INEX: INTRINSIC AND EXTRINSIC EVIDENCE FOR TRUSTWORTHY EVALUATION OF LATENTLY VERIFIABLE TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

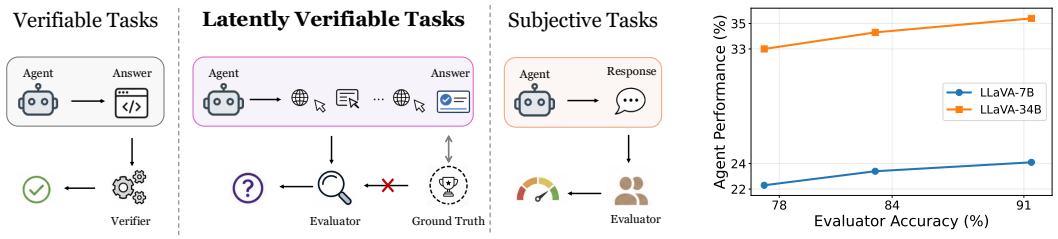
Many real-world agent tasks are *latently verifiable*: a correct answer or success condition exists in principle, but the corresponding ground truth is not cheaply available during evaluation. Reliable evaluation must therefore determine whether an agent's claimed outcome is supported by evidence rather than merely judge the plausibility of its final answer. We introduce InEx, an evidence-based evaluator that combines two complementary sources: intrinsic evidence already contained in the submitted trajectory and extrinsic evidence acquired through additional environment interaction. InEx-I reconstructs task-relevant intrinsic evidence by inspecting local observation-action transitions and determines whether the resulting evidence chain supports the claimed outcome. When intrinsic evidence alone is insufficient, InEx-E selectively interacts with the environment to acquire the missing extrinsic evidence. A scaffolded handoff carries the structured intrinsic analysis into extrinsic verification so that the two components form a continuous evidence chain. We implement InEx-I by specializing an open vision-language model through supervised fine-tuning, reinforcement learning, and confidence-aware learning from imperfect supervision, and implement InEx-E with a frontier model capable of environment interaction. We evaluate InEx on a diverse suite of agent benchmarks spanning web, mobile, and desktop interaction. InEx consistently achieves the highest evaluator accuracy among the compared methods and produces more reliable rewards that improve downstream agent training.

1 INTRODUCTION

Large language models (LLMs) are increasingly used as agents that interact with external environments, navigate websites, call tools, and complete long-horizon tasks (Yao et al., 2023; Gur et al., 2024; Liu et al., 2024). Alongside this shift, reinforcement learning with verifiable rewards (RLVR) has become an important paradigm for post-training LLMs and agents (Guo et al., 2025; Kimi Team et al., 2025; Da et al., 2025; Wei et al., 2025). While much of this literature focuses on reinforcement learning (RL) algorithms, exploration, and rollout design, the verified reward itself is equally central: it determines which agent behaviors are reinforced. In domains such as math and coding, reward design is often relatively straightforward, since correctness can be checked against known answers, symbolic rules, execution results, or unit tests (Cobbe et al., 2021; Chen et al., 2021). However, many realistic agent tasks are harder to verify. A web agent may need to search dynamic online content, interact with changing interfaces, collect evidence across multiple pages, or determine that no valid result exists. In these settings, the desired answer may be objective, but producing an accurate reward can require nontrivial verification. Figure 1(b) provides a motivating observation for this point: when evaluators are used as reward sources under the same agent-training framework, higher evaluator accuracy is associated with higher downstream agent task success.

We focus on *latently verifiable agent tasks*: tasks for which a correct answer or success condition exists in principle, but the corresponding ground truth is not cheaply available during evaluation. This setting is distinct from subjective preference-based tasks, for which a single objective answer may not exist (Christiano et al., 2017; Ouyang et al., 2022), and from automatically verifiable domains such as many math and coding tasks, where correctness can be checked cheaply through symbolic rules, execution, or unit tests (Cobbe et al., 2021; Chen et al., 2021). Such tasks arise naturally in

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.



(a) Task verifiability spectrum. (b) Better evaluators improve agents.

Figure 1: **Overview of latently verifiable agent task evaluation. (a)** Automatically verifiable tasks admit cheap checkers, while subjective tasks rely on preference or quality judgments. Latently verifiable tasks lie between these extremes: an objective success condition exists, but the corresponding ground truth is not cheaply available during evaluation. **(b)** Improved evaluator reliability leads to better downstream web-agent training. When different evaluators are used as reward sources under the same training framework, higher evaluator accuracy corresponds to higher agent task success.

agentic settings where the evidence needed to determine success is distributed across long multimodal trajectories, dynamic environment states, or follow-up interactions. Existing work often treats this verification problem as a benchmark-side necessity handled through human annotation or task-specific scripts. We instead study how to construct a reliable evaluator when the outcome is objective but the evidence required to verify it is not directly available.

A common design in current model-based agent evaluation is the model-as-a-judge pipeline: the evaluator packs the task instruction, trajectory, visual context, and final answer into a prompt and asks an LLM or VLM to predict whether the task was completed (Zheng et al., 2023; Kim et al., 2024; He et al., 2024; Pan et al., 2024; Zhou et al., 2025). While practical, this design is often too coarse for latently verifiable agent tasks. Context-window limits may force the evaluator to discard early trajectory information, while holistic judgment over a flattened context obscures whether the agent's reasoning and final answer are grounded in its observations. Such evaluators can therefore be misled by plausible but unsupported answers, hallucinated evidence, or right-answer-wrong-reason trajectories. Moreover, a trajectory-only evaluator cannot resolve claims whose necessary evidence was never recorded. For example, when an agent claims that no valid result exists, the evaluator may need to revisit the environment to distinguish a genuine impossibility from an incomplete search.

We propose InEx, an evidence-based evaluator that treats evaluation as evidence reconstruction rather than direct final-answer classification. We distinguish two complementary evidence sources. *Intrinsic evidence* comprises the observations, actions, reasoning traces, and final answer already contained in the submitted trajectory, whereas *extrinsic evidence* is acquired through additional interaction with the environment after the trajectory is submitted. InEx-I performs bottom-up intrinsic evaluation by analyzing local observation-action transitions, extracting task-relevant evidence, and determining whether the resulting evidence chain supports or contradicts the claimed outcome. When intrinsic evidence leaves important claims unresolved, InEx-E acquires the missing extrinsic evidence from the environment. A scaffolded handoff passes the structured intrinsic analysis to InEx-E, allowing it to continue the same evidence chain rather than restart evaluation from the raw trajectory.

In our implementation, InEx-I is an open vision-language model specialized for intrinsic evidence reconstruction through supervised fine-tuning, reinforcement-learning post-training, and a confidence-aware strategy for imperfect supervision. InEx-E uses a frontier model with environment-interaction capabilities. The intrinsic component first produces a structured evidence analysis and preliminary verdict; the extrinsic component inherits this state, acquires additional observations only when needed, and produces the final verdict. These components and their scaffolded handoff jointly form InEx, with training serving to specialize the intrinsic evidence-reconstruction component of the evaluator.

We evaluate InEx on a suite of agent benchmarks spanning web, mobile, and desktop interaction. We reuse the same trained InEx-I checkpoint across all benchmarks without environment-specific retraining, while InEx-E operates only through each benchmark's native observation and action interface. InEx achieves the highest evaluator accuracy among the compared methods in every setting and substantially reduces false-positive success judgments. Moreover, using InEx-derived rewards for downstream web-agent training yields stronger agents, demonstrating the practical value of improved evaluator reliability.

Our **contributions** are threefold. **(i)** We formulate *latently verifiable agent tasks* as an important evaluation setting in which an objective answer or success condition exists, but the corresponding ground truth is not cheaply available during evaluation. **(ii)** We introduce InEx, an evidence-based evaluator that reconstructs action-conditioned intrinsic evidence, selectively acquires extrinsic evidence, and preserves the resulting evidence chain through a scaffolded handoff. **(iii)** We evaluate the same implementation, without environment-specific retraining, on a suite of agent benchmarks spanning web, mobile, and desktop interaction. InEx consistently achieves the highest evaluator accuracy among the compared methods and provides higher-quality rewards that improve downstream agent training.

2 Related Work

Interactive Agents and Latently Verifiable Tasks. LLM-based agents increasingly operate across web, mobile, and desktop interfaces, motivating benchmarks for long-horizon decision making under partial observability and changing environment state (Yao et al., 2023; Gur et al., 2024; Deng et al., 2023; Zhou et al., 2024; He et al., 2024; Rawles et al., 2025; Xie et al., 2024). Most prior work centers on agent policies, prompting, training, or benchmark construction. We instead study evaluation when task success is objective but not cheaply observable: verification may require reconstructing evidence distributed across a multimodal trajectory or revisiting the resulting environment state. We view interactive-agent evaluation as a representative setting for latently verifiable tasks and treat the evaluator, rather than the acting policy, as the primary object of study.

Trajectory-Based Model Evaluation. LLMs are widely used to judge generation quality and instruction following (Zheng et al., 2023; Li et al., 2023; Kim et al., 2024), and model-based evaluators increasingly assess agent task completion. WebVoyager judges success from the instruction, answer, and visual interaction record (He et al., 2024); AER studies autonomous evaluation and refinement of digital agents (Pan et al., 2024); and PAE incorporates a VLM evaluator into agent skill discovery and training (Zhou et al., 2025). More structured approaches exploit trajectory organization in different ways. NNetNav retroactively labels explored browser trajectories to construct and filter synthetic demonstrations, rather than evaluating a submitted agent run (Murty et al., 2025). WebJudge, our closest trajectory-only baseline, derives task key points, selects relevant screenshots, and aggregates them into a success verdict (Xue et al., 2025). In contrast, InEx-I reconstructs support from action-conditioned multimodal transitions rather than treating selected observations only as verdict context; the complete InEx evaluator can additionally obtain evidence absent from the submitted trajectory.

Interactive and Tool-Augmented Verification. Tool-using and browsing methods employ environment interaction primarily to solve tasks (Nakano et al., 2021; Schick et al., 2023; Yao et al., 2023); a growing line instead makes evaluation itself agentic. Agent-as-a-Judge evaluates code-generation agents against structured requirements using an evaluator with workspace tools (Zhuge et al., 2025), while recent GUI evaluators study milestone-based trajectory inspection, execution-video reward modeling, and proactive environment probing (Li et al., 2026; Song et al., 2026; Dai et al., 2026; Cui et al., 2026; Shi et al., 2026). InEx combines the two evidence sources in one workflow: a trained multimodal evaluator first reconstructs intrinsic evidence from the submitted trajectory, and InEx-E selectively acquires only the environment evidence needed to resolve remaining uncertainty through a scaffolded handoff.

3 Problem Formulation

We study the problem of designing evaluators for *latently verifiable agent tasks*. Each task instance consists of an instruction x, a latent ground-truth object G, and an agent submission (τ, y), where y is the final answer and $\tau = \{(o_t, r_t, a_t, o_{t+1})\}_{t=1}^{T}$ is the interaction trajectory. Here, o_t and o_{t+1} are the observations before and after step t, r_t is the agent's reasoning or decision text when available, and a_t is the agent action. Evidence such as retrieved content, URLs, tool outputs, and environment-dependent information is derived from these recorded transitions. The latent ground truth $G \in \mathcal{G}$ may take a task-dependent structured form, such as a canonical answer, a set of acceptable answers, a checklist of required conditions, or an environment state that determines success. If G were available,

it would induce an ideal binary success verdict; however, G is not provided to the evaluator:

$$v^\star = \Gamma(x, \tau, y; G), \qquad \hat{v} = E(x, \tau, y), \qquad v^\star, \hat{v} \in \{0, 1\}. \tag{1}$$

Here, $v^\star = 1$ indicates that the agent successfully completed the task, while $\hat{v}$ is the evaluator's predicted verdict based on the submitted trajectory, final answer, and any evidence accessible through the evaluation process.

The defining property of this task class is that an objective ground truth exists in principle but is not cheaply available during evaluation. This distinguishes our setting from purely subjective open-ended tasks, where correctness is primarily a matter of preference or response quality (Christiano et al., 2017; Ouyang et al., 2022), and from automatically verifiable tasks such as many math or coding problems, where correctness can often be checked by a rule-based evaluator, symbolic solver, execution script, or unit test (Cobbe et al., 2021; Chen et al., 2021). This setting is common in modern agentic systems, where task success may depend on dynamic environments, evolving external information, and nontrivial evidence gathering. We study this setting across web navigation, mobile-device interaction, and desktop interaction, where the evidence needed to determine success may be distributed across long multimodal trajectories and evolving application states (He et al., 2024; Zhou et al., 2024; Rawles et al., 2025; Xie et al., 2024). Human outcomes for WebVoyager and official benchmark-evaluator outcomes for WebArena, AndroidWorld, and OSWorld are used only for offline meta-evaluation. They are not provided to InEx or any baseline, which must instead reconstruct evidence from the submitted trajectory or acquire it through the benchmark-supported environment interface.

4 InEx: Intrinsic and Extrinsic Evidence for Trustworthy Evaluation

In this section, we present InEx, an evidence-based evaluator for latently verifiable agent tasks. Our objective is to determine whether an agent has completed a task when the ground truth exists in principle but is not cheaply available during evaluation. Rather than treating evaluation as final-answer classification, InEx reconstructs what the submitted trajectory actually establishes and selectively acquires additional evidence when the trajectory alone cannot support a reliable verdict. This reduces unsupported judgments and hallucinated evaluations, and is especially important when evaluator outputs become reward signals for downstream agent training.

The central idea of InEx is to organize evaluation around two complementary sources of evidence. *Intrinsic evidence* is already contained in the submitted trajectory: the observations, actions, collected information, reasoning traces, and final answer that record what the agent saw, what it did, and how it arrived at its response. *Extrinsic evidence* is acquired by the evaluator beyond that trajectory through additional environment interaction or follow-up queries. Trustworthy evaluation should first exploit the evidence already present in the submission and, when that evidence is insufficient, acquire the missing evidence needed for a reliable verdict.

Following this principle, InEx consists of two complementary components and a mechanism for connecting them. InEx-I reconstructs a structured evidence chain from the submitted trajectory, while InEx-E performs selective active verification for claims that remain unresolved. A scaffolded handoff carries the intrinsic analysis into extrinsic verification so that InEx-E continues, rather than repeats, the evidence-reconstruction process.

4.1 Intrinsic and Extrinsic Evidence Evaluation

4.1.1 InEx-I: Intrinsic Evidence Reconstruction

The first procedure of InEx, InEx-I, judges task success using evidence already contained in the submitted agent trajectory. Even when the latent ground truth G is unavailable, the trajectory records how the final answer was produced: what the agent observed, what actions it took, what evidence it collected, and how it reasoned over that evidence. A trustworthy evaluator should therefore ask not only whether the final answer appears plausible, but whether the trajectory itself provides sufficient support for it.

Many existing model-based evaluators for agent tasks can be viewed as coarse intrinsic-evidence evaluators: they take the task instruction, trajectory, and final answer as input, and prompt an LLM or VLM to produce an overall success judgment (Pan et al., 2024; Zhou et al., 2025). This flatten-and-judge workflow is often insufficient. Context-window constraints may force the evaluator to discard early trajectory information, and holistic judgment over a single prompt makes it difficult to inspect whether the agent's intermediate process is grounded. As a result, such evaluators may accept hallucinated evidence, overlook missing verification steps, or mark a plausible final answer as correct even when it is reached for the wrong reason.

In latently verifiable tasks, failures often occur locally before they become visible in the final answer: the agent may misread an observation, take an action that does not have the intended effect, or reason from evidence that never appeared in the environment. A fluent final response can obscure these failures when the trajectory is judged only as a whole. InEx-I therefore decomposes intrinsic evaluation into step-level evidence inspection followed by trajectory-level aggregation. For each local transition $\Delta_t = (o_t, r_t, a_t, o_{t+1})$, where o_t and o_{t+1} are observations before and after the action, a_t is the action, and r_t is the agent's reasoning trace, InEx-I produces a step-level evidence record

$$z_t = \Phi_{\mathrm{I}}(x, \Delta_t). \tag{2}$$

Each record summarizes the local intent, supporting or contradicting evidence, grounding status, and action outcome. The records are not independent verdicts; they preserve local evidence for trajectory-level aggregation.

After inspecting local transitions, InEx-I selects the task-relevant subset of steps $\mathcal{P} \subseteq \{1, \ldots, T\}$, reconstructs how the agent arrives at the final answer, and produces both a structured intrinsic evidence chain H_{I} and an intrinsic verdict:

$$(H_{\mathrm{I}}, \hat{v}_{\mathrm{I}}) = \Psi_{\mathrm{I}}\left(x,\, y,\, \{(z_t, \Delta_t)\}_{t \in \mathcal{P}}\right), \tag{3}$$

where $\hat{v}_{\mathrm{I}} \in \{0, 1\}$. This procedure grounds the task-level judgment in a coherent sequence of local transitions rather than an unstructured summary. We provide the full intrinsic-evidence workflow in Appendix A.

4.1.2 InEx-E: Selective Extrinsic Evidence Acquisition

Intrinsic evaluation makes full use of the evidence already contained in the submitted trajectory, but that trajectory may not always support a reliable verdict. For example, an agent may claim that a task is impossible or that no valid answer exists. The trajectory can reveal that the claim is unsupported, yet still be insufficient to determine whether it reflects a real limitation of the environment or merely an agent failure. More broadly, trustworthy evaluation of capable agents may itself require agentic capabilities: the evaluator should be able to revisit the environment and gather additional evidence when the submitted record is incomplete or ambiguous.

InEx-E formalizes this second stage as selective extrinsic evidence acquisition. Given the instruction, submitted trajectory, final answer, and intrinsic evidence chain H_{I}, it can use an environment-appropriate action space $\mathcal{A}_{\mathrm{E}}$ to obtain evidence $\mathcal{E}_{\mathrm{E}}$ for unresolved claims. When H_{I} already supports a reliable judgment, the procedure may terminate without an environment action; otherwise, it queries or interacts with the environment until the missing evidence is resolved or the interaction budget is exhausted.

This turns evaluation into active verification rather than passive trajectory reading. Instead of relying only on what the evaluated agent chose to submit, InEx-E can independently test whether the environment supports the final answer, whether a required condition still holds, or whether an apparent impossibility is caused by the agent rather than the task. Appendix C specifies the inputs, action space, stopping criteria, and optional agent-query protocol used in our implementation.

4.2 Implementation and Scaffolded Handoff

In our implementation, InEx-I uses Qwen2.5-VL-7B-Instruct (Bai et al., 2025b), specialized through supervised fine-tuning and reinforcement-learning post-training to follow the bottom-up intrinsic-evidence workflow. InEx-E uses a frontier model with environment-interaction capabilities to resolve claims that cannot be verified from the submitted trajectory. The complete data-construction and

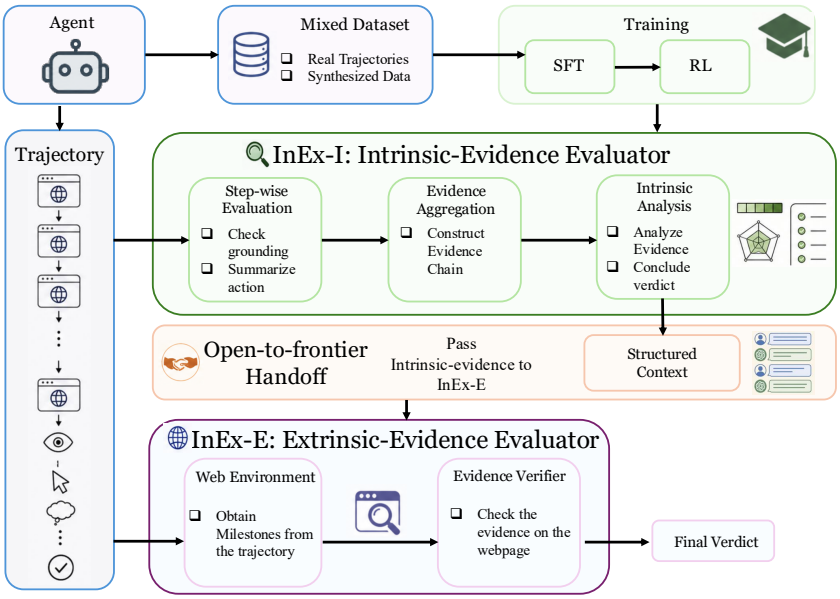


Figure 2: **Workflow of InEx.** InEx-I reconstructs a structured intrinsic evidence chain from action-conditioned trajectory transitions. A scaffolded handoff passes this analysis to InEx-E, which selectively acquires missing environment evidence and produces the final verdict.

training procedure for InEx-I is provided in Appendix B, while the operational details for InEx-E are provided in Appendix C.

For every submitted instance, InEx-I first produces the intrinsic scaffold $S_{\mathrm{I}} = (H_{\mathrm{I}}, \hat{v}_{\mathrm{I}})$. We convert this scaffold into the preceding evaluation state for InEx-E, allowing the frontier component to continue the same evidence chain rather than rediscovering the relevant transitions and unresolved claims from the raw trajectory:

$$S_{\mathrm{I}} = \text{InEx-I}(x, \tau, y), \qquad (\mathcal{E}_{\mathrm{E}}, \hat{v}) = \text{InEx-E}(x, \tau, y; S_{\mathrm{I}}, \mathcal{A}_{\mathrm{E}}), \tag{4}$$

where $\mathcal{E}_{\mathrm{E}}$ denotes any additional environment evidence acquired by InEx-E, possibly empty, and $\hat{v} \in \{0, 1\}$ is the final InEx verdict. The handoff occurs for every instance, but environment actions are taken only when the intrinsic scaffold is insufficient. Appendix D details the conversation construction and handoff execution.

5 Experiments

We design experiments to answer three questions: **(1)** Does InEx produce more reliable success judgments than generic and specialized model-based evaluators on latently verifiable agent trajectories? **(2)** Does improved evaluator reliability translate into more effective reward signals for downstream agent training? **(3)** How do the training of InEx-I, the InEx-E component, and the scaffolded handoff contribute to the complete evaluator?

5.1 Experiment Setup

We evaluate across four agent environments: WebVoyager (He et al., 2024) and WebArena (Zhou et al., 2024) for web interaction, AndroidWorld (Rawles et al., 2025) for mobile-device interaction, and OSWorld (Xie et al., 2024) for desktop interaction. Each evaluation set contains 1,000 complete trajectories. For WebVoyager, we use the held-out trajectories described in Appendix E, with human-annotated task-completion labels. For WebArena, AndroidWorld, and OSWorld, the corresponding official benchmark evaluator supplies the reference task-completion outcome. These task-specific evaluators may inspect benchmark state unavailable to a model-based evaluator and serve only as hidden measurement oracles: their outputs and implementations are never exposed to InEx or any baseline during inference.

		WebVoyager *Web*	WebArena *Web*	AndroidWorld *Mobile*	OSWorld *Desktop*	Macro Avg.
Frontier	Claude 3.5 Sonnet	78.7±0.8	82.3±1.2	83.9±0.8	75.2±1.5	80.0
	GPT-5.0	81.3±0.7	84.5±0.9	83.2±0.7	77.1±1.8	81.5
Open-source	Qwen3-VL-8B	75.3±1.9	76.9±1.8	81.2±0.8	70.3±3.5	75.9
Structured	WebJudge	86.7±0.6	87.0±0.9	87.8±0.6	83.7±1.5	86.3
Ours	InEx-I	90.1±0.3	91.1±0.7	92.1±0.3	84.6±1.3	89.5
	InEx	**91.4±0.3**	**93.0±0.5**	**94.5±0.3**	**88.2±1.0**	**91.8**

Table 1: **Evaluator accuracy across web, mobile, and desktop agent environments (in %).** Each benchmark contains 1,000 trajectories. WebVoyager uses human task-completion labels; WebArena, AndroidWorld, and OSWorld use reference outcomes from their official benchmark evaluators. Benchmark results are mean ± standard deviation over three runs. *Macro Avg.* is the unweighted mean of the four benchmark-level means and therefore has no independently estimated error bar. Reference outcomes are used only for offline scoring and are never exposed to the evaluated methods.

All evaluated methods receive the same task instruction, submitted trajectory, and final answer within each benchmark. We reuse a single trained InEx-I checkpoint across all four environments and perform no environment-specific fine-tuning or other parameter updates. InEx-E operates through each benchmark's native observation interface and only the actions officially supported by that environment. This changes the interface through which evidence is acquired, not the trained InEx-I parameters or the intrinsic–extrinsic evaluation procedure. The full protocol is summarized in Appendix E.4.

We compare InEx against representative model-as-a-judge baselines. The frontier-model baselines are Claude 3.5 Sonnet (Anthropic, 2024) and GPT-5.0 (OpenAI, 2025); the open-source baseline is Qwen3-VL-8B (Bai et al., 2025a). We additionally compare with WebJudge (Xue et al., 2025), a structured evaluator that extracts task key points and aggregates outcome-relevant screenshots. We report InEx-I as the intrinsic-only configuration and InEx as the complete evaluator, which adds InEx-E through the scaffolded handoff.

Our primary metric is evaluator accuracy: $\mathrm{Acc} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[\hat{v}_i = v_i^\star]$, where $\hat{v}_i$ is the evaluator's predicted success verdict and $v_i^\star$ is the held-out reference outcome. We report mean ± standard deviation over three evaluator runs. Because evaluator outputs are also used as reward signals, we additionally conduct a reward-relevant diagnostic analysis in Appendix F.2. This analysis reports success precision, success recall, failure recall, and balanced accuracy to characterize false-positive and false-negative reward errors.

5.2 Main Results

Table 1 reports evaluator accuracy across two web environments, Android mobile applications, and desktop applications. InEx achieves the highest accuracy in every environment, outperforming the strongest baseline, WebJudge, by 4.7 points on WebVoyager, 6.0 on WebArena, 6.7 on AndroidWorld, and 4.5 on OSWorld. Its macro average is 91.8%, compared with 86.3% for WebJudge. Because the same trained InEx-I checkpoint is reused without environment-specific fine-tuning, the results show that its intrinsic evidence reconstruction transfers across these interaction interfaces: InEx-I alone exceeds WebJudge in all four settings.

Adding selective extrinsic verification improves InEx-I by 1.3, 1.9, 2.4, and 3.6 points on WebVoyager, WebArena, AndroidWorld, and OSWorld, respectively. Thus, active evidence acquisition provides additional gains in every environment while preserving a common intrinsic evaluator. Appendix F.1 provides the full five-category WebVoyager breakdown.

A broader reward-relevant error analysis is provided in Appendix F.2. As shown in Table 7, InEx achieves 97.6% success precision, 96.7% failure recall, and 92.3% balanced accuracy. These results show that its aggregate gain is driven in particular by fewer false-positive success judgments—the error type most likely to corrupt evaluator-derived rewards.

Further analysis of reward-relevant error behavior, selective extrinsic verification, and downstream agent-training gains is provided in Appendix F.5.

Appendix F.5 also presents two audited qualitative cases illustrating the complementary roles of the two evidence sources: an InEx-I-style intrinsic audit traces an unsupported claim to the action that produced it, while the InEx-E verification procedure acquires a timestamped counterexample unavailable in the submitted trajectory.

5.3 Downstream Agent Training

We next evaluate whether improved evaluator reliability translates into stronger downstream web-agent training. To isolate the effect of the reward evaluator, we build on the PAE training framework (Zhou et al., 2025) and keep the agent backbone, task generation procedure, rollout collection, optimization pipeline, and final evaluation protocol fixed. The only change is the evaluator used to assign trajectory-level success rewards during training.

In the original PAE setup, agent-generated trajectories are labeled by a VLM-as-Judge evaluator, and the resulting binary success signal is used as the reward for agent optimization. In our variant, we replace this reward source with InEx. This comparison directly tests whether the evaluator improvements observed in Section 5.2 lead to more effective training supervision for web agents.

Table 2 reports downstream agent success rates under different reward sources. Replacing the VLM-as-Judge reward with InEx improves the LLaVA-7B agent on both WebArena and WebVoyager. Since all other components of the PAE training framework are held fixed, these gains isolate the effect of using a more reliable evaluator as the reward source.

These results support the broader motivation of InEx: evaluator quality matters not only for measuring agent behavior, but also for shaping the supervision used to train agents. Compared with a generic VLM-as-Judge reward, InEx reduces false-positive success labels and grounds reward assignment in both intrinsic and extrinsic evidence, leading to more effective downstream web-agent learning.

5.4 Ablation Studies

We conduct ablation studies to isolate the contributions of the main components in InEx. Unless otherwise specified, all ablations are evaluated on the same human-annotated WebVoyager evaluation set described in Section 5.1, and all numbers report evaluator accuracy against human labels.

Component-wise ablation. We first study how each training and evidence component contributes to evaluator reliability. Starting from the open-source backbone used for InEx-I, we progressively add a structured trajectory-inspection prompt, supervised fine-tuning, reinforcement learning post-training, and finally extrinsic-evidence evaluation through the scaffolded open-to-frontier handoff.

Table 3 shows that prompting the backbone with a more structured trajectory-inspection format provides only a small improvement, indicating that prompting alone is insufficient for reliable evaluation of latently verifiable trajectories. Supervised fine-tuning produces a much larger gain by teaching the model the intrinsic-evidence evaluation workflow. Reinforcement learning further improves evaluator accuracy, showing the benefit of post-training the evaluator toward task-level correctness. Finally, adding extrinsic-evidence evaluation through the scaffolded handoff yields the strongest result, demonstrating that InEx benefits from combining trained trajectory-internal evidence inspection with environment-grounded evidence acquisition.

Appendix F.6 further evaluates the pipeline with a smaller Qwen2.5-VL-3B intrinsic evaluator across all four environments.

Ablating the scaffolded open-to-frontier handoff. We next isolate the role of the handoff mechanism. A natural alternative to InEx is to ask the frontier model to perform the entire evaluation process directly: first inspect the submitted trajectory, then acquire extrinsic evidence, and finally produce a verdict. This baseline uses frontier-model capability and access to extrinsic evidence, but it does not receive the structured intrinsic-evidence scaffold produced by InEx-I.

Table 4 separates the benefit of frontier-model interaction from the benefit of the scaffolded handoff. Allowing the frontier model to perform intrinsic and extrinsic evaluation from scratch improves over direct model-as-a-judge evaluation, but remains well below the trained intrinsic-evidence evaluator. This suggests that frontier-model capability alone does not recover the specialized trajectory-inspection workflow learned by InEx-I. InEx achieves the best result by combining both strengths:

Reward Source	Agent Backbone	WebArena	WebVoyager	Avg.	Δ Avg.
VLM-as-Judge	LLaVA-7B	24.6	22.3	23.5	–
InEx	LLaVA-7B	**25.9**	**24.1**	**25.0**	**+1.6**
VLM-as-Judge	LLaVA-34B	–	33.0	33.0	–
InEx	LLaVA-34B	–	**35.4**	**35.4**	**+2.4**

Table 2: **Downstream agent training with evaluator-derived rewards.** We train agents under the same PAE framework while changing only the reward source. Success rates report full task completion; Δ *Avg.* is the absolute improvement over the corresponding VLM-as-Judge baseline.

Configuration	Acc.
Qwen2.5-VL-7B	71.6
+ structured trajectory-inspection prompt	72.8
+ SFT	85.2
+ RL post-training, InEx-I	90.1
+ extrinsic-evidence handoff, InEx	**91.4**

Table 3: **Component-wise ablation of InEx.** We start from the open-source backbone and progressively add structured prompting, supervised fine-tuning, reinforcement learning post-training, and scaffolded extrinsic-evidence evaluation.

Configuration	Intrinsic Source	Extrinsic Evidence	Acc.
Frontier model-as-judge	Frontier prompt	No	81.3
Frontier I+E w/o scaffold	Frontier prompt	Yes	86.8
InEx-I only	Trained InEx-I	No	90.1
InEx	InEx-I scaffold	Yes	**91.4**

Table 4: **Ablation of the scaffolded open-to-frontier handoff.** We compare direct frontier judging, frontier intrinsic-plus-extrinsic evaluation without the trained scaffold, the intrinsic-evidence evaluator alone, and the full scaffolded handoff.

InEx-I provides a structured intrinsic-evidence analysis, and InEx-E conditions on this scaffold to perform extrinsic evidence acquisition. The gain over both frontier I+E without scaffold and InEx-I alone shows that the handoff is useful not merely because the frontier model can interact with the environment, but because it inherits a trained evaluation state before acquiring extrinsic evidence.

6 Conclusion

We introduce InEx, an evidence-based evaluator for latently verifiable agent tasks, where an objective answer or success condition exists in principle but the corresponding ground truth is not cheaply available during evaluation. InEx replaces coarse final-answer judgment with a structured process that reconstructs intrinsic evidence from the submitted trajectory, selectively acquires extrinsic evidence from the environment, and preserves the resulting evidence chain through the final verdict.

InEx combines the trained InEx-I component, the frontier-model InEx-E component, and a scaffolded handoff between them. Across 4,000 trajectories from two web environments, one mobile environment, and one desktop environment, we reuse the same trained InEx-I checkpoint without environment-specific fine-tuning. InEx outperforms generic and specialized evaluator baselines in every environment, including gains of 4.5–6.7 percentage points over WebJudge. Downstream experiments further show that improved evaluator-derived rewards lead to stronger web-agent training.

These results establish intrinsic–extrinsic evidence evaluation across distinct web, mobile, and desktop interaction interfaces while preserving a common trained intrinsic evaluator and evidence-reconstruction procedure. Future work can extend InEx to additional latently verifiable task domains, including data-analysis agents over live or private databases, scientific assistants operating over evolving experimental records, and enterprise agents operating over dynamic organizational state.

AI USE STATEMENT

Generative AI tools were used to assist with drafting and revising portions of the manuscript, improving language and organization, and checking formatting. All AI-assisted content was reviewed, verified, and revised by the authors. The authors take full responsibility for the final content of the paper, including all text, claims, citations, and reported experimental results.

REFERENCES

Anthropic. Claude 3.5 Sonnet model card addendum. Anthropic, 2024. 7, 19

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv*, 2025a. 7

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv*, 2025b. 5, 14

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv*, 2021. 1, 4

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *NeurIPS*, 2017. 1, 4

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv*, 2021. 1, 4

Chaoqun Cui, Jing Huang, Shijing Wang, Liming Zheng, Qingchao Kong, and Zhixiong Zeng. Agentic reward modeling: Verifying GUI agent via progressive trajectory-grounded interaction. In *NeurIPS*, 2026. 3

Jeff Da, Clinton Wang, Xiang Deng, Yuntao Ma, Nikhil Barhate, and Sean Hendryx. Agent-RLVR: Training software engineering agents via guidance and environment rewards. *arXiv*, 2025. 1

Gaole Dai, Shiqi Jiang, Ting Cao, Yuqing Yang, Yuanchun Li, Rui Tan, Mo Li, and Lili Qiu. ProRe: A proactive reward system for GUI agents via reasoner–actor collaboration. In *ICLR*, 2026. 3

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *NeurIPS*, 2023. 3

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 2025. 1

Izzeddin Gur, Hiroki Furuta, Austin V. Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world WebAgent with planning, long context understanding, and program synthesis. In *ICLR*, 2024. 1, 3

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. In *ACL*, 2024. 2, 3, 4, 6, 19

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. In *ICLR*, 2024. 2, 3

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye, Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong, Weiran He, Weixiao Huang, Weixin Xu, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang Liu, Yanru Chen, Yejie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, Zonghan Yang, and Zongyu Lin. Kimi k1.5: Scaling reinforcement learning with LLMs. *arXiv*, 2025. 1

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaEval: An automatic evaluator of instruction-following models. GitHub, 2023. 3

Zehao Li, Zhenyu Wu, Yibo Zhao, Bowen Yang, Jingjing Xie, Zhaoyang Liu, Zhoumianze Liu, Kaiming Jin, Jianze Liang, Zonglin Li, Feng Wu, Bowen Zhou, Zun Wang, and Zichen Ding. OS-Themis: A scalable critic framework for generalist GUI rewards. *arXiv*, 2026. 3

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *ICLR*, 2024. 1

Shikhar Murty, Hao Zhu, Dzmitry Bahdanau, and Christopher D. Manning. NNetNav: Unsupervised learning of browser agents through environment interaction in the wild. In *ICLR Workshop*, 2025. 3

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. WebGPT: Browser-assisted question-answering with human feedback. *arXiv*, 2021. 3

OpenAI. GPT-5 system card. OpenAI, 2025. 7

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022. 1, 4

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. In *COLM*, 2024. 2, 3, 5

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E. Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Kenji Toyama, Robert James Berry, Divya Tyamagundlu, Timothy P. Lillicrap, and Oriana Riva. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *ICLR*, 2025. 3, 4, 6, 19

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, 2023. 3

Chenrui Shi, Yuwei Wu, Yang Liu, Ruining Feng, Zirui Shang, Zhi Gao, Lifeng Fan, and Che Sun. Interactive reward agent: GUI task evaluation via environment-state verification. *arXiv*, 2026. 3

Linxin Song, Jieyu Zhang, Huanxin Sheng, Taiwei Shi, Rahul Gupta, Yang Liu, Ranjay Krishna, Jian Kang, and Jieyu Zhao. Reward modeling for computer-using agent from video execution. In *ECCV*, 2026. 3

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. WebAgent-R1: Training web agents via end-to-end multi-turn reinforcement learning. In *EMNLP*, 2025. 1

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *NeurIPS*, 2024. 3, 4, 6, 19

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. An illusion of progress? assessing the current state of web agents. In *COLM*, 2025. 3, 7

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *ICLR*, 2023. 1, 3

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *NeurIPS*, 2023. 2, 3

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *ICLR*, 2024. 3, 4, 6, 19

Yifei Zhou, Qianlan Yang, Kaixiang Lin, Min Bai, Xiong Zhou, Yu-Xiong Wang, Sergey Levine, and Li Erran Li. Proposer-agent-evaluator (PAE): Autonomous skill discovery for foundation model internet agents. In *ICML*, 2025. 2, 3, 5, 8, 15

Mingchen Zhuge, Changsheng Zhao, Dylan R. Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-Judge: Evaluate agents with agents. In *ICML*, 2025. 3

A INTRINSIC-EVIDENCE EVALUATION WORKFLOW

This appendix provides additional details for the intrinsic-evidence evaluation workflow defined by InEx-I. The goal of this workflow is to transform a long submitted trajectory into a structured evidence chain that supports a final task-completion verdict.

A.1 INPUT FORMAT

Given a task instruction x, a submitted trajectory τ, and a final answer y, the intrinsic-evidence evaluator receives the trajectory as an ordered sequence of local transitions. We write each transition as

$$\Delta_t = (o_t, r_t, a_t, o_{t+1}), \tag{5}$$

where o_t and o_{t+1} are the observations before and after the agent action, a_t is the action taken by the agent, and r_t is the agent's reasoning trace or decision text associated with step t. When explicit reasoning traces are unavailable, r_t is treated as empty and the evaluator relies on observations and actions.

A.2 STEP-LEVEL EVIDENCE INSPECTION

The evaluator first inspects each transition independently in its local context. For transition Δ_t, the evaluator infers what the agent is attempting to do, whether the reasoning is consistent with the observed environment, and what effect the action has achieved. This produces a step-level evidence record

$$z_t = \Phi_{\mathrm{I}}(x, \Delta_t). \tag{6}$$

Conceptually, z_t contains the agent's local intent, the evidence observed at the step, the grounding status of the reasoning, the action outcome, and any local failure or unsupported claim. These records are not treated as independent task-level verdicts. Instead, they serve as intermediate summaries that preserve local evidence for later aggregation.

A.3 TRAJECTORY-LEVEL AGGREGATION

After constructing step-level records, the evaluator identifies the subset of steps that are relevant to the claimed task completion:

$$\mathcal{P} \subseteq \{1, \ldots, T\}. \tag{7}$$

The selected set $\mathcal{P}$ is used to reconstruct an evidence chain connecting the agent's intermediate observations, actions, and reasoning to its final answer. This aggregation step is important because task completion is often supported by evidence distributed across multiple trajectory steps, while failures may arise from local inconsistencies that are easy to miss in a flattened trajectory prompt.

A.4 FINAL INTRINSIC VERDICT

The final intrinsic evidence chain and verdict are produced from the instruction, final answer, selected step-level records, and the original transition details for the selected steps:

$$(H_{\mathrm{I}}, \hat{v}_{\mathrm{I}}) = \Psi_{\mathrm{I}}\left(x,\, y,\, \{(z_t, \Delta_t)\}_{t \in \mathcal{P}}\right), \tag{8}$$

where H_{I} is the structured intrinsic evidence chain and $\hat{v}_{\mathrm{I}} \in \{0, 1\}$ is the predicted intrinsic success verdict. A positive verdict indicates that the final answer is supported by the trajectory-internal evidence; a negative verdict indicates that the submitted process is missing necessary evidence, contains unsupported reasoning, or contradicts the claimed answer.

B TRAINING DETAILS FOR INEX-I

This appendix provides additional details for training InEx-I. We initialize it from Qwen2.5-VL-7B-Instruct (Bai et al., 2025b) and train it in two stages: supervised fine-tuning followed by reinforcement learning post-training. We use the Qwen2.5-VL family because its open weights support both training stages and its native interleaved image-text interface matches screenshot-action trajectories. We use

the 7B variant for the primary experiments to balance multimodal reasoning capacity with tractable training, keep it fixed across environments for controlled comparison, and report a 3B variant in Appendix F.6 to examine size sensitivity. The goal is to teach the model a bottom-up intrinsic-evidence evaluation workflow while improving its ability to distinguish grounded task completion from unsupported, hallucinated, or incomplete trajectories. This specialization is important because a generic prompted model may default to judging whether the final response appears plausible instead of reconstructing whether the action-conditioned trajectory actually supports it.

B.1 Supervised Fine-Tuning Data

The supervised fine-tuning dataset is designed to teach InEx-I the intrinsic-evidence workflow described in Appendix A. Each training example consists of an input (x, τ, y), where x is the task instruction, τ is the submitted trajectory, and y is the final answer, together with an intrinsic-evidence evaluation trace h and a binary verdict v.

The SFT dataset contains two types of examples. The first type consists of verified evaluation traces. We start from human-labeled web-agent trajectories from PAE (Zhou et al., 2025). For each trajectory, we run a frontier model with our intrinsic-evidence evaluation workflow to generate step-level analysis, trajectory-level reasoning, and a final verdict. We retain only examples whose generated verdict agrees with the human label, treating them as high-quality demonstrations of intrinsic-evidence evaluation.

The second type consists of synthetic hard negatives. These examples are constructed to expose the evaluator to plausible but unsupported agent behavior. Such cases are important because latently verifiable agent trajectories often contain fluent final answers even when the underlying process is incorrect or ungrounded.

B.2 Synthetic Negative Construction

We construct synthetic negatives using two strategies.

First, starting from successful trajectories, we randomly select task-relevant steps and perturb the corresponding reasoning traces or actions. These perturbations create inconsistencies between the local observation, the agent's stated reasoning, and the action outcome. The resulting trajectories often remain superficially plausible, but their process no longer supports the original success label. We label these examples as failures, encouraging the evaluator to detect local grounding errors and right-answer-wrong-reason trajectories.

Second, we train a non-interactive trajectory generator that receives only the task instruction and directly outputs a complete trajectory and final answer without actually interacting with the web environment. The generator is trained to imitate successful trajectories, but because it does not observe the environment, its outputs often contain fabricated evidence, unsupported process claims, or hallucinated intermediate observations. These generated trajectories provide realistic hard negatives for training the evaluator to reject ungrounded evidence chains.

Formally, the SFT dataset is

$$\mathcal{D}_{\text{sft}} = \mathcal{D}_{\text{ver}} \cup \mathcal{D}_{\text{syn}}, \tag{9}$$

where $\mathcal{D}_{\text{ver}}$ denotes verified demonstrations and $\mathcal{D}_{\text{syn}}$ denotes synthesized examples. We train InEx-I with the standard next-token prediction objective:

$$\mathcal{L}_{\text{sft}}(\theta) = -\mathbb{E}_{(x,\tau,y,h,v)\sim\mathcal{D}_{\text{sft}}} \log p_\theta(h, v \mid x, \tau, y). \tag{10}$$

This stage teaches the model to follow the structured evaluation format and to connect local transition analysis with the final task-level judgment.

B.3 Reinforcement Learning Post-Training

After supervised fine-tuning, we further optimize InEx-I with reinforcement learning post-training. At each training iteration, an agent generates trajectories on a set of seed web-agent tasks, and the current evaluator produces evaluation traces and verdicts for these trajectories. The evaluator then receives reward feedback indicating whether its verdict is correct.

The reward-labeling process uses two sources of supervision. For a subset of seed tasks, we manually construct rule-based ground-truth checkers. These checkers specify task-specific success conditions, such as whether the final answer matches a required value, whether a required webpage state is reached, or whether a necessary condition is satisfied in the environment. For these tasks, the reward is determined by agreement with the rule-based success label:

$$R_i = \mathbb{I}[\hat{v}_i = v_i^{\text{rule}}], \qquad c_i = 1, \tag{11}$$

where R_i is the reward and c_i is the confidence score.

For tasks where reliable rule-based labels are unavailable, we use frontier models as reward judges. Given the evaluator's reasoning process and verdict, the frontier model assesses whether the evaluation is correct and assigns both a reward R_i and a confidence score $c_i \in [0, 1]$. This provides imperfect but useful supervision for tasks whose ground truth cannot be cheaply specified.

B.4 Confidence-Aware Imperfect Supervision

Because some reward labels are produced by imperfect model-based judges, we weight each evaluator rollout by a scalar learning value:

$$\omega_i = \text{Value}(R_i, c_i, \rho_i), \tag{12}$$

where R_i is the reward, c_i is the confidence score, and ρ_i denotes a task-distribution factor associated with the training instance. Intuitively, high-confidence correct evaluations should provide stronger learning signals, while uncertain or lower-value samples should contribute less to the update.

We then use these values as weighted RL feedback to update the intrinsic-evidence evaluator:

$$\max_{\theta} \ \mathbb{E}_{(x_i,\tau_i,y_i)} \left[\omega_i \, \mathcal{R}_\theta(x_i, \tau_i, y_i)\right], \tag{13}$$

where $\mathcal{R}_\theta$ denotes the RL objective induced by the evaluator rollout. This confidence-aware procedure allows InEx-I to improve beyond supervised demonstrations while accounting for the imperfect nature of many reward labels.

B.5 Compute Requirements

Training InEx-I requires substantial compute because the full pipeline includes supervised fine-tuning, reinforcement learning post-training, repeated evaluator rollouts, and reward labeling over generated web-agent trajectories. In our implementation, running the complete training pipeline requires 32 NVIDIA A100 GPUs with 40GB memory. This compute requirement mainly comes from long-context trajectory processing, multi-rollout RL post-training, and parallel reward evaluation. Evaluation-only experiments are substantially cheaper, but reproducing the full training setup requires comparable multi-GPU infrastructure.

C Extrinsic-Evidence Evaluation Details

This appendix provides additional details for InEx-E. Unlike intrinsic-evidence evaluation, which only reasons over the submitted trajectory, extrinsic-evidence evaluation allows the evaluator to acquire information beyond the trajectory through evaluator-initiated environment interaction. We implement this component with a frontier model whose general reasoning and interaction capabilities complement the specialized trajectory inspection learned by InEx-I. For example, when a submitted trajectory claims that no valid result exists, the intrinsic analysis may reveal that the claim is unsupported but cannot determine whether the environment truly contains no result or whether the evaluated agent searched incompletely; InEx-E can revisit the environment to resolve the claim.

C.1 Extrinsic-Evidence Inputs

For each evaluation instance, InEx-E receives the task instruction x, the submitted trajectory τ, the final answer y, the intrinsic scaffold S_{I}, and the benchmark-native environment references associated with the agent's interaction:

$$\mathcal{R}^{(b)} = \{r_t^{(b)}\}_{t=1}^{T}, \tag{14}$$

Here, b indexes the benchmark and $r_t^{(b)}$ denotes the environment reference associated with step t, such as a URL or browser state for web tasks, an Android application state, or a desktop application state. These references expose only information available through the corresponding official benchmark interface; the evaluator never receives the official reference verdict.

C.2 Action Space

InEx-E is given the official interaction space of benchmark b:

$$\mathcal{A}_{\mathrm{E}}^{(b)} = \mathcal{A}_{\mathrm{official}}^{(b)} \cup \{\texttt{stop}\}. \tag{15}$$

For WebVoyager and WebArena, these are the benchmark-supported browser operations, including navigation, clicking, and typing. AndroidWorld and OSWorld analogously expose only their official mobile and desktop operations. The evaluator can therefore collect decision-relevant evidence without introducing an unofficial control channel. The `stop` action indicates that the intrinsic scaffold and any additionally acquired evidence are sufficient for a verdict; InEx-E may stop without issuing an environment action.

C.3 Final Verdict

After reviewing the intrinsic scaffold and, when needed, acquiring additional environment evidence, InEx-E outputs the final binary verdict

$$\hat{v} \in \{0, 1\}. \tag{16}$$

A positive verdict indicates that the available intrinsic and environment evidence supports successful task completion; a negative verdict indicates that the evidence contradicts the final answer, shows missing required conditions, or fails to support the agent's claimed completion. In InEx, this final reasoning is conditioned on the intrinsic scaffold produced by InEx-I, as described in Appendix D.

C.4 Interaction Budget and Stopping Criteria

During extrinsic-evidence evaluation, the evaluator is encouraged to acquire only decision-relevant evidence. It does not indiscriminately rerun the submitted task: it first reviews the intrinsic scaffold, terminates without an environment action when that evidence is sufficient, and otherwise targets the unresolved claims most likely to change the verdict. It stops when it has enough information to determine whether the submitted answer is supported by the environment, or when the interaction budget is exhausted. The WebVoyager configuration uses a 20-step cap; all environments retain the restrictions of their official interfaces. Typical stopping conditions include verifying that an environment state supports the final answer, finding evidence that contradicts the submitted answer, confirming that a claimed no-result or impossible condition is valid, or determining that further interaction is unlikely to change the verdict.

C.5 Agent-Query Variant

We also consider an optional form of extrinsic evidence based on interaction with the evaluated agent. In this variant, the evaluated agent does not terminate immediately after submitting its final answer. Instead, it remains available for follow-up questions from the evaluator while being restricted from further Internet access. The evaluator receives an additional action,

$$\texttt{ask_agent}(q), \tag{17}$$

which allows it to ask the agent about specific claims, intermediate decisions, or missing evidence in the submitted trajectory.

The agent's responses may provide useful clarification and can help the evaluator identify which parts of the trajectory require further verification. However, this protocol requires modifying the evaluated agent and may introduce bias toward agents with stronger post-hoc explanation or dialogue capabilities. It is also less practical for some open-source training-based web agents. For these reasons, we do not use the agent-query variant in the main experiments and focus on environment-based extrinsic verification.

D SCAFFOLDED OPEN-TO-FRONTIER HANDOFF DETAILS

This appendix provides additional details for the scaffolded open-to-frontier handoff used to combine InEx-I and InEx-E in InEx. The goal of the handoff is to allow a frontier model to benefit from the trained intrinsic-evidence workflow learned by the open-source evaluator, rather than restarting evaluation from scratch.

D.1 INTRINSIC ANALYSIS SCAFFOLD

Given an instruction x, submitted trajectory τ, and final answer y, InEx-I first produces an intrinsic-evidence analysis:

$$(H_{\mathrm{I}}, \hat{v}_{\mathrm{I}}) = \text{InEx-I}(x, \tau, y), \qquad S_{\mathrm{I}} = (H_{\mathrm{I}}, \hat{v}_{\mathrm{I}}). \tag{18}$$

Here, H_{I} is the structured intrinsic evidence chain assembled from step-level records and task-relevant transitions, and $\hat{v}_{\mathrm{I}}$ is the intrinsic verdict. Together they form S_{I}, which summarizes how the trajectory supports or contradicts the final answer and identifies unresolved or ambiguous claims that may require extrinsic evidence.

D.2 CONVERSATION CONSTRUCTION

We convert S_{I} into a conversation history for the frontier model. The frontier model is given the original evaluation input, including the task instruction, submitted trajectory, final answer, and available environment references. The intrinsic analysis is then inserted into the conversation as a structured prior evaluation. The frontier model is prompted to continue from this state, treating the intrinsic analysis as an initial scaffold rather than as a final answer.

The scaffold contains three types of information:

- **Selected evidence chain:** task-relevant trajectory steps and step-level summaries;
- **Unresolved claims:** claims whose truth cannot be determined from the submitted trajectory alone;
- **Intrinsic verdict and rationale:** the intrinsic-evidence evaluator's preliminary success judgment and supporting reasoning.

D.3 HANDOFF EXECUTION

Conditioned on the original input and the intrinsic scaffold, InEx-E performs extrinsic evidence acquisition through the official interface of the current benchmark. It may revisit browser, mobile, or desktop application state to verify the unresolved or decision-relevant claims identified by InEx-I. The final verdict is then produced by combining the intrinsic scaffold S_{I} with any acquired extrinsic evidence $\mathcal{E}_{\mathrm{E}}$.

In the complete InEx pipeline, this handoff is applied to every instance and InEx-E owns the final verdict. Handoff does not, however, imply environment interaction on every trajectory: the frontier component may confirm or revise the intrinsic judgment directly from the scaffold and acquires new observations only when additional evidence is needed.

The full process can be written as

$$S_{\mathrm{I}} = (H_{\mathrm{I}}, \hat{v}_{\mathrm{I}}) = \text{InEx-I}(x, \tau, y), \qquad \hat{v} = \text{InEx-E}(x, \tau, y, \mathcal{R}^{(b)}, \mathcal{A}_{\mathrm{E}}^{(b)}; S_{\mathrm{I}}). \tag{19}$$

This formulation makes the role of the handoff explicit: the frontier evaluator is not merely prompted to judge the trajectory from scratch, but is conditioned on a structured intrinsic-evidence state produced by a trained evaluator.

D.4 RATIONALE FOR THE HANDOFF

The handoff is motivated by the trainability–capability trade-off. Open-source models can be fine-tuned to follow a specialized intrinsic-evidence workflow, but may lack strong environment-interaction capabilities. The frontier service used in our implementation provides stronger general reasoning and

interaction abilities but is accessed without task-specific fine-tuning. By passing the trained intrinsic analysis to the frontier model, InEx combines the strengths of both: specialized trajectory inspection from InEx-I and environment-grounded evidence acquisition from InEx-E.

E Evaluation Sets and Reference Outcomes

E.1 Evaluation-Set Construction

Our evaluator benchmark is derived from WebVoyager (He et al., 2024). We run Claude 3.5 Sonnet (Anthropic, 2024) on WebVoyager tasks and retain 1,000 complete web-agent trajectories. Each instance contains the task instruction, the ordered multimodal interaction trajectory, the agent's submitted final answer, and a human binary task-completion label. The set spans 13 websites. These labels are used only to measure evaluator predictions; no evaluated method receives the label as input.

This construction targets evaluation errors that cannot be diagnosed reliably from a final answer alone. A trajectory preserves the observations available to the agent, its intermediate reasoning or decision text when present, its actions, and the resulting visual states. Consequently, the benchmark supports both holistic judging baselines and methods, such as InEx-I, that audit whether the claimed outcome follows from the recorded process.

E.2 Task-Level Split Control

We separate evaluator training and evaluation at the underlying task level rather than only at the trajectory level. If any trajectory generated from a WebVoyager task is used to construct SFT or RL data for InEx-I, every trajectory from that task is excluded from the 1,000-instance evaluation set.

E.3 Human Reference Judgments

For each evaluation trajectory, the human label records whether the agent completed the requested task, not whether its final response merely appears plausible. The evaluator predicts this instance-level reference outcome without receiving the label at inference time. This distinction is central to our formulation: the benchmark curator has a human reference judgment, while the evaluator lacks an inexpensive oracle that exposes it for a new submitted trajectory.

E.4 Cross-Environment Evaluation Protocol

We supplement the 1,000 WebVoyager trajectories with 1,000 complete trajectories each from WebArena (Zhou et al., 2024), AndroidWorld (Rawles et al., 2025), and OSWorld (Xie et al., 2024). WebVoyager and WebArena provide two distinct web-interaction settings, AndroidWorld provides mobile-device interaction, and OSWorld provides desktop interaction. For the three additional benchmarks, we run the official task evaluator on each submitted trajectory to obtain a fixed binary reference outcome. These task-specific evaluators may inspect benchmark state that is unavailable to a model-based evaluator. Their outcomes are used only after model inference to score candidate predictions; neither the verdict nor the official evaluator implementation is available to any candidate evaluator during inference.

For each added environment, trajectories are collected with the benchmark-provided standard agent under its official rollout protocol, using Claude 3.5 Sonnet as in our WebVoyager collection. We follow the default task distribution without manual rebalancing and use the official benchmark evaluator without replacing it with a learned judge.

We evaluate the same trained InEx-I checkpoint in all four environments without environment-specific fine-tuning or other parameter updates. InEx-E uses each benchmark's native observation format and only operations supported by its official environment interface. Thus, the environment wrapper changes with the interaction surface, while the trained intrinsic evaluator and the intrinsic–extrinsic evidence procedure remain fixed. All reported results are mean $\pm$ standard deviation across three evaluator executions on a fixed 1,000-trajectory set.

Benchmark	Surface	N	Reference Outcome	InEx-E Interface
WebVoyager	Web	1,000	Human judgment	Official browser operations
WebArena	Web	1,000	Official task evaluator	Official web operations
AndroidWorld	Mobile	1,000	Official task evaluator	Official Android operations
OSWorld	Desktop	1,000	Official task evaluator	Official desktop operations

Table 5: **Cross-environment evaluation protocol.** Reference outcomes are hidden from all evaluated methods. No environment-specific parameter training is performed for the three added environments.

E.5 Website-Function Reporting Groups

For compact presentation, we aggregate the 13 websites into five fixed functional groups: Commerce/Services, Knowledge/Learning, Developer Platforms, Search/Navigation, and News/Sports. The grouping changes only how results are displayed; the overall accuracy in Table 6 is computed directly over the 1,000 trajectories and is not an unweighted mean of the five group accuracies.

Release artifacts. An auditable release will include a de-identified instance manifest, the task-level train/evaluation exclusion list, the website-to-group mapping, label and adjudication records, and the script used to reproduce per-group and overall metrics, subject to source-website terms, benchmark licensing, privacy constraints, and venue anonymity requirements.

F Additional Results

F.1 Detailed WebVoyager Results

		Commerce Services	Knowledge Learning	Developer Platforms	Search Navigation	News Sports	Overall
Frontier	Claude 3.5 Sonnet	75.1±0.8	80.8±1.3	81.3±0.6	82.5±1.1	73.9±0.5	78.7±0.8
	GPT-5.0	76.7±0.7	85.2±1.0	86.9±0.6	88.5±0.9	74.1±0.4	81.3±0.7
Open-source	Qwen3-VL-8B	74.2±1.9	76.2±2.4	75.5±1.4	76.5±2.7	73.1±1.6	75.3±1.9
Structured	WebJudge	83.6±0.6	90.1±0.8	88.7±0.4	89.8±0.7	83.9±0.5	86.7±0.6
Ours	InEx-I	87.0±0.4	92.4±0.5	91.6±0.3	92.1±0.4	86.3±0.2	90.1±0.3
	InEx	**89.7±0.3**	**93.6±0.4**	**92.0±0.2**	**92.3±0.4**	**88.2±0.2**	**91.4±0.3**

Table 6: **Evaluator accuracy on WebVoyager by website-function group (in %).** Results are mean ± standard deviation over three runs. *Overall* is computed over all 1,000 trajectories rather than as an unweighted mean of the five displayed groups.

F.2 Reward-Relevant Evaluator Diagnostics

Because evaluator outputs are used not only for benchmarking but also as reward signals for downstream agent training, aggregate accuracy alone does not fully characterize reward quality. In particular, false positives assign positive rewards to failed or unsupported trajectories, while false negatives discard useful successful behavior. We therefore report diagnostic metrics that separate these reward-relevant error types.

We treat success as the positive class and define true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) with respect to the human-annotated success label. We report success precision, success recall, failure recall, and balanced accuracy:

$$\text{Success Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}, \qquad \text{Success Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}, \tag{20}$$

$$\text{Failure Recall} = \frac{\text{TN}}{\text{TN}+\text{FP}}, \qquad \text{Balanced Acc.} = \frac{1}{2}\left(\text{Success Recall} + \text{Failure Recall}\right). \tag{21}$$

Success precision measures the reliability of positive success predictions, success recall measures how often truly successful trajectories are accepted, and failure recall measures how often failed trajectories are correctly rejected.

		Acc.	Success Precision	Success Recall	Failure Recall	Balanced Acc.
Frontier	Claude 3.5 Sonnet	78.7	83.4	80.7	75.6	78.2
	GPT-5.0	81.3	85.7	82.7	79.1	80.9
Open-source	Qwen3-VL-8B	75.3	78.8	80.7	67.1	73.9
Ours	InEx-I	90.1	93.7	**89.5**	91.0	90.2
	InEx	**91.4**	**97.6**	87.9	**96.7**	**92.3**

Table 7: **Reward-relevant evaluator diagnostics.** We report evaluator diagnostics on the full WebVoyager evaluation set. All values are percentages. InEx achieves the highest success precision, failure recall, and balanced accuracy, indicating that it is more selective in assigning positive success labels and better at rejecting failed trajectories.

As shown in Table 7, InEx achieves the best overall accuracy, success precision, failure recall, and balanced accuracy. Compared with InEx-I, the complete InEx evaluator is more selective: it slightly reduces success recall, but substantially improves success precision and failure recall. This behavior is desirable for reward assignment, since false-positive success predictions would reinforce failed trajectories during downstream agent training.

F.3 Standalone Extrinsic-Evidence Evaluation

Evaluator	Commerce Services	Knowledge Learning	Developer Platforms	Search Navigation	News Sports	Overall
InEx-E only	82.4	86.1	86.3	87.2	80.9	83.5

Table 8: **Standalone InEx-E accuracy on WebVoyager (in %).** The extrinsic evaluator is effective on its own, while remaining complementary to the structured intrinsic-evidence stage.

For this standalone ablation, InEx-E receives the task, trajectory, final answer, and environment references directly, without the InEx-I scaffold. Standalone InEx-E reaches 83.5% overall accuracy, showing that active environment inspection can independently recover useful verification evidence. Its accuracy nevertheless remains below InEx-I (90.1%) and the complete InEx evaluator (91.4%), supporting its intended role as a targeted source of additional evidence when the submitted trajectory is insufficient, rather than as a replacement for structured intrinsic verification.

F.4 Evaluation Cost and Reproducibility

Our repeated-run results report mean and standard deviation over three evaluator executions. We will release the corresponding predictions, aggregation code, prompts, model versions, and preprocessing details to support exact reproduction of the comparisons and decision-level analyses.

On WebVoyager, extrinsic verification is selective: environment actions are invoked for 23.6% of trajectories, and action-taking cases average 5 steps and 29.1 seconds under a 20-step cap. These measurements characterize the interaction overhead of the evaluated WebVoyager configuration; absolute latency may vary with the environment, network conditions, and frontier-model service.

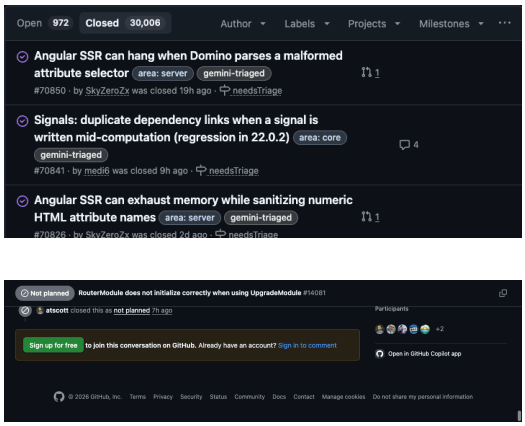
F.5 Analysis of Evaluator Decisions

Intrinsic evidence exposes an unsupported count. In the Amazon case in Figure 3a, the agent answers "six colors." Its sole basis is a Step 3 attempt to count sidebar items 47–52. The harness rejects this action, and the observation shows five gallery photographs plus a video thumbnail—not six color selectors. No later step establishes the total. Applying the InEx-I evidence-reconstruction procedure therefore exposes an unsupported claim without inferring an alternative; the record does not determine the true color count.

Extrinsic evidence reveals an omitted recent issue. In the GitHub case in Figure 3b, the agent reports #70850, #70841, and #70826 as the three most recently closed issues in `angular/angular`.



(a) **Intrinsic evidence.** The actor treated the six labeled gallery items (47–52) as color options, although they are product photographs and a video thumbnail.



(b) **Extrinsic evidence.** The submitted trajectory shows three closed issues (top), whereas targeted verification finds omitted issue #14081 and its closure event (bottom).

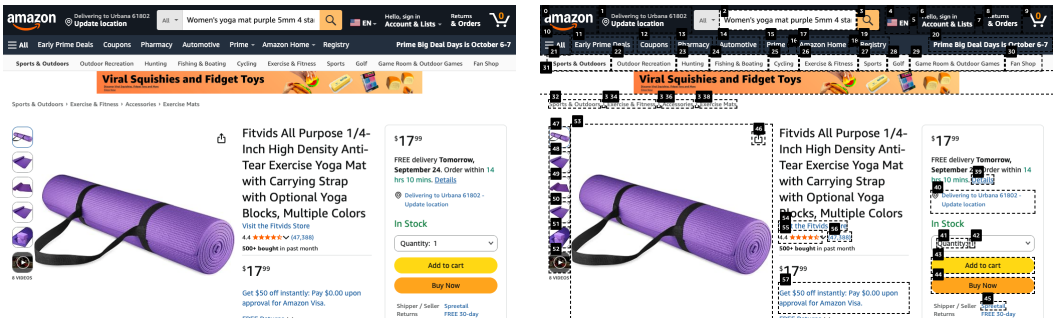
Figure 3: **Audited examples of evidence-based evaluation.** A screenshot-based GPT-4o evaluator using the official WebVoyager prompt marked both answers successful in all three seeded runs. Intrinsic inspection exposes an unsupported count in (a), while targeted extrinsic verification supplies a decisive missing fact in (b).

The visible list confirms their existence but not their recency. The InEx-E verification procedure checks timestamped histories and finds that omitted issue #14081 closed at 2026-09-22 23:38 UTC—later than all three returned issues and before the agent visit at 2026-09-23 04:26 UTC. Thus, the answer was already incomplete rather than made stale by a later update. This one counterexample suffices to reject the claimed set without reconstructing the complete top three.

Reward-relevant error profile. The comprehensive diagnostic results in Appendix F.2 (Table 7) show that InEx improves not only aggregate accuracy but also success precision, failure recall, and balanced accuracy. In particular, its 97.6% success precision and 96.7% failure recall show that the primary gain comes from reducing false-positive success judgments—the error type most likely to corrupt evaluator-derived rewards.

Selective extrinsic verification. For every input, InEx-I first produces a structured intrinsic analysis, which is passed to InEx-E. On the WebVoyager evaluation set, InEx-E invokes environment actions for only 23.6% of trajectories; among these action-taking cases, interaction averages 5 steps under a 20-step cap. Thus, the extrinsic component is a targeted complement to intrinsic verification rather than an additional full rollout for every example.

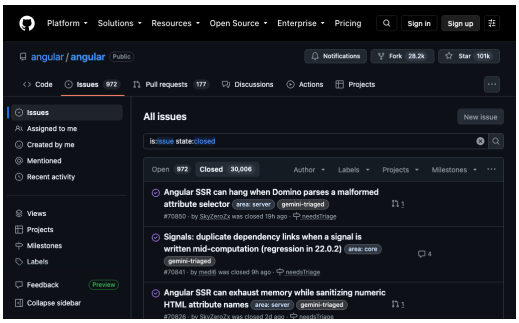
Impact on downstream agent training. Under the matched PAE setup in Table 2, replacing the generic judge with InEx improves the LLaVA-7B agent from 24.6 to 25.9 on WebArena and from 22.3 to 24.1 on WebVoyager. The same replacement improves the LLaVA-34B agent from 33.0 to 35.4 on WebVoyager. The gains across two agent capacities and two benchmarks show that more reliable evaluator-derived rewards translate into stronger downstream agents.



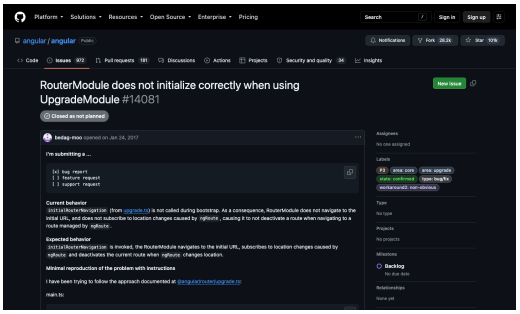
(a) Uncropped product-page observation. (b) Exact numbered observation supplied to the actor.

Raw record	Verbatim excerpt
Actor, Step 3	*Thought:* "The product page shows there are more color options available, visible on the left sidebar." *Action:* "Count the number of color options by analyzing the sidebar images. (Identified labeled elements: 47, 48, 49, 50, 51, 52)."
Actor, Step 7	"The sidebar shows six different color options." *Final answer:* "The women's yoga mat is available in six colors. It offers free delivery with the option of a 30-day return."
Screenshot-only judge	"The screenshot shows multiple color options, supporting the claim of multiple colors, though the exact number isn't visible." **Verdict: SUCCESS**
Audited check	The six numbered items are five gallery photographs and one video tile. The Step 3 action was rejected, and no subsequent observation establishes the color total.

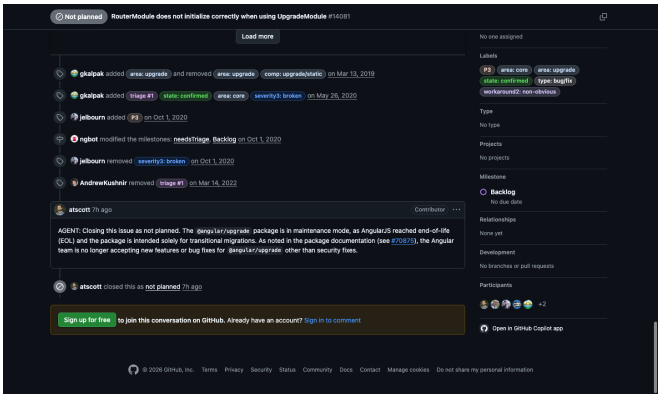
Figure 4: **Raw trajectory record for the intrinsic-evidence case.** The first row of panels preserves the uncropped product page and exact numbered actor input; the table reproduces the decisive actor and baseline-evaluator excerpts. The screenshot-only GPT-4o judge returned SUCCESS for seeds 42–44. The record supports the narrower audit conclusion that the six-color claim is unsupported; it does not establish the true number of colors.



(a) Uncropped closed-issue list in the submitted trajectory.



(b) The omitted issue #14081 and its closed state.



(c) Raw issue history showing the closing comment and event.

Verified closure events (UTC)

Issue	Answer	Closed
#70850	Yes	Sep 22 08:34:02
#70841	Yes	Sep 22 19:24:20
#70826	Yes	Sep 21 08:54:20
#14081	**No**	**Sep 22 23:38:00**

Actor visit: Sep 23 04:26:24–04:27:13 UTC. The omitted closure therefore predates the visit and postdates every returned issue.

Raw record	Verbatim excerpt or preserved check
Actor, Step 6	"I can see the closed issues now. I will note the last three closed issues displayed in the list." The final answer names the visible entries corresponding to #70850, #70841, and #70826.
Screenshot-only judge	"The information in the result response matches the details in the screenshots." **Verdict: SUCCESS**
Audited extrinsic check	Public, read-only issue histories show that omitted #14081 closed later than all three returned issues and before the actor visit. A single verified counterexample is sufficient to reject the claimed latest-three set.

Figure 5: **Raw trajectory and external record for the extrinsic-evidence case.** Panels (a)–(c) expose the submitted observation, omitted issue identity, and preserved closure event. Absolute event times were rechecked against public GitHub issue histories; relative-age labels from screenshots captured at different times are not compared. The screenshot-only GPT-4o judge returned SUCCESS for seeds 42–44.

F.6 Effectiveness with a Smaller Intrinsic Evaluator

To test whether the gains are tied to the 7B intrinsic evaluator used in our primary experiments, we additionally instantiate the pipeline with Qwen2.5-VL-3B. We evaluate the base model as a direct judge, apply the InEx-I training procedure, and then connect the trained intrinsic evaluator to InEx-E through the scaffolded handoff. Here, "3B" refers only to the trainable intrinsic component; the complete pipeline retains the frontier-model InEx-E component described in Section 4.1.2. Table 9 reports the corresponding accuracy on the four evaluation sets from Table 1.

Configuration	WebVoyager *Web*	WebArena *Web*	AndroidWorld *Mobile*	OSWorld *Desktop*	Macro Avg.
Qwen2.5-VL-3B direct judge	60.1±3.6	61.3±2.8	58.6±4.2	56.2±4.5	59.1
+ intrinsic-evaluator training (InEx-I)	76.1±2.5	77.2±2.2	80.3±1.5	72.6±2.5	76.6
+ scaffolded InEx-E handoff (InEx)	**80.2±1.5**	**85.9±2.0**	**86.2±1.8**	**81.5±1.3**	**83.5**

Table 9: **Results with a smaller Qwen2.5-VL-3B intrinsic evaluator (in %).** Accuracy is reported as mean ± standard deviation over three runs. *Macro Avg.* is the unweighted mean of the four environment-level means.

Training the 3B intrinsic evaluator with InEx-I improves its macro-average accuracy from 59.1% to 76.6%, with gains in every environment. Adding the scaffolded InEx-E handoff yields a further 6.9-point improvement to 83.5%. These consistent gains complement the primary 7B results and show that the benefits of intrinsic-evidence training and extrinsic-evidence acquisition are not confined to a single intrinsic-evaluator scale.

G Limitations and Future Work

While InEx improves evaluator reliability for latently verifiable agent tasks, several limitations remain.

First, InEx relies on a frontier model for InEx-E and extrinsic-evidence evaluation. This design is motivated by the strong general reasoning and interaction capabilities of frontier models, but it also introduces cost, latency, and reproducibility challenges. The exact behavior of closed-source frontier models may change over time, and access to these models may not be available in all deployment settings. Future work should investigate stronger open-source extrinsic-evidence evaluators, cheaper evidence-acquisition policies, and distillation methods that reduce reliance on frontier-model APIs.

Second, extrinsic evidence acquisition depends on the availability and stability of the evaluation environment. Websites, mobile applications, and desktop applications may change over time, become inaccessible, require authentication, or behave nondeterministically. These changes can affect both evaluation reproducibility and the interpretation of success labels. Although our experiments are conducted through official benchmark interfaces, applying InEx in live environments would require careful handling of dynamic content, environment versioning, interaction budgets, and irreversible actions.

Third, our training procedure for InEx-I uses a mixture of verified labels, synthetic negatives, rule-based checkers, and confidence-weighted frontier-model feedback. This improves scalability under limited ground-truth supervision, but the resulting evaluator can still inherit biases or errors from imperfect supervision. In particular, model-based reward judges may occasionally approve incorrect evaluations or penalize valid but unusual reasoning traces. More principled uncertainty estimation, active human annotation, and calibration of evaluator confidence are promising directions for improving robustness.

Finally, because InEx outputs can serve as reward signals for downstream agent training, evaluator errors can have training-time consequences. False positives may reinforce failed or unsupported trajectories, while false negatives may discard useful successful behavior. Our diagnostic analysis shows that InEx improves reward-relevant error behavior, but no learned evaluator is immune to reward hacking or adversarial trajectories. Future work should study evaluator robustness under adversarial agent behavior, distribution shift, and iterative optimization against the evaluator.

H BROADER IMPACT

This work aims to improve the reliability of evaluation and reward construction for agentic systems. Since evaluator outputs can be used as reward signals during RLVR-style post-training, more accurate and evidence-grounded evaluators can help reduce the reinforcement of hallucinated answers, unsupported completion claims, and superficially successful but ungrounded trajectories.

Potential benefits. InEx can support safer and more scalable agent development by encouraging evaluators to ground their judgments in both trajectory-internal evidence and independently acquired environment evidence. This can improve benchmark reliability, reduce reliance on costly human annotation, and provide higher-quality reward signals for downstream agent training.

Risks and safeguards. Improved reward construction may also accelerate the training of more capable agents, including agents that could be misused for unauthorized or harmful web activities. Moreover, any evaluator used as a reward model may become a target for reward hacking. We therefore view deployments of InEx-based evaluators as one component of a broader safety stack that should include environment access controls, interaction budgets, anomaly detection, human oversight for high-impact tasks, and periodic human auditing of evaluator decisions.

Privacy considerations. Extrinsic-evidence evaluation may involve revisiting webpages or interacting with applications, files, and device state. Outside controlled benchmarks, this could raise privacy, consent, or terms-of-service concerns, especially when the environment contains user-specific information. Future applications should ensure that evaluators operate only on authorized data sources, avoid unnecessary storage of sensitive content, and respect user consent boundaries.

Outlook. We believe trustworthy agent training requires trustworthy evaluation. Evidence-based evaluators can make reward signals more reliable, but they should be deployed with clear scope, calibrated uncertainty, and safeguards against misuse.