HiRUB: Hierarchical Rubric Rewards for Long-Horizon Large Language Model Agents

Anonymous authors
Paper under double-blind review

Abstract

Long-horizon agent reinforcement learning (RL) relies on informative trajectory-level rewards that indicate how a complete interaction advances the requested outcome. Success in these tasks can depend on a chain of interdependent decisions, with different actions contributing unequally to the eventual result. Consequently, terminal feedback alone is often too coarse to distinguish useful progress from ineffective behavior. LLM judges can provide richer trajectory assessments, yet placing them inside the RL loop requires repeated inference for every rollout and becomes costly as trajectories and rollout groups grow. To make rubric-style feedback practical without retaining this online expense, we present HiRUB, a hierarchical rubric reward that uses an LLM to generate task-aware criteria before training and applies them through executable checks during RL. These criteria are related to a task schema, allowing each complete rollout to receive a task-aware score that combines partial progress, semantic constraints, successful completion, and penalties. This design keeps the reward computation lightweight and compatible with different RL backbones while avoiding online LLM judging. Experiments across ALFWorld and WebShop with 1.5B and 7B parameters show consistent gains over the evaluated RL baselines. In a matched ALFWorld comparison against an online LLM-judge reward, HiRUB improves validation success from 90.1% to 92.2% and reduces total training time from 31.1 to 19.6 hours. Together, these results show that task-aware trajectory scoring can improve agent learning without placing judge inference in the training loop.

1 Introduction

Large language model (LLM) agents can act interactively by reading observations, producing actions, and receiving environment feedback. In RL training, each interaction history is evaluated as a trajectory whose return provides the optimization target for the policy update (Shao et al., 2024; Ahmadian et al., 2024; Feng et al., 2025; Jin et al., 2025; Luo et al., 2025). The reward is therefore the interface that converts environment interaction into an optimization signal. Before the optimizer can learn which behavior to reinforce, the reward has to decide which sampled trajectories contain useful task progress. Long-horizon tasks put particular pressure on this interface. A rollout may span many turns, while the most reliable environment feedback is often a binary success flag at the end of the episode (Feng et al., 2025). In this setting, terminal feedback may assign the same value to a trajectory that made useful progress before a late mistake and one that spent its steps repeating ineffective actions.

A natural way to improve reward quality is to describe trajectory success with rubrics. A rubric can, for example, distinguish whether the trajectory reached the requested target and whether it completed the required state change. This gives the reward designer a richer vocabulary than a scalar terminal outcome and a more inspectable structure than an opaque reward-model score. Recent rubric-generation and rubric-aware reward-model pipelines use structured criteria to guide evaluation (Anugraha et al., 2025; Liu et al., 2025; Xie et al., 2025). A two-stage configuration can ask one LLM to synthesize a rubric and another LLM or reward model to evaluate candidates against the generated criteria (Liu et al., 2025). Applying this pattern directly to agent RL, however, creates a practical bottleneck. The generated criteria still need an online judge call for every sampled rollout. This is burdensome for long-horizon RL tasks that require many environment interactions, and the

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.

cost is amplified in group-relative methods (Shao et al., 2024; Feng et al., 2025), where each update compares multiple trajectories from the same or related task context.

Another line of work improves credit assignment or makes reward propagation denser, including methods that compare rollouts, redistribute trajectory-level feedback, or exploit step structure in long-horizon episodes (Shao et al., 2024; Feng et al., 2025; Wang et al., 2025). These methods can improve how a reward is used, but they do not resolve a weak or ambiguous reward definition. A reward that gives credit to any pick or put action can teach generic manipulation rather than task completion. A target-aware reward that only detects late successful interactions can be too sparse for early exploration. Existing directions therefore leave complementary gaps: judge-based rewards are expensive to call at scale, while density and credit-assignment improvements still depend on reward quality.

HiRUB addresses this gap by keeping rubric generation outside the RL inner loop and turning the generated criteria into a structured trajectory reward. Before training, an LLM produces hierarchy-aware criteria from the task family, instruction, and related task information. We then map the rubric to a task schema and trajectory checks. The schema records what the task asks for, while the checks find evidence of those requirements in observations, actions, and environment feedback. Thus, the criteria are evaluated by the scorer itself rather than by an LLM judge during RL. The score is aggregated over the whole trajectory, so partial credit reflects task-relevant evidence accumulated across the rollout rather than isolated actions.

The reward organizes evidence through two complementary hierarchies. The progress hierarchy tracks whether an episode is moving toward completion, from discovery to interaction and verification. The semantic hierarchy refines this evidence from general validity signals to workflow-family and target-instance constraints. The reward therefore distinguishes whether an action is merely plausible from whether it advances the requested task instance.

The resulting trajectory-level reward is compatible with different policy-update procedures, which lets us evaluate HiRUB across multiple RL backbones. To examine whether this interface transfers across interaction paradigms, we study it in ALFWorld and WebShop (Shridhar et al., 2021; Yao et al., 2022). Across the compared settings, HiRUB achieves the strongest ALFWorld results at both model scales, raising GiGPO from 86.5% to 92.2% with 1.5B parameters and from 90.8% to 94.0% with 7B parameters. It also improves both WebShop score and success in the corresponding GiGPO runs. In the matched judge comparison, validation success increases from 90.1% to 92.2%, while total training time decreases from 31.1 to 19.6 hours. The main contributions are:

- We introduce a target-aware hierarchical rubric reward for long-horizon LLM agents, combining progress hierarchy and semantic hierarchy in a single online scoring function.
- We use generated criteria to define a low-cost, schema-grounded trajectory scorer. Online reward computation therefore avoids an LLM judge while preserving similar trajectory preferences.
- We evaluate HiRUB on ALFWorld and WebShop with multiple RL backbones and model scales, showing consistent gains over strong RL baselines.

2 Related Work

RL for LLM agents. RL is widely used in LLM post-training to convert scalar or preference-derived feedback into policy updates (Schulman et al., 2017; Christiano et al., 2017; Ouyang et al., 2022; Rafailov et al., 2023; Shao et al., 2024; DeepSeek-AI, 2025; Team et al., 2025; Zeng et al., 2025; Yu et al., 2025). GRPO estimates relative advantages within a sampled group, while GiGPO adapts group comparison to long-horizon agent rollouts (Shao et al., 2024; Feng et al., 2025). Recent agent RL systems study search-augmented reasoning, multi-turn agent training, and reusable training interfaces (Jin et al., 2025; Wang et al., 2025; Luo et al., 2025). These optimizers improve how reward values are normalized and used during learning, but the reward function remains a separate design choice. In long-horizon environments, the reward has to decide which observations, actions, and state changes indicate task progress. HiRUB is therefore complementary to these RL backbones: it supplies a structured trajectory reward that can be consumed by different update rules.

Reward design with models and code. Process supervision supplies signals at intermediate reasoning steps rather than only at the final answer (Wu et al., 2023; Lightman et al., 2024;

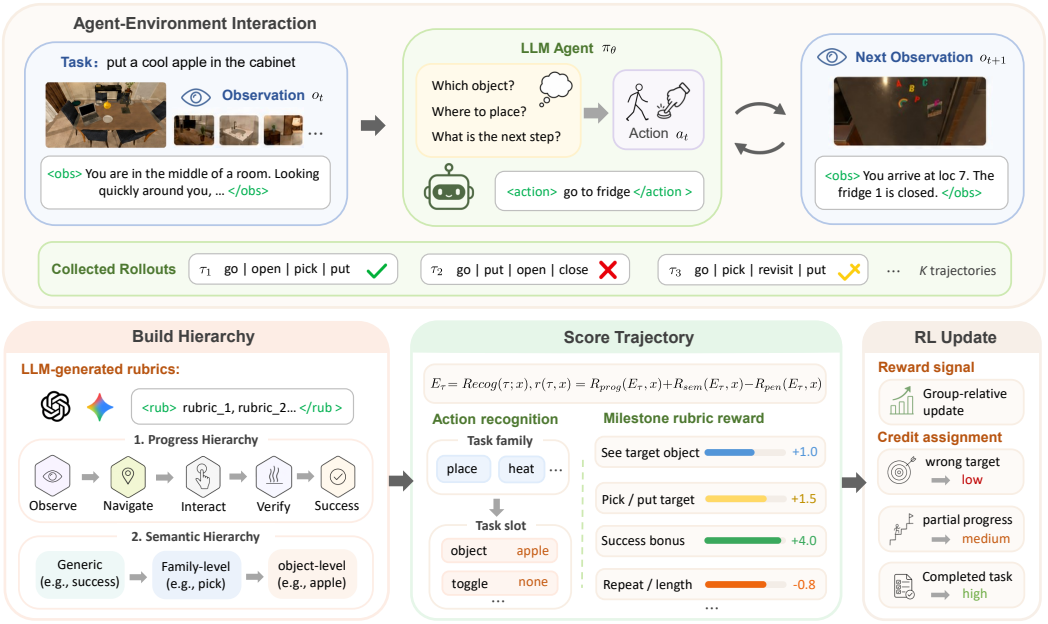


Figure 1: Overview of HiRUB. The agent interacts with the environment to collect rollouts. An LLM generates hierarchical rubrics. The generated rubrics provide criteria for environment-specific checks over complete rollouts. HiRUB's lightweight scorer aggregates the resulting evidence into the reward consumed by the RL optimizer, avoiding an online LLM judge.

Wang et al., 2023a; Setlur et al., 2025; Dai et al., 2024; Lai et al., 2024a; Luo et al., 2024), and language-model critics, hierarchical reward structures, and multi-objective designs translate such feedback into denser training rewards (Cao et al., 2024; Lai et al., 2024b; Wang et al., 2024a;b); recent variants study dynamically selected process rewards, criteria-based reward trees, lower-annotation and bounded designs, and turn-level rewards for long-horizon agents (Yin et al., 2025; Liang et al., 2025; Yuan et al., 2024a; Cui et al., 2025; Gao et al., 2024; Wei et al., 2025; Wang et al., 2025; Luo et al., 2025). A second family instead asks an LLM to write the reward itself: Eureka (Ma et al., 2024) and Text2Reward (Xie et al., 2023) generate reward code over simulator state, and Eureka improves that code through in-context evolutionary search with reward reflection, which requires training a policy for every candidate reward and access to the environment's state variables. Rubric-based reward models take a third route: OpenRubrics (Liu et al., 2025) generates criteria offline yet still evaluates each candidate with a trained reward model or a judge. HiRUB also generates criteria offline, but compiles them into deterministic checks grounded in the environment's task grammar, so reward computation invokes no model. As with process rewards, the score may use milestone evidence from intermediate states, but credit is assigned once at the trajectory level and tied to the task schema.

Rubrics, reward models, and LLM judges. LLM-as-a-judge evaluation is widely used when task quality is difficult to capture with a single exact-match label, with judge calibration, pairwise comparison, and benchmark design becoming active topics (Zheng et al., 2023; Liu et al., 2023; Wang et al., 2023b; Zhu et al., 2023; Dubois et al., 2024; Liu et al., 2024; Lambert et al., 2024; Bavaresco et al., 2024; Tan et al., 2024). Rubrics make judge behavior more explicit by decomposing quality into criteria. Recent work examines LLM-generated criteria, rubric-conditioned reward modeling, and LLM judging for instruction-following or reasoning outputs (Kim et al., 2023; 2024; Cui et al., 2023; Wang et al., 2024c; Jiang et al., 2023; Qin et al., 2024; Anugraha et al., 2025; Liu et al., 2025; Xie et al., 2025; Yuan et al., 2024b; Chen et al., 2025). The closest related approaches use generated criteria to structure a learned reward model or an LLM judge, and then score individual responses against those criteria (Gunjal et al., 2025; Xu et al., 2026). These works highlight the value of rubric structure, but typically keep the criteria inside model-based judging or response-level evaluation. HiRUB instead uses generated rubrics to define trajectory checks over complete rollouts, without an online judge.

3 PROBLEM SETUP

We consider episodic interactive tasks in which an LLM policy observes the environment, emits a textual action, and receives a new observation. A rollout is

$$\tau = \{(o_1, a_1), \ldots, (o_T, a_T)\}. \tag{1}$$

Here o_t and a_t are the observation and action at turn t, and T is the rollout length. Each task x has a natural-language instruction whose relevant structure can be summarized by a task schema,

$$z(x) = \big(f(x), s(x), c(x), \mathcal{M}(x)\big), \tag{2}$$

where x is the task, $z(x)$ is its schema, f is the workflow family, s stores target entities or attributes, c stores constraints and admissibility conditions, and $\mathcal{M}$ stores milestone events. ALFWorld fills these fields with objects, receptacles, and state changes, while WebShop uses search terms, product preferences, page actions, and purchase state. For example, for "put a cool apple in the cabinet," the schema records a placement family, apple and cabinet as the target entities, cooling as a task constraint, and discovery, navigation, pickup, placement, and verification as milestones.

4 METHOD

4.1 OVERVIEW

Figure 1 illustrates the HiRUB pipeline. Following the two-stage rubric-generation setup used in OpenRubrics (Liu et al., 2025), GPT-5.2 first generates hierarchical rubrics covering progress, workflow requirements, target or attribute constraints, and common failure modes. The prompt template is shown in Appendix E. The task schema is derived from the instruction and environment interface. HiRUB connects the rubric criteria to this schema and turns them into checks over observations, actions, and outcomes. The scorer uses the check results to form progress, semantic, success, and penalty terms for each complete rollout, so reward computation does not call an LLM judge. Appendix B gives a concrete ALFWorld example.

4.2 TASK SCHEMA AND RUBRIC GROUNDING

HiRUB treats the rubric as a specification and compiles it, once and offline, into environment-specific checks grounded in the environment's task grammar rather than in individual task instances. In ALFWorld, instructions are produced by a fixed library of goal templates and the environment exposes a fixed action space, so the compiled scorer inverts this grammar. It resolves the task schema $z(x)$ from the instruction by reading the workflow family and the target object, receptacle, and toggle, and it parses each action into a verb and its arguments. The family then selects the matching template and milestone set.

4.3 PROGRESS AND SEMANTIC HIERARCHIES

The progress hierarchy groups milestones by their role in task completion, including finding a relevant entity or page, navigating or searching toward it, matching a target or constraint, executing the required interaction, and verifying the resulting state. First-hit aggregation avoids increasing the score when the agent repeats the same evidence.

The semantic hierarchy organizes the same evidence from general conditions to workflow-family and instance-level constraints. General signals capture completion, validity, length, and repetition. Workflow-family signals check the required procedure, while instance-level signals check whether detected entities, attributes, and state transitions match the requested task. These hierarchies are structural groupings of criteria with different scopes and weights, rather than hard temporal prerequisites. The ablations test whether their levels provide complementary supervision.

4.4 TRAJECTORY SCORE

Let $E(\tau, x)$ denote the set of detected trajectory events after reading actions and observations. For each milestone m, we define a first-hit indicator

$$\phi_m(\tau, x) = \mathbb{I}\left[\exists t \le T : \; g_m(o_t, a_t, z(x)) = 1\right], \tag{3}$$

where m denotes a milestone, ϕ_m is its first-hit indicator, $\mathbb{I}[\cdot]$ is the indicator, and g_m checks for evidence of it. Because ϕ_m is binary, repeated actions or observations that satisfy the same criterion do not accumulate additional positive reward. Repeated consecutive actions are instead counted separately by the penalty term. HiRUB decomposes the trajectory score into progress, semantic, success, and penalty components:

$$R_{\mathrm{H}}(\tau, x) = \left[R_{\mathrm{prog}} + R_{\mathrm{sem}} + R_{\mathrm{succ}}(\tau, x) - R_{\mathrm{pen}}\right]_+ . \tag{4}$$

Here R_{H} is the HiRUB score, R_{prog} and R_{sem} are its progress and semantic components, R_{succ} is the terminal-success component, R_{pen} is the penalty, and $[v]_+ = \max(v, 0)$ clips a scalar v at zero. This lower bound prevents penalties from turning a trajectory with no useful evidence into a negative learning signal. The progress component aggregates milestone evidence related to task completion:

$$R_{\mathrm{prog}}(\tau, x) = \sum_{m \in \mathcal{M}_{\mathrm{prog}}} \alpha_m \phi_m(\tau, x). \tag{5}$$

Here $\mathcal{M}_{\mathrm{prog}} \subseteq \mathcal{M}(x)$ is the set of progress milestones and α_m is the coefficient for milestone m. It includes discovery, directed navigation or search, target matching, required interaction, and verification milestones. Equation 5 does not impose an ordering constraint between milestones: the hierarchy specifies their category and relative weighting. The semantic component activates workflow-family and instance-level constraints:

$$R_{\mathrm{sem}}(\tau, x) = \sum_{h \in \mathcal{H}(z(x))} \beta_h \psi_h(E, z(x)), \tag{6}$$

where $\mathcal{H}(z(x))$ is the set of semantic checks selected by the schema, $\psi_h(E, z(x))$ is the value of check h, and β_h is its coefficient. When a task requires multiple instances, event counts are capped by the requested counts:

$$\psi_{\mathrm{multi}}(\tau, x) = \sum_{u \in \mathcal{U}(x)} \beta_u \min(c_u^z, q_u). \tag{7}$$

Here $\mathcal{U}(x)$ denotes the required event types, c_u^z is the number of distinct matched event instances of type u under schema $z(x)$, q_u is the requested count, and β_u is the coefficient for that event type. For single-instance criteria, the implementation uses a first-hit indicator. For multi-instance criteria, it accepts only distinct target-aware events up to q_u, so repeating the same action to inflate the count does not add credit. The penalty component discourages repetitive or unnecessarily long behavior:

$$R_{\mathrm{pen}}(\tau) = \lambda_{\mathrm{rep}} \rho(c_{\mathrm{rep}}) + \lambda_{\mathrm{len}} \ell(T), \tag{8}$$

where λ_{rep} and λ_{len} weight repetition and length penalties, c_{rep} is the consecutive repeat count, and $\rho(\cdot)$ and $\ell(\cdot)$ are clipped penalty functions. Environment-specific coefficients and budgets are listed in Appendix H.

4.5 Applying HiRUB to ALFWorld and WebShop

The common interface is a schema-conditioned trajectory scorer, but the checks change across environments. In ALFWorld, positive evidence comes from target discovery, directed navigation, object interaction, required operations such as cleaning or cooling, verification, and terminal success:

$$\begin{aligned} R_{\mathrm{pos}}^{\mathrm{AW}} = {} & w_o \phi_{\mathrm{obj}} + w_r \phi_{\mathrm{rec}} + w_g \phi_{\mathrm{goto}} + w_p \phi_{\mathrm{pick}} \\ & + w_{\mathrm{put}} \phi_{\mathrm{put}} + w_c \psi_{\mathrm{cond}} + w_v \psi_{\mathrm{verify}} + w_s y. \end{aligned} \tag{9}$$

Here $R_{\mathrm{pos}}^{\mathrm{AW}}$ is the positive ALFWorld score, ϕ_{obj}, ϕ_{rec}, ϕ_{goto}, ϕ_{pick}, and ϕ_{put} are first-hit events for the corresponding discovery, navigation, pickup, and placement milestones, and ψ_{cond} and ψ_{verify} measure the required state change and its verification. The symbols w and $y \in \{0, 1\}$ denote their coefficients and terminal success.

In WebShop, the schema is expressed through the shopping goal. The semantic portion scores search, browsing, option matching, action validity, and purchase evidence:

$$R_{\mathrm{sem}}^{\mathrm{WS}} = w_q q_{\mathrm{search}} + w_b q_{\mathrm{browse}} + w_o q_{\mathrm{option}} + w_a q_{\mathrm{action}} + w_p q_{\mathrm{purchase}}. \tag{10}$$

Here $R_{\mathrm{sem}}^{\mathrm{WS}}$ is the WebShop semantic score, q_{search}, q_{browse}, q_{option}, q_{action}, and q_{purchase} measure task-relevant search, browsing, option matching, action validity, and purchase evidence, and w_q, w_b,

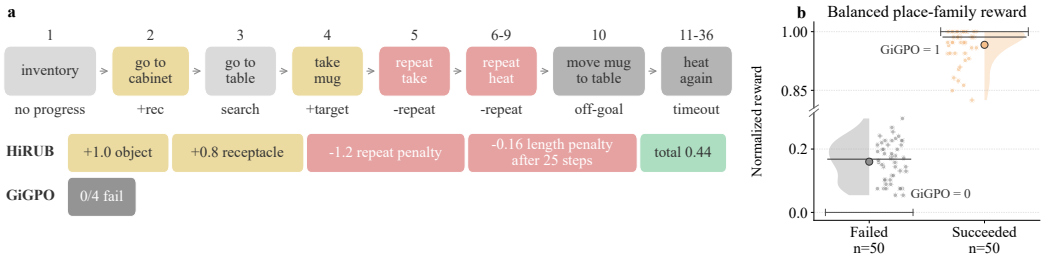


Figure 2: Place-family trajectory case study. Panel (a) decomposes one incomplete rollout into target-detection rewards, repeat and length penalties, and the resulting partial HiRUB score. The binary GiGPO reference remains zero. Panel (b) shows normalized HiRUB reward distributions for balanced failed and successful place trajectories, with binary GiGPO references for comparison.

w_o, w_a, and w_p are their coefficients. The sparse outcome and the environment-specific coefficients and budgets are listed in Appendix H.

Figure 2 gives a place-family case study. Panel (a) decomposes an incomplete rollout into target evidence, penalties, and final score, while panel (b) shows that HiRUB separates failed and successful place trajectories without collapsing all failed rollouts to the same value. Additional case-study notes are provided in Appendix D.

4.6 Using HiRUB with Group-Relative RL

HiRUB produces a scalar trajectory score, so it can be used with different RL optimizers. In group-relative methods, a rollout group $\mathcal{G} = \{\tau_i\}_{i=1}^{K}$ is scored by R_{H} and normalized within the group:

$$r_i = R_{\mathrm{H}}(\tau_i, x), \qquad \mu_{\mathcal{G}} = K^{-1}\sum_{j=1}^{K} r_j, \qquad A_i = \frac{r_i - \mu_{\mathcal{G}}}{\sigma_{\mathcal{G}} + \epsilon}. \tag{11}$$

Here K is the number of trajectories in group $\mathcal{G}$, τ_i is trajectory i, r_i is its HiRUB score, $\mu_{\mathcal{G}}$ and $\sigma_{\mathcal{G}}$ are the mean and standard deviation of scores in the group, $\epsilon > 0$ is a numerical stabilizer, and A_i is the normalized reward signal used by the group-relative update. HiRUB changes the reward signal supplied to the update, not the optimizer.

5 Experiments

5.1 Environments and Models

We evaluate primarily on ALFWorld, which contains six task families: pick-and-place, look-in-light, clean-and-place, heat-and-place, cool-and-place, and two-object placement. We report success rate by family and overall. We also evaluate on WebShop, reporting both dense score and success rate. We use Qwen2.5-Instruct backbones with 1.5B and 7B parameters, comparing prompting and RL baselines with their HiRUB variants. Implementation details are provided in Appendix I.

5.2 Main Results

Table 1 reports ALFWorld success rates together with WebShop score and task success. Prompting-only agents remain far behind RL-trained agents, showing that inference-time reasoning patterns alone are not enough for these long-horizon interaction tasks. At 1.5B, adding HiRUB improves both GRPO and GiGPO on the aligned ALFWorld and WebShop results, indicating that the reward design is not tied to one particular optimizer. The same trend appears at the larger model scale for the stronger GiGPO backbone, where the remaining errors are more concentrated in task-specific execution details. The ALFWorld task-family columns further show that HiRUB is especially useful when success depends on more than a single interaction, because state changes and repeated object manipulation require the reward to distinguish target-aware progress from merely valid actions.

Table 1: Main results on ALFWorld and WebShop. ALFWorld columns report success rates by task family, while WebShop columns report dense score and task success. RL entries with uncertainty report mean and standard deviation across three seeds. Best results are highlighted in bold, and second-best results are underlined.

Type	Method	ALFWorld							WebShop	
		Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Success
Closed-Source Model										
Prompting	GPT-4o	75.3	60.8	31.2	56.7	21.6	49.8	48.0	31.8	23.7
Prompting	Gemini-2.5-Pro	<u>92.8</u>	<u>63.3</u>	62.1	69.0	26.6	58.7	60.3	42.5	35.9
Qwen2.5-1.5B-Instruct										
Prompting	Qwen2.5	5.9	5.5	3.3	9.7	4.2	0.0	4.1	23.1	5.2
Prompting	ReAct	17.4	20.5	15.7	6.2	7.7	2.0	12.8	40.1	11.3
Prompting	Reflexion	35.3	22.2	21.7	13.6	19.4	3.7	21.8	55.8	21.9
RL Training	PPO (with critic)	$64.8_{\pm 3.5}$	$40.5_{\pm 6.9}$	$57.1_{\pm 4.9}$	$60.6_{\pm 6.6}$	$46.4_{\pm 4.0}$	$47.4_{\pm 1.9}$	$54.4_{\pm 3.1}$	$73.8_{\pm 3.0}$	$51.5_{\pm 2.9}$
RL Training	RLOO	$88.3_{\pm 3.0}$	$52.8_{\pm 8.6}$	$71.0_{\pm 5.9}$	$62.8_{\pm 8.7}$	$66.4_{\pm 5.5}$	$56.9_{\pm 4.7}$	$69.7_{\pm 2.5}$	$73.9_{\pm 5.6}$	$52.1_{\pm 6.7}$
RL Training	GRPO	$77.5_{\pm 1.7}$	$33.3_{\pm 19.1}$	$93.6_{\pm 2.2}$	$52.4_{\pm 4.8}$	$66.7_{\pm 7.6}$	$57.9_{\pm 13.9}$	$69.3_{\pm 2.7}$	$75.8_{\pm 3.5}$	$56.8_{\pm 3.8}$
RL Training	GRPO + **HiRUB**	$81.1_{\pm 6.4}$	$62.5_{\pm 12.5}$	$87.5_{\pm 7.8}$	$61.5_{\pm 10.0}$	$79.8_{\pm 13.0}$	$62.5_{\pm 8.8}$	$75.0_{\pm 4.7}$	$80.7_{\pm 0.9}$	$57.8_{\pm 4.7}$
RL Training	GiGPO	$92.0_{\pm 9.3}$	$62.5_{\pm 12.5}$	<u>94.9</u>$_{\pm 5.9}$	<u>78.4</u>$_{\pm 2.3}$	<u>86.7</u>$_{\pm 5.8}$	<u>82.4</u>$_{\pm 7.4}$	<u>86.5</u>$_{\pm 1.6}$	<u>83.1</u>$_{\pm 1.6}$	<u>65.0</u>$_{\pm 3.2}$
RL Training	GiGPO + **HiRUB**	**98.0**$_{\pm 1.7}$	**79.2**$_{\pm 7.2}$	**96.3**$_{\pm 3.7}$	**90.7**$_{\pm 8.5}$	**89.8**$_{\pm 0.3}$	**86.1**$_{\pm 8.1}$	**92.2**$_{\pm 1.4}$	**83.6**$_{\pm 0.6}$	**69.3**$_{\pm 2.8}$
Qwen2.5-7B-Instruct										
Prompting	Qwen2.5	33.4	21.6	19.3	6.9	2.8	3.2	14.8	26.4	7.8
Prompting	ReAct	48.5	35.4	34.3	13.2	18.2	17.6	31.2	46.2	19.5
Prompting	Reflexion	62.0	41.6	44.9	30.9	36.3	23.8	42.7	58.1	28.8
RL Training	PPO (with critic)	$92.3_{\pm 4.0}$	$64.0_{\pm 8.4}$	$92.5_{\pm 2.4}$	**89.5**$_{\pm 7.0}$	$80.3_{\pm 2.0}$	$68.8_{\pm 8.3}$	$80.4_{\pm 2.7}$	$81.4_{\pm 3.1}$	$68.7_{\pm 5.1}$
RL Training	RLOO	$87.6_{\pm 4.3}$	<u>78.2</u>$_{\pm 8.3}$	$87.3_{\pm 5.8}$	$81.3_{\pm 7.6}$	$71.9_{\pm 5.2}$	$48.9_{\pm 8.4}$	$75.5_{\pm 4.6}$	$80.3_{\pm 3.2}$	$65.7_{\pm 4.0}$
RL Training	GRPO	$90.8_{\pm 5.1}$	$66.1_{\pm 6.7}$	$89.3_{\pm 5.4}$	$74.7_{\pm 6.9}$	$72.5_{\pm 5.4}$	$64.7_{\pm 7.3}$	$77.6_{\pm 5.2}$	$79.3_{\pm 2.8}$	$66.1_{\pm 3.7}$
RL Training	GiGPO	<u>97.7</u>$_{\pm 1.6}$	**82.7**$_{\pm 7.9}$	<u>98.8</u>$_{\pm 1.6}$	$83.7_{\pm 7.2}$	<u>89.3</u>$_{\pm 8.2}$	<u>79.2</u>$_{\pm 6.6}$	<u>90.8</u>$_{\pm 1.3}$	<u>84.4</u>$_{\pm 2.9}$	<u>72.8</u>$_{\pm 3.2}$
RL Training	GiGPO + **HiRUB**	**100.0**$_{\pm 0.0}$	$75.0_{\pm 12.5}$	**100.0**$_{\pm 0.0}$	<u>87.2</u>$_{\pm 7.1}$	**91.7**$_{\pm 7.6}$	**93.7**$_{\pm 11.0}$	**94.0**$_{\pm 1.2}$	**87.5**$_{\pm 0.9}$	**73.7**$_{\pm 1.6}$

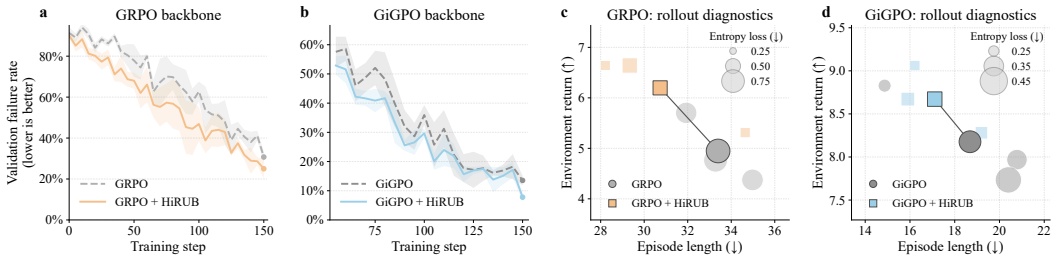


Figure 3: ALFWorld training and rollout diagnostics. Panels (a) and (b) show validation failure dynamics for GRPO and GiGPO backbones, while panels (c) and (d) show rollout episode length versus reward, with marker size encoding entropy loss. Curves, shaded regions, and error bars summarize three seeds where shown.

Figure 3 provides the training-dynamics view behind the final table. Panel (a) tracks validation failure for the GRPO backbone, while panel (b) repeats the comparison for GiGPO. In both cases, HiRUB lowers failure during training and the shaded regions summarize variation across three seeds. Panels (c) and (d) plot rollout episode length against rollout reward for GRPO and GiGPO, respectively. Marker shape identifies the baseline and HiRUB variant, vertical position measures reward, horizontal position measures episode length, and marker size represents policy entropy loss. HiRUB moves toward shorter, higher-reward rollouts, indicating changes in success, trajectory quality, and learning efficiency.

5.3 Rubric-Conditioned Judges and Reward Efficiency

For Figure 4, we use Gemini-2.5-Pro with GPT-5.2-generated rubrics, the best-average pair in Table C.1 in Appendix C. To test whether rubric structure can be retained without online judge inference, we compare the HiRUB slot scorer with the selected rubric-conditioned LLM judge and a direct judge. The rubric-conditioned judge is asked to reward successful completion, target-aware progress toward the correct object and receptacle, coherent action sequences, and to penalize

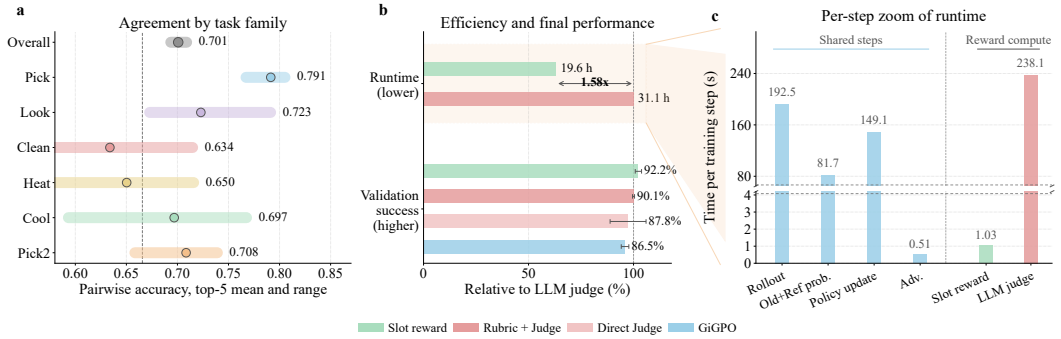


Figure 4: Agreement and efficiency of the slot-based HiRUB reward versus online LLM-judge rewards. Panel (a) reports top-5 pairwise agreement by task family over 200 trajectory samples. Markers show mean agreement and colored spans show the observed range. Panel (b) normalizes runtime to the rubric-conditioned LLM judge and compares validation success for slot reward, rubric + judge, direct judge, and GiGPO. Error bars on validation success summarize three seeds. Panel (c) decomposes mean non-evaluation training-step time, separating shared computations from reward computation. In panel (c), Rollout denotes environment rollout, Old+Ref prob. denotes old-policy and reference-model log-probability computation, and Adv. denotes advantage calculation. Prompt and timing details are provided in Appendix F.

repetition and unnecessarily long trajectories. The direct judge instead makes a holistic assessment without predefined criteria. The two prompt templates are given in Appendix E.

Figure 4(a) reports top-5 pairwise agreement by task family. The roughly 0.7-level agreement indicates similar but not identical preferences: the LLM judge can value global coherence and explanation-like trajectory quality, while the slot reward emphasizes explicit task-schema evidence. Panel (b) separates runtime from validation success. The slot scorer completes training in less time than the rubric-conditioned judge and attains the strongest validation success among the four compared reward or baseline settings. The rubric-conditioned judge also exceeds the direct judge, supporting the value of criteria even when both alternatives retain online model inference. Panel (c) shows that reward computation is small for the slot scorer but dominates the online-judge pipeline.

6 Ablation Study

6.1 Hierarchy and Reward Design

This study tests whether the gains depend on the organization and task awareness of the reward rather than on density alone. We compare HiRUB with variants that remove one hierarchy or replace the full design with simpler alternatives under the same evaluation protocol. Figure 5(a) shows that removing early milestones slows validation learning. The gap is largest during the early part of training, which suggests that discovery and interaction evidence helps the policy distinguish useful partial trajectories before successful completion becomes frequent. This comparison tests the role of progress structure, rather than merely asking whether adding more positive terms improves the learning curve. Panel (b) isolates the semantic hierarchy. Removing target-instance matching weakens performance on task families with distractors or specific state changes, indicating that recognizing an action type without checking its target and task role provides an incomplete learning signal. The contrast also suggests that task-conditioned evidence matters when several objects or state transitions share the same broad action description. Panels (c) and (d) then compare minimal-rubric, step-slot, and no-repeat variants. These alternatives test whether generic rubric density, local slot matching, or positive milestone terms alone can explain the gains. Their weaker or less balanced learning curves and task-family profiles indicate that the full reward benefits from combining progress evidence, target-aware semantics, successful completion, and controls against repeated or unnecessarily long behavior. In particular, the step-slot variant shows why recognizing isolated local events is not equivalent to evaluating whether a trajectory advances the requested task.

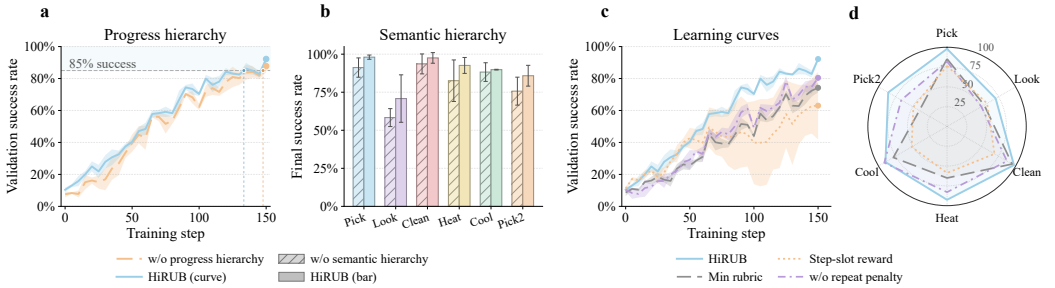


Figure 5: Ablation analysis for HiRUB. Panels (a) and (b) isolate the progress and semantic hierarchies. Panels (c) and (d) compare minimal-rubric, step-slot, and no-repeat reward variants. Shaded regions and error bars summarize three seeds where shown.

Table 2: Reward-coefficient ablation on ALFWorld and WebShop with the 1.5B backbone. Values are three-seed means and standard deviations. Detailed settings are in Appendix I.

Weight setting	**ALFWorld**							**WebShop**	
	Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Success
GiGPO + HiRUB	$98.0_{\pm 1.7}$	$79.2_{\pm 7.2}$	$96.3_{\pm 3.7}$	$90.7_{\pm 8.5}$	$89.8_{\pm 0.3}$	$86.1_{\pm 8.1}$	$92.2_{\pm 1.4}$	$83.6_{\pm 0.6}$	$69.3_{\pm 2.8}$
Milestone profiles									
Uniform	$94.1_{\pm 5.1}$	$58.3_{\pm 14.4}$	$92.3_{\pm 6.7}$	$84.1_{\pm 16.7}$	$78.3_{\pm 15.3}$	$63.2_{\pm 9.1}$	$82.8_{\pm 5.1}$	$80.5_{\pm 3.4}$	$62.4_{\pm 6.0}$
Strong hierarchy	$87.3_{\pm 6.1}$	$62.5_{\pm 12.5}$	$96.2_{\pm 0.0}$	$79.4_{\pm 2.7}$	$86.7_{\pm 7.6}$	$61.4_{\pm 8.0}$	$82.3_{\pm 1.2}$	$82.2_{\pm 3.7}$	$65.6_{\pm 4.3}$
Inverted hierarchy	$97.1_{\pm 2.9}$	$58.3_{\pm 7.2}$	$94.9_{\pm 2.2}$	$79.4_{\pm 19.2}$	$88.2_{\pm 7.5}$	$68.9_{\pm 0.9}$	$85.7_{\pm 2.5}$	$82.2_{\pm 0.1}$	$61.2_{\pm 5.2}$
Scale and perturbation									
$2.0\times$ shaping	$88.2_{\pm 7.8}$	$50.0_{\pm 12.5}$	$96.2_{\pm 3.8}$	$82.5_{\pm 14.5}$	$80.0_{\pm 5.0}$	$63.2_{\pm 10.5}$	$81.5_{\pm 7.9}$	$79.4_{\pm 2.0}$	$57.8_{\pm 2.1}$
Random perturbation	$97.1_{\pm 5.1}$	$66.7_{\pm 14.4}$	$97.4_{\pm 2.2}$	$93.7_{\pm 5.5}$	$93.2_{\pm 2.8}$	$79.4_{\pm 9.6}$	$91.4_{\pm 4.9}$	$83.5_{\pm 2.1}$	$64.6_{\pm 4.7}$

6.2 Reward-Coefficient Sensitivity

We next test sensitivity to shaping weights. Table 2 reports three-seed ALFWorld task-family success together with WebShop score and success. Uniform and hierarchy variants test milestone ordering, while scaling and perturbation variants test coefficient magnitude and balance. The comparison separates a structural effect from a favorable but highly specific numerical choice. The alternatives remain useful but trade performance across task families and WebShop success, whereas the default retains the strongest overall profile. Thus, coefficient scale and distribution affect learning dynamics without making the reward depend on one brittle setting.

The task-family columns show why aggregate performance is insufficient. Equalized or reordered weights weaken families that require several distinct milestones, with larger variation on Heat and Pick2. On WebShop, alternatives can remain competitive on dense score without preserving task success. The results therefore support using the rubric hierarchy to guide coefficient allocation rather than treating all detected events as interchangeable.

7 Conclusion

We presented HiRUB, a hierarchical rubric reward for long-horizon LLM agents. HiRUB uses an LLM before training to generate a hierarchical rubric, then grounds its criteria in a low-cost trajectory scorer for RL. The rubric's progress and semantic hierarchies allow partial credit to reflect task-relevant milestones, target constraints, completion, and bounded penalties. Experiments in embodied and web interaction environments show improvements across RL backbones and model scales. The LLM-judge comparison and ablations further suggest that reward quality is complementary to optimizer design. Future work can make the rubric-to-scorer construction more automatic for less templated environments.

8 AI DISCLOSURE

We used generative AI tools for feedback on research methodology and experiments, implementation of methods including rubric generation, translation, interpretation of results, suggestions for experimental parameters, software-code creation and editing, brainstorming, sourcing and searching for information, identification of relevant literature, reference formatting, suggestions for the paper structure, title, and keywords, drafting parts of the paper, and editing the paper to improve readability. We also used generative AI tools to help formulate parts of the rubric prompts and to inspect intermediate experimental outputs. All technical decisions, reported results, citations, code changes, and manuscript revisions were reviewed by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, provide essential ingredients for proving mathematical claims, assist in writing proofs, propose or refine hypotheses, clean or reformat datasets, support qualitative or thematic data analysis, formulate survey or interview questions, create or modify scientific figures or images, create other research artifacts, transcribe recordings, summarize or analyze existing literature, discover research topics or identify gaps, or create artifacts beyond the software and manuscript work described above. We also did not use generative AI tools to provide a complete automated analysis in place of author interpretation. The authors reviewed AI-assisted code for correctness, checked the generated rubric content and prompts against the implemented reward logic, verified experimental settings and results, and manually checked references and claims.

All other required disclosure categories not explicitly identified above are not applicable to this work.

9 REPRODUCIBILITY STATEMENT

We provide the implementation code and supporting materials in the Supplementary Material. Additional implementation details, including model configurations, reward coefficients, prompt templates, training settings, and evaluation procedures, are provided in the appendices.

REFERENCES

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

David Anugraha, Zilu Tang, Lester James Validad Miranda, Hanyang Zhao, MohammadRifqi Farhansyah, Garry Kuwanto, Derry Wijaya, and Genta Indra Winata. R3: Robust rubric-agnostic reward models. *ArXiv*, abs/2505.13388, 2025. URL https://api.semanticscholar.org/CorpusID:278768432.

Anna Bavaresco, Raffaella Bernardi, Leonardo Bertolazzi, Desmond Elliott, Raquel Fernández, Albert Gatt, E. Ghaleb, Mario Giulianelli, Michael Hanna, Alexander Koller, Andr'e F. T. Martins, Philipp Mondorf, Vera Neplenbroek, Sandro Pezzelle, Barbara Plank, David Schlangen, Alessandro Suglia, Aditya K Surikuchi, Ece Takmaz, and Alberto Testoni. Llms instead of human judges? a large scale empirical study across 20 nlp evaluation tasks. In *Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://api.semanticscholar.org/CorpusID:270738074.

Meng Cao, Lei Shu, Lei Yu, Yun Zhu, Nevan Wichers, Yin-Xiao Liu, and Lei Meng. Enhancing reinforcement learning with dense rewards from language model critic. In *Conference on Empirical Methods in Natural Language Processing*, 2024. URL https://api.semanticscholar.org/CorpusID:273901456.

Nuo Chen, Zhiyuan Hu, Qingyun Zou, Jiaying Wu, Qian Wang, Bryan Hooi, and Bingsheng He. Judgelrm: Large reasoning models as a judge. *ArXiv*, abs/2504.00050, 2025. URL https://api.semanticscholar.org/CorpusID:277467872.

Paul Francis Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems*, 2017. URL https://api.semanticscholar.org/CorpusID:4787508.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guo Tong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with scaled ai feedback. In *International Conference on Machine Learning*, 2023. URL https://api.semanticscholar.org/CorpusID:283290883.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qi-Xin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *ArXiv*, abs/2502.01456, 2025. URL https://api.semanticscholar.org/CorpusID:276107672.

Ning Dai, Zheng Wu, Renjie Zheng, Ziyun Wei, Wenlei Shi, Xing Jin, Guan-Lin Liu, Chen Dun, Liang Huang, and Lin Yan. Process supervision-guided policy optimization for code generation. *ArXiv*, abs/2410.17621, 2024. URL https://api.semanticscholar.org/CorpusID:273532499.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv*, abs/2501.12948, 2025. URL https://api.semanticscholar.org/CorpusID:284488789.

Yann Dubois, Bal'azs Galambosi, Percy Liang, and Tatsunori Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *ArXiv*, abs/2404.04475, 2024. URL https://api.semanticscholar.org/CorpusID:269004605.

Lang Feng, Zhenghai Xue, Ting-Cong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *ArXiv*, abs/2505.10978, 2025. URL https://api.semanticscholar.org/CorpusID:278715074.

Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weiling Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. On designing effective rl reward at training time for llm reasoning. *ArXiv*, abs/2410.15115, 2024. URL https://api.semanticscholar.org/CorpusID:273502123.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Bing Liu, and Sean M. Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *ArXiv*, abs/2507.17746, 2025. URL https://api.semanticscholar.org/CorpusID:280101783.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. Followbench: A multi-level fine-grained constraints following benchmark for large language models. In *Annual Meeting of the Association for Computational Linguistics*, 2023. URL https://api.semanticscholar.org/CorpusID:264802282.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *ArXiv*, abs/2503.09516, 2025. URL https://api.semanticscholar.org/CorpusID:276937772.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, S. Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. *ArXiv*, abs/2310.08491, 2023. URL https://api.semanticscholar.org/CorpusID:265675839.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. *ArXiv*, abs/2405.01535, 2024. URL https://api.semanticscholar.org/CorpusID:269502688.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xi Peng, and Jiaya Jia. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *ArXiv*, abs/2406.18629, 2024a. URL https://api.semanticscholar.org/CorpusID:270764693.

Yuhang Lai, Siyuan Wang, Shujun Liu, Xuanjing Huang, and Zhongyu Wei. Alarm: Align language models via hierarchical rewards modeling. In *Annual Meeting of the Association for Computational Linguistics*, 2024b. URL https://api.semanticscholar.org/CorpusID:268357678.

Nathan Lambert, Valentina Pyatkin, Jacob Daniel Morrison, Lester James Validad Miranda, Bill Yuchen Lin, Khyathi Raghavi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hanna Hajishirzi. Rewardbench: Evaluating reward models for language modeling. *ArXiv*, abs/2403.13787, 2024. URL https://api.semanticscholar.org/CorpusID:268537409.

Xiaobo Liang, Haoke Zhang, Juntao Li, Kehai Chen, Qiaoming Zhu, and Min Zhang. Generative reward modeling via synthetic criteria preference learning. In *Annual Meeting of the Association for Computational Linguistics*, 2025. URL https://api.semanticscholar.org/CorpusID:280016851.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *International Conference on Learning Representations*, 2024. URL https://api.semanticscholar.org/CorpusID:258987659.

Tianci Liu, Ran Xu, Tony Yu, Ilgee Hong, Carl Yang, Tuo Zhao, and Haoyu Wang. Openrubrics: Towards scalable synthetic rubric generation for reward modeling and llm alignment. In *Annual Meeting of the Association for Computational Linguistics*, 2025. URL https://api.semanticscholar.org/CorpusID:281951535.

Yan-Tao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. Rm-bench: Benchmarking reward models of language models with subtlety and style. *ArXiv*, abs/2410.16184, 2024. URL https://api.semanticscholar.org/CorpusID:273507377.

Yang Liu, Dan Iter, Yichong Xu, Shuo Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://api.semanticscholar.org/CorpusID:257804696.

Liangchen Luo, Yin-Xiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision. *ArXiv*, abs/2406.06592, 2024. URL https://api.semanticscholar.org/CorpusID:270379625.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Si-Yun Zhao, Dongsheng Li, Lu-Na K. Qiu, and Yuqing Yang. Agent lightning: Train any ai agents with reinforcement learning. *ArXiv*, abs/2508.03680, 2025. URL https://api.semanticscholar.org/CorpusID:280526917.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *International Conference on Learning Representations*, 2024. URL https://api.semanticscholar.org/CorpusID:264306288.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Daniel Morrison, Noah A. Smith, Hanna Hajishirzi, and Nathan Lambert. Rewardbench 2: Advancing reward model evaluation. *ArXiv*, abs/2506.01937, 2025. URL https://api.semanticscholar.org/CorpusID:279119102.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 2022. URL https://api.semanticscholar.org/CorpusID:246426909.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. Infobench: Evaluating instruction following ability in large language models. In *Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://api.semanticscholar.org/CorpusID:266844311.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 2023. URL https://api.semanticscholar.org/CorpusID:258959321.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017. URL https://api.semanticscholar.org/CorpusID:28695052.

Amrith Rajagopal Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *International Conference on Learning Representations*, 2025. URL https://api.semanticscholar.org/CorpusID:273233462.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Jun-Mei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv*, abs/2402.03300, 2024. URL https://api.semanticscholar.org/CorpusID:267412607.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.03768.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William YM Tang, Alejandro Cuadron, Chenguang Wang, Raluca A. Popa, and Ion Stoica. Judgebench: A benchmark for evaluating llm-based judges. *ArXiv*, abs/2410.12784, 2024. URL https://api.semanticscholar.org/CorpusID:273374769.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Feng Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Haochen Ding, Hao-Xing Hu, Haoming Yang, Hao Zhang, Haotian Yao, Hao-Dong Zhao, Haoyu Lu, Hao-Ze Li, Hao Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianling Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Li-Na Shi, Li-Tao Ye, Long Yu, Meng xiao Dong, Neo Y. Zhang, Ning Ma, Qi Pan, Qucheng Gong, Shaowei Liu, Shen Ma, Shu-Yan Wei, Sihan Cao, Si-Da Huang, Tao Jiang, Wei-Wei Gao, Weiming Xiong, Weiran He, Weixiao Huang, Wenhao Wu, Wen He, Xian sen Wei, Xian-Xian Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xue biao Pan, Y. Charles, Yang Li, Yan-Ling Hu, Yangyang Liu, Yanru Chen, Ye-Jia Wang, Yi-Bo Liu, Yidao Qin, Yifeng Liu, Yingbo Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhengxin Zhu, Zheng Zhang, Zhe-Xu Wang, Zhilin Yang, Zhiqi Huang, Zi-Hao Huang, Ziya Xu, and Zong-Han Yang. Kimi k1.5: Scaling reinforcement learning with llms. *ArXiv*, abs/2501.12599, 2025. URL https://api.semanticscholar.org/CorpusID:275789974.

Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. In *Conference on Empirical Methods in Natural Language Processing*, 2024a. URL https://api.semanticscholar.org/CorpusID:270562658.

Peifeng Wang, Austin Xu, Yilun Zhou, Caiming Xiong, and Shafiq Joty. Direct judgement preference optimization. *ArXiv*, abs/2409.14664, 2024b. URL https://api.semanticscholar.org/CorpusID:272827021.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y.Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *ArXiv*, abs/2312.08935, 2023a. URL https://api.semanticscholar.org/CorpusID:266209760.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xingxu Xie, Wei Ye, Shikun Zhang, and Yue Zhang. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. *ArXiv*, abs/2306.05087, 2023b. URL https://api.semanticscholar.org/CorpusID:259108266.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. Helpsteer2: Open-source dataset for training top-performing reward models. *ArXiv*, abs/2406.08673, 2024c. URL https://api.semanticscholar.org/CorpusID:270440126.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Monica S. Lam, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Fei-Fei Li, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *ArXiv*, abs/2504.20073, 2025. URL https://api.semanticscholar.org/CorpusID:278170861.

Quan Wei, Siliang Zeng, Chenliang Li, Zhong-Ruo Wang, William Brown, Oana Frunza, Wei Deng, Anderson Schneider, Yuriy Nevmyvaka, Yang Zhao, Alfredo García, and Mingyi Hong. Reinforcing multi-turn reasoning in llm agents via fine-grained reward structure and credit assignment. 2025. URL https://api.semanticscholar.org/CorpusID:282304438.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A. Smith, Mari Ostendorf, and Hannaneh Hajishirzi. Fine-grained human feedback gives better rewards for language model training. *Advances in Neural Information Processing Systems*, 2023. URL https://api.semanticscholar.org/CorpusID:259064099.

Lipeng Xie, Sen Huang, Zhuo Zhang, Anni Zou, Yun-Peng Zhai, Dingchao Ren, Ke-Zun Zhang, Hao-Yuan Hu, Bo-Yin Liu, Hao-Ran Chen, Zhaoyang Liu, and Bolin Ding. Auto-rubric: Learning from implicit weights to explicit rubrics for reward modeling. 2025. URL https://api.semanticscholar.org/CorpusID:285303608.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Reward shaping with language models for reinforcement learning. In *International Conference on Learning Representations*, 2023. URL https://api.semanticscholar.org/CorpusID:270063618.

Ran Xu, Tianci Liu, Zi-Han Dong, T. You, Ilgee Hong, Carl Yang, Linjun Zhang, Tao Zhao, and Haoyu Wang. Alternating reinforcement learning for rubric-based reward modeling in non-verifiable llm post-training. *ArXiv*, abs/2602.01511, 2026. URL https://api.semanticscholar.org/CorpusID:285269626.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 2022. URL https://api.semanticscholar.org/CorpusID:250264533.

Zhangyue Yin, Qiu-Shi Sun, Zhiyuan Zeng, Qinyuan Cheng, Xipeng Qiu, and Xuanjing Huang. Dynamic and generalizable process reward modeling. *ArXiv*, abs/2507.17849, 2025. URL https://api.semanticscholar.org/CorpusID:280017924.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tian-Tian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Honglin Yu, Weinan Dai, Yuxuan Song, Xiang Wei, Haodong Zhou, Jingjing Liu, Wei Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yong-Xu Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. *ArXiv*, abs/2503.14476, 2025. URL https://api.semanticscholar.org/CorpusID:277104124.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. *ArXiv*, abs/2412.01981, 2024a. URL https://api.semanticscholar.org/CorpusID:274445748.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason E. Weston. Self-rewarding language models. *International Conference on Machine Learning*, 2024b. URL https://api.semanticscholar.org/CorpusID:267035293.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *ArXiv*, abs/2503.18892, 2025. URL https://api.semanticscholar.org/CorpusID:277940848.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Da-Cheng Li, Eric P. Xing, Haotong Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 2023. URL https://api.semanticscholar.org/CorpusID:259129398.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *ArXiv*, abs/2311.07911, 2023. URL https://api.semanticscholar.org/CorpusID:265157752.

Liang-Hui Zhu, Xinggang Wang, and Xin-Long Wang. Judgelm: Fine-tuned large language models are scalable judges. *ArXiv*, abs/2310.17631, 2023. URL https://api.semanticscholar.org/CorpusID:264490588.

APPENDIX

APPENDIX A: ENVIRONMENT SCHEMATICS

Figures A.1 and A.2 summarize the two interaction settings. ALFWorld requires target-aware household manipulation through textual observations and actions, whereas WebShop requires search, product-attribute matching, and purchase decisions. Together, they expose distinct task schemas while retaining the same trajectory-level reward interface.

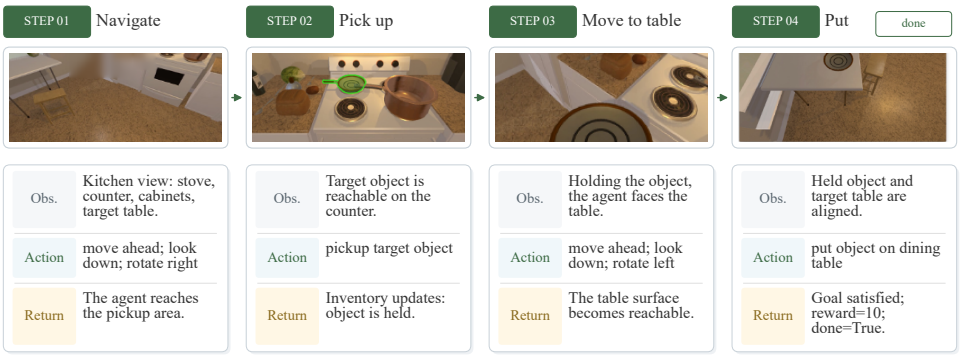


Figure A.1: ALFWorld environment schematic. The instruction defines target slots and task-family requirements, while the agent interacts through textual observations and actions until success or the step limit.

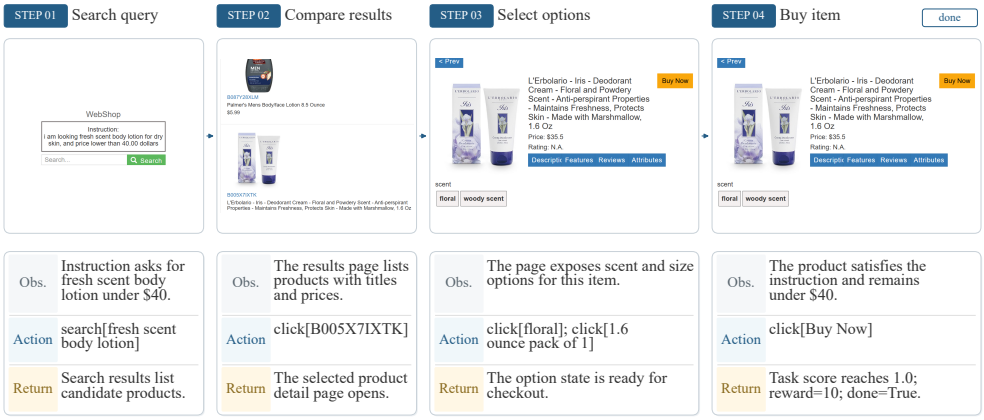


Figure A.2: WebShop environment schematic. The instruction specifies purchase constraints, and the agent completes the task by navigating product pages, matching attributes, selecting options, and issuing the final purchase action.

APPENDIX B: EXECUTABLE CHECK EXAMPLE

Consider the ALFWorld instruction "put a cool apple in the cabinet." The rubric generator supplies hierarchical criteria, while the task schema records the target object, receptacle, required cooling state, and placement workflow. The scorer evaluates the following checks on the complete rollout. Each row maps a rubric criterion to the schema field it binds and to the check that operationalizes it. Each positive check is registered once for a single-instance criterion, so a repeated action that does not create a new target-aware event leaves the corresponding positive term unchanged.

Rubric criterion	Schema field	Check	Evidence required from the rollout	Aggregation
The requested entity is located	s: object	Object discovery	An observation identifies the requested apple	First hit
The destination is located	s: receptacle	Receptacle discovery	An observation identifies the requested cabinet	First hit
The agent moves toward the target	f: place	Directed progress	A navigation action moves toward the relevant object or cabinet	First hit
The requested object is acquired	s: object	Correct pickup	The action picks up the requested apple	First hit
The cooling requirement is satisfied	c: cool	Cooling	The apple is exposed to the required cooling operation or state change	First hit
The object reaches the destination	s: receptacle	Correct placement	The requested apple is placed in the requested cabinet	First hit
The task completes	y	Terminal success	The environment reports task completion	Binary success bonus
Repetition is bounded	—	Repeat penalty	Consecutive duplicate actions	0.08 per repeat, cap 1.2
Length is bounded	—	Length penalty	Rollout length	0.015 per step past 25, cap 0.75

APPENDIX C: RUBRIC-CONDITIONED JUDGE BENCHMARKS

We use reward-model and judge benchmarks to compare candidate judge models under rubric-based evaluation. The benchmark suite includes RM-Bench chat comparisons (Liu et al., 2024), RewardBench2 precise-instruction and focus slices (Malik et al., 2025), and IF Evaluation Bench subsets for PPE-style and general instruction-following constraints (Zhou et al., 2023). Earlier RewardBench chat and chat-hard slices are excluded from the reported averages, so the average reflects only the columns shown in Table C.1. These benchmarks are not agent environments, but they characterize rubric-conditioned judging behavior and motivate the offline rubric-generation setup used by HiRUB. The two-stage rubric-match prompt is provided in Appendix E.

Table C.1: Reward-judge benchmark results comparing direct and rubric-conditioned evaluation. The listed model is the judge or reward model being evaluated. In the Gemini-2.5-Pro rubric-conditioned setting, GPT-5.2 generates the rubrics. In the Gemini-2.5-FlashLite rubric-conditioned setting, GPT-4.1-Mini generates the rubrics. Some baseline results follow the OpenRubrics setting (Liu et al., 2025). The average is computed over the five displayed benchmark slices. Best results are highlighted in bold, and second-best results are underlined.

Model	Setting	**RM-Bench**	**RewardBench2**		**IF Evaluation Bench**		**Avg.**
		Chat	Precise IF	Focus	PPE-IFEval	IFBench	
JudgeLRM-7B	Direct Judge	55.4	9.4	29.1	46.0	47.5	37.5
Qwen-3-8B	Direct Judge	63.9	8.1	44.0	57.0	59.5	46.5
Gemini-2.5-FlashLite	Direct Judge	67.2	13.2	63.4	59.2	60.4	52.7
Gemini-2.5-FlashLite	Rubric + Judge	67.9	42.5	79.6	61.0	66.2	63.4
Rubric-RM-8B	Rubric + Judge	65.7	34.4	82.2	67.0	65.4	62.9
Gemini-2.5-Pro	Direct Judge	**75.3**	**70.4**	<u>86.8</u>	<u>67.3</u>	<u>68.7</u>	<u>73.7</u>
Gemini-2.5-Pro	Rubric + Judge	<u>73.0</u>	<u>69.6</u>	**90.2**	**74.0**	**69.4**	**75.2**

Table C.1 separates a direct-judge setting, in which the listed model makes an unconstrained judgment, from a two-stage setting, in which one model first generates rubric criteria and the listed model then evaluates the candidate responses against them. The Gemini-2.5-Pro comparison illustrates the value of this separation: direct judging is stronger on RM-Bench and the precise-instruction slice, whereas rubric conditioning improves the focus and instruction-following slices and yields the best reported average. In this rubric-conditioned row, GPT-5.2 is the generator used to produce the

criteria. We use the same model to generate the task rubrics that are subsequently grounded into the HiRUB trajectory scorer. Gemini-2.5-FlashLite follows the same two-stage design with GPT-4.1-Mini as the rubric generator. Several baseline rows are drawn from or aligned with the OpenRubrics evaluation setting (Liu et al., 2025). Appendix D gives PPE-IFEval and ALFWorld examples of how rubric matching changes the evidence considered in a preference decision. Appendix F further evaluates repeated direct and rubric-guided judgments of fixed rollouts. It measures reward-signal stability rather than end-to-end task success.

APPENDIX D: CASE STUDIES

Trajectory decomposition. Figure 2 is intended to show how a trajectory can receive useful partial credit while still being clearly separated from a successful completion. The rollout observes or approaches task-relevant entities, so the reward records early milestone evidence rather than treating the failed episode as uninformative. Credit is attached to the object and receptacle requested by the instruction, and manipulation actions receive credit when they are consistent with the required workflow and task schema. Repetition and excessive length reduce the final score, while the terminal success component remains zero for the incomplete trajectory. The distribution plot in Figure 2(b) follows the same reading: failed trajectories can obtain nonzero reward when they collect correct evidence, but successful trajectories receive the completion component and are shifted upward.

PPE-IFEval rubric-match example. The following case study corresponds to the reward-judge benchmark setting in Table C.1. It illustrates why the rubric-generation model and the judging model are separated in the two-stage evaluation.

Field	Content
Prompt	It is extremely important that you follow all instructions exactly: Write an essay about how aluminium cans are used in food storage. Do not forget to include the keywords "waste," "material," and "meal." Have more than 30 sentences in your response.
Resp A (snippet)	"The use of aluminium cans in food storage has become a ubiquitous practice in the modern era. The versatility and durability of these cans make them an ideal material for packaging a wide range of food products. ... The use of aluminium cans also has a significant impact on waste reduction. ... In the context of a meal, the use of aluminium cans can have a significant impact on the overall dining experience. ..." The response contains more than 30 sentences but is repetitive and contains some questionable factual claims.
Resp B (snippet)	"Aluminum cans are widely used in food storage, thanks to their exceptional properties ... Another significant benefit of aluminum cans is their ability to reduce waste ... In terms of material costs ... when it comes to meal preparation ..." The response is more concise and polished but contains fewer than 30 sentences.
Ground Truth	Resp A.
Gemini-2.5-Pro (Direct)	"Assistant A provided a well-written, factually accurate, and properly structured essay ... Assistant B's response contains significant factual errors ... and its structure is poor. For these reasons, Assistant A is clearly superior." In the original A/B ordering, Assistant A corresponds to Resp B here. Gemini prioritizes factuality and writing quality but overlooks the explicit requirement of more than 30 sentences. (Choose B)
GPT-5.2-Generated Rubric	1. The response must be an essay focused on how aluminium cans are used in food storage. 2. The response must contain more than 30 sentences. 3. The response must include the keyword "waste" at least once. 4. The response must include the keyword "material" at least once. 5. The response must include the keyword "meal" at least once.
Gemini-2.5-Pro Rubric Judge	"Response A followed the length constraint of having more than 30 sentences, whereas Response B failed this instruction. Both responses address the requested topic and include the required keywords. The explicit sentence-count requirement determines the result." (Choose A)

ALFWorld successful trajectory example. The following case study shows a successful ALFWorld placement trajectory for the task *Put a toiletpaper in toiletpaperhanger*. The result is `success=true`. The rollout has `episode_reward=10.0`, contains six steps, and all actions are legal.

Environment (Step 1)

Task: Put a toiletpaper in toiletpaperhanger.
You are in the middle of a room. You see a bathtubbasin, two cabinets, a countertop, a garbagecan, a sinkbasin, a toilet, a toiletpaperhanger, and a towelholder.

Agent (Step 1)

<think>
The goal is to put a toiletpaper in the toiletpaperhanger.
The only place where toiletpapers can be found is on the toilet or countertops.
I need to find a toiletpaper and then move it to the toiletpaperhanger.
</think><action>go to toilet 1</action>

Environment (Step 2)

You arrive at toilet 1. On the toilet 1, you see a cloth 1.

Agent (Step 2)

<think>
The goal is to put a toiletpaper in the toiletpaperhanger.
I need to find a toiletpaper first before placing it on the hanger.
</think><action>go to cabinet 1</action>

Environment (Step 3)

You arrive at cabinet 1. The cabinet 1 is closed.

Agent (Step 3)

<think>
The current situation is that I am at cabinet 1, which is closed.
I need to open the cabinet to find a toiletpaper.
</think><action>open cabinet 1</action>

Environment (Step 4)

You open cabinet 1. In it, you see a candle, a cloth, a soapbar, a spraybottle, and a toiletpaper 1.

Agent (Step 4)

<think>
The goal is to put a toiletpaper in the toiletpaperhanger.
I have now found a toiletpaper in cabinet 1, so I should pick it up.
</think><action>take toiletpaper 1 from cabinet 1</action>

Environment (Step 5)

You pick up the toiletpaper 1 from cabinet 1.

Agent (Step 5)

<think>
My goal is to put the toiletpaper in toiletpaperhanger 1.
I have the toiletpaper in my possession, so I need to go to the toiletpaperhanger.
</think><action>go to toiletpaperhanger 1</action>

Environment (Step 6)

You arrive at toiletpaperhanger 1. On the toiletpaperhanger 1, you see nothing.

Agent (Step 6)

<think>
I have picked up toiletpaper 1 and arrived at toiletpaperhanger 1.
I should now move the toiletpaper to the toiletpaperhanger to complete the task.
</think><action>move toiletpaper 1 to toiletpaperhanger 1</action>

Environment (Final)

The action succeeds. `env_reward=1.0`, `won=true`, `episode_reward=10.0`.

APPENDIX E: PROMPT TEMPLATES

This section gives the reconstructed prompt template used to ask an LLM to generate task-conditioned rubric criteria for HiRUB, followed by the two-stage rubric-match prompt used for the model-selection results in Table C.1. In the two-stage benchmark prompt, rubric generation and judging can be performed by different models, as described in Table C.1. The final two boxes show the ALFWorld online judge prompts used for the agreement and efficiency comparison in Figure 4. Prompt Box A requests suggested coefficients together with checkable milestones and trajectory checks. The coefficients reported in Appendix H are the LLM's original suggestions and were not manually adjusted.

Prompt Box A: LLM-generated HiRUB rubrics.

System prompt.
You design hierarchical reward rubrics for long-horizon large language model agents in interactive environments. Given an environment description, task family, action format, observation format, and success condition, produce a compact rubric that can later guide the construction of trajectory checks. The rubric should include progress milestones, semantic criteria, target or attribute constraints, common failure modes, bounded penalties, and a suggested numeric coefficient for each criterion. Each criterion should be atomic, checkable from trajectory text or environment metadata, and should avoid rewarding local actions that are unrelated to the task instance. Return a JSON object with keys `progress_hierarchy`, `semantic_hierarchy`, `trajectory_checks`, `penalties`, `coefficients`, and `aggregation_notes`. Return the suggested coefficient for each criterion as part of the JSON object. Wrap the JSON object inside `<rubrics>` and `</rubrics>`.

User prompt.
Environment: {`environment`}
Task family or workflow: {`task_family`}
Task instruction: {`task`}
Observation format: {`observation_format`}
Action format and admissible actions: {`action_format`}
Success condition: {`success_condition`}
Known distractors or failure modes: {`failure_modes`}
Output only the tagged JSON rubric.

The rubric-match prompt has two stages: first a rubric-generation model creates a short rubric list from the benchmark user prompt, then a separate judge or reward model evaluates whether each candidate response satisfies each rubric item.

Prompt Box B: Two-stage rubric-match prompt used for Table C.1.

Stage 1 system prompt: rubric generation.
You generate evaluation rubrics. Given a USER PROMPT, create a short rubric list tailored to it. Return a JSON object with exactly one key: `rubrics`, a list of 4 to 8 strings. Each rubric should be atomic, checkable, non-overlapping, and phrased as a requirement. Wrap the JSON object inside `<rubrics>` and `</rubrics>`. Do not output any other text and do not repeat the instructions.

Stage 1 user prompt.
USER PROMPT:
{`prompt`}
Output only the tagged JSON object.

Stage 2 system prompt: A/B matching.
You evaluate responses using a rubric list. For each rubric item i, decide if RESPONSE A satisfies it, using 1 for satisfied and 0 otherwise, and do the same for RESPONSE B. Be conservative: mark 1 only if the rubric item is clearly satisfied. Return a JSON object with exactly these keys: `matches_a`, `matches_b`, `tie_break_winner`, and `tie_break_reason`. The two match lists should have length exactly equal to the rubric length. If total scores are equal, set `tie_break_winner` to A or B using overall instruction-following and correctness. Otherwise still output a reasonable value. Wrap the JSON object inside `<rubric_match>` and `</rubric_match>`. Do not output any other text and do not repeat the instructions.

Stage 2 user prompt.
USER PROMPT:
{`prompt`}
RUBRICS:
1. {`rubric_1`}
2. {`rubric_2`}
...
RESPONSE A:
{`response_a`}
RESPONSE B:
{`response_b`}
Output only the tagged JSON object.

The following ALFWorld prompts are used to compare the HiRUB reward with online LLM-judge rewards. Prompt Box C uses an explicit rubric-style scoring instruction. Prompt Box D asks for a direct holistic judgment without a predefined rubric.

Prompt Box C: ALFWorld rubric-conditioned LLM judge.

System prompt.
You are a strict ALFWorld trajectory judge. Follow the requested rubric and output only the requested tags.

User prompt.
Evaluate the following ALFWorld rollout with a rubric.

Scoring principle:
- Reward successful completion the most.
- Reward target-aware progress toward the correct object, receptacle, and required condition.
- Penalize repetitive actions, ineffective searching, and unnecessarily long trajectories.
- Give a higher score when the action sequence is coherent and goal-directed.

Return only the following tags and nothing else:
`<score>FLOAT</score>`
`<reason>ONE SHORT SENTENCE</reason>`

Task: {`task_desc`}
Episode outcome: {`success_or_failure`}
Episode length: {`episode_length`}

Trajectory:
Step 1:
Observation: {`step_1_observation`}
Action: {`step_1_action`}
Step 2:
Observation: {`step_2_observation`}
Action: {`step_2_action`}
...
Step N:
Observation: {`step_n_observation`}
Action: {`step_n_action`}

Prompt Box D: ALFWorld direct LLM judge.

System prompt.
You are a strict ALFWorld trajectory judge. Output only the requested tags.

User prompt.
Directly assess the overall quality of the following ALFWorld rollout. Use your holistic judgment rather than a predefined rubric or a list of scoring criteria. Return the result in the required tagged format.

Return only the following tags and nothing else:
`<score>FLOAT</score>`
`<reason>ONE SHORT SENTENCE</reason>`

Task: {`task_desc`}
Episode outcome: {`success_or_failure`}
Episode length: {`episode_length`}

Trajectory:
Step 1:
Observation: {`step_1_observation`}
Action: {`step_1_action`}
Step 2:
Observation: {`step_2_observation`}
Action: {`step_2_action`}
...
Step N:
Observation: {`step_n_observation`}
Action: {`step_n_action`}

Repeated-evaluation invocation settings. The repeated-evaluation study in Appendix F uses `gpt-5.4-2026-03-05` with `reasoning_effort=medium` and a maximum of 512 completion tokens. Both modes receive the same task description, outcome, episode length, and observation–action trajectory, and use the same XML output fields. Temperature is not explicitly set, so requests use the service default. Transient API failures and malformed outputs are reissued until a valid tagged score is obtained. The direct and rubric-conditioned modes differ in the judging instructions reproduced in Prompt Boxes C and D.

APPENDIX F: REPEATED LLM-JUDGE STABILITY

This experiment isolates the repeatability of the online judge itself. We fix eight ALFWorld task groups with eight rollouts per group, yielding 64 trajectories sampled from the same checkpoint. Each trajectory retains its original instruction, success outcome, episode length, observations, and actions. We then query the same LLM judge independently 20 times with either the direct prompt in Prompt Box D or the rubric-guided prompt in Prompt Box C, for 2,560 valid judge calls in total. The requests are shuffled across the two modes.

For a trajectory i, we compute the population standard deviation of its 20 scores. To assess the consequence for group-relative updates, we also standardize the eight scores within each group at each repeat:

$$\widetilde{A}_i^{(r)} = \frac{s_i^{(r)} - \overline{s}_g^{(r)}}{\mathrm{Std}_{j \in g}\left(s_j^{(r)}\right) + 10^{-6}}. \tag{F.1}$$

where r indexes the repeated judge call, $s_i^{(r)}$ is the score assigned to trajectory i at repeat r, g is the rollout group containing trajectory i, $\overline{s}_g^{(r)}$ is the group mean score at repeat r, $\mathrm{Std}_{j \in g}(\cdot)$ is the standard deviation over trajectories in that group, 10^{-6} prevents division by zero, and $\widetilde{A}_i^{(r)}$ is the normalized judge score used for stability analysis. We report score stability, pairwise ranking flips, and changes in the sign of $\widetilde{A}_i^{(r)}$ across repeated evaluations.

Table F.1: Repeated LLM-judge stability on 64 fixed ALFWorld trajectories (8 groups $\times$ 8 rollouts, with 20 independent evaluations per trajectory and prompt). Metrics are grouped by preferred direction: lower score variation and flip rates are better, while higher top-choice consistency is better. The better value in each row is highlighted in bold, and the final column reports the relative change of the rubric-guided judge from the direct judge.

Metric	Direct judge	Rubric judge	Rubric change
Score dispersion (lower is better)			
Mean within-trajectory score std.	0.037	**0.023**	-37.5%
Median within-trajectory score std.	0.033	**0.022**	-32.8%
Mean absolute deviation	0.031	**0.019**	-37.8%
Mean score range	0.141	**0.084**	-40.3%
Preference and update variation (lower is better)			
Pairwise ranking flip rate	14.5%	**11.1%**	-23.0%
Advantage sign flip rate	16.3%	**10.1%**	-37.9%
Mean advantage std.	0.290	**0.211**	-27.3%
Top-choice consistency (higher is better)			
Top-1 modal agreement	70.6%	**73.8%**	$+4.4\%$

Table F.1 shows that the explicit rubric reduces score variation for identical trajectories. Direct judgments have a larger standard deviation on 55 of 64 trajectories, compared with nine trajectories for rubric-guided judgments. The more consequential group-level measures move in the same direction: rubric guidance lowers ranking flips by 23.0%, reduces normalized-advantage sign flips by 37.9%, and lowers the average standard deviation of the relative training signal by 27.3%. Thus, with a direct judge, a fixed rollout is more likely to receive opposite relative feedback across repeated calls even though the rollout itself has not changed.

The effect is consistent across task groups. Treating the eight groups as the sampling units, the mean reduction in within-trajectory score standard deviation is 0.014, with a group-cluster bootstrap 95% confidence interval of $[0.007, 0.025]$. All eight groups favor rubric-guided scoring (one-sided sign test $p = 0.004$). The reduction is stronger for successful rollouts (0.037 to 0.019) than for failed rollouts (0.039 to 0.037), suggesting that the generic rubric anchors clear completion evidence more effectively than ambiguous partial failure states. The two modes also use different score calibrations, so the relative ranking and advantage measures are more informative here than their mean scores. Because the policy and trajectories are held fixed, this study establishes repeated-evaluation stability only. It does not by itself establish an end-to-end improvement in final task success.

APPENDIX G: ALGORITHM DETAILS

Algorithm G.1 HiRUB trajectory reward construction

Require: Task instruction x, trajectory $\tau = \{(o_t, a_t)\}_{t=1}^{T}$, success flag y.
Ensure: Trajectory reward $R_{\mathrm{H}}(\tau, x)$.
1: Ground the hierarchical rubric in schema $z(x) = (f, s, c, \mathcal{M})$ and select checks $\mathcal{H}(z)$
2: Initialize event set $E \leftarrow \emptyset$ and repeat count $c_{\mathrm{rep}} \leftarrow 0$
3: **for** $t = 1$ to T **do**
4: Match (o_t, a_t) to progress, semantic, validity, and completion evidence
5: Add newly satisfied checks to E
6: Update consecutive repeat count c_{rep}
7: **end for**
8: Compute R_{prog} and R_{sem} from E and $z(x)$
9: Compute terminal component R_{succ} from y
10: Compute penalty R_{pen} from c_{rep} and T
11: **return** $[R_{\mathrm{prog}} + R_{\mathrm{sem}} + R_{\mathrm{succ}} - R_{\mathrm{pen}}]_{+}$

APPENDIX H: REWARD COEFFICIENTS

The main method describes the reward symbolically. The following tables list the environment-specific coefficients and clipping constants used in our experiments.

ALFWorld instantiation.

Component	Term	Value
Terminal	success bonus	4.0
Progress	seen target object	1.0
Progress	seen target receptacle	0.8
Progress	seen target toggle	0.8
Progress	target-directed goto	0.5
Interaction	correct target pick	1.5
Interaction	correct target place	1.5
Family	condition action	1.5
Family	light-task inspect	0.8
Two-object	each correct pick	0.9
Two-object	each correct put	0.9
Penalty	repeat coefficient / cap	0.08 / 1.2
Penalty	length threshold	25 steps
Penalty	length coefficient / cap	0.015 / 0.75

WebShop instantiation. The WebShop scorer retains the sparse environment outcome and adds evidence for search quality, relevant browsing, preference matching, action validity, and purchase timing. Repetition and unnecessary length are penalized after the positive terms are normalized.

Component	Term	Value
Sparse	environment completion	0.60
Semantic	search query overlap	0.12
Semantic	relevant page overlap	0.10
Semantic	preference option match	0.08
Validity	admissible/projection action	0.05
Completion	purchase timing	0.05
Penalty	repeat coefficient	0.08
Penalty	length coefficient	0.08
Budget	target step budget	8

Table I.1: Training configurations for both model scales and environments.

Setting	1.5B ALFWorld	7B ALFWorld	1.5B WebShop	7B WebShop
Train / validation batch size	16 / 128	16 / 128	16 / 128	16 / 128
Maximum prompt / response length	2048 / 512	2048 / 512	4096 / 512	4096 / 512
Maximum environment steps	50	50	15	15
Rollout group size	8	8	8	8
Learning rate	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
Discount factor	0.95	0.95	0.95	0.95
PPO mini / micro batch size	256 / 32	256 / 8	64 / 8	64 / 8
Rollout / reference log-probability micro batch	32 / 32	8 / 8	16 / 16	8 / 8
KL type / coefficient	low-variance / 0.01	low-variance / 0.01	low-variance / 0.01	low-variance / 0.01
Validation sampling (temperature)	sample (0.4)	sample (0.4)	sample (0.4)	sample (0.4)
Rollout dtype / memory utilization	bfloat16 / 0.6	bfloat16 / 0.6	bfloat16 / 0.6	bfloat16 / 0.6
Invalid-action penalty coefficient	0.1	0.1	0.1	0.1
Tensor parallel size	2	4	2	4
Actor parameter / optimizer offload	false / false	true / true	false / false	true / true
Reference parameter offload	true	true	true	true
Configured GPUs	2	8	2	8
Training epochs	150	150	150	150

Table I.2: Reward-coefficient ablation settings used for Table 2. The listed changes apply to positive shaping terms unless stated otherwise.

Setting	Coefficient construction
Reference	LLM-suggested coefficients listed in Appendix H
Uniform	Positive milestone coefficients set to a common value, with success and penalties unchanged
Strong hierarchy	Lower-level milestone weights reduced and higher-level milestone weights increased
Inverted hierarchy	Lower-level milestone weights increased and higher-level milestone weights reduced
Global scaling	Shaping coefficients scaled by 0.5 or 2.0, with success and penalties fixed
Random perturbation	Each shaping coefficient multiplied by an independent $\mathcal{U}(0.75, 1.25)$ factor

APPENDIX I: TRAINING DETAILS

The main text omits numerical training settings for readability. Table I.1 reports the complete configurations used at both model scales and in both environments. We adapt the training entry points from the public `langfengQ/verl-agent` implementation accompanying GiGPO (Feng et al., 2025). The dataset split and the remaining environment and rollout settings follow that project. For ablations, we alter only the reward construction while leaving the training entry point and main hyperparameters unchanged. The 1.5B and 7B columns use Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct, respectively. The 1.5B runs use 2 GPUs with tensor parallel size 2 and no actor or optimizer offload. The 7B runs use 8 GPUs with tensor parallel size 4 and actor and optimizer offload enabled. The reward coefficients are listed separately in Appendix H.

Reward-coefficient ablation protocol. The coefficient study uses the Qwen2.5-1.5B ALFWorld and WebShop configurations above. All variants use the same generated rubric, target checks, data split, optimizer, rollout budget, and three-seed protocol as the main HiRUB run. We change only the coefficient pattern. The original setting is the reference. The uniform setting assigns the same positive weight to milestone terms. Strong hierarchy increases the contrast between lower-level and higher-level milestone terms, while inverted hierarchy reverses that contrast. Global scaling changes the shaping scale while leaving the success bonus and penalty structure fixed. Random perturbation multiplies each shaping coefficient by an independent factor sampled from $\mathcal{U}(0.75, 1.25)$, with the success bonus held fixed.

Judge timing and API concurrency. The Figure 4 comparison uses matched 150-step ALFWorld runs on two NVIDIA H20 GPUs. The online-judge runs issue requests at an API concurrency of two. Because service-side batching is external to our implementation, we report measured wall-clock and reward-computation time. The judge run takes 31.1 hours in total and 238.1 seconds per step for reward computation, while the HiRUB run takes 19.6 hours and 1.0 second per step.

APPENDIX J: LIMITATIONS

HiRUB relies on a task schema and trajectory checks that expose the relevant evidence in each environment. This is practical for ALFWorld and WebShop, where instructions, actions, and observations have stable structure, but it may require additional engineering in more open-ended settings. The coverage and granularity of the rubric remain dependent on the generating LLM. It may omit evidence that matters for a particular task or phrase a criterion too broadly to separate genuine progress from incidental behavior. Finally, our judge comparison measures preference similarity and runtime in a controlled setting rather than replacing broader human evaluation. Future work can study robustness to noisier observations, broader task formats, and human validation of learned behaviors.