AGENTJUDGE: FROM REAL AGENT TRAJECTORIES TO MULTI-GRANULARITY JUDGING AND AGENT–JUDGE CO-EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Local action quality, overall task performance, and evolving trajectory states are interdependent in agent execution, yet static, outcome-centered judging struggles to capture these signals jointly. We introduce **AgentJudge**, a framework for **multi-granularity judging and agent–judge co-evolution** built on real execution trajectories. We collect trajectories from search and coding tasks and derive supervision for local steps, dynamic prefixes, and final outcomes, unifying local action assessment, global outcome evaluation, and execution-state tracking. We adopt **progressive SFT–DPO–GRPO training**: multi-granularity supervised fine-tuning establishes basic judging capabilities; future-masked retrospective preference learning uses complete trajectories to identify potential misjudgments and constructs revised judgments from evidence visible in the current prefix for preference optimization; reward-based optimization further reinforces structured judging against supervision references. We further design a bidirectional update mechanism in which judge feedback guides agent optimization, while newly generated trajectories supply supervision for subsequent judge updates, forming a loop of trajectory generation, judge learning, and feedback-driven optimization. Experiments show that AgentJudge outperforms the base model on general response judging and agent trajectory judging, with gains in both agent success and judge accuracy over two co-evolution rounds.

1 INTRODUCTION

LLM-as-a-Judge provides feedback for model evaluation and optimization by interpreting task requirements, assessing response quality, and comparing candidate outputs (Zheng et al., 2023). As language models take on multi-step interactive tasks such as search and code development (Du et al., 2026; Jimenez et al., 2024), evaluation extends from standalone responses to execution trajectories comprising planning, tool use, and environmental feedback. Agent judges extend this paradigm to agent execution: they build on the semantic understanding and quality assessment capabilities of general-purpose LLM judges while modeling relationships among local actions, accumulated execution states, and final outcomes to provide process-level evaluation and feedback (Zhuge et al., 2024; Xia et al., 2025).

Toward these goals, related work has expanded from scoring and comparing general responses to evaluating tool use, local steps, and full trajectories (Choudhury, 2025; Lù et al., 2025; Wang et al., 2026). These assessments provide complementary information: a locally reasonable action can still leave a trajectory on a previously misguided path, whereas a trajectory containing local errors can recover and ultimately complete the task. Agent judging thus calls for linking local decisions to global performance and tracking execution states as more of the trajectory becomes visible. Errors, deviations, and recoveries observed in real executions provide direct learning signals for such multi-granularity, dynamic judging.

We therefore introduce **AgentJudge**, which turns real agent execution experience into multi-granularity judging capabilities. We collect trajectories from search and coding tasks to construct supervision for local steps, dynamic prefixes, and final outcomes, complemented by relative preference judgments between execution outcomes. We adopt progressive SFT–DPO–GRPO training:

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

multi-granularity supervised fine-tuning establishes basic judging capabilities; future-masked retrospective preference learning uses complete trajectories to identify potential misjudgments and constructs revised judgments grounded in evidence visible in the current prefix; reward-based optimization further reinforces structured judging against supervision references. AgentJudge thereby learns to assess local action quality, accumulated execution states, and task outcomes within a unified model.

We further integrate judge learning and agent optimization into a bidirectional update framework. Judge feedback guides improvements in agent execution, and updated agents generate new trajectories that provide further learning experience for the judge. By alternating trajectory generation, judge learning, and feedback-driven optimization, the framework translates judging improvements into better agent execution while using new execution experience to support subsequent judge updates, forming a mutually reinforcing mechanism for agent–judge co-evolution.

Our main contributions are as follows:

1. **We construct AgentJudge-19K, a real-trajectory-driven multi-granularity judging dataset.** It contains 19,106 judging records derived from real search and coding agent trajectories, covering step-level, dynamic-prefix-level, and final-outcome-level supervision together with pairwise preference judgments for local, dynamic, and global judging.
2. **We train AgentJudge for multi-granularity dynamic judging.** Compared with conventional judge models focused on response quality and final outcomes, AgentJudge uses progressive learning to extend judging to local action assessment, dynamic trajectory-state tracking, and global outcome evaluation. Experiments show improvements over the base model across multiple benchmarks for general response judging and agent trajectory judging.
3. **We propose a framework for agent–judge co-evolution.** The framework unifies judge learning and agent optimization in a closed loop, continually converting execution experience into multi-granularity supervision. This allows real-trajectory supervision to support both agent-specific and general-purpose judging, while feedback-driven optimization promotes the co-evolution of agents and judges.

2 Related Work

Judging data and process supervision. Judging data span response quality, reasoning steps, and agent execution. UltraFeedback (Cui et al., 2023) uses model feedback to construct multidimensional response evaluation data, PRM800K (Lightman et al., 2024) provides human correctness annotations for mathematical reasoning steps, and AgentRewardBench (Lù et al., 2025) provides expert annotations of task success, side effects, and repetitive behavior in web-agent trajectories for evaluating automated judges. We jointly derive step-level, dynamic-prefix-level, and final-outcome-level supervision from real search and coding trajectories, complemented by pairwise preference judgments. These data organize execution experience into multi-granularity training signals connecting local actions, accumulated execution states, and task outcomes.

LLM judges and agent judges. LLM-as-a-Judge (Zheng et al., 2023) uses language models to assess response quality. JudgeLM (Zhu et al., 2025b) trains specialized judges on judgments produced by strong models, while Prometheus 2 (Kim et al., 2024) supports direct scoring and pairwise comparison under custom evaluation criteria. Agent judges retain the principle of assessing quality against task requirements and semantic evidence, while extending evaluation to actions, environmental feedback, and their cumulative effects. Agent-as-a-Judge (Zhuge et al., 2024) evaluates requirement fulfillment by inspecting execution artifacts and trajectories, while AgentRM (Xia et al., 2025) learns reward models to guide candidate selection and step-level search. We use progressive learning to unify local action, dynamic prefix, and final outcome assessment. Future-masked retrospective preference learning further uses later execution to identify potential misjudgments and construct revised judgments grounded in currently visible evidence.

Judge feedback and co-evolution. Reflexion (Shinn et al., 2023) improves subsequent execution through verbal feedback and reflective memory. Self-Rewarding Language Models (Yuan

et al., 2024) uses self-generated evaluations for iterative preference optimization, while Meta-Rewarding (Wu et al., 2025) evaluates judgments themselves to improve judging capability. In agent settings, AgentPRM (Choudhury, 2025) estimates process-reward targets through interaction sampling and uses them to optimize the policy. We build a bidirectional learning loop around real search and coding execution: judge feedback supports agent improvement, and newly generated trajectories become supervision for subsequent judge updates, linking agent optimization with judge learning.

3 METHOD

AgentJudge transforms real agent execution experience into multi-granularity judging supervision, learns process and outcome assessment through progressive training, and uses judge feedback for agent optimization. As shown in Figure 1, the framework comprises real-trajectory collection, multi-granularity judge learning, and bidirectional agent–judge updates.

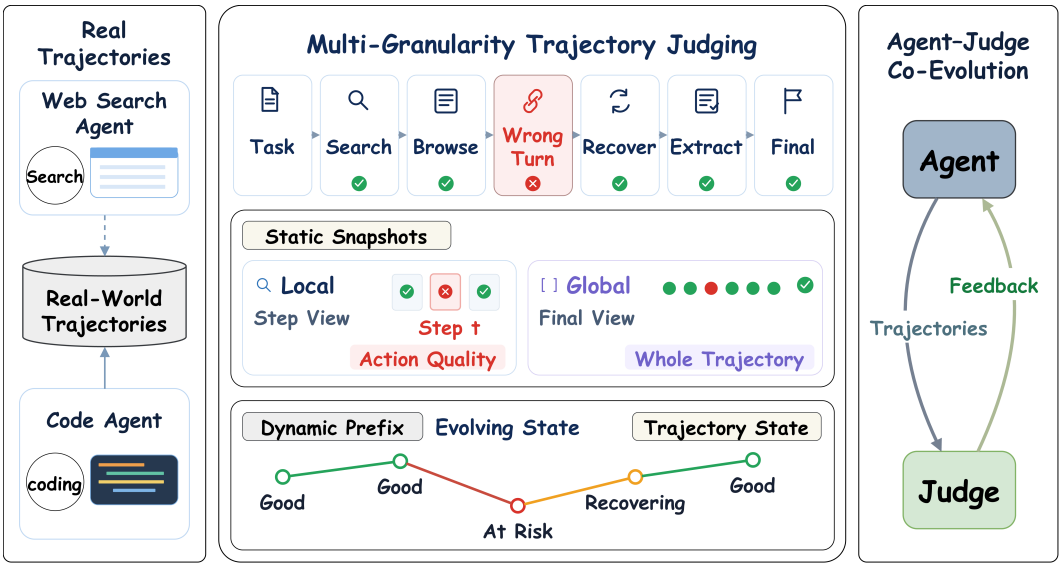

Figure 1: **AgentJudge overview. Left:** real execution trajectories are collected from search and coding agents. **Center:** an example search trajectory illustrates deviation and recovery. Its steps appear at the top; local and global snapshots assess action quality and overall task performance in the middle; dynamic-prefix judging tracks changes in execution state at the bottom. **Right:** agent-generated trajectories support judge learning, while judge feedback guides agent optimization, forming a bidirectional update loop.

3.1 REAL TRAJECTORIES AND MULTI-GRANULARITY SUPERVISION

For a task q, let $\tau = (s_1, \ldots, s_T)$ denote an execution trajectory, where s_t contains the action and corresponding execution feedback at step t, and $\tau_{\le t}$ is the prefix through that step. We collect search and coding trajectories from public execution records and our own agent runs to construct step-level, dynamic-prefix-level, and final-outcome-level supervision, complemented by pairwise judgments. Multi-step trajectories support process and outcome assessment, while constructed single-turn examples primarily support final-outcome assessment.

Local-step and dynamic-prefix judging. These tasks assess current action quality and accumulated execution state, respectively. Let ι_{step} and ι_{prefix} denote their judging instructions. The inputs are

$$x_t^{\text{step}} = (q, \tau_{<t}, s_t; \iota_{\text{step}}), \qquad x_t^{\text{prefix}} = (q, \tau_{\le t}; \iota_{\text{prefix}}). \tag{1}$$

Step-level judging assesses tool selection, argument use, local decisions, and their contribution to task progress. Dynamic-prefix judging integrates prior decisions and environmental feedback to assess the current execution state. Both use records up to the same time point but evaluate different

targets: a reasonable current action may leave an earlier deviation unresolved. As new steps become available, dynamic judgments reflect deviation, recovery, and continued progress, as illustrated in the center of Figure 1. Some training records additionally provide a task-level reference answer; Appendix B.1 describes the input organization.

Final-outcome and pairwise judging. Final-outcome supervision assesses the correctness, completeness, and requirement fulfillment of final outputs; pairwise supervision compares execution outcomes for the same task. Denoting the four datasets by $\mathcal{D}_{\text{step}}$, $\mathcal{D}_{\text{prefix}}$, $\mathcal{D}_{\text{final}}$, and $\mathcal{D}_{\text{pair}}$, we obtain

$$\mathcal{D}_{\text{SFT}} = \text{Clean}(\mathcal{D}_{\text{step}} \cup \mathcal{D}_{\text{prefix}} \cup \mathcal{D}_{\text{final}} \cup \mathcal{D}_{\text{pair}}), \tag{2}$$

where Clean standardizes data formats and filters examples for quality. After transformation and filtering, the main SFT corpus contains 19,106 records; we refer to it as **AgentJudge-19K**. The dataset organizes local actions, accumulated states, final outcomes, and relative preferences into shared judging supervision.

Figure 2 shows how real trajectories yield three types of training signals: the SFT branch supplies multi-granularity reference judgments, the DPO branch constructs retrospective judgment preferences, and the GRPO branch reuses existing supervision for reward-based optimization.

Figure 2. From Trajectories to Training Data

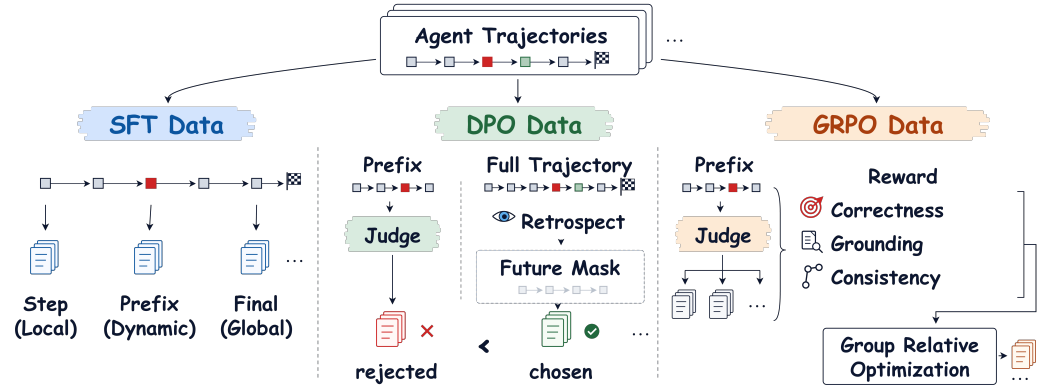

Figure 2: **Constructing judging supervision from trajectories. Left:** local steps, dynamic prefixes, and final outcomes provide multi-granularity SFT data. **Center:** an initial prefix-based judgment is reviewed retrospectively using the complete trajectory. Future-masked rewriting and validation produce a revised judgment supported by prefix-visible evidence, which is paired with the initial judgment for DPO. **Right:** groups of candidate judgments are sampled for the same input and scored against structured supervision references for group-relative optimization.

3.2 Future-Masked Retrospective Preference Learning

Complete trajectories contain subsequent execution and final outcomes, providing additional cues for reviewing earlier judgments. We introduce future-masked retrospective preference learning to turn misjudgment cues from complete executions into supervision for prefix-based judging.

Prefix-only review and global retrospection. Let x_t contain the task, current prefix, and judging instruction. An initial judge J_{loc} generates a judgment y_t^0. A retrospective operation $\mathcal{R}$ then reviews it using the complete trajectory and produces diagnostic information d_t. A teacher model applies a future-masked rewriting operation $\mathcal{M}$ to generate a candidate revision $\widetilde{y}_t$:

$$y_t^0 \sim J_{\text{loc}}(\cdot \mid x_t), \qquad d_t = \mathcal{R}(x_t, \tau, y_t^0), \qquad \widetilde{y}_t = \mathcal{M}(x_t, y_t^0, d_t). \tag{3}$$

Retrospection identifies evidence overlooked or misinterpreted in the initial judgment. Rewriting reconstructs the verdict and explanation using evidence already visible in the current prefix.

Dual validation and preference construction. Two validators independently review each candidate revision. Let $v_{j,t} \in \{0, 1\}$ indicate acceptance by validator j. We retain records accepted by

both validators to construct

$$\mathcal{D}_{\text{DPO}} = \left\{ (x_t, \widetilde{y}_t, y_t^0) \mid v_{1,t} = v_{2,t} = 1 \right\}. \tag{4}$$

For each retained example, the revised judgment is $y_t^+ = \widetilde{y}_t$ and the initial judgment is $y_t^- = y_t^0$. Both share the same prefix input. Preference improvements include verdict correction, reduced confidence, and evidence refinement that preserves the original verdict.

Thus, pairwise SFT learns relative quality across execution outcomes, whereas retrospective DPO learns relative quality across judgments of the same visible evidence, translating retrospective insights into judging capability during execution.

3.3 Progressive Judge Training

As shown in Figure 3, AgentJudge undergoes SFT, DPO, and GRPO in sequence to establish multi-granularity judging capabilities, learn validated judgment preferences, and reinforce existing evaluation objectives.

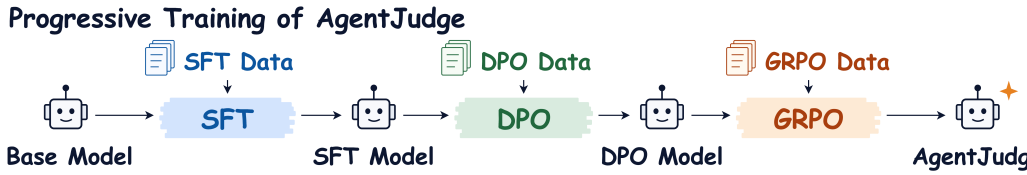

Figure 3: **Progressive training of AgentJudge.** **SFT** learns local-action, dynamic-prefix, and final-outcome assessment, together with relative preferences, from multi-granularity data. **DPO** then learns validated retrospective judgment preferences, and **GRPO** reinforces structured judging against supervision references. Each stage initializes from the preceding checkpoint, yielding the final AgentJudge model.

Multi-granularity supervised fine-tuning. Let $J_\theta(y \mid x)$ denote the judge with parameters θ, where x specifies the evaluation target and y is the reference judgment. SFT minimizes the negative log-likelihood (Ouyang et al., 2022) on the mixed multi-granularity data:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{SFT}}} \left[\log J_\theta(y \mid x) \right]. \tag{5}$$

This stage unifies local-action, dynamic-prefix, final-outcome, and pairwise assessment in one model, establishing a common foundation for subsequent preference learning.

Retrospective preference optimization. After supervised fine-tuning, we use $\mathcal{D}_{\text{DPO}}$ to increase the model's relative preference for revised judgments. With a fixed reference model J_{ref}, the objective (Rafailov et al., 2023) is

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(x,y^+,y^-)\sim\mathcal{D}_{\text{DPO}}} \log \sigma \left(\beta \left[\log \frac{J_\theta(y^+ \mid x)}{J_{\text{ref}}(y^+ \mid x)} - \log \frac{J_\theta(y^- \mid x)}{J_{\text{ref}}(y^- \mid x)} \right] \right) \tag{6}$$

where σ is the sigmoid function and $\beta > 0$ is a scaling coefficient. The objective directly contrasts validated revisions with initial judgments, teaching the model to distinguish judgment quality under the same prefix.

Reward optimization with supervision references. GRPO (Shao et al., 2024) organizes existing supervision into judging inputs x and reward references ρ. For each input, the model generates a group of candidate judgments and scores their agreement with the reference under the corresponding output schema:

$$y_i \sim J_{\theta_{\text{old}}}(\cdot \mid x), \qquad r_i = R(x, y_i; \rho), \qquad i = 1, \ldots, G. \tag{7}$$

Here, G is the group size and $J_{\theta_{\text{old}}}$ is the policy used to generate the current group. Within-group relative rewards guide policy optimization to reinforce structured judgment accuracy. Appendix B details the reward references, scoring rules, and group-relative objective.

3.4 AGENT–JUDGE CO-EVOLUTION

The right side of Figure 1 connects agent execution and judge learning through bidirectional updates. Let A_k and J_k denote the agent and judge at iteration k, and let $\mathcal{Q}_k$ be the task set for trajectory sampling in that iteration. We express the mechanism through the following framework-level updates:

$$A_{k+1} = \mathcal{U}_A(A_k; J_k), \qquad \mathcal{T}_{k+1} = \mathrm{Rollout}(A_{k+1}, \mathcal{Q}_k), \qquad J_{k+1} = \mathcal{U}_J(J_k; \Phi(\mathcal{T}_{k+1})). \tag{8}$$

Here, $\mathcal{U}_A$ updates the agent using judge feedback, $\mathrm{Rollout}$ executes tasks and collects the new trajectory set $\mathcal{T}_{k+1}$, Φ constructs supervision following Sections 3.1–3.2, and $\mathcal{U}_J$ updates the judge using this supervision.

For agent optimization, J_k provides feedback on local actions, execution states, and final outcomes to guide improvements in A_k. For judge updates, new trajectories yield multi-granularity supervision, while misjudgments exposed during execution become preference data through the retrospective pipeline. By alternating feedback-driven optimization, trajectory generation, and judge updates, the framework incorporates new execution experience into a unified judge and uses its judging capabilities for subsequent agent optimization, forming a bidirectional learning loop.

4 EXPERIMENTS

We evaluate AgentJudge through four questions:

- **Can real-trajectory supervision improve agreement with human preferences and strong-model judgments, and support broader applications such as reward evaluation?**
- **Can AgentJudge connect local behavior to overall execution quality and recognize cross-step patterns such as recovery and repetitive stagnation?**
- **How do supervised fine-tuning, retrospective preference learning, and reward optimization contribute to multi-granularity judging?**
- **Can agents and judges improve together, and how do bidirectional updates compare with updating either component alone?**

4.1 EXPERIMENTAL SETUP

Models and training. AgentJudge starts from Qwen3-VL-8B-Instruct and follows SFT–DPO–GRPO training. SFT uses AgentJudge-19K (19,106 records); DPO uses 311 preference pairs with $\beta = 0.1$. All stages use LoRA (Hu et al., 2021) and run for one, three, and three epochs, respectively. GRPO scores groups of judgments against structured references. Hardware and hyperparameters appear in Appendix B.6.

Baselines. Smaller baselines cover 7–14B models from Qwen3-VL (Bai et al., 2025), GLM-4 (Glm et al., 2024), LLaVA-1.5 (Liu et al., 2024b), and InternVL3 (Zhu et al., 2025a), alongside Flex-VL-7B (Ko et al., 2026). Larger baselines include GLM-4-32B-0414, GLM-4.5-Air-106B, Qwen2.5-72B-Instruct (Qwen et al., 2025), Qwen3-VL-235B-A22B (Bai et al., 2025), Hunyuan-A13B-80B (Team et al., 2026), and 671B DeepSeek models (Liu et al., 2024a; Guo et al., 2025; Liu et al., 2025). JudgeLM-7B (Zhu et al., 2025b) and PandaLM-7B (Wang et al., 2024) provide specialized-judge comparisons.

Benchmarks and metrics. PandaLM (Wang et al., 2024) and JudgeLM (Zhu et al., 2025b) assess agreement with human preferences and GPT-4 judgments, respectively. RewardBench (Lambert et al., 2025) tests reward-preference evaluation as a broader application. Plan-RewardBench (Wang et al., 2026) and AgentRewardBench (Lù et al., 2025) assess trajectory preferences and execution behaviors. Captions in Tables 1–3 define the metrics. Co-evolution tests compare joint updates with fixed-judge and fixed-agent branches, reporting success, judge accuracy, and mean tool calls. Scores are percentages; gains are percentage points.

Table 1: **General response judging and reward evaluation as a broader application. PandaLM** measures agreement with human response preferences, while **JudgeLM** measures agreement with GPT-4 reference judgments. Both report agreement, precision, recall, and F1. **RewardBench** evaluates reward preferences across dialogue, safety, and reasoning. Comparisons include similarly sized general-purpose models, larger models, and specialized judges. AgentJudge uses an 8B base model. Agr., P, and R denote agreement, precision, and recall. All scores are percentages; higher is better.

	PandaLM Test Set				JudgeLM Validation				RewardBench
Model	Agr.	P	R	F1	Agr.	P	R	F1	Score
LLaVA-1.5-7B	37.13	40.11	39.24	28.98	40.80	43.11	41.16	34.34	42.07
JudgeLM-7B	66.77	63.83	71.95	63.92	78.00	66.02	67.61	66.66	46.57
PandaLM-7B	65.82	44.56	48.93	46.24	67.42	44.99	48.53	46.66	53.00
Flex-VL-7B	71.34	64.30	62.18	63.00	77.44	64.82	57.49	57.42	75.55
Qwen3-VL-8B	70.05	63.69	64.80	63.79	72.15	58.77	58.78	58.72	78.69
GLM-4-9B	70.02	69.86	57.69	59.55	72.42	59.06	53.31	52.54	68.24
LLaVA-1.5-13B	29.89	43.16	41.67	28.28	48.59	41.61	40.45	39.69	40.28
InternVL3-14B	66.67	65.07	68.67	62.27	71.96	62.26	64.69	61.23	75.23
Qwen3-VL-235B-A22B	69.74	65.20	69.39	64.45	72.54	62.91	66.32	62.28	**89.30**
DeepSeek-V3-671B	64.76	58.48	61.13	58.83	72.04	59.38	60.37	59.75	83.34
AgentJudge-8B	**73.39**	**84.64**	**73.42**	**78.58**	**79.12**	**85.56**	**79.15**	**82.01**	82.55

4.2 Generalization and Broader Applications of Judging

Table 1 tests whether judging learned from real execution experience generalizes across evaluation standards and task formats. AgentJudge achieves F1 scores of **78.58** on PandaLM and **82.01** on JudgeLM, improving over the base model by **14.79** and **23.29** points. It also exceeds the specialized JudgeLM-7B by 14.66 and 15.35 points. Agreement reaches 73.39 and 79.12, the highest values in the table. On RewardBench, AgentJudge scores **82.55**, a 3.86-point gain over the base model and close to DeepSeek-V3-671B at 83.34. These results show that trajectory supervision improves alignment with human preferences and agreement with strong-model judgments. Its benefits also extend to reward evaluation, demonstrating generalization across evaluation standards and task formats.

4.3 Agent Trajectory Judging

Table 2: **Full-trajectory preference evaluation on Plan-RewardBench.** The **planning subsets** s-easy and s-hard cover easy and hard single-turn planning; m-easy and m-hard cover their multi-turn counterparts. **Recovery** evaluates recovery after tool errors, empty results, or partial failures. **Tool-IR** covers decisions when tools are irrelevant or unavailable. **Safety** distinguishes safe refusals from unsafe execution. Each subset reports accuracy in selecting the reference-preferred trajectory. Avg. is the mean of the seven subset scores. Higher is better.

Model	s-easy	s-hard	m-easy	m-hard	Recovery	Tool-IR	Safety	Avg.
Flex-VL-7B	72.60	70.90	70.20	61.00	74.10	**71.80**	55.90	68.06
GLM-4-32B-0414	75.69	70.99	68.58	68.49	69.44	60.18	54.41	66.83
Qwen2.5-72B-Instruct	**84.85**	77.22	71.72	**73.68**	**76.62**	63.26	48.00	70.76
Hunyuan-A13B-80B	72.57	71.20	64.47	65.00	70.31	58.03	51.47	64.72
GLM-4.5-Air-106B	75.87	69.15	69.72	65.41	69.81	60.09	47.06	65.30
DeepSeek-R1-671B	83.10	72.00	66.84	71.43	73.04	66.24	48.86	68.79
DeepSeek-V3.2-671B	80.65	76.58	67.82	63.46	74.01	66.03	30.23	65.54
AgentJudge-8B	81.60	**77.85**	**76.15**	71.92	76.18	66.36	**80.39**	**75.78**

Overall trajectory quality and local execution behavior. Table 2 tests whether AgentJudge integrates local tool behavior, task constraints, and cross-step execution into overall trajectory judgments. Its average accuracy reaches **75.78**, exceeding Qwen2.5-72B-Instruct, the strongest listed baseline at 70.76, by **5.02** points. AgentJudge leads on s-hard, m-easy, and Safety with scores of **77.85**, **76.15**, and **80.39**, respectively, and scores 76.18 on Recovery. These results cover single-turn and multi-turn planning, safety constraints, and adjustment after errors. They support incorporating

local actions and their execution effects into global assessment, connecting specific behavior quality to overall task performance.

Table 3: **Multi-dimensional trajectory judging on AgentRewardBench. Success P, SideEffect P, and Looping P** report positive-class precision for task success, side effects, and repetitive behavior without progress, respectively. **Opt. Acc.** is trajectory-optimality classification accuracy. **Overall** is the mean of these four scores. The dimensions assess goal completion, execution effects, repetitive stagnation, and overall execution quality. Higher is better.

Model	Success P	SideEffect P	Looping P	Opt. Acc.	Overall
Flex-VL-7B	**82.30**	7.20	68.40	40.30	49.60
GLM-4-32B-0414	72.80	7.10	84.70	50.80	53.90
Qwen2.5-72B-Instruct	66.70	**12.10**	78.30	50.50	51.90
Hunyuan-A13B-80B	58.60	8.80	81.50	38.80	46.90
GLM-4.5-Air-106B	74.14	8.40	81.32	41.54	51.35
DeepSeek-R1-671B	70.50	10.50	79.00	49.70	52.40
DeepSeek-V3.2-671B	80.70	9.40	77.60	47.70	53.90
AgentJudge-8B	69.00	5.60	**91.70**	**54.50**	**55.20**

Task outcomes and cross-step states. Table 3 examines both final task outcomes and behavioral patterns during execution. AgentJudge scores 69.00, 5.60, **91.70**, and **54.50** on Success P, SideEffect P, Looping P, and Opt. Acc., respectively. Its Overall score of **55.20** exceeds the best baseline score of 53.90, shared by GLM-4-32B-0414 and DeepSeek-V3.2-671B, by **1.30** points. Looping precision improves over the best baseline by **7.00** points, and optimality accuracy is also the highest in the table. Together with the recovery results in Table 2, these findings show an ability to distinguish effective adjustment from persistent stagnation, linking outcome assessment with an understanding of execution processes.

4.4 Ablation of Progressive Training

Table 4: **Effects of progressive training on trajectory preference evaluation. Base** is the base model, and **Full** uses the complete SFT–DPO–GRPO sequence. **w/o SFT, w/o DPO, and w/o GRPO** omit the indicated stage. The seven subsets follow Table 2, and Avg. is their mean. The average compares complete and reduced training configurations; subset scores show changes in planning, recovery, tool selection, and safety judgments.

Setting	s-easy	s-hard	m-easy	m-hard	Recovery	Tool-IR	Safety	Avg.
Base	77.43	75.50	73.39	75.36	71.72	62.18	39.00	67.80
w/o SFT	78.82	80.38	72.48	73.97	72.99	71.64	59.80	72.87
w/o DPO	77.78	**81.21**	69.72	**76.06**	72.82	71.09	56.86	72.22
w/o GRPO	80.21	79.11	72.02	75.34	73.41	**71.70**	60.78	73.23
Full	**81.60**	77.85	**76.15**	71.92	**76.18**	66.36	**80.39**	**75.78**

Table 4 tests the joint contribution of supervised learning, retrospective preference optimization, and reward-based reinforcement. Full raises average accuracy from **67.80 to 75.78**, a **7.98**-point gain. Omitting SFT, DPO, or GRPO reduces the average to 72.87, 72.22, or 73.23, respectively, corresponding to drops of **2.91**, **3.56**, and **2.55** points. Full reaches 76.18 on Recovery, above the 72.82–73.41 range of the stage-omission variants, and 80.39 on Safety. These results support complementary roles for the three stages. Multi-granularity supervision establishes the evaluation foundation, retrospective preferences refine evidence-based judgments, and reward optimization reinforces structured assessment. Together, they improve judgments of process behavior and overall trajectory quality.

Table 5 shows that Full improves Overall from **51.70 to 55.20**, outperforming all stage-omission variants. Relative to Base, success precision, looping precision, and optimality accuracy increase by **4.10**, **5.90**, and **3.90** points, respectively. These gains indicate that progressive training improves complementary outcome- and process-level judgments within a single model.

Table 5: **Effects of progressive training on multi-dimensional trajectory judgments.**

Setting	Success P	SideEffect P	Looping P	Opt. Acc.	Overall
Base	64.90	5.30	85.80	50.60	51.70
w/o SFT	64.90	**7.50**	91.20	49.40	53.20
w/o DPO	66.90	6.20	**93.20**	49.80	54.00
w/o GRPO	66.10	7.10	89.70	50.90	53.50
Full	**69.00**	5.60	91.70	**54.50**	**55.20**

4.5 AGENT–JUDGE CO-EVOLUTION

Table 6: **Bidirectional versus one-sided updates. (a)** Agent and judge are updated together. **(b)** Only the agent is updated with fixed J_0. **(c)** Only the judge is updated with fixed R0. Success, accuracy, and mean tool calls are reported per task.

State	Agent Success Rate	Judge Accuracy	Mean Tool Calls / Task
(a) Bidirectional updates			
R0	20.83%	82.81%	22.50
R1	29.17%	84.38%	22.58
R2	**41.67%**	**88.28%**	23.83
(b) Fixed judge J_0			
R0	20.83%	82.81%	22.50
R1	25.00%	82.81%	23.96
R2	8.33%	82.81%	24.17
(c) Fixed agent R0			
J_0	20.83%	82.81%	22.50
J_1	20.83%	**85.16%**	22.50
J_2	20.83%	84.77%	22.50

Bidirectional updates. Table 6(a) tracks both components when they are updated together. After two rounds, agent success increases from **20.83% to 41.67%**, while judge accuracy rises from **82.81% to 88.28%**. Both metrics improve at each round. Mean tool calls change from 22.50 to 23.83. This pattern shows that execution and judging improve together across the two iterations. **One-sided updates.** With J_0 fixed, R1 raises agent success from **20.83% to 25.00%**, but the next R2 candidate falls to 8.33%; judge accuracy remains unchanged. With R0 fixed, J_1 improves judge accuracy from **82.81% to 85.16%**. The next candidate J_2 reaches 84.77%, below the retained J_1, and is therefore not used to replace it. These controls show that either component can improve while the other is fixed, but successive one-sided proposals do not necessarily continue that gain.**Co-evolution.** Across the three branches, the bidirectional setting is the one that improves both agent success and judge accuracy over both rounds. The fixed-side controls isolate the gains available from updating the agent or the judge alone and show that these gains need not persist across successive proposals. This comparison is consistent with the co-evolution design: judge feedback improves agent execution, while new execution experience supplies the next source of judge supervision.

5 CONCLUSION

We present AgentJudge, a unified framework for agent judging from real execution trajectories. By combining multi-granularity supervision with progressive SFT–DPO–GRPO training, AgentJudge supports local, dynamic, and global trajectory evaluation within one model. Experiments show gains on general judging, trajectory judging, and agent–judge co-evolution, suggesting that real trajectories provide an effective foundation for both specialized and broader judging capabilities.

AI USE STATEMENT

We used generative AI tools during manuscript preparation for language polishing, translation, and drafting assistance. All AI-assisted content was reviewed, revised, and verified by the authors, who take full responsibility for the final wording, technical content, and research claims. Generative models and judge models used as components of the research methodology are described separately in the paper; this statement concerns only generative-AI assistance in preparing the manuscript.

ETHICS STATEMENT

This study uses public benchmarks, open-source resources, and agent trajectories generated in search and coding task environments. It does not involve human subjects or target the collection of private personal information. Because automated judges can produce incorrect evaluations and agents may propagate inaccurate feedback, the reported results should not be interpreted as reliability guarantees for safety-critical deployment.

REPRODUCIBILITY STATEMENT

We report the trajectory sources, multi-granularity data construction pipeline, SFT–DPO–GRPO training objectives, and major training and evaluation settings in the main paper and appendix. The final supplementary material will additionally provide prompts, preprocessing details, key hyperparameters, and implementation configurations. Where permitted by the licenses of the underlying resources, we will also provide processed data and code needed to reproduce the experiments.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Sanjiban Choudhury. Process reward models for llm agents: Practical framework and directions. *arXiv preprint arXiv:2502.10325*, 2025.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, et al. Ultrafeedback: Boosting language models with scaled ai feedback. *arXiv preprint arXiv:2310.01377*, 2023.

Yuwen Du, Rui Ye, Shuo Tang, Xinyu Zhu, Yijun Lu, Yuzhu Cai, and Siheng Chen. Openseeker: Democratizing frontier search agents by fully open-sourcing training data. *arXiv preprint arXiv:2603.15594*, 2026.

James Ferguson, Matt Gardner, Hannaneh Hajishirzi, Tushar Khot, and Pradeep Dasigi. Iirc: A dataset of incomplete information reading comprehension questions. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 1137–1147, 2020.

Team Glm, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

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. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4334–4353, 2024.

Jongwoo Ko, Sungnyun Kim, Sungwoo Cho, and Se-Young Yun. Flex-judge: Text-only reasoning unleashes zero-shot multimodal evaluators. *Advances in Neural Information Processing Systems*, 38:108720–108755, 2026.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1755–1797, 2025.

Zhuofeng Li, Dongfu Jiang, Xueguang Ma, Haoxiang Zhang, Ping Nie, Yuyu Zhang, Kai Zou, Jianwen Xie, Yu Zhang, and Wenhu Chen. Openresearcher: A fully open pipeline for long-horizon deep research trajectory synthesis. *arXiv preprint arXiv:2603.20278*, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26286–26296. IEEE, 2024b.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stańczak, Peter Shaw, Christopher J Pal, and Siva Reddy. Agentrewardbench: Evaluating automatic evaluations of web agent trajectories. *arXiv preprint arXiv:2504.08942*, 2025.

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

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

MiroMind Team, Song Bai, Lidong Bing, Carson Chen, Guanzheng Chen, Yuntao Chen, Zhe Chen, Ziyi Chen, Jifeng Dai, Xuan Dong, et al. Mirothinker: Pushing the performance boundaries of open-source research agents via model, context, and interactive scaling. *arXiv preprint arXiv:2511.11793*, 2025.

Tencent Hunyuan Team, Ao Liu, Botong Zhou, Can Xu, Chayse Zhou, ChenChen Zhang, Chengcheng Xu, Chenhao Wang, Decheng Wu, Dengpeng Wu, Dian Jiao, Dong Du, Dong Wang, Feng Zhang, Fengzong Lian, Guanghui Xu, Guanwei Zhang, Hai Wang, Haipeng Luo, Han Hu, Huilin Xu, Jiajia Wu, Jianchen Zhu, Jianfeng Yan, Jiaqi Zhu, Jihong Zhang, Jinbao Xue, Jun Xia, Junqiang Zheng, Kai Liu, Kai Zhang, Kai Zheng, Kejiao Li, Keyao Wang, Lan Jiang, Lixin Liu, Lulu Wu, Mengyuan Huang, Peijie Yu, Peiqi Wang, Qian Wang, Qianbiao Xiang, Qibin Liu, Qingfeng Sun, Richard Guo, Ruobing Xie, Saiyong Yang, Shaohua Chen, Shihui Hu, Shuai Li, Shuaipeng Li, Shuang Chen, Suncong Zheng, Tao Yang, Tian Zhang, Tinghao Yu, Weidong Han, Weijie Liu, Weijin Zhou, Weikang Wang, Wesleye Chen, Xiao Feng, Xiaoqin Ren, Xingwu Sun, Xiong Kuang, Xuemeng Huang, Xun Cao, Yanfeng Chen, Yang Du, Zhen Yang, Yangyu Tao, Yaping Deng, Yi Shen, Yigeng Hong, and Yiqi Chen. Hunyuan-a13b technical report, 2026. URL https://arxiv.org/abs/2609.27284.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Jiaxuan Wang, Yulan Hu, Wenjin Yang, Zheng Pan, Xin Li, and Lan-Zhe Guo. Aligning agents via planning: A benchmark for trajectory-level reward modeling. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23174–23200, 2026.

Yidong Wang, Zhuohao Yu, Wenjin Yao, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, et al. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. In *International Conference on Learning Representations*, volume 2024, pp. 43573–43593, 2024.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason E Weston, and Sainbayar Sukhbaatar. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 11548–11565, 2025.

Yu Xia, Jingru Fan, Weize Chen, Siyu Yan, Xin Cong, Zhong Zhang, Yaxi Lu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Agentrm: Enhancing agent generalization with reward modeling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19277–19290, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

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

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

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025a.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. Judgelm: Fine-tuned large language models are scalable judges. In *International Conference on Learning Representations*, volume 2025, pp. 51257–51296, 2025b.

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, et al. Agent-as-a-judge: Evaluate agents with agents. *arXiv preprint arXiv:2410.10934*, 2024.

APPENDIX CONTENTS

A Corpus and Supervision Statistics 15
A.1 Source Composition and Tool-Use Distribution 15
A.2 Supervision Composition and Preference Labels 16
B Training Objectives and Reward Implementation 17
B.1 Judging Inputs and Supervision Targets 17
B.2 Supervised Fine-Tuning 17
B.3 Retrospective Preference Optimization 17
B.4 Reward References and Structured Scoring 18
B.5 Group-Relative Policy Optimization 18
B.6 Training Configuration 19
C Examples of Training Supervision 20
C.1 Pointwise Trajectory-State Supervision 20
C.2 Pairwise Trajectory Supervision 21
C.3 Prefix-Grounded Risk Refinement 22
C.4 Trajectory-Preference Reward Reference 23
D Qualitative Judging Comparisons 24
D.1 General Response Judging 24
D.2 Agent Trajectory Judging 26

A Corpus and Supervision Statistics

Figure 4 summarizes source provenance and tool use, while Figure 5 describes the transformed supervision. Source records and derived judging examples are reported separately because they use different counting units.

A.1 Source Composition and Tool-Use Distribution

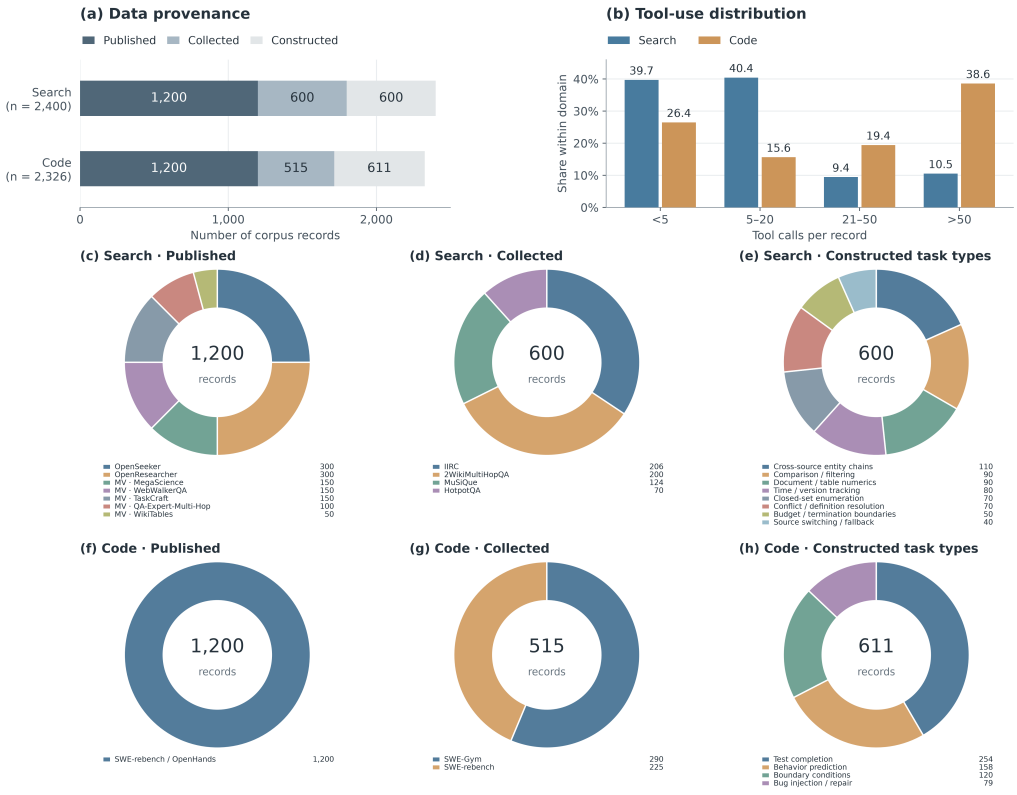

Figure 4: **Source-corpus composition and tool-use distribution.** **(a)** Search and code contain 2,400 and 2,326 records, respectively, partitioned into published, collected, and constructed sources. **(b)** Tool-call bins show within-domain proportions using the data-construction counter; all 611 constructed code records have zero tool calls. **(c)–(e)** The search corpus is broken down by published source (Du et al., 2026; Li et al., 2026), collected source (Ferguson et al., 2020; Ho et al., 2020; Trivedi et al., 2022; Yang et al., 2018), and constructed task type. **(f)–(h)** Corresponding code breakdowns include SWE-rebench/OpenHands for published records and SWE-Gym and SWE-rebench for collected records. Ring centers give record totals, and legends list source or task-type counts; MV denotes MiroVerse (Team et al., 2025). These statistics describe the frozen source corpus rather than the final mixture of transformed and filtered training examples.

A.2 Supervision Composition and Preference Labels

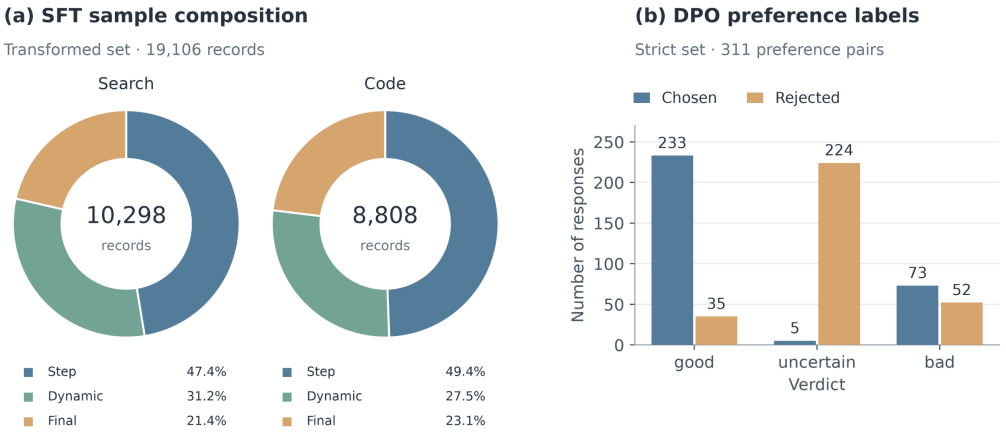

Figure 5: **Supervised fine-tuning composition and preference-verdict distributions. Left (a):** the displayed transformed set contains 19,106 records: 10,298 from search and 8,808 from code. Step-level, dynamic-prefix-level, and final-outcome-level examples account for 47.4%, 31.2%, and 21.4% of search records, and 49.4%, 27.5%, and 23.1% of code records, respectively. **Right (b):** verdicts within chosen and rejected judgments are counted separately for the strict set of 311 direct preference optimization (DPO) pairs. Each side contains 311 judgments, categorized as `good`, `uncertain`, or `bad`. These are judgment counts, not counts of pairs in each category. A chosen judgment is preferred for its relative quality; it need not assign a positive verdict to the evaluated behavior.

B Training Objectives and Reward Implementation

This section expands the supervised fine-tuning, retrospective preference optimization, and group-relative policy optimization objectives in the main text, connecting judging inputs, output supervision, and reward computation. Following the main-text notation, $J_\theta(y \mid x)$ denotes the conditional probability of judgment y given input x under judge parameters θ.

B.1 Judging Inputs and Supervision Targets

For task q, write the execution trajectory as $\tau = (s_1, \ldots, s_T)$ with $s_t = (a_t, o_t)$, where a_t contains an action and its arguments and o_t is the corresponding observation. Let ι_m specify the evaluation task and output schema. The input views are

$$\begin{aligned} x_t^{\text{step}} &= (q, \tau_{<t}, s_t; \iota_{\text{step}}), \\ x_t^{\text{prefix}} &= (q, \tau_{\le t}; \iota_{\text{prefix}}), \\ x^{\text{final}} &= (q, \tau; \iota_{\text{final}}). \end{aligned} \tag{B.1}$$

Step-level and dynamic-prefix judging both use actions and observations through the current step. Step-level judging assesses the current action–observation unit in context, while dynamic-prefix judging assesses the accumulated execution state. Final-outcome judging evaluates task completion using the full execution record. Pairwise inputs contain two candidate execution records for the same task together with a comparison instruction. Training records that supply a task-level reference answer retain it as additional evaluation information in the input.

The target judgment y follows the record's output schema and may contain a verdict, a rationale, a risk level, and other structured fields. Appendix C illustrates pointwise state judgments, pairwise trajectory preferences, and risk refinement.

B.2 Supervised Fine-Tuning

Let $\mathcal{B} \subset \mathcal{D}_{\text{SFT}}$ be a minibatch and z_b the serialized prompt–response sequence for example b. With $m_{b,j}$ indicating whether token j belongs to the supervised response, the token-level objective is

$$\mathcal{L}_{\text{SFT}}(\theta; \mathcal{B}) = -\frac{\sum_{b \in \mathcal{B}} \sum_{j} m_{b,j} \log J_\theta(z_{b,j} \mid z_{b,<j})}{\sum_{b \in \mathcal{B}} \sum_{j} m_{b,j}}. \tag{B.2}$$

Response tokens receive mask value one, and prompt and padding tokens receive zero. Pointwise and pairwise records share this objective, with their differences represented in the input organization and target sequence. When a supervised response includes a rationale, its tokens are optimized jointly with the structured fields under the same response-token likelihood objective.

B.3 Retrospective Preference Optimization

Retrospective preference data contain triples (x, y^+, y^-), where y^+ and y^- are the chosen and rejected judgments for the same input. For prefix-based examples, both judgments use the same visible execution record; their preference may reflect a corrected verdict, adjusted risk, or refined evidence-based explanation. Define the response sequence log probability as

$$\ell_\theta(y \mid x) = \sum_{j=1}^{|y|} \log J_\theta(y_j \mid x, y_{<j}). \tag{B.3}$$

With fixed reference model J_{ref}, the reference-adjusted margin between chosen and rejected judgments is

$$\begin{aligned} \Delta_\theta(x, y^+, y^-) = &\big[\ell_\theta(y^+ \mid x) - \ell_{\text{ref}}(y^+ \mid x)\big] \\ &- \big[\ell_\theta(y^- \mid x) - \ell_{\text{ref}}(y^- \mid x)\big]. \end{aligned} \tag{B.4}$$

The corresponding preference objective (Rafailov et al., 2023) is

$$\mathcal{L}_{\mathrm{DPO}}(\theta) = -\mathbb{E}_{(x,y^+,y^-)\sim\mathcal{D}_{\mathrm{DPO}}}\left[\log\sigma\big(\beta\Delta_\theta(x,y^+,y^-)\big)\right], \tag{B.5}$$

where $\beta > 0$ follows the main-text definition. Pairwise SFT learns preferences across execution records, whereas retrospective DPO learns preferences across judgments of the same input. The latter can improve risk assessment or reasoning while retaining the original verdict. In Appendix C.3, for example, both judgments retain `good`, but the chosen judgment more fully accounts for an unresolved premise in the prefix.

B.4 Reward References and Structured Scoring

GRPO uses records consisting of a judging input and a reward reference:

$$\mathcal{D}_{\mathrm{GRPO}} = \{(x_j, \rho_j)\}_{j=1}^{N}, \qquad r_{j,i} = R(x_j, y_{j,i}; \rho_j). \tag{B.6}$$

Here, x_j is the input used to generate judgments, ρ_j is the supervision reference supplied to the reward evaluator, and $y_{j,i}$ is a candidate judgment. Existing SFT targets or accepted DPO judgments can be converted into references under the corresponding reward schema. The evaluator scores candidates against the reference fields; the reward reference is kept separate from the generation input.

Task-level reference answers and reward references serve different roles: the former provide evaluation information in records containing that field, while the latter specify judgment targets for training-time scoring. Appendix C.4 illustrates a trajectory-preference reference expressed by a `winner` field. The four-label configuration uses the scoring rule below.

Four-label reward. Let $\mathcal{K} = \{\mathrm{success}, \mathrm{side}, \mathrm{optimal}, \mathrm{loop}\}$ represent task success, side effects, trajectory optimality, and looping. Parser P_k extracts and normalizes field k, returning $\perp$ for a missing field. The reward combines field agreement with output completeness:

$$\begin{aligned} R_{\mathrm{four}}(y;\rho) = \frac{1}{4}\sum_{k\in\mathcal{K}} \mathbf{1}[P_k(y) \neq \perp \ \wedge\ P_k(y) = P_k(\rho)] \\ + 0.05\,\mathbf{1}[\forall k \in \mathcal{K},\ P_k(y) \neq \perp]. \end{aligned} \tag{B.7}$$

The first term measures the fraction of fields matching the supervision reference. The second rewards outputs containing all four fields, giving a reward range of $[0, 1.05]$. This configuration turns structured judgment correctness and output completeness into computable training feedback.

B.5 Group-Relative Policy Optimization

For the same judging input x, sample G candidate judgments from the old policy $J_{\theta_{\mathrm{old}}}$ and compute their rewards r_i. The within-group mean and normalized advantage (Shao et al., 2024) are

$$\bar{r} = \frac{1}{G}\sum_{i=1}^{G} r_i, \qquad \widehat{A}_i = \frac{r_i - \bar{r}}{\mathrm{std}(r_1,\ldots,r_G) + \delta}, \tag{B.8}$$

where $\delta > 0$ is a numerical stability term. The token-level importance ratio for token j of candidate y_i is

$$w_{i,j}(\theta) = \frac{J_\theta(y_{i,j} \mid x, y_{i,<j})}{J_{\theta_{\mathrm{old}}}(y_{i,j} \mid x, y_{i,<j})}. \tag{B.9}$$

Let n_i be the number of valid response tokens in candidate i. Define the clipped surrogate term as

$$s_{i,j}(\theta) = \min\Big\{w_{i,j}(\theta)\widehat{A}_i,\ \mathrm{clip}\big(w_{i,j}(\theta), 1-\epsilon, 1+\epsilon\big)\widehat{A}_i\Big\}. \tag{B.10}$$

For the configuration with zero Kullback–Leibler (KL) coefficient, the objective is

$$\mathcal{L}_{\mathrm{GRPO}}(\theta) = -\mathbb{E}_{x,\{y_i\}_{i=1}^{G}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{n_i}\sum_{j=1}^{n_i} s_{i,j}(\theta)\right]. \tag{B.11}$$

Valid response tokens within a candidate share its completion-level advantage, and the loss is normalized by candidate response length. The old policy generates the current sample group and supplies the denominator of the importance ratio; the DPO reference model remains fixed during preference optimization. The four-label configuration above uses $G = 4$, $\epsilon = 0.2$, and a zero KL coefficient.

B.6 TRAINING CONFIGURATION

Table 7 summarizes the hardware and optimization settings for SFT, DPO, and GRPO. All three stages use LoRA, with stage-specific learning rates and epoch counts.

Table 7: **Training configurations for SFT, DPO, and GRPO.** Each stage uses LoRA on eight RTX 4090 GPUs with 24 GB of memory per device, with batch size 4. The parameters r and α denote the LoRA rank and scaling coefficient. Learning rates, dropout, and epoch counts are listed for each stage. SFT runs for one epoch; DPO and GRPO each run for three.

Stage	LoRA	Batch Size	Device	Number of GPUs	Learning Rate	LoRA r	LoRA α	Dropout	Epochs
SFT	Yes	4	RTX 4090 24 GB	8	10^{-5}	16	32	0.05	1
DPO	Yes	4	RTX 4090 24 GB	8	5×10^{-7}	16	32	0.05	3
GRPO	Yes	4	RTX 4090 24 GB	8	10^{-6}	16	32	0.05	3

C EXAMPLES OF TRAINING SUPERVISION

Figures 6–9 illustrate stored records used for supervised fine-tuning, direct preference optimization, and group relative policy optimization. They distinguish visible inputs, execution evidence, and supervision targets for the data-construction and training stages.

C.1 POINTWISE TRAJECTORY-STATE SUPERVISION

SFT / Pointwise trajectory-state judgment

One task + one observed prefix -> a structured supervision target

Task (abridged)

Do Payment on Demand and My Cousin from Warsaw have directors from the same country? Use the four required local documents and cite all four in the final answer.

Stored input also supplies gold_answer: "yes". Evaluation point: state_index = 4.

Observed prefix

(selected evidence; original indices)

0 | Delegate the four-document research task. (action summary)
The delegation returns no conclusions; completion_assessment is "failed". (observation summary)

1 | Read the Curtis Bernhardt document. (action summary)
"... was a German film director born in Worms, Germany ..."

2 | Read the Carl Boese document. (action summary)
"... was a German film director, screenwriter and producer. ..."

3 | Read the Payment on Demand document. (action summary)
"... Payment on Demand is a 1951 drama film directed by Curtis Bernhardt ..."

4 | Read the My Cousin from Warsaw document. (action summary)
"... My Cousin from Warsaw is a 1931 German comedy film directed by Carl Boese ..."

Stored assistant target

(selected fields and verbatim summary excerpt)

```
{
  "per_state": [
    {"state_index": 4, "action_judge": "continue", "risk": "low"}
  ],
  "pointwise_score": 0.9, "trajectory_verdict": "good",
  "schema_version": "dynamic-replay-v1"
}
```

"All four documents have been successfully retrieved, providing clear evidence that both directors are German."

Additional target fields and document text omitted. The stored reference answer is visible to this judge; no final agent answer is shown.

Figure 6: **Pointwise trajectory-state supervision for supervised fine-tuning. Top:** the task asks whether two film directors come from the same country, using four specified documents. **Middle:** the visible prefix contains an unsuccessful delegation followed by four document reads, with original indices retained. **Bottom:** at state index 4, the stored target specifies continuing execution, low risk, a score of 0.9, and a `good` trajectory verdict. Its summary identifies evidence that both directors are German. The example shows how accumulated retrieval evidence becomes structured prefix-state supervision. The reference answer `yes` is visible to the judge in this training record; no final agent answer is shown, and the displayed assessment concerns the current trajectory state.

C.2 PAIRWISE TRAJECTORY SUPERVISION



Figure 7: **Pairwise supervision linking evidence acquisition to execution quality. Top:** the task identifies a band's lead singer by connecting two specified documents. **Left:** trajectory A reads both documents directly, using two child tool calls with eight remaining. **Right:** trajectory B exhausts ten child tool calls, encounters a budget-exhaustion error on an additional read, and then relies on excerpts previously returned by search. Both answer Eddie Argos and reference the required documents. **Bottom:** the stored preference and original rationale favor A for its direct tool use, fuller evidence coverage, and cleaner termination. The example distinguishes execution quality despite a shared final answer. The input includes a reference answer, and trajectory B already contains middle truncation in the stored record.

C.3 PREFIX-GROUNDED RISK REFINEMENT

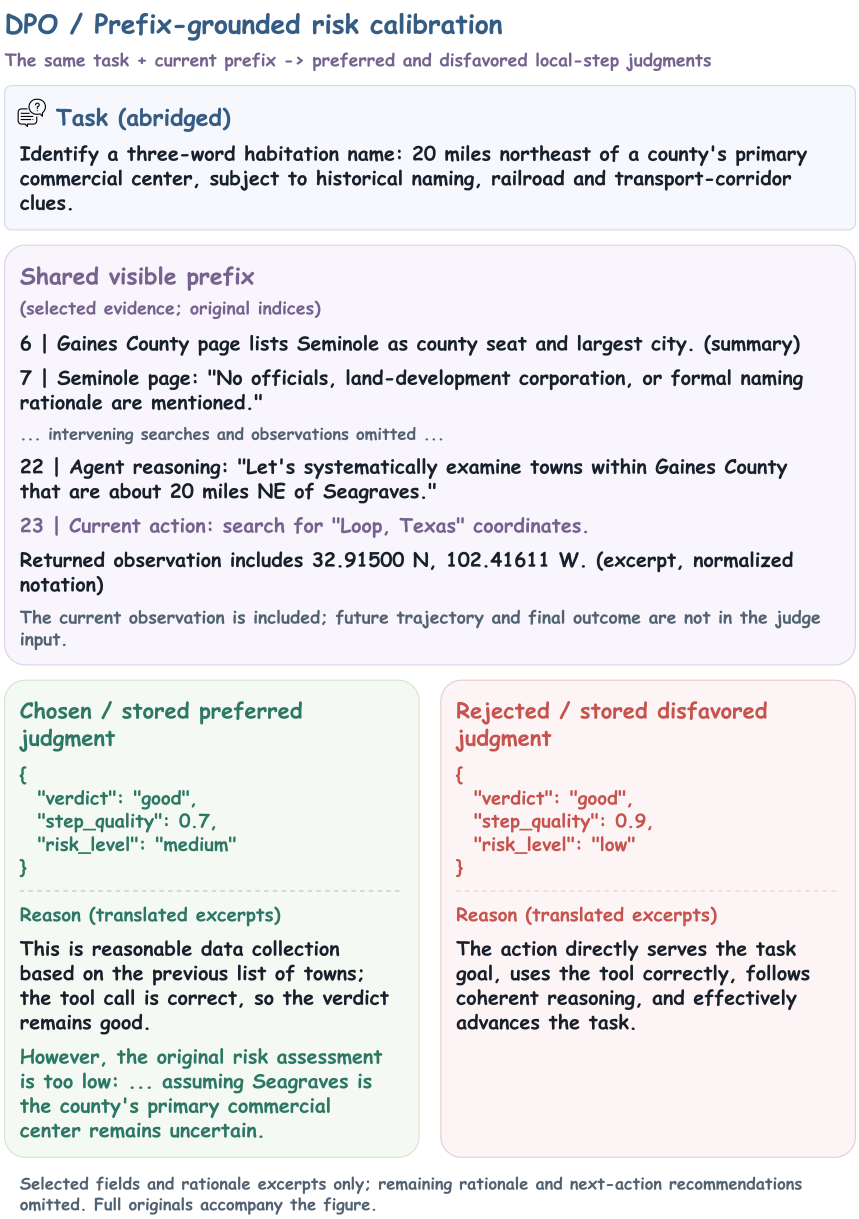

Figure 8: **Prefix-grounded risk refinement in a preference pair. Top:** the shared input contains the geographic task, earlier evidence, and the coordinate search for Loop, Texas at index 23, including its observation; future execution and the final outcome are absent. **Left:** the chosen judgment retains a `good` verdict but assigns a score of 0.7 and `medium` risk because identifying Seagraves as the county's primary commercial center remains uncertain. **Right:** the rejected judgment assigns 0.9 and `low` risk without reflecting this unresolved premise. The pair separates reasonable tool use from accumulated reasoning risk, illustrating a preference for refined risk assessment rather than a forced verdict reversal. Selected fields and translated rationale excerpts are shown.

C.4 TRAJECTORY-PREFERENCE REWARD REFERENCE

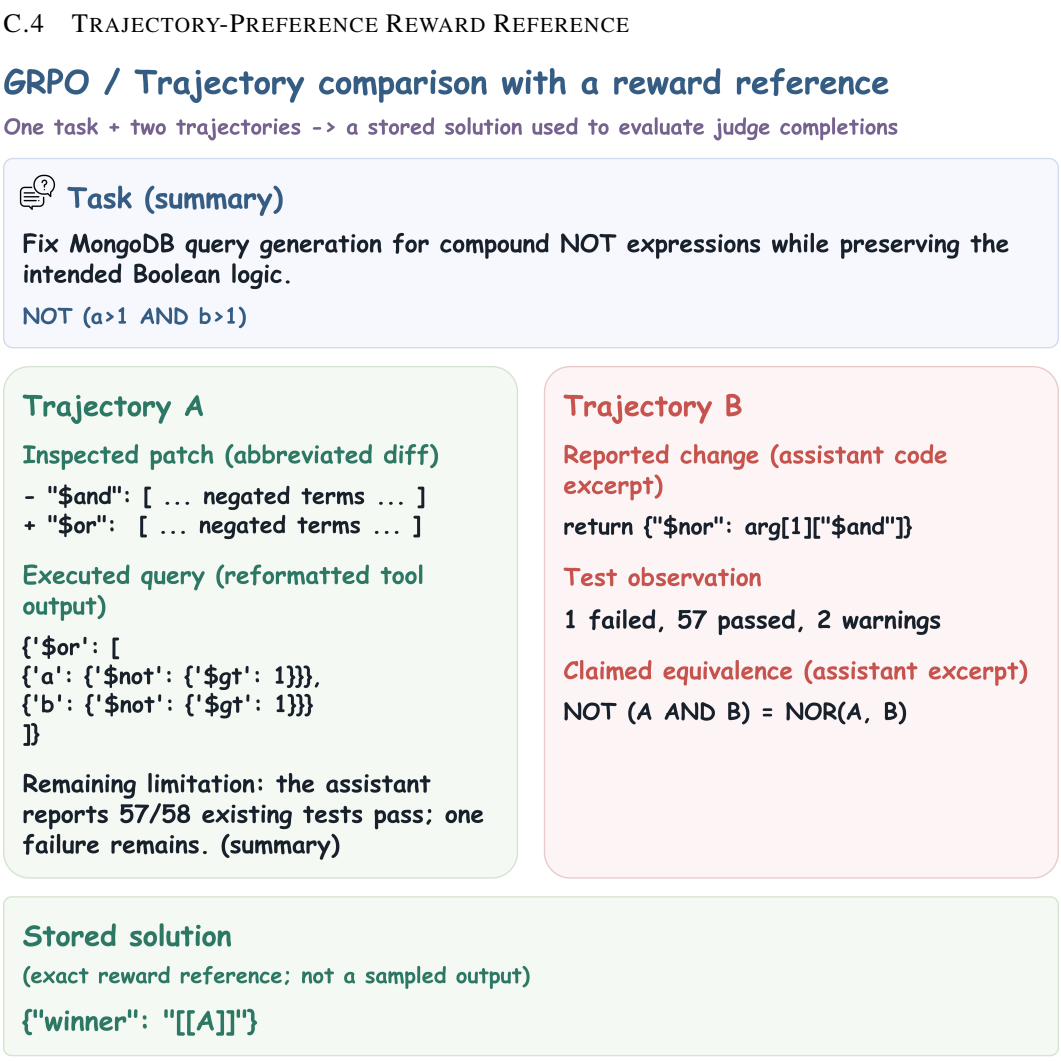

Figure 9: **A trajectory-preference reward reference for group relative policy optimization. Top:** the coding task repairs MongoDB query generation for compound negation while preserving Boolean logic. **Left:** trajectory A changes the connective between negated terms from `$and` to `$or`; the executed query reflects this structure, although one existing test failure remains reported. **Right:** trajectory B reports using `$nor` and gives the displayed incorrect equivalence, with a test observation of one failure and 57 passes. **Bottom:** the stored `winner = [[A]]` supplies the preference reference for evaluating judge completions. This is a reward target, not a sampled judge output or a measured reward. Both stored trajectories contain middle truncation.

D Qualitative Judging Comparisons

Figures 10–13 compare the base model and AgentJudge on general responses and agent trajectories. Each selected case presents task evidence, saved judgment excerpts, and a reference label to illustrate a specific judgment difference.

D.1 General Response Judging



Figure 10: **Factual discrimination under distracting content in JudgeLM** (Zhu et al., 2025b). **Top:** case 1252 asks for three animal classifications. Answer A correctly identifies the rabbit as a mammal, whereas Answer B calls it a reptile. Both include incomplete code omitted from the display, and the reference preference is A. **Lower left:** the base model calls both classifications incorrect, contradicts itself about the rabbit, and selects no winner. **Lower right:** AgentJudge identifies A's correct classifications and selects A, matching the reference. The case highlights the factual distinction determining relative response quality, with the omitted code disclosed as additional context.

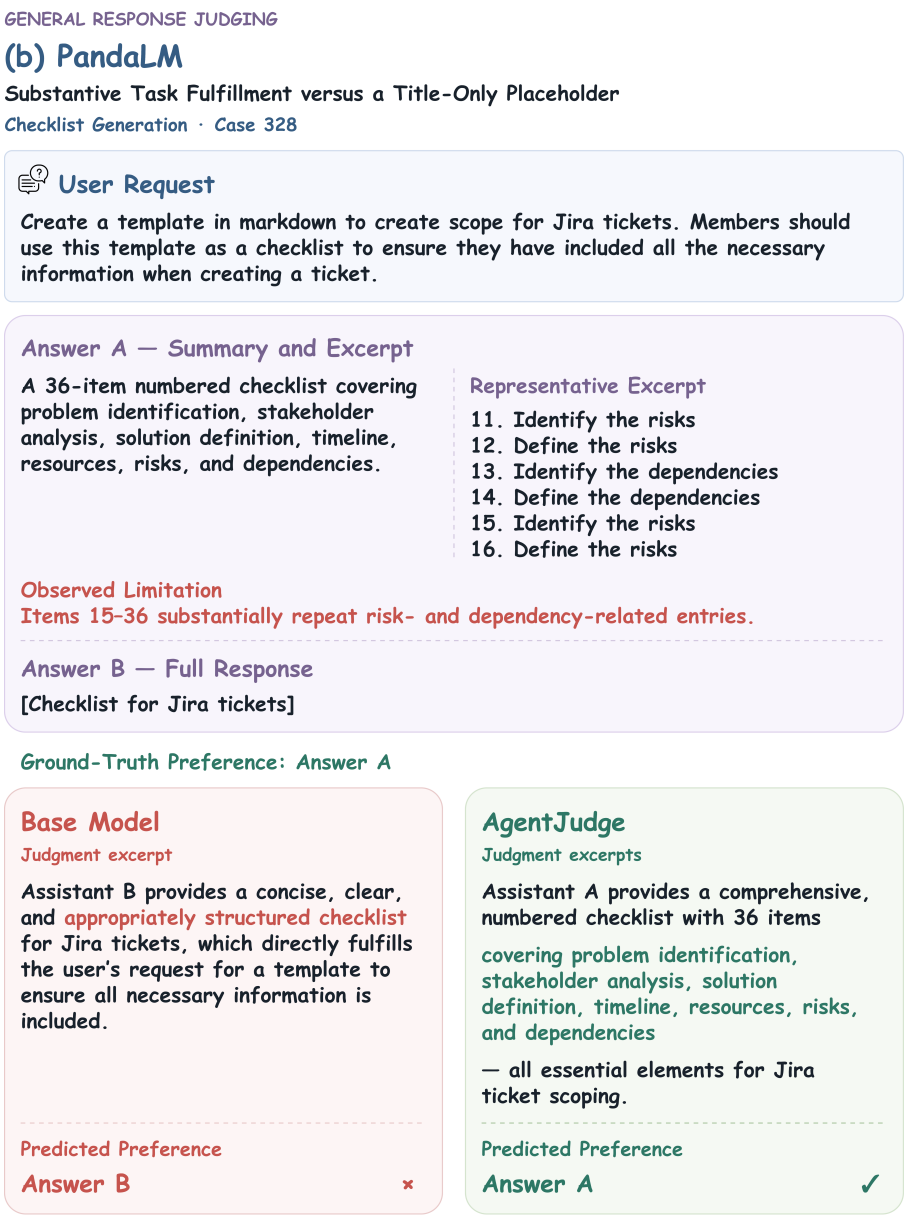

Figure 11: **Substantive task fulfillment versus a title-only placeholder in PandaLM** (Wang et al., 2024). **Top:** case 328 requests a Jira ticket-scoping checklist. Answer A supplies 36 items covering relevant aspects but substantially repeats entries in items 15–36; Answer B's entire response is a title-only placeholder. **Lower left:** the base model describes B as an appropriately structured checklist and prefers it. **Lower right:** AgentJudge prefers A for its substantive coverage, agreeing with the reference. The figure retains A's repetition to make the comparison explicit. AgentJudge's displayed explanation emphasizes coverage without identifying that repetition, distinguishing a correct relative preference from a complete account of the preferred answer's limitations.

D.2 AGENT TRAJECTORY JUDGING



Figure 12: **Repetitive behavior and execution stagnation in AgentRewardBench** (Lù et al., 2025). **Top:** a WorkArena agent attempts to create a hardware asset, scrolling 13 times at steps 10–22 and five more times at steps 26–30 before ending on the new-record form with its submit button still visible. The two intervals are separated to preserve the intervening actions. **Lower left:** the base model acknowledges redundant scrolling but predicts `Looping = No`. **Lower right:** AgentJudge links repeated scrolling to the unfilled Vendor field and predicts `Yes`, agreeing with the human annotation. The case illustrates how repeated actions and unresolved task requirements jointly inform a complete-trajectory looping judgment.



Figure 13: **Direct execution versus avoidable error-and-retry behavior in Plan-RewardBench** (Wang et al., 2026). **Top:** a `planning_single_easy` task requests New York time and a conversion to Tokyo time. Trajectory A uses accepted time-zone identifiers in two tool calls; trajectory B incurs two invalid-time-zone calls before correcting its arguments, requiring four calls overall. **Lower left:** the base model favors B, treating recovery from these errors as greater adaptability. **Lower right:** AgentJudge favors A because direct execution avoids unnecessary errors and retries, matching the reference preference. Although both trajectories provide final answers, their local argument choices and additional tool-call costs distinguish overall execution quality.