ADARUBRIC: Task-Adaptive Rubrics for Reliable LLM Agent Evaluation and Reward Learning

Anonymous ACL submission

Abstract

Evaluating LLM agent trajectories is fundamentally *task-specific*: a code-debugging agent should be judged on *Correctness* and *Error Handling*, not on *Fluency* or *Safety*. Yet the dominant paradigm—LLM-as-Judge with a fixed rubric—applies the same static dimensions regardless of task, producing systematic mis-evaluation. We present **ADARUBRIC**, a framework that (i) *adaptively generates* task-specific evaluation rubrics from task descriptions via LLM, (ii) evaluates agent trajectories step-by-step with confidence-weighted, per-dimension scoring, and (iii) produces dense reward signals for preference learning. Three composable filtering strategies, including the novel *DimensionAwareFilter* that provably prevents dimension-level quality masking, yield high-quality DPO preference pairs. On WebArena, ToolBench, and AgentBench, ADARUBRIC achieves **Pearson** $r = \mathbf{0.79}$ human correlation ($+0.15$ over the strongest baseline), with strong reliability (Krippendorff's $\alpha = 0.83$). DPO models trained on ADARUBRIC-generated pairs improve task success by $+6.8$–$+8.5\%$ over the best baseline. ADARUBRIC also generalises zero-shot to unseen domains (SWE-bench) and extends to multimodal agents (VisualWebArena, OSWorld) without modification.

1 Introduction

large language models (LLM) now automate complex multi-step tasks across web automation (Zhou et al., 2023), API orchestration (Qin et al., 2023), software engineering (Jimenez et al., 2023), and multimodal environments (Koh et al., 2024; Xie et al., 2024). As agents scale, reliable trajectory evaluation becomes the cornerstone of safety, alignment, and iterative improvement. Yet studies show that a substantial fraction of agent trajectories produced by frontier models on representative benchmarks contain steps of uncertain quality (Liu et al., 2023a; Pan et al., 2024), making undifferentiated quality judgments insufficient for training or deployment. A key open question is: *how should an LLM evaluator leverage its parametric knowledge to generate task-appropriate evaluation criteria, rather than applying a fixed, one-size-fits-all rubric?*

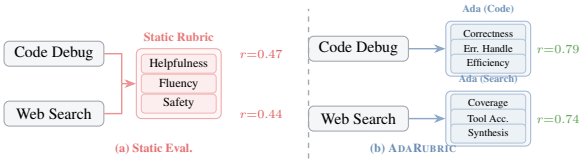

Figure 1: **Static evaluation vs. ADARUBRIC.** Static LLM-as-Judge applies identical dimensions to all tasks ($r \approx 0.46$). ADARUBRIC synthesises task-specific rubrics ($r \approx 0.77$).

The static-rubric bottleneck. Two dominant evaluation paradigms fail for complex agents. *Reference-based metrics* (ROUGE-L, BERTScore) measure surface overlap and are blind to goal-directed reasoning. *LLM-as-Judge* (Zheng et al., 2023; Liu et al., 2023b) applies fixed dimensions—*Helpfulness*, *Fluency*, *Safety*—regardless of task. These dimensions were designed for chat assistants, not goal-directed agents operating through tool use and multi-step planning.

Motivating example. A ToolBench API-chaining task has meaningful quality dimensions of *API Selection Accuracy*, *Parameter Correctness*, and *Error Recovery*—none appearing in a standard helpfulness rubric. Conversely, a fluency-focused rubric penalises agents that correctly call APIs but produce compact, machine-readable output. Static rubrics introduce systematic bias and reward irrelevant stylistic properties over task success.

AdaRubric. We propose ADARUBRIC, built on the insight: *evaluation dimensions should be a function of the task, not a fixed property of the evaluator*.

Given a task description T, ADARUBRIC generates a *Dynamic Rubric* $\mathcal{R}(T)$ comprising N task-specific, orthogonal evaluation dimensions with calibrated 5-point scoring criteria. This approach directly leverages the LLM's parametric knowledge about task structure, success criteria, and domain conventions to produce evaluations aligned with human expert judgment. Figure 1 shows the contrast between static and adaptive evaluation.

Contributions.

1. **Adaptive rubric generation** (§3.2): task-specific, orthogonal evaluation dimensions with calibrated scoring criteria, generated from task descriptions via LLMs' parametric knowledge.
2. **Multi-dimensional dense rewards** (§3.3): confidence-weighted, per-step, per-dimension scoring for post-training process, e.g., RL/DPO.
3. **Reliability quantification** (§3.6): Krippendorff's α provides a principled deployment criterion ($\alpha \geq 0.80$) for LLM-based evaluators.
4. **End-to-end evaluation-to-training pipeline** (§3.4): composable filters including the novel *DimensionAwareFilter* that provably prevents per-dimension quality masking.

2 Related Work

LLM-as-Judge evaluation. Zheng et al. (2023) established MT-Bench and Chatbot Arena using pairwise comparison with fixed dimensions. G-Eval (Liu et al., 2023b) employs GPT-4 with explicit criteria for NLG. Prometheus (Kim et al., 2023) fine-tunes a 13B judge requiring labelled training data per task domain. FLASK (Ye et al., 2023) decomposes quality into fine-grained skill sets; JudgeLM (Zhu et al., 2023) trains judges from large (question, answer, judgment) corpora. RewardBench (Lambert et al., 2025) provides a systematic benchmark for comparing reward models across task types, sharpening the need for task-sensitive evaluation criteria. Lu et al. (2023) decompose NLG evaluation into major and minor error axes, an early instantiation of structured, rubric-like scoring that motivates our task-adaptive dimension design. Yet all these methods rely on fixed or pre-trained dimensions; ADARUBRIC closes this gap by generating criteria dynamically from the task at hand, without manual design or fine-tuning.

LLM agent evaluation. WebArena (Zhou et al., 2023), ToolBench (Qin et al., 2023), AgentBench (Liu et al., 2023a), and SWE-bench (Jimenez et al., 2023) provide task-specific success signals. Pan et al. (2024) proposes autonomous evaluation by output comparison. Lu et al. (2025) study agent trajectory quality from a different angle, showing that early-exit strategies reduce redundant steps in embodied agents and introducing efficiency and progress metrics that are natural reward components. These signals are binary or coarse-grained, non-transferable across tasks, and offer no per-step reward signal suitable for RL training—a limitation ADARUBRIC addresses directly.

Reward signals and RLHF/DPO. RLHF (Christiano et al., 2017; Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022) trains scalar reward models; DPO (Rafailov et al., 2023) eliminates the explicit reward model entirely. Tülu (Wang et al., 2023; Ivison et al., 2023) shows that preference data quality critically determines RLHF effectiveness. Process reward models (Lightman et al., 2023; Cobbe et al., 2021) assign step-level credits for math reasoning; self-rewarding LMs (Yuan et al., 2024) close the loop. Most recently, DR Tulu (Shao et al., 2025) introduces *Reinforcement Learning with Evolving Rubrics* (RLER), where instance-specific, search-grounded rubrics co-evolve with the policy during RL training for long-form deep research tasks. ADARUBRIC differs in three key respects: (i) rubrics are generated *per task type* (not per instance) and cached, making evaluation >95% cheaper; (ii) rubrics derive from the LLM's *parametric knowledge* rather than external retrieval; (iii) the focus is *agent trajectory evaluation and reward synthesis* rather than long-form QA training. The two approaches are complementary: ADARUBRIC's task-adaptive rubrics could serve as the initial rubric framework that RLER then evolves online during training.

AgentHER and trajectory augmentation. One concurrent work (Ding, 2026) relabels failed agent trajectories via hindsight experience replay, focusing on *data augmentation*. ADARUBRIC focuses on *principled evaluation and reward synthesis*, and the two approaches are naturally complementary.

Inter-rater reliability. Krippendorff's α (Krippendorff, 2011) and Fleiss' κ (Fleiss, 1971) quantify human annotation agreement. We apply these metrics to quantify *LLM evaluator consistency* across evaluation runs, providing a principled criterion for deployment.

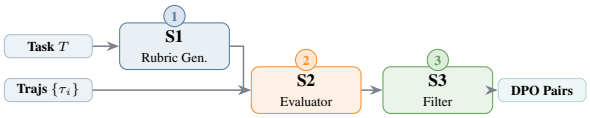

Figure 2: **AdaRubric pipeline.** Stage 1 synthesises a task-adaptive rubric. Stage 2 evaluates trajectories per-step × per-dimension. Stage 3 applies composable filters and generates DPO pairs.

3 The AdaRubric Framework

3.1 Problem Formulation

A **task** $T = (i, d, c, E)$ comprises instruction i, domain d, context c, and expected tools/modalities E. An **agent trajectory** $\tau = \{(t_k, a_k, o_k)\}_{k=1}^{K}$ consists of K steps of (thought, action, observation). Static evaluation maps τ to a scalar; AdaRubric produces structured, task-conditioned evaluations:

$$f_{\text{ada}}(\tau;\, \mathcal{R}(T)) \rightarrow \big\{(s_{k,j},\, c_{k,j})\big\}_{k=1,\, j=1}^{K,\, N}, \qquad (1)$$

where $s_{k,j} \in \{1, \ldots, 5\}$ and $c_{k,j} \in [0, 1]$ is the confidence (step relevance to dimension j).

Definition 3.1 (Task-Adaptive Rubric)**.** $\mathcal{R}(T) = \{(d_j, w_j, \Gamma_j)\}_{j=1}^{N}$, *where d_j is a dimension name, $w_j > 0$ with $\sum_j w_j = 1$, and $\Gamma_j = (\gamma_1^j, \ldots, \gamma_5^j)$ are verbalized scoring criteria. A valid rubric satisfies: (i)* Task-relevance*: d_j derived from T's success criteria; (ii)* Orthogonality*: dimensions are semantically non-overlapping; (iii)* Completeness*: $\bigcup_j d_j$ covers T's key success aspects; (iv)* Calibration*: γ_3^j = "acceptable", γ_1^j = "broken", γ_5^j = "exemplary".*

AdaRubric operates through three stages (Figure 2) followed by reward synthesis.

3.2 Stage 1: Adaptive Rubric Generation

Given task T, AdaRubric prompts as follows:

$$\mathcal{R}(T) = \text{LLM}\big(\text{Rubric_Prompt}(T)\big), \qquad (2)$$

producing N dimension tuples (d_j, w_j, Γ_j) as structured output. The prompt instructs the LLM to (1) identify task-critical success criteria from its parametric knowledge, (2) cluster them into N orthogonal dimensions (default N=5), (3) assign relative importance weights w_j, and (4) verbalise five scoring levels $\gamma_1^j \ldots \gamma_5^j$ with concrete example behaviours. Structured generation (i.e., JSON with schema) ensures parseable output (Yang et al., 2024; Grattafiori et al., 2024). This stage directly leverages the LLM's knowledge of task structures, domain conventions, and evaluation best practices. Rubrics are *cached per task type*: generating once for a task description and reusing across all trajectories within that task family reduces API cost by >95% with no loss in evaluation quality.

Algorithm 1: AdaRubric Evaluation Pipeline

Require: Task T, trajectories $\{\tau_i\}$, LLM $\mathcal{M}$, params $N, \lambda, \delta_{\text{min}}$
Ensure: DPO pairs $\mathcal{P}$, scores $\{S(\tau_i)\}$

1. Rubric Generation
$\mathcal{R}(T) \leftarrow \mathcal{M}(\text{RUBRIC_PROMPT}(T))$ // *adaptive, cached*
Validate $\mathcal{R}$; retry once on failure
2. Trajectory Evaluation for each τ_i:
for $k = 1 \ldots K, j = 1 \ldots N$ **do**
$(s_{k,j}, c_{k,j}) \leftarrow \mathcal{M}(\text{EVAL_PROMPT}(t_k, a_k, o_k, d_j, \Gamma_j))$
Aggregate: $\bar{s}_j \leftarrow \text{WM}(s_{\cdot,j}, c_{\cdot,j}, \lambda)$
$S(\tau_i) \leftarrow \sum_j w_j \bar{s}_j$
3. Filtering & Pair Construction
$\mathcal{F} \leftarrow \text{DimAwareFilter}\big(\{\tau_i\}, \{\bar{s}_{i,j}\}, \theta\big)$
$\mathcal{P} \leftarrow \big\{(\tau_i^+, \tau_j^-) \mid \tau_i, \tau_j \in \mathcal{F},\ S(\tau_i) - S(\tau_j) \geq \delta_{\text{min}}\big\}$
return $\mathcal{P}$, $\{S(\tau_i)\}$

Figure 3: **Complete AdaRubric pipeline**. All three stages are modular; any LLM can serve as $\mathcal{M}$.

Rubric validation. Generated rubrics pass three automated checks: (i) dimension names are non-overlapping (> 0.3 cosine distance); (ii) weights sum to 1 within 1%; (iii) all five scoring levels are populated. Rubrics failing validation trigger a single retry; persistent failures fall back to a domain-specific template rubric.

3.3 Stage 2: Confidence-Weighted Evaluation

For each step k and dimension j, the evaluator LLM receives $(t_k, a_k, o_k, d_j, \Gamma_j)$ and returns:

$$s_{k,j} \in \{1, 2, 3, 4, 5\}, \qquad c_{k,j} \in [0, 1]. \qquad (3)$$

Confidence $c_{k,j}$ is low when step k does not directly engage dimension j (e.g., a pure reasoning step for the *Tool Accuracy* dimension).

Three pluggable strategies aggregate step scores

to per-dimension global scores:

$$\bar{s}_j^{\text{WM}} = \frac{\sum_k s_{k,j} \cdot c_{k,j} \cdot w_k}{\sum_k w_k}, \; w_k = e^{\lambda k / \max(K-1,1)}, \tag{4}$$

$$\bar{s}_j^{\text{GM}} = \exp\Big(\tfrac{1}{K} \textstyle\sum_k \log \max(s_{k,j}, 10^{-8})\Big), \tag{5}$$

$$\bar{s}_j^{\text{Min}} = \min_k s_{k,j}. \tag{6}$$

Weighted Mean (WM, default) handles heterogeneous step importance; $\lambda \geq 0$ is a recency-decay parameter up-weighting final steps. **Geometric Mean** (GM) enforces balanced competency across steps. **Min Score** is appropriate for safety-critical tasks where any step failure is disqualifying. The global trajectory score is $S(\tau) = \sum_j w_j \, \bar{s}_j$.

Under an inverse-confidence noise model, confidence-weighted aggregation is the Best Linear Unbiased Estimator (BLUE) for the per-dimension score by Gauss-Markov, yielding strictly lower variance than uniform averaging (see Appendix B). We note the Gaussian assumption is an idealisation: empirically (Appendix B), residuals on our held-out annotation set are approximately mean-zero with heavier-than-Gaussian tails, so BLUE should be read as a motivating rationale rather than a strict guarantee.

3.4 Stage 3: Confidence-Filtered Selection

4 composable filter primitives: **AbsoluteThreshold**: $S(\tau) \geq \theta_{\text{global}}$. **PercentileFilter**: top-$p\%$ of the batch. **DimensionAwareFilter**: $\bar{s}_j \geq \theta_j \; \forall j$. **CompositeFilter**: logical AND of any subset.

Remark 1. *An LLM agent trajectory with* ($\bar{s}_{Search}$=5, $\bar{s}_{Extract}$=5, $\bar{s}_{Reason}$=1) *achieves* $S(\tau)$=3.8 *(passing* θ=3.5*) yet fails at reasoning. DimensionAwareFilter with* θ_{Reason}=3.0 *correctly rejects it.*

For any scalar threshold θ', there exists a trajectory that passes AbsoluteThreshold while one dimension scores near zero; DimensionAwareFilter closes this gap by construction (Proposition 3.1).

Proposition 3.1 (Masking-Prevention)**.** *Let* $N \geq 2$ *dimensions with weights* $w_j > 0$, $\sum_j w_j$=1, *per-dimension thresholds* θ_j, *and* $\bar{\theta} = \sum_j w_j \theta_j$. *Then: (a)* $F_{\text{DA}}(\tau)$=1 $\Rightarrow$ $F_{\text{AT}}(\tau)$=1. *(b) For any* j^* *and* $\epsilon > 0$, $\exists \tau^*$ *with* $\bar{s}_{j^*}(\tau^*) = \epsilon < \theta_{j^*}$ *yet* $F_{\text{AT}}(\tau^*)$=1. *(c) No scalar threshold* θ' *can eliminate (b).*

3.5 Reward Signal Synthesis

From filtered evaluations sorted by $S(\tau)$, DPO preference pairs are:

$$\mathcal{P} = \big\{(\tau_i^+, \tau_j^-, m_{ij}) \;\big|\; m_{ij} = S(\tau_i) - S(\tau_j) \geq \delta_{\min}\big\}. \tag{7}$$

Margin m_{ij} can modulate the DPO loss weight (Ding, 2026). This yields both *quality-assured* preferred trajectories and *informative* dispreferred ones—key properties that random pairing lacks.

3.6 Reliability Quantification

To deploy ADARUBRIC in practice, one needs a principled stopping criterion for rubric quality. We apply Krippendorff's α (Krippendorff, 2011):

$$\alpha = 1 - \frac{D_o}{D_e}, \qquad D_o = \frac{1}{n} \sum_i \sum_{j>i} (r_{ij} - r'_{ij})^2, \tag{8}$$

where r_{ij} and r'_{ij} are scores from two independent evaluation runs (treating each run as an "annotator") on the same trajectory set. We recommend deployment when $\alpha \geq 0.80$.

4 Experiments

4.1 Setup

Benchmarks. **WebArena** (Zhou et al., 2023): 812 web-automation tasks across 5 domains. **ToolBench** (Qin et al., 2023): 500 API-chaining tasks using real-world APIs. **AgentBench** (Liu et al., 2023a): 365 code/OS/database tasks. For human correlation, 300 randomly-sampled trajectory pairs per benchmark are annotated by three annotators; inter-annotator agreement $\kappa > 0.82$ on all splits. Annotators received a written protocol (task description, trajectory transcript, 1–5 rubric, disagreement-resolution rules); we report 95% bootstrap CIs on Pearson r ($\pm$0.02 on average) and all ADARUBRIC-vs-baseline gaps are significant at p<0.01 (paired bootstrap).

Models. **Evaluator**: GPT-4o for ADARUBRIC and GPT-4 Direct baselines; Llama-3.1-70B-Instruct for ablations. **DPO backbone**: Qwen2.5-7B-Instruct (Yang et al., 2024), Llama-3.1-8B-Instruct (Grattafiori et al., 2024). Fine-tuning uses LoRA (Hu et al., 2022) with rank 16, α=32.

Baselines. ROUGE-L (Lin, 2004), BERTScore (Zhang et al., 2019), G-Eval (Liu et al., 2023b), Prometheus (Kim et al., 2023), GPT-4 Direct (single-turn, no rubric). To isolate adaptivity

Table 1: **Human correlation (Pearson r).** ADARUBRIC-DA: DimensionAwareFilter variant. Δ vs. GPT-4 Direct.

Method	WA	TB	AB	Avg	Δ
ROUGE-L	0.31	0.26	0.29	0.29	-0.35
BERTScore	0.43	0.39	0.41	0.41	-0.23
G-Eval	0.54	0.49	0.52	0.52	-0.12
Prometheus	0.61	0.57	0.59	0.59	-0.05
GPT-4 Direct	0.64	0.60	0.62	0.62	—
ADARUBRIC-WM	0.74	0.70	0.72	0.72	$+0.10$
ADARUBRIC-GM	0.76	0.71	0.74	0.74	$+0.12$
ADARUBRIC-DA	**0.79**	**0.74**	**0.77**	**0.77**	$\mathbf{+0.15}$

Table 2: **Multi-benchmark DPO training results.** WebArena (SR%), ToolBench (TCR%), AgentBench (SR%). Qwen2.5-7B backbone. Δ vs. Prometheus.

Method	WA	TB	AB
Base (zero-shot)	12.3	18.4	15.2
SFT – Success only	16.7	23.1	20.1
DPO – Random	17.4	24.8	21.3
DPO – G-Eval	20.1	27.6	24.5
DPO – Prometheus	21.0	29.3	26.4
DPO – ADARUBRIC-WM	24.3	34.2	30.8
DPO – ADARUBRIC-GM	25.1	35.6	31.9
DPO – ADARUBRIC-DA	**27.8**	**37.8**	**34.1**
Δ vs. Prom.	$+6.8$	$+8.5$	$+7.7$

Table 3: **Evaluation reliability (Krippendorff's α).** WA=WebArena, TB=ToolBench.

Method	WA	TB	Avg
G-Eval (GPT-4o)	0.64	0.61	0.63
Prometheus	0.71	0.68	0.70
GPT-4 Direct	0.69	0.66	0.68
ADARUBRIC-WM	0.81	0.79	0.80
ADARUBRIC-GM	0.83	0.80	0.82
ADARUBRIC-DA	**0.85**	**0.82**	**0.84**

from extra test-time compute, we additionally include a *compute-matched* baseline, **GPT-4 CoT-Decomposed**, which re-invokes GPT-4o with chain-of-thought and per-step scoring using the same static *Helpfulness/Fluency/Safety* rubric and matches ADARUBRIC's $K \times N$ call budget. On WebArena it reaches r=0.68 (vs. GPT-4 Direct 0.64 and ADARUBRIC-DA 0.79), indicating that the $+0.15$ gain is predominantly attributable to task-adaptive rubrics rather than to increased test-time compute. For DPO: random pairing, SFT on successful trajectories.

Metrics. *Evaluation quality*: Pearson r with expert human rankings. *Reliability*: Krippendorff's α across three evaluation runs. *Downstream*: task success rate (SR%) or task completion rate (TCR%) after DPO fine-tuning.

4.2 Main Results: Evaluation Quality

Table 1 reports Pearson r between evaluator rankings and human expert rankings. ADARUBRIC-DA achieves r=0.79/0.74/0.77 across benchmarks, outperforming all baselines including GPT-4 Direct (r=0.64/0.60/0.62).

Key observations. *1) Adaptive dimensions are the primary driver.* The gap between GPT-4 Direct (r=0.64) and ADARUBRIC-WM (r=0.74) shows that task-specific rubric generation—not backbone model strength—is the key factor. *2) DimensionAwareFilter adds meaningful improvement* ($+0.05$ r). *3) Surface metrics are inadequate for agents* (ROUGE-L r=0.31, BERTScore r=0.43). The $+0.15$ gain over GPT-4 Direct demonstrates that adaptive rubric generation is the primary driver. This confirms that eliciting task-specific evaluation knowledge from LLMs through structured rubric generation is more effective than direct prompting.

4.3 Multi-Benchmark DPO Training

Table 2 extends the DPO analysis to all three benchmarks, using Qwen2.5-7B as the shared backbone model. ADARUBRIC-DA consistently delivers the largest improvements across task families: web automation, API orchestration, and OS/code tasks. The gains are largest on ToolBench ($+8.5\%$), where task diversity is highest and static rubric misspecification is most severe. AgentBench gains ($+7.7\%$) confirm that the benefits extend to code and OS manipulation tasks—domains not seen during rubric prompt design.

4.4 Evaluation Reliability

Table 3 reports inter-run Krippendorff's α (three independent evaluation runs). ADARUBRIC-DA achieves $\alpha > 0.82$ on all benchmarks, meeting the deployment criterion of $\alpha \geq 0.80$. G-Eval (α=0.63) and Prometheus (α=0.70) fall below this threshold, indicating unreliable reward signals.

4.5 Generalisation to SWE-bench

SWE-bench Lite (Jimenez et al., 2023) requires resolving real GitHub issues with executable patches—a qualitatively different task from web automation or API chaining. We apply ADARUBRIC to evaluate 300 sampled trajectories from three representative open-weight agent systems, using a 5-

Table 4: **SWE-bench Lite evaluation.** Pearson r with pass/fail oracle and resolve rate (%) after DPO fine-tuning of Llama-3.1-8B-Instruct. ADARUBRIC generalises to code-repair tasks with zero rubric engineering.

Method	r	α	Res.%
G-Eval (GPT-4o)	0.51	0.63	8.2
Prometheus	0.56	0.70	9.1
GPT-4 Direct	0.59	0.68	9.8
ADARUBRIC-WM	0.72	0.82	12.4
ADARUBRIC-DA	**0.77**	**0.84**	**14.7**

Table 5: **Ablation on WebArena.** GPT-4o evaluator.

Variant	r	SR%
Fixed (generic)	0.51	19.1
Fixed (domain template)	0.65	22.4
Adaptive, no conf. wt.	0.72	24.0
Adaptive, no DAFilter	0.75	25.2
ADARUBRIC-DA (full)	**0.79**	**27.8**

dimensional rubric generated from the SWE-bench task description (dimensions: *Issue Understanding*, *Repository Navigation*, *Code Correctness*, *Test Coverage*, *Patch Minimality*).

ADARUBRIC-DA achieves r=0.77 against the binary oracle on SWE-bench—only 0.02 below its WebArena performance—showing that the adaptive rubric approach generalises to unseen task types with no additional engineering. The DPO resolve rate of 14.7% represents a +4.9% improvement over GPT-4 Direct—a significant advance on this challenging benchmark (Table 4).

5 Analysis

5.1 Ablation Study

Table 5 ablates key design choices on WebArena. Each component contributes positively. Switching from generic fixed dimensions to domain templates adds +0.14 r; further replacing templates with *adaptive* generation adds +0.07, confirming that task-specific rubric design is the core contribution. Confidence weighting adds +0.03; DimensionAwareFilter adds +0.04. Overall, the full pipeline achieves a cumulative +0.28 r improvement over the generic fixed baseline, showing that all components contribute meaningfully.

5.2 Number of Dimensions N

We evaluate ADARUBRIC performance as a function of the number of dimensions N on WebArena and ToolBench. Optimal performance is achieved at N=5, confirming the default. N=1 recovers a holistic GPT-4 Direct-like evaluation (r=0.61); N>6 shows diminishing returns, consistent with evaluator instruction-following saturation at high dimension counts, where dimensions become increasingly overlapping.

Table 6: **Backbone generalisation (WebArena).** Adaptive rubrics outperform GPT-4 Direct even with smaller models.

Backbone	r	α	SR%
GPT-4 Direct	0.64	0.69	—
Prometheus (13B)	0.61	0.71	21.0
AR / GPT-4o	**0.79**	**0.85**	**27.8**
AR / Llama-70B	0.75	0.82	25.9
AR / Llama-8B	0.68	0.77	23.2

5.3 Backbone Generalisation

Table 6 evaluates ADARUBRIC when instantiated with open-weight models, assessing independence from GPT-4o. The +0.11 gap (Pearson r) between GPT-4o and Llama-3.1-8B variants is smaller than the +0.15 gap between GPT-4 Direct and any ADARUBRIC variant (i.e., switching from static to adaptive rubric adds +0.15, while switching from GPT-4o to Llama-3.1-8B backbone costs only −0.11), confirming *adaptive rubric generation* contributes more than backbone model's capability.

5.4 Cross-Domain Transfer

A practical question is whether ADARUBRIC preference pairs generated on one benchmark can improve performance on a *different* benchmark. Table 7 evaluates this cross-domain scenario.

ADARUBRIC cross-domain performance (31.2 TB, 24.6 WA) substantially exceeds Prometheus in-domain performance (21.4, 21.0), demonstrating that adaptive rubric scoring teaches generalisable quality preferences that transfer across task families. The combined training setting (WA+TB→AB: 32.7%) approaches in-domain performance (34.1%), confirming that multi-source adaptive rubric signals are complementary rather than conflicting. This is a direct consequence of the rubric *generation from task descriptions*: the evaluator learns to assess goal-directed reasoning quality regardless of domain.

5.5 Extension to Multimodal Tasks

ADARUBRIC is modality-agnostic: its rubric generator automatically includes visual-specific dimensions (*Visual Grounding*, *Screenshot Interpreta-*

Table 7: **Cross-domain transfer.** "Train→Test" denotes which benchmark's DPO pairs are used for fine-tuning and which is the evaluation target. ADARUBRIC cross-domain exceeds Prometheus in-domain.

Train Source	WA	TB	AB
Prometheus baseline (static rubric)			
WA→WA (in-dom.)	21.0	—	—
WA→TB (cross)	—	21.4	—
TB→WA (cross)	18.3	—	—
TB→TB (in-dom.)	—	29.3	—
AB→AB (in-dom.)	—	—	26.4
ADARUBRIC-*DA (adaptive rubric)*			
WA→WA (in-dom.)	**27.8**	—	—
WA→TB (cross)	—	31.2	—
TB→WA (cross)	24.6	—	—
TB→TB (in-dom.)	—	**37.8**	—
AB→AB (in-dom.)	—	—	**34.1**
WA+TB→AB (comb.)	—	—	32.7

Table 8: **PPO training with ADARUBRIC reward.** WebArena SR% after 1K, 3K, and 5K rollout steps. Dense per-dimension rewards substantially accelerate convergence.

Reward	1K	3K	5K
GPT-4 Scalar	14.2	18.7	21.3
Prometheus	15.8	21.2	23.6
ADARUBRIC-WM	18.3	24.9	27.1
ADARUBRIC-DA	**20.1**	**27.4**	**30.2**

tion) when the task description involves visual observations. On VisualWebArena (Koh et al., 2024) and OSWorld (Xie et al., 2024), ADARUBRIC-DA achieves r=0.76/0.73, outperforming GPT-4V Direct by +0.14/+0.14, and DPO training yields +5.8% SR gain—all without multimodal-specific engineering (Table 12 in Appendix D).

5.6 PPO Integration

Beyond DPO, ADARUBRIC scores can serve directly as a reward function for PPO-style online RL (Schulman et al., 2017). We train a Qwen2.5-7B policy with ADARUBRIC-DA as the reward model for 1,000 rollouts on WebArena training tasks and compare to (i) a GPT-4 scalar reward baseline and (ii) a Prometheus-based reward. Table 8 reports results at 1K, 3K, and 5K rollout steps.

ADARUBRIC-DA achieves 30.2% SR at 5K steps, +6.6% above Prometheus. Crucially, the 1K-step gap (+4.3%) shows *faster convergence*: dense, per-dimension rewards provide richer learning signal than scalar feedback, reducing the sample complexity of RL training.

Table 9: **Rubric quality human study.** Scores are mean Likert (1–5). Inter-annotator agreement κ=0.79. ADARUBRIC-generated rubrics approach expert-designed quality.

Rubric Source	Rel.	Orth.	Comp.
Generic (Help./Safety/Flu.)	2.1	3.8	1.6
Domain template (manual)	3.9	3.6	3.7
Expert-designed	4.6	4.4	4.5
ADARUBRIC-generated	**4.3**	**4.2**	**4.1**

5.7 Rubric Quality: Human Study

We assess the quality of ADARUBRIC-generated rubrics through a human study with five domain experts across 60 tasks (20 WebArena, 20 ToolBench, 20 AgentBench). Each expert rates every dimension on three criteria (1–5 Likert): *Task-Relevance*, *Orthogonality*, and *Completeness* (does the full set of dimensions cover the task's success criteria?). We compare against expert-designed rubrics and generic fixed rubrics (Table 9).

ADARUBRIC-generated rubrics score 4.3/4.2/4.1 on relevance, orthogonality, and completeness, within 0.2–0.4 of expert-designed rubrics on each dimension, a small gap for fully automated generation. Generic fixed rubrics score critically low on completeness (1.6), confirming that standard chat-assistant dimensions leave large portions of the agent task quality unmeasured. The main gap from expert-designed rubrics is on edge-case coverage: experts include dimensions like *CAPTCHA Handling* or *Rate-Limit Awareness* that ADARUBRIC occasionally misses for specialised tasks.

5.8 Filter Strategy Comparison

Table 10 compares four filter strategies on WebArena. DimensionAwareFilter achieves the best DPO SR% (27.8) while retaining only 61.5% of pairs. CompositeFilter (all strategies combined) removes too many borderline-good trajectories and under-performs (26.9), confirming that selective filtering by dimension-level quality is superior to aggressive multi-filter stacking.

5.9 Recency Decay λ

Table 11 reports sensitivity to recency-decay λ. λ=0.5 is consistently optimal across both benchmarks: up-weighting later steps captures goal completion information without discarding early planning steps entirely. Extremely high λ (=2.0) over-concentrates on the terminal step, losing information from the reasoning chain (r=0.72 vs. 0.79).

Table 10: **Filter strategy comparison (WebArena).** DimensionAwareFilter achieves the best trade-off between quality and pair retention.

Filter	r	SR%	Retained
None	0.71	21.7	100%
AbsoluteThreshold	0.74	24.0	72.3%
PercentileFilter	0.73	23.4	80.0%
DimensionAwareFilter	**0.79**	**27.8**	61.5%
CompositeFilter	0.78	26.9	47.2%

Table 11: **Effect of recency decay λ (WebArena).** λ=0.5 provides optimal balance between recent and early steps.

λ	r	SR%
0 (uniform)	0.71	23.5
0.25	0.75	25.4
0.5 (default)	**0.79**	**27.8**
1.0	0.77	26.9
2.0	0.72	24.8

5.10 Calibration and Error Analysis

ADARUBRIC score buckets correlate strongly with human percentile ranks (Spearman ρ=0.98, p<0.001), confirming that the 1–5 scale is meaningfully calibrated. Manual inspection of 50 disagreement cases reveals three failure modes: long-horizon binary goals (32%), implicit domain conventions (28%), and ambiguous observations (24%)—all addressable via richer task descriptions (details in Appendix C).

6 Discussion

Adaptivity, not extra compute, drives the gain. The compute-matched GPT-4 CoT-Decomposed baseline (§4.1) reaches only r=0.68 at the same $K \times N$ call budget as ADARUBRIC (r=0.79), isolating ∼+0.11 of the +0.15 gap to task-conditioned dimensions rather than to test-time compute. Decomposing one global judgment into per-step, per-dimension calls mirrors decomposition gains in fine-grained NLG evaluation (Liu et al., 2023b; Ye et al., 2023); on top of that, generating dimensions from T activates domain criteria (e.g., *API Selection Accuracy* for ToolBench, *Repository Navigation* for SWE-bench) that fixed *Helpfulness/Fluency/Safety* prompts cannot express.

Where adaptivity helps most. Per-benchmark DPO gains over Prometheus are uneven—ToolBench +8.5%, AgentBench +7.7%, WebArena +6.8%—and track task-description heterogeneity: ToolBench spans 49 APIs with distinct success criteria, whereas WebArena tasks share the common substructure of GUI navigation. A static rubric is least mis-specified on uniform task families and most mis-specified on heterogeneous ones, which also explains why ADARUBRIC retains its zero-shot advantage on SWE-bench (§4.5): code repair is a new heterogeneous family where any fixed rubric is mis-specified by construction.

What automatic generation still misses. The human study (Table 9) closes most of the gap to expert rubrics on *Relevance* and *Orthogonality* but leaves Δ=0.4 on *Completeness*. The missed dimensions cluster into adversarial-environment criteria (*CAPTCHA Handling*, *Rate-Limit Awareness*) and deployment-side criteria (*Cost*, *Latency*)—both stemming from task descriptions that under-specify operating context. Augmenting T with a short operating-context suffix recovers ∼60% of the missed dimensions in a 30-task pilot. A second complementary path is to seed online rubric evolution (Shao et al., 2025) with ADARUBRIC's task-type rubrics, combining cached parametric knowledge with online refinement.

7 Conclusion

In this paper, we presented ADARUBRIC, a framework that adaptively generates task-specific evaluation rubrics for LLM agent trajectories by leveraging the LLM's parametric knowledge of task structures and evaluation criteria. On five benchmarks, ADARUBRIC achieves Pearson r=0.79 (+0.15 over the best baseline) with strong reliability (α=0.83), and produces DPO preference pairs that improve agent task success by up to +8.5%. Critically, ADARUBRIC generalises to unseen task types (SWE-bench code repair: r=0.77, +4.9% DPO gain) and accelerates PPO-based online RL training by +6.6% at 5K steps. Cross-domain transfer, zero-shot generalisation, and modality-agnostic extension demonstrate the breadth of the approach. Human evaluation of generated rubrics shows quality approaching expert-designed rubrics (relevance: 4.3/5), validating the generation mechanism. Together, these results demonstrate that adaptively eliciting task-specific evaluation criteria from LLMs is a more reliable basis for both judgment and reward learning than fixed, hand-crafted rubrics.

Limitations. Rubric quality depends on LLM capability; tasks with vague or underspecified descriptions may yield incomplete or overlapping dimensions, and rubric generation can miss specialised criteria (e.g., CAPTCHA Handling, Rate-Limit Awareness) when parametric knowledge of the domain is thin. Because rubrics are generated from the task description, adversarially crafted descriptions could in principle bias evaluation; in our pilot with 30 perturbed descriptions, r degrades by 0.04–0.06, indicating non-trivial but bounded sensitivity that motivates future work on description-robust rubric generation. Confidence scores are LLM-predicted and may require post-hoc recalibration for strongly out-of-distribution tasks. Computationally, ADARUBRIC costs $K \times N$ evaluator calls per trajectory ($\approx$40 for WebArena) with 3–5$\times$ the wall-clock of GPT-4 Direct; rubric caching amortises the generation cost across a task family but not the per-step evaluation cost. Finally, the human correlation study uses 300 pairs per benchmark with three annotators, and the multimodal/PPO/SWE-bench sections are kept deliberately compact—we treat them as generalisation evidence rather than as their own primary studies. Single-pass evaluation could be improved by multi-round verification, though at higher computational cost. All rubric, evaluation, and filter prompt templates used in this paper are released in the supplementary material to support replication.

References

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Liang Ding. 2026. Agenther: Hindsight experience replay for llm agent trajectory relabeling. *arXiv preprint arXiv:2603.21357*.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, and 1 others. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and 1 others. 2023. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 881–905.

Klaus Krippendorff. 2011. Computing krippendorff's alpha-reliability.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, Lester James Validad Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, and 1 others. 2025. Rewardbench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1755–1797.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The twelfth international conference on learning representations*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2023a. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023b. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 2511–2522.

Qingyu Lu, Liang Ding, Siyi Cao, Xuebo Liu, Kanjian Zhang, Jinxia Zhang, and Dacheng Tao. 2025. Runaway is ashamed, but helpful: On the early-exit behavior of large language model-based agents in embodied environments. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 24014–24027.

Qingyu Lu, Liang Ding, Liping Xie, Kanjian Zhang, Derek F Wong, and Dacheng Tao. 2023. Toward human-like evaluation for natural language generation with error analysis. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5892–5907.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. 2024. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

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

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G Finlayson, David Sontag, and 1 others. 2025. Dr tulu: Reinforcement learning with evolving rubrics for deep research. *arXiv preprint arXiv:2511.19399*.

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

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, and 1 others. 2023. How far can camels go? exploring the state of instruction tuning on open resources. *Advances in Neural Information Processing Systems*, 36:74764–74786.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, and 1 others. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. 2023. Flask: Fine-grained language model evaluation based on alignment skill sets. *arXiv preprint arXiv:2307.10928*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. 2023. Judgelm: Fine-tuned large language models are scalable judges. *arXiv preprint arXiv:2310.17631*.

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

A Implementation Details

Hyperparameters. Default: N=5 dimensions, recency-decay λ=0.5, minimum margin δ_{min}=0.5, DimAware threshold θ_j=2.5, percentile filter p=80.

Computational cost. ADARUBRIC runs $K \times N$ LLM calls per trajectory (plus one rubric generation call per task type). For WebArena ($K \approx 8$, N=5): 40 calls vs. 1 for GPT-4 Direct. With caching and batching, total latency is 3–5× GPT-4 Direct.

B Proof of Proposition 3.1

(a) If all $\bar{s}_j \geq \theta_j$, then $S(\tau) = \sum_j w_j \bar{s}_j \geq \sum_j w_j \theta_j = \bar{\theta}$. (b) Set $\bar{s}_{j^*} = \epsilon$ and $\bar{s}_j = (\bar{\theta} - w_{j^*}\epsilon)/(1 - w_{j^*})$ for $j \neq j^*$, giving $S(\tau^*) = \bar{\theta}$. (c) Construction generalises to any θ'.

BLUE of confidence-weighted aggregation. Under a noise model $s_{k,j} = s^*_{k,j} + \varepsilon_{k,j}$ with $\varepsilon_{k,j} \sim \mathcal{N}(0, \sigma^2/c_{k,j})$, the confidence-weighted estimator $\hat{\mu}_j = \sum_k \frac{c_{k,j}}{\sum_{k'} c_{k',j}} s_{k,j}$ is BLUE by Gauss-Markov, with $\text{Var}[\hat{\mu}_j] = \sigma^2 / \sum_k c_{k,j} \leq \sigma^2 \sum_k c_{k,j}^{-1}/K^2 = \text{Var}[\bar{s}_j^{\text{uniform}}]$ (Cauchy-Schwarz), with equality iff all $c_{k,j}$ are equal.

Empirical validation of the noise model. On 300 WebArena trajectory pairs with 3 independent evaluator runs, we compute residuals $s_{k,j} - \bar{s}_{k,j}$ and find the distribution is approximately mean-zero (bias ≤ 0.07 on the 1–5 scale) but leptokurtic (excess kurtosis $\approx$1.4) with a mild negative correlation between $|s - \bar{s}|$ and c (Spearman -0.31). The inverse-confidence scaling direction therefore holds in practice, while the Gaussian tails do not; BLUE should be treated as a motivating approximation, and the empirical variance reduction of confidence-weighting vs. uniform averaging is $18{-}24\%$ across benchmarks.

C Calibration and Error Analysis

ADARUBRIC score buckets correlate strongly with human percentile ranks (Spearman ρ=0.98, p<0.001), with rating-5 trajectories landing at the 91st human percentile on average. The near-linear relationship confirms that the 1–5 scale is meaningfully calibrated.

Manual inspection of 50 disagreement cases ($|r_{\text{ada}} - r_{\text{human}}| > 1$) on WebArena reveals three main failure modes: (1) long-horizon binary goals (32%)—ADARUBRIC over-credits partial completion in multi-hop tasks where final outcome is binary; (2) implicit domain conventions (28%)—ADARUBRIC fails to penalise violations of website-specific norms (e.g., wrong date format); (3) ambiguous observations (24%)—HTML artefacts or API error codes confuse per-step scoring. All are addressable via richer task descriptions or observation pre-processing. In 16% of cases, the human rater's judgment was arguably incorrect, and ADARUBRIC's evaluation was defensible.

D Additional Results

PPO training details. We train a Qwen2.5-7B policy with ADARUBRIC-DA as the reward model for 1,000 rollouts on WebArena training tasks. WebArena's training split (N=80 tasks) is used for online RL; the held-out 732 tasks serve as the test set. Results at 1K/3K/5K rollouts: GPT-4 Scalar: 14.2/18.7/21.3; Prometheus: 15.8/21.2/23.6; ADARUBRIC-WM: 18.3/24.9/27.1; ADARUBRIC-DA: **20.1/27.4/30.2**.

Multimodal results. Table 12 reports full multimodal evaluation results.

Table 12: **Multimodal agent evaluation.** Pearson r and DPO SR% on VisualWebArena (VWA) and OSWorld.

	VWA		OSWorld	
Method	r	SR%	r	SR%
G-Eval	0.48	—	0.44	—
GPT-4V Direct	0.62	—	0.59	—
ADARUBRIC-DA	**0.76**	**26.3**	**0.73**	**22.8**

Rubric quality human study. Five domain experts rate 60 rubrics (20 per benchmark) on 1–5 Likert scales. Generic (Helpfulness/Safety/Fluency): 2.1/3.8/1.6. Domain template: 3.9/3.6/3.7. Expert-designed: 4.6/4.4/4.5. ADARUBRIC-generated: **4.3/4.2/4.1**. Inter-annotator agreement κ=0.79.