Lin, Chankyu Lee, Yangyi Chen, Dongfu Jiang, Jiafan He, Renjie Pi, Grace Lam, Nayeon Lee, Alexander Bukharin, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nemotron-Cascade 2: Post-training LLMs with cascade RL and multi-domain on-policy distillation. *arXiv preprint arXiv:2603.19220*, 2026b.

Shunyu Yao, Howard Chen, John Yang, and Karthik R Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Proc. NeurIPS*, 2022.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Proc. NeurIPS*, 2025.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In *Proc. ACL Findings*, 2024.

Hanchen Zhang, Xiao Liu, Bowen Lv, Xueqiao Sun, Bohao Jing, Iat Long Iong, Zhenyu Hou, Zehan Qi, Hanyu Lai, Yifan Xu, Rui Lu, Hongning Wang, Jie Tang, and Yuxiao Dong. AgentRL: Scaling agentic reinforcement learning with a multi-turn, multi-task framework. *arXiv preprint arXiv:2510.04206*, 2025.

Zhen Zhang, Kaiqiang Song, Xun Wang, Yebowen Hu, Weixiang Yan, Chenyang Zhao, Henry Peng Zou, Haoyun Deng, Sathish Reddy Indurthi, Shujian Liu, Simin Ma, Xiaoyang Wang, Xin Eric Wang, and Song Wang. CM2: Reinforcement learning with checklist rewards for multi-turn and multi-step agentic tool use. *arXiv preprint arXiv:2602.12268*, 2026.

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 *Proc. ICML*, 2026.

Haizhong Zheng, Yang Zhou, Brian Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts. In *Proc. NeurIPS*, 2025.

Xinyu Zhou, Boyu Zhu, Haotian Zhang, Huiming Wang, and Zhijiang Guo. Efficient RLVR training via weighted mutual information data selection. *arXiv preprint arXiv:2603.01907*, 2026.

Erle Zhu, Dazhi Jiang, Yuan Wang, Xujun Li, Jiale Cheng, Yuxian Gu, Yilin Niu, Aohan Zeng, Jie Tang, Minlie Huang, and Hongning Wang. Data efficient RLVR via off-policy influence guidance. In *Proc. ACL*, pp. 46167–46192, 2026a.

Kejian Zhu, Zhuoran Jin, Dongqi Huang, Hongbang Yuan, Yupu Hao, Kang Liu, and Jun Zhao. Beyond simply environment scaling: Designing effective environment distributions for multimodal agent learning. *arXiv preprint arXiv:2608.03571*, 2026b.

Heming Zou, Yixiu Mao, Yun Qu, Qi Wang, and Xiangyang Ji. Utility-diversity aware online batch selection for LLM supervised fine-tuning. In *Proc. ICML*, 2026.

A Rubrics Construction and Tagging Details

A.1 Further Details on Vocabulary Construction

We provide additional details on vocabulary construction below. We take rollouts from one warmup training run with model `Qwen-2.5-3B-instruct` at step 100 and ask `Gemini-2.5-pro` for rubric names. The warmup model is trained on 4 environments, with details of training data as described in App. B.1. The unique rubric names yield roughly 1,150. These are embedded with `all-MiniLM-L6-v2` and grouped by clustering on cosine distance (threshold 0.5, average linkage) into about 194 clusters, each represented by its centroid-closest member. Skills are introduced afterwards. We present the resulting clusters to the judge in separate batches to keep the prompt within the context limit. For each batch, the judge assigns a canonical rubric name, merges clusters that describe the same behaviour, and groups the resulting rubrics under higher-level skill headings. Because the batches are processed independently, the same capability can be named twice, so a second clustering pass over skill-name embeddings (cosine distance threshold 0.5) merges the duplicates. Skills are therefore *discovered* from the rubrics rather than imposed on them. Overall, this yields an initial vocabulary of 59 rubrics across 11 skills on 4 environments.

Because the initial vocabulary is extracted from rollouts of a weak warm-up policy, it provides substantially better coverage of failure modes than of successful behaviours. We therefore augment it using two sources for each environment: 100 warm-up rollouts from `Qwen2.5-3B-Instruct` and 100 teacher rollouts generated by `Claude-Opus-4.6`. The former primarily expose failure behaviours, whereas the latter provide successful behaviours that the weak policy rarely exhibits. Each rollout is processed twice by `Gemini-2.5-pro`: once using the current vocabulary as a closed taxonomy and once using free-form rubric extraction. The free-form pass identifies behaviours missing from the existing vocabulary. Across these rollouts, it produces 1,565 distinct rubric names, which we reduce to 60 semantic clusters. We admit clusters that have no counterpart in the existing vocabulary. We generate descriptions via `Gemini-2.5-pro` for both the original and newly added rubrics using three representative annotated rollouts per rubric. Each description summarizes the behaviour and can include short abstracted examples of environment-specific actions. This procedure yields a final vocabulary of 82 rubrics grouped into 11 skills across four environments.[3] Rubric descriptions are not used for tagging or sample selection; they are used for teacher conditioning and training-time rollout guidance(Sec. 3). Note that all rollouts used for vocabulary construction and augmentation are generated from training-split prompts; validation examples are excluded.

Free-form extraction (Huang et al., 2026; Rezaei et al., 2026) is useful for vocabulary construction, but independently generated names require semantic alignment before rollouts can be compared. In preliminary experiments, clustering free-form annotations yielded between 25 and 44 skills across checkpoints, making comparisons across steps and environments difficult. This motivated establishing a shared vocabulary across environments for primary tagging.

A.2 Examples of Rubrics and Skills

We illustrate the vocabulary through examples of its two-level structure and a summarized sample of behaviours it captures. A skill groups related rubrics under a broader capability, while individual rubrics describe more specific behavioural patterns. For example, skill `search_and_navigation` includes query formulation in WebShop, object search in ALFWorld, and schema exploration in DBBench (Tab. 3).

The vocabulary captures both desired and undesired behaviours. Tab. 4 illustrates these through two contrasting pairs. Error recovery distinguishes revising a plan after receiving feedback from repeating an unsuccessful action. Constraint satisfaction distinguishes performing a required state change from proceeding without it. These contrasts may occur within one skill or span different skills.

The examples also illustrate the environment-general and environment-specific rubrics. The rubrics `revises_plan_after_failure` and `repeats_failed_action` describe responses to fail-

[3] In the main experiments, we use data from three environments. After removing the rubrics and skill specific to the excluded environment, the resulting vocabulary contains 59 rubrics grouped into 10 skills.

Table 3: Example rubrics grouped under the shared skill `search_and_navigation`. Behaviour summaries are illustrative.

Environment	Rubric	Illustrative behaviour
WebShop	`formulates_constrained_search_query`	Constructs a search query using relevant task constraints.
ALFWorld	`systematic_object_search`	Searches plausible locations systematically to find a target object.
DBBench	`explores_database_schema`	Inspects the database structure to identify relevant tables and columns.

ure that apply across environments, such as an unsuccessful search, a failed object interaction, or an invalid database query. In contrast, `explores_database_schema` captures a behaviour specific to DBBench among the evaluated environments. Thus, a skill such as `search_and_navigation` can contain rubrics tailored to particular environments.

Table 4: Positive and negative rubric examples for two ALFWorld tasks. Within each task, the first row illustrates a positive rubric and the second a negative rubric. Actions are illustrative rather than verbatim rollout excerpts.

Skill	Rubric	Illustrative behaviour
ALFWorld: taking an apple fails because the fridge is closed.		
`error_recovery_and_exploration`	`revises_plan_after_failure`	Opens the fridge before retrying the action.
`error_recovery_and_exploration`	`repeats_failed_action`	Repeats the failed action without opening the fridge.
ALFWorld: the task requires placing a hot mug on a side table.		
`object_manipulation`	`uses_appliance_for_state_change`	Heats the mug using a microwave before placing it on the table.
`instruction_following_and_grounding`	`ignores_task_constraint`	Places the mug on the table without first heating it.

A.3 Main Prompts Used for Rubric Generation

Our rubric-extraction prompt is inspired by the skill-distillation prompts in SkillRL (Xia et al., 2026b), which summarize agent experiences into concise, transferable behavioural principles. Initially, directly extracting positive rubrics from successful rollouts proved difficult. Adapting SkillRL's skill-generation prompt made these rubrics easier to summarize: our prompt first traces backward from the goal-achieving action to identify its prerequisite steps, removes irrelevant actions, and abstracts the resulting sequence into a generalizable planning pattern Xia et al. (2026b). It then extracts one to three positive rubrics describing observable correct behaviours. For negative rubrics, a rollout may contain many distinct erroneous behaviours, so we instruct the LLM to count their occurrences and report the two to four most frequent patterns. Partially successful rollouts undergo both procedures to capture demonstrated and violated rubrics.

Both prompts, one used for open ended rubrics generation (Sec. 2.2) and one used to tag rollouts (Sec. 2.4), are used as system messages. The user message is identical in both cases:

```
ENVIRONMENT: {task_name}
REWARD: {reward}
TURNS: {num_turns}
TASK GOAL:
{task_goal}
rollout:
{conversation}
```

Rubrics Construction prompt (initial vocabulary construction) Used for the first free-form pass, whose emitted names were clustered into the initial vocabulary. No taxonomy is supplied, and no skill field is requested. Skill headings were assigned subsequently by the LLM, with duplicate headings merged through clustering.

Prompt

```
You are an expert **rollout Analysis Engine**.

Analyze a single RL training rollout from an agent operating in
interactive environments (ALFWorld, WebShop, DBBench, AppWorld).

Use a DIFFERENT analysis strategy depending on the rollout outcome.


  ======================================================================
IF FULL SUCCESS (reward = 1.0)

  ======================================================================

**Phase 1: The Refinement Algorithm (Strictly Backward Logic)**
1. Start Point: Identify the Last Successful Step (the step that
  achieved
   the goal). Call this Current_Step.
2. Recursive Step (The Loop):
   * Analyze Dependency: Determine the strict Required
  Precondition for
     Current_Step. Ask: "What specific state, information, or
  object
     availability was necessary for this action to occur?"
   * Scan Backwards: Traverse the log in reverse order from
  Current_Step.
   * Find Nearest Enabler: Identify the nearest preceding step
     (Preceding_Step) that explicitly generated this Precondition.
   * Prune Noise: Mark all steps logically located between
  Preceding_Step
     and Current_Step as NOISE (irrelevant actions) and exclude
  them.
   * Update: Set Current_Step = Preceding_Step.
   * Repeat recursively until the start of the episode is reached.

**Phase 2: The Abstraction Layer (Strict Generalization Rules)**
Once the causal chain is identified, rewrite all content using
  these
environment-specific rules:

1. Entity Abstraction (Environment Specific):
   * IF ALFWorld: keep the Specific Object Name but REMOVE
  Instance IDs
     (e.g. "mug 1" -> "mug", "cabinet 2" -> "cabinet").
   * IF WebShop: replace specific product titles or IDs with the
  Broad
     Item Category (e.g. "Samsung Galaxy S21" -> "Smartphone").
   * IF DBBench: keep table/column structure but REMOVE specific
  values
     (e.g. "SELECT name FROM users WHERE age > 25"
        -> "SELECT [column] FROM [table] WHERE [condition]").
   * IF AppWorld: keep API structure but REMOVE specific IDs/names
     (e.g. "GET /users/12345/playlists"
        -> "GET /[resource]/[id]/[sub_resource]").

2. Attribute Abstraction (Constraint Categories):
```

Prompt (continued)

```
     * NEVER output specific numbers or values (e.g. "$50.00",
    "Blue").
     * ALWAYS replace them with the Constraint Type placeholder:
       "$50.00" -> [Price_Constraint], "Red" -> [Color_Constraint],
       "Small" -> [Size_Constraint].

**Phase 3: Extract Planning Pattern**
Abstract the refined causal chain into a high-level Logical Chain.
* Syntax: use " -> " to separate major steps.
* NEVER use specific names. Replace with placeholders: [Object],
  [Location], [Constraint], [Query], [Tool], [Destination].

**Phase 4: Extract Rubrics**
From the planning pattern, identify 1-3 rubrics the agent
  demonstrated.
Each rubric is a specific observable correct behaviour
  (snake_case).
  - role: "demonstrates"
  - count: 1


  ========================================================================
IF FAILURE (reward <= 0)

  ========================================================================

Do NOT trace a causal chain. Instead:

Step A -- Scan and Classify: read the entire rollout. For every
  wrong
  behaviour you observe, give it a short snake_case name and count
  how
  many times it occurs.
Step B -- Rank by Frequency: sort by count descending. The failure
  that
  repeats 5 times matters far more than one that happens once.
Step C -- Report Top 2-4: output the 2-4 most frequent failure
  patterns
  (most frequent first).

  ========================================================================
IF PARTIAL SUCCESS (0 < reward < 1)

  ========================================================================

Do BOTH:
  - Apply the SUCCESS strategy for rubrics that WORKED
  - Apply the FAILURE strategy for rubrics that FAILED
  Report both demonstrated and violated rubrics.

================================================================
RUBRIC NAMING GUIDELINES
================================================================

Rubric names should be generalizable across environments -- they
  should
describe observable behaviours that could appear in any agentic AI
  system.

Use snake_case for all names. If a rubric is truly
  environment-specific,
```

Prompt (continued)

```
  prefix with the env name:
    e.g. "WebShop__price_operator_in_search",
    "DBBench__wrong_sql_syntax"

  ====================================================================
  OUTPUT FORMAT (strict JSON, nothing else):
  ====================================================================

  {
    "outcome": "success|failure|partial",
    "planning_pattern": "<abstract rule with [Placeholders] and ->,
                         null if failure>",
    "rubrics": [
      {
        "rubric": "<snake_case rubric name>",
        "role": "demonstrates|violates",
        "count": 1
      }
    ],
    "learning_stage": "random_exploration|skill_acquisition|
                       near_competent|competent",
    "descriptions": {}
  }
```

Rubrics Tagging Prompt (training-time tagging) Used for all tagging once the vocabulary is fixed, including online tagging during training. The taxonomy is injected as {taxonomy}: one line per skill listing its rubric names. The analysis strategy branches on the rollout outcome.

Prompt

```
You are an expert **Trajectory Analysis Engine**.

Analyze a single RL training trajectory from an agent operating in
interactive environments (ALFWorld, WebShop, DBBench, AppWorld).

Use a DIFFERENT analysis strategy depending on the trajectory
    outcome.

====================================================================
IF FULL SUCCESS (reward = 1.0)
====================================================================

**Phase 1: The Refinement Algorithm (Strictly Backward Logic)**
1.  **Start Point:** Identify the **Last Successful Step** (The step
    that achieved the goal). Call this `Current_Step`.
2.  **Recursive Step (The Loop):**
    * **Analyze Dependency:** Determine the strict **Required
    Precondition** for `Current_Step`. Ask: "What specific state,
    information, or object availability was necessary for this
    action to occur?"
    * **Scan Backwards:** Traverse the log in reverse order from
    `Current_Step`.
    * **Find Nearest Enabler:** Identify the **nearest** preceding
    step (`Preceding_Step`) that explicitly generated this
    Precondition.
    * **Prune Noise:** Mark all steps logically located *between*
    `Preceding_Step` and `Current_Step` as NOISE (irrelevant
    actions) and exclude them.
    * **Update:** Set `Current_Step` = `Preceding_Step`.
```

Prompt (continued)

```
    * **Repeat:** Continue this process recursively until the start
    of the episode is reached.

**Phase 2: The Abstraction Layer (Strict Generalization Rules)**

Once the causal chain is identified, rewrite all content using these
    environment-specific rules:

1.  **Entity Abstraction (Environment Specific):**
    * **IF ALFWorld:** Keep the **Specific Object Name** but
    **REMOVE Instance IDs** (e.g., "mug 1" -> "mug").
    * **IF WebShop:** Replace specific product titles with **Broad
    Item Category** (e.g., "Samsung Galaxy S21" -> "Smartphone").
    * **IF DBBench:** Keep **table/column structure** but **REMOVE
    specific values** (e.g., "SELECT name FROM users WHERE age > 25"
    -> "SELECT [column] FROM [table] WHERE [condition]").
    * **IF AppWorld:** Keep **API structure** but **REMOVE specific
    IDs/names** (e.g., "GET /users/12345/playlists" -> "GET
    /[resource]/[id]/[sub_resource]").

**Phase 3: Extract Rubrics**
From the refined causal chain, identify 1-3 rubrics the agent
    demonstrated correctly.
The predefined taxonomy below may not cover limited success
    behaviors - feel free to use
"discovered::" for positive rubrics that describe what the agent did
    right.

================================================================
IF FAILURE (reward <= 0)
================================================================

Do NOT trace a causal chain. Instead:

**Step A - Scan and Classify:**
Read the entire trajectory. For every wrong behavior you observe,
classify it and count how many times it occurs.

**Step B - Rank by Frequency:**
Sort by count descending. The failure that repeats 5 times matters
far more than one that happens once.

**Step C - Report Top 2-4:**
Output the 2-4 most frequent failure patterns (most frequent first).

================================================================
IF PARTIAL SUCCESS (0 < reward < 1, e.g. AppWorld passing some test
    cases)
================================================================

Do BOTH:
  - Apply the SUCCESS strategy for rubrics that WORKED (what got
    partial credit)
  - Apply the FAILURE strategy for rubrics that FAILED (what
    prevented full credit)
  Report both demonstrated and violated rubrics.
  For positive rubrics (demonstrates), feel free to use
    "discovered::" if the taxonomy
  does not cover the successful behavior observed.

================================================================
```

Prompt (continued)

```
PREDEFINED SKILL + RUBRIC TAXONOMY
================================================================

{taxonomy}

================================================================
RULES
================================================================

- Use EXACT predefined skill and rubric names from the taxonomy when
    they match.
  Do not paraphrase or create synonyms for predefined names (e.g.,
    use "premature_termination"
  not "ends_task_early" or "fakes_task_completion" if
    premature_termination fits).
- If you observe a behavior NOT covered by the taxonomy above,
  identify patterns not addressed by current skills/rubrics and
    propose new ones:
  - New skill: use "discovered::snake_case_name"
  - New rubric: use "discovered::snake_case_name"
- When discovering a new rubric, prefer assigning it to an EXISTING
    skill if one
  is a reasonable fit. Only discover a new skill if no existing
    skill covers it.
- Discovered names should be general and environment-agnostic (e.g.
    "corrects_action_format"
  not "fixes_ALFWorld_put_syntax"). Use common, widely-applicable
    names.
- Skills are ABSTRACT capabilities. Rubrics are SPECIFIC observable
    behaviors.
- 1-5 rubrics per trajectory

OUTPUT FORMAT (strict JSON, nothing else):
{{
  "outcome": "success|failure|partial",
  "rubrics": [
    {{
      "skill": "<skill_name or discovered::new_name>",
      "rubric": "<rubric_name or discovered::new_name>",
      "role": "demonstrates|violates",
      "count": 1
    }}
  ],
  "descriptions": {{}}
}}

Rules:
- skill and rubric: REQUIRED
- descriptions: one-sentence definition for any discovered:: name;
    {{}} otherwise
- No text outside the JSON
```

A.4 Further Detail on Rubric Tagging.

We implement rubric tagging using an asynchronous primary path and a retry path. The primary prompt always contains the predefined vocabulary and successfully processes approximately 98% of rollouts. Both paths use a tagging template that prioritizes supplied rubric names but permits additional names for uncovered behaviours. Such discovered names are rare, accounting for less than 1% of emitted rubric names. To reduce retry overhead, the retry path uses a shorter candidate list refreshed every ten steps from frequently observed rubric names in the preceding ten steps;

Table 5: Training and validation data for the four environments. Sizes count task instances, not generated rollouts. AppWorld training includes only easy and medium tasks, whereas its validation set includes all difficulty levels. The main experiments and ablations use the first three environments; the preliminary correlation analysis and additional experiments use all four. AG denotes AgentBench

Environment	Train	Validation	Training Data / Selection	Validation Data / Selection
ALFWorld	3,150	109	AG `train_valid`	AG `new_std`
DBBench	4,803	300	AG `db_out_new.jsonl`	AG `standard.jsonl`
WebShop	11,000	200	AG indices $[1000, 12000)$	AG indices $[0, 200)$
AppWorld	72	57	Easy and medium training tasks	Complete dev split

this list may include discovered names. These updates affect only the retry candidate list, not the vocabulary supplied to the primary path. Before constructing selection profiles, returned names are normalized and mapped to predefined rubric identifiers using edit-distance matching. Teacher conditioning and rollout guidance use the predefined descriptions.

B Training Details

B.1 Training Datasets and Environments

Our main experiments and ablation studies use three interactive environments: ALFWorld (Shridhar et al., 2021), DBBench (Liu et al., 2024), and WebShop (Yao et al., 2022), covering household manipulation, database operations, and online shopping, respectively. The preliminary correlation analysis in Sec. 2.3 and the additional experiments in the Sec. 5 include AppWorld (Trivedi et al., 2024) as a fourth environment, extending the analysis to application-based digital tasks. We therefore summarize all four environments in Tab. 5.

For ALFWorld, DBBench, and WebShop, we adopt the training data and environment implementations provided through AgentBench, which integrates with AgentRL (Zhang et al., 2025). For AppWorld, we follow the training-task selection of Chen et al. (2025) and retain only easy and medium tasks (difficulty levels 1 and 2), yielding 72 training tasks across 24 scenarios. For validation, we use the designated validation set of each environment without subsampling. Specifically, we use the AgentBench evaluation splits for ALFWorld, DBBench, and WebShop, and the full released development split for AppWorld. The latter contains 57 tasks spanning all difficulty levels; the difficulty restriction applies only to AppWorld training. Dataset sizes denote the numbers of task instances, each defined by an instruction and its associated initial environment state. Multiple interaction rollouts can be generated for the same task. All four environments support multi-turn interaction, allowing agents to use feedback from previous actions to guide subsequent decisions.

B.2 Training Hyperparameters

Detailed hyperparameters used for the training are provided in Tab. 6.

B.3 Training Loss

We follow the GRPO formulation and training setup of AgentRL (Zhang et al., 2025). For each selected group $g = (\tau_{g,i})_{i=1}^{n} \in \mathcal{S}_t$, let $r_{g,i}$ be the rollout reward and define the group-relative advantage

$$\hat{A}_{g,i} = \frac{r_{g,i} - \operatorname{mean}_j r_{g,j}}{\operatorname{std}_j r_{g,j} + \epsilon_A}.$$

Following AgentRL, the group-relative advantages are subsequently normalized over generated tokens within each environment. Let $\tilde{A}_{g,i}$ denote the resulting advantages used for optimization. For exposition, we present a compact group-indexed form of the GRPO objective; details of token-level loss computation and normalization can be found in the AgentRL implementation. We use the same GRPO hyperparameters as AgentRL, with backbone-specific settings reported in Tab. 6.

Table 6: Training configuration for the two backbones. Values shared by both are given once; where they differ, both are listed.

	`Qwen2.5-3B-Instruct`	`Qwen3-4B`
Rollout		
Sampling temperature	0.8	0.6
Max new tokens per turn	1024	2048
Reasoning mode	—	disabled
Max turns per episode	20	
Max episode length (tokens)	8192	
Unfinished-episode penalty	-0.2	
Optimisation		
Prompts per step	64	
Rollouts per prompt n	8	
rollouts per step	512 (from 1024 candidates)	
Learning rate	1×10^{-6}	
Weight decay	0	
KL-to-reference coefficient	1×10^{-4}	3×10^{-4}
Loss reduction	token-mean	seq-mean
Tokens per micro-batch	16,384	
Advantage estimator	GRPO, per-task normalised	
Distillation coefficient λ	0.01	
Actor precision	`float32`	
Training steps	1000	1000
Validation		
Interval	every 10 steps	
Rollouts per task	4	
Rubric tagging		
Tagger	`Qwen3-32B`, tensor parallel 8	
Vocabulary	59 rubrics, 10 skills	
rollout truncation	16k characters	

Let

$$\rho_{g,i}(\theta) = \frac{\pi_\theta(y_{g,i} \mid z_g)}{\pi_{\theta_{\text{old}}}(y_{g,i} \mid z_g)}.$$

Using the loss-minimization convention, the GRPO objective is

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}_{g \in \mathcal{S}_t}\left[\frac{1}{n}\sum_{i=1}^{n} \min\Big(\rho_{g,i}(\theta)\tilde{A}_{g,i}, \text{clip}(\rho_{g,i}(\theta), 1-\epsilon, 1+\epsilon)\,\tilde{A}_{g,i}\Big)\right] + \beta\, D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})\,, \tag{3}$$

where θ_{old} denotes the rollout policy and π_{ref} is the frozen reference policy.

B.4 Algorithm Pseudo-code

Alg 1 presents the conceptual training procedure for RISED. Our experiments implement it using AgentRL's rollout and optimization conventions. An overview of the pipeline is also provided in Fig. 5.

C Additional empirical analysis

Unless otherwise specified, all experiments in this section are conducted on `Qwen2.5-3B-Instruct`.

Algorithm 1 One training step for RISED

Require: policy parameters θ, environments $\mathcal{E}$, group budget k, rollouts per group n, rubric vocabulary $\mathcal{R}$, descriptions D, distillation coefficient λ
Persistent state: environment-specific negative-rubric blocks $A_t[e]$, initialized empty at $t = 0$

$\theta_{\text{old}} \leftarrow \theta$; $\bar{\theta} \leftarrow \text{sg}(\theta_{\text{old}})$
sample $2k$ prompts in round-robin order across $\mathcal{E}$
Rollout. For each sampled prompt x_g from environment e, update its negative-rubric block to $A_t[e]$ when $A_t[e] \neq \emptyset$; let z_g denote the steered prompt and generate a group $g = (\tau_{g,i})_{i=1}^{n}$ of n under rollout pipeline.
$\mathcal{B}_t \leftarrow$ sampled groups; tag all $\tau \in \mathcal{B}_t$ against $\mathcal{R}$
selection (§2.4)
$\mathcal{S} \leftarrow \emptyset$
for $e \in \mathcal{E}$ cyclically until $|\mathcal{S}| = k$ **do**
  $g^\star \leftarrow \arg\max_{g \in \mathcal{B}_t \setminus \mathcal{S}:\, e(g)=e} \text{score}(g \mid \mathcal{S})$; $\mathcal{S} \leftarrow \mathcal{S} \cup \{g^\star\}$
end for
distillation (§3)
for $g \in \mathcal{S}$ and rollout $\tau \in g$ **do**
  $R_\tau^+ \leftarrow \text{Top}_3(\textsf{pos}(\tau)) \cup \text{Top}_3(\textsf{pos}(g))$; $c(\tau) \leftarrow D[R_\tau^+]$ rendered as a privileged context
end for
update θ on $\mathcal{S}$ with $\mathcal{L}_{\text{GRPO}} + \lambda \mathcal{L}_{\text{SD}}$
negative steering for next step $t+1$ (§3)
for $e \in \mathcal{E}$ **do**
  $\mathcal{C}_t(e) \leftarrow \bigcup_{g \in \mathcal{B}_t : e(g)=e} g$; $(n_e^-, n_e^+) \leftarrow \sum_{\tau \in \mathcal{C}_t(e)} \big(\mathbf{1}[\textsf{neg}(\tau) \neq \emptyset], \mathbf{1}[\textsf{pos}(\tau) \neq \emptyset]\big)$
  $A_{t+1}[e] \leftarrow \begin{cases} D[\text{Top}_3(\textsf{neg}(\mathcal{C}_t(e)))], & n_e^- > n_e^+, \\ \emptyset, & \text{otherwise.} \end{cases}$
end for
return θ, A_{t+1}

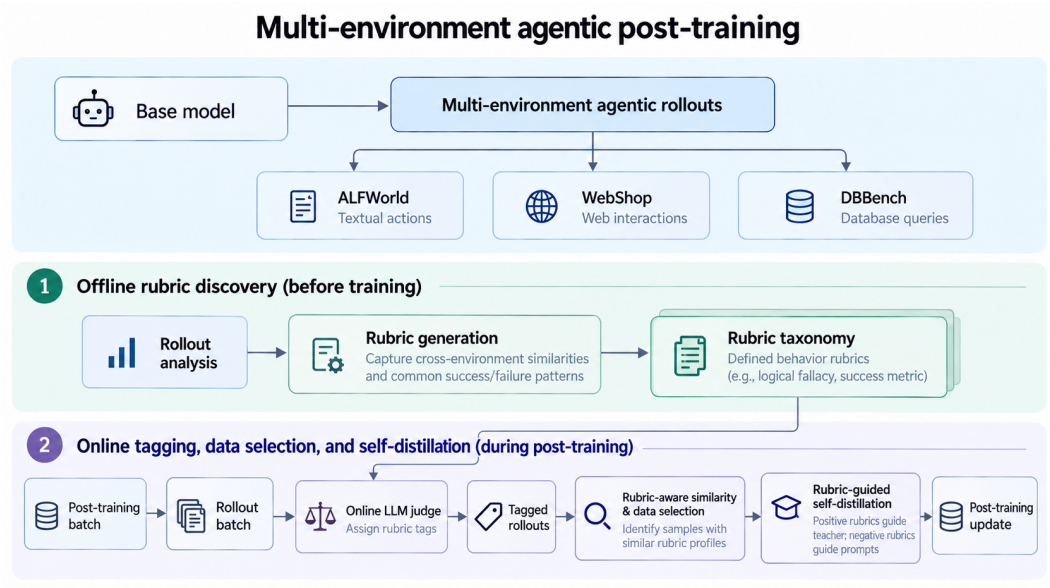

Figure 5: Overview of RISED.

C.1 Additional Cross-Environment Transfer Analysis

We repeat the analysis in Sec. 2.3 with DBBench as the sole training environment. The evaluation environments, profile construction, and correlation protocol remain unchanged; the held-out pool now excludes DBBench.

As shown in Tab. 7, all-tag rubric similarity achieves a higher held-out correlation than all-tag skill similarity and gradient similarity. Positive-only profiles achieve higher correlations in three of the four comparisons, whereas combined profiles attain the highest all-environment Pearson correlation.

Table 7: Correlation between similarity signals and reward improvement under DBBench-only training. Spearman (ρ) and Pearson (r) correlations are pooled over checkpoint–environment pairs at steps 50–500, relative to the step-25 baseline. Held-out results exclude DBBench. Positive-only, negative-only, and combined profiles are denoted by pos, neg, and all, respectively.

		Rubric-level			Skill-level			Gradient
Val. envs		pos	neg	all	pos	neg	all	full layer.
Held-out	ρ	$+\mathbf{0.50}$	$+0.32$	$\underline{+0.47}$	$+0.46$	$+0.40$	$+0.40$	$+0.20$
	r	$+\mathbf{0.54}$	$+0.42$	$+0.43$	$\underline{+0.51}$	$+0.36$	$+0.38$	$+0.18$
All env	ρ	$+\mathbf{0.77}$	$+0.69$	$\underline{+0.76}$	$+0.74$	$+0.73$	$+0.72$	$+0.41$
	r	$+0.80$	$\underline{+0.89}$	$+\mathbf{0.92}$	$+0.63$	$+0.68$	$+0.79$	$+0.41$

Positive-only profiles perform well under DBBench training but show negative held-out correlations under WebShop training. Combined profiles remain among the strongest positively correlated signals in both settings, motivating our use of both positive and negative rubric roles.

Additionally, we would like to clarify some empirical details on gradient-based signals here. We compute training and validation policy gradients weighted by raw environment rewards, using the same rollout pools as the rubric analysis. We average the gradients within each pool and compute the cosine similarity between the resulting vectors.

C.2 Tagger reliability

We assess the reproducibility of the LLM tagger by tagging the same rollouts in two separate runs with identical model and tagging settings, using different inference endpoints.

Tag and profile agreement. The two runs produce identical tag sets for 68.8% of rollouts, with a mean Jaccard overlap of 0.856 and a median of 1.000. Despite this rollout-level variation, aggregate profiles remain highly consistent: mean cross-run cosine similarity ranges from 0.9622 to 0.9992 across the training and validation profiles.

Downstream stability. Tab. 8 shows that the resulting similarity signals and transfer correlations are also stable. All reported Spearman correlations change by at most 0.060, preserving their signs and the ranking of feature variants. The held-out correlation for the rubric-all changes only slightly from 0.595 to 0.588. These results support the repeatability of the environment-level similarity and transfer-correlation analysis.

Table 8: Tagger reproducibility across two runs. Signal agreement is measured over checkpoint–environment pairs using Pearson correlation (r) and mean absolute difference (MAD). Transfer correlations are pooled Spearman coefficients (ρ), shown as Run 1 / Run 2.

	Signal agreement		Held-out ρ		All env ρ		
Feature	r	mean $\lvert\Delta\rvert$	run 1	run 2	run 1	run 2	max $\lvert\Delta\rho\rvert$
Rubric, all	0.9992	0.0115	$+0.595$	$+0.588$	$+0.814$	$+0.809$	0.007
Rubric, violates	0.9959	0.0260	$+0.040$	$+0.052$	$+0.570$	$+0.576$	0.012
Rubric, demonstrates	0.9663	0.0502	-0.522	-0.474	$+0.320$	$+0.345$	0.048
Skill, all	0.9986	0.0142	$+0.431$	$+0.426$	$+0.750$	$+0.746$	0.005
Skill, violates	0.9919	0.0312	$+0.302$	$+0.362$	$+0.688$	$+0.712$	0.060
Skill, demonstrates	0.9622	0.0525	-0.687	-0.681	$+0.250$	$+0.252$	0.006

C.3 Role Filter in the Teacher Context

We retain rubric-guided selection and compare three uses of rubric feedback: conditioning the teacher on both positive and negative rubrics; conditioning it on aggregated positive rubrics (deduplicated union of its top-3 positive (*demonstrate*) rubrics and the top-3 positive rubrics of its group);

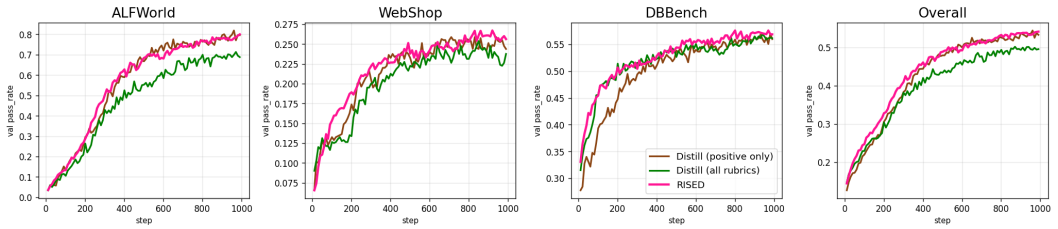

Figure 6: **Rubric roles in teacher conditioning on Qwen2.5-3B-Instruct.** We show positive-only conditioning without negative prompt injection (brown) and all-rubric conditioning with selection (green). Full RISED (pink) is included for reference. Positive-only conditioning outperforms all-rubric conditioning overall, particularly on ALFWorld, while full RISED learns faster initially than the positive-only variant, most clearly on DBBench.

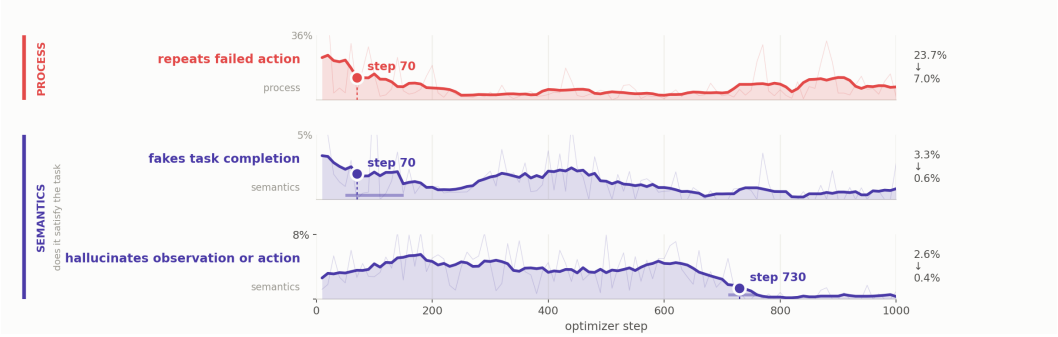

Figure 7: **Negative rubrics reveal distinct stages of behavioural improvement in DBBench.** Curves track the fraction of rollouts exhibiting each undesired behaviour. Markers indicate when each undesired behaviour rate suddenly falls to half its initial level. `Repeated failed actions` and `false completion claims` reach this milestone substantially earlier than `hallucinated observations or actions`.

and RISED, which combines positive-only teacher conditioning with negative-rubric injection into rollout prompts. The first two variants use no rubric guidance in rollout prompts; only full RISED injects negative rubrics during rollout generation. Note that we distinguish the all-rubric baseline in Tab. 2 from RISED without selection in Fig. 2 that these two do not have the selection component.

Fig. 6 supports our decision to exclude negative rubrics from the teacher's privileged context. The all-rubric variant achieves lower overall performance than positive-only conditioning, particularly on ALFWorld. A possible explanation is that instructions to avoid observed violations shift the teacher toward alternative actions, creating tension with distillation on the sampled rollout. Positive-only conditioning instead highlights demonstrated behaviours to reinforce.

Negative rubrics can nevertheless provide useful guidance during rollout generation. Full RISED improves faster initially than the positive-only variant, which does not have negative rubric steering, most clearly on DBBench. There, positive tags occur very few times, whereas negative rubrics identify actionable failures such as `sql_syntax_or_execution_error` and `missing_commit`. Together, these observations motivate assigning the two rubric roles to different stages: positive rubrics guide distillation, while negative rubrics guide subsequent exploration.

C.4 Additional Qualitative Analysis

Rubrics reveal behavioural curriculum. Fig. 7 shows that different negative (failure and hence undesired) behaviours in DBBench disappear at markedly different stages of training. `Repeating failed actions` and `falsely claiming task completion` claims fall below half of their initial rates around step 70, whereas `hallucinated observations or actions` do not reach the same sustained reduction until step 730. This separation suggests an emergent behavioural progression in which some failure modes are resolved early while others persist until

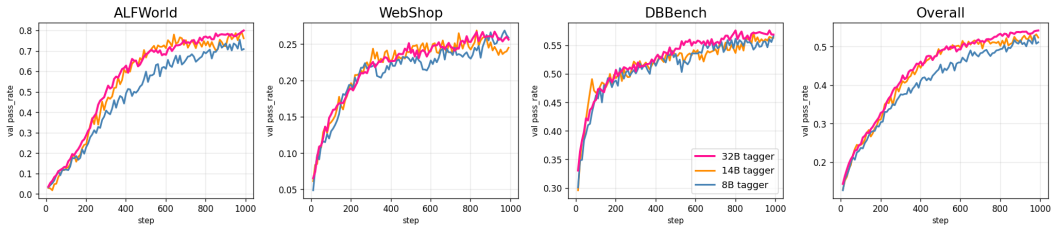

Figure 8: **Effect of rubric judge size.** Validation pass rate for RISED with 8B, 14B, and 32B judges. The 14B judge remains competitive, while the 32B judge achieves the best overall performance.

Table 9: Per-step wall-clock time (seconds) for `Qwen2.5-3B-Instruct` across three environments using 32 H100 GPUs for training and rollout generation. Methods requiring rubric annotations additionally use a dedicated eight-GPU judge server. Reported values are averaged over first 300 steps. Roll. wait measures blocking on batch retrieval; Tag-wait measures blocking to complete rubric annotations. Ref computes reference-policy log probabilities for sampled rollouts; Teacher computes rubric-conditioned teacher log probabilities for distillation. Actor includes the policy forward pass, backpropagation, and optimizer update. Other is the remaining step time, including checkpointing and trainer overhead.

Method	Roll. wait	Tag-wait	Ref	Teacher	Actor	Other	**Step**	Relative
GRPO-64	0.5	–	25.7	–	89.7	39.6	**155.5**	1.00×
GRPO-128	0.8	–	48.0	–	167.0	33.8	**249.6**	1.61×
Global Variance	16.5	–	27.9	–	95.6	39.1	**179.1**	1.15×
Round-robin Variance	21.9	–	26.5	–	88.4	40.5	**177.3**	1.14×
RISED	8.6	63.0	27.2	30.6	92.1	31.4	**252.9**	1.63×

much later in training. Rubric-level rollouts therefore reveal changes in agent behaviour that are obscured by outcome rewards alone.

C.5 Effect of judge model size

We compare 8B, 14B, and 32B rubric judges while keeping the remaining RISED pipeline fixed (Fig. 8). The 32B judge achieves the highest overall validation performance, while the 14B judge retains most of the gain and remains competitive across environments. The 8B judge performs consistently worse overall. Interestingly, a smaller judge does not necessarily reduce wall-clock time in our serving setup: the 8B configuration incurs greater rubric-tagging wait time. Thus, judge size exhibits a performance–serving trade-off in our experiments, with 14B providing a competitive intermediate configuration.

C.6 Training-Time Analysis

Tab. 9 reports per-step wall-clock time under the resource allocations described in Sec. 4.1. Rubric-guided selection itself takes less than $0.1\,\mathrm{s}$ per step. RISED's additional latency comes primarily from rubric tagging and the teacher forward pass. Its actor-update time remains close to GRPO-64, while its total step time is $1.63\times$ that of GRPO-64 and comparable to GRPO-128. Because RISED additionally uses a dedicated judge server, these measurements compare elapsed training time rather than total compute.

AgentRL's asynchronous pipeline overlaps rollout generation with optimization, so rollout wait measures exposed blocking time rather than total generation cost. Although RISED generates $2k$ candidate groups and selects k, its additional tagging and distillation computation gives background rollouts more time to complete before the next batch is requested. This may explain its lower rollout wait relative to the variance-based selectors and should not be interpreted as faster rollout generation. In our implementation, rubric tagging is the largest source of additional blocking latency, suggesting that more efficient judge serving could further reduce RISED's wall-clock overhead.

D Additional Literature Review

Multi-environment agentic RL. Joint training with multiple environments has been found to have the potential of matching environment-specific specialists (Cheng et al., 2025; Wang & Ammanabrolu, 2025; Zhang et al., 2025). Cross-domain studies also report positive transfer between tasks: In a five-domain comparison, mixed RL achieves performance comparable to separately trained and merged models at lower total training cost (Wang et al., 2026b). For interactive agents, Liu et al. (2026b) identify state information richness and planning complexity as important correlates of cross-domain generalization. Joint training on a mixture is therefore raising more and more attention. Training strategies include sequential domain stages (NVIDIA, 2025a; Ren et al., 2026) and joint optimization over mixed-environment batches (NVIDIA, 2025b; Zhang et al., 2025). Nemotron-Cascade 2 orders stages to reduce interference, but notes that suitable orderings depend on the model's learning dynamics and that capability drift remains (Yang et al., 2026b). We follow the mixed-batch approach, holding environment proportions fixed.

Curriculum and domain allocation. RL-related curriculum methods control sample difficulty, category or domain allocation, and environment membership. Nemotron 3 Nano schedules sample difficulty using checkpoint-based pass-rate estimates, which are refreshed when training progress plateaus (NVIDIA, 2025b). Nemotron-Cascade 2 orders training stages to mitigate inter-domain interference (Yang et al., 2026b). MoDoMoDo optimizes domain mixtures using predicted training outcomes from pilot runs (Liang et al., 2025). Online adaptive methods include SEC's bandit-based category sampling (Chen et al., 2026), RuleReasoner's reward-history-based domain reweighting (Liu et al., 2026a), and TAC's use of gradient alignment to select the domain supplying each training batch (Yang et al., 2026a). These methods primarily determine category or domain allocation, stage ordering, or sample difficulty. Our focus is complementary: selecting prompt groups from current mixed-environment rollouts under a prescribed environment allocation, with selection accounting for both the batch's behavioural composition and previously selected groups. Closest to our rubric profiles as selection reference, AES aggregates trajectory-derived abilities into environment profiles and selects environment subsets using coverage, redundancy, and gradient conflict (Zhu et al., 2026b), but note that they select on environment level but not prompt groups level. In our AgentRL-based implementation (Zhang et al., 2025), we fix the candidate environment mixture and RISED's round-robin selection schedule to focus on the problem of prompt-group selection. Directly applying domain schedulers or environment-subset selection would change our experimental setting. We acknowledge that jointly optimizing these decisions remains a parallel but important direction which we currently do not focus on. Global Variance nevertheless allows selected environment proportions to vary with observed reward variance, providing a comparison with adaptive batch composition within the fixed candidate pool.

Rubrics as reward. Rubrics decompose response quality into interpretable criteria and provide reward supervision for open-ended generation (Gunjal et al., 2026; Jia et al., 2026), agentic tool use (Zhang et al., 2026), and image captioning (Huang et al., 2026). Later work improves both the criteria and their use in optimization: RLCER evolves rubric criteria during training (Sheng et al., 2026), POW3R adapts criterion weights according to rollout variation (Tyagi et al., 2026), and CriPO uses self-distillation and advantage correction to recover learning signals for unexplored or suppressed criteria (Xia et al., 2026a). These methods primarily construct or adapt rubric-based rewards, while CriPO additionally improves optimization under those rewards through self-distillation and advantage correction. RISED retains environment-provided rewards and instead uses behavioural rubrics for prompt-group selection, teacher conditioning, and rollout guidance. Our focus is therefore how behavioural rubrics organize training data and provide auxiliary supervision, rather than how rubric scores construct the reward.

On-policy distillation and its selective variants. Distillation trains a student to match a teacher's output distribution, providing dense token-level supervision. On-policy distillation applies this supervision to student-generated rollouts, reducing the distribution mismatch of off-policy imitation (Agarwal et al., 2024). Recent on-policy self-distillation methods construct a teacher from the student model, conditioning it on privileged information or feedback to provide dense supervision on student-generated rollouts (Hübotter et al., 2026; Zhao et al., 2026). Li et al. (2026b) link successful distillation to teacher–student compatibility and additional teacher capabilities, with learning

concentrated on shared high-probability tokens. Several variants refine supervision through uncertainty and disagreement (Xu et al., 2026), token teachability (Wang et al., 2026c), trajectory filtering and token reweighting (Li et al., 2026c), or trust-region restrictions on teacher supervision (Xing et al., 2026). For multi-turn agents, Skill-SD constructs teacher context from rollout-derived skill summaries and derives the importance-weighted reverse-KL objective (Wang et al., 2026a); SDAR uses token-level gating to stabilize joint RL and self-distillation (Lu et al., 2026), while SERL studies which environmental feedback to distill and where to apply it (Li et al., 2026a). We adopt Skill-SD's objective, but construct privileged context from a shared cross-environment rubric vocabulary whose annotations also guide prompt-group selection. Our contribution is to use a shared behavioural rubric vocabulary jointly for prompt-group selection and self-distillation. The component ablations (Sec. 5) show that combining the two yields the strongest overall performance, supporting their complementary roles. Token-selection rules and alternative distillation losses address complementary optimization choices. Closest to our distillation component, RuPI supplies rubric criteria, including positive and negative criteria, as privileged teacher context (Bablani et al., 2026). Our All-Rubric SD baseline adapts this approach to agentic RL by retaining GRPO and using rollout-derived rubrics. RISED differs by combining distillation with prompt-group selection, while assigning positive rubrics to teacher conditioning and negative rubrics to rollout guidance.

E FAQ

E.1 Why use a fixed rubric vocabulary?

Our goal requires comparing rubric profiles across environments and training steps. Dynamic generation of free-form rubric assigns semantically equivalent behaviours different names and therefore requires repeated clustering or alignment before profiles can be compared. In preliminary experiments, clustering free-form annotations yielded between 25 and 44 skills across checkpoints, making comparisons across steps and environments difficult. The verification of each step's new rubrics can also be difficult. A fixed vocabulary instead provides a stable coordinate system for online comparison.

E.2 Will the warm-up process in building vocabulary introduce additional time cost?

The warm-up run, taking only $1/10$ steps of full training, introduces a one-time preprocessing cost that is amortized across subsequent training runs:

1. **Reuse across backbones.** The warm-up policy provides failure trajectories used to discover negative rubrics. Once constructed, the resulting vocabulary is reused across training runs and model backbones, so a separate warm-up run is not required for each backbone.
2. **Reduced recurring overhead.** A predefined vocabulary reduces online tagging to label selection. In contrast, free-form extraction would require repeated name alignment or clustering to reconcile semantically equivalent labels and preserve comparability across rollouts and training steps.
3. **Offline use of larger models.** Rubric consolidation and description generation are performed offline using a larger model, while online tagging uses a locally served judge. This avoids repeated large-model inference during training and reduces throughput overhead.

E.3 Why evaluate rubric similarity at the environment level?

We use environment-level analysis as a computationally tractable diagnostic of whether rubric similarity tracks reward improvements during single-environment training. Directly estimating the marginal benefit of individual prompt groups would require more expensive controlled updates and evaluations. The aggregate analysis motivates the predefined and shared rubric representation; it does not directly estimate each group's contribution. We assess the practical utility of prompt-group selection through the main experiment and the end-to-end component ablations.

E.4 Does reward-conditioned tagging affect the interpretation of rubric similarity?

The judge receives rollout rewards and uses outcome-dependent instructions to identify desired and undesired behaviours. The resulting rubric profiles therefore reflect both behavioural content and outcome composition. The gradient baseline also incorporates rewards through its objective. Our comparison evaluates how well these two reward-informed representations track observed environment-level reward improvements. The stronger correlations support rubric similarity as an empirical diagnostic in this setting, but do not establish that its association is independent of reward composition.

E.5 Are negative rubrics injected during validation?

No. Negative rubrics guide training rollouts only. All reported validation results use the original task prompts without rubric injection. The role-filter ablation in App. C.3 and main results in Sec. 4.3 evaluate these training choices under the same unguided validation protocol. Its results show that the benefits of training-time guidance persist when the guidance is removed at evaluation.

E.6 Does every group with identical rewards receive positive-rubric supervision?

Not necessarily. Positive-rubric self-distillation can provide additional supervision for groups with identical rewards when positive rubrics are available within the group. Groups without positive rubrics lack this privileged behavioral context, and the distillation term has zero loss and gradients when teacher and student parameters and inputs coincide. However, negative rubrics can still provide a complementary mechanism by guiding subsequent rollout generation.

E.7 Why not evaluate larger policy backbones?

Agent RL is already computationally expensive at the 3–4B scale. For `Qwen2.5-3B-Instruct`, each run comprises 1000 training steps on 32 H100 GPUs. Based on the measured average step times, even GRPO-64 requires approximately 1.8 days per run (155.5 s/step), while RISED requires approximately 2.9 days (252.9 s/step). Validation every 10 steps further increases the total runtime. Repeating training across baselines, ablations, and seeds incurs substantial computational cost. We therefore prioritize controlled comparisons across provided two backbones.