Selecting LLM Judges for Agent Evaluation Pipelines: Accuracy Masking and Review-Burden Tradeoffs

Punitha Ponnuraj
Independent Researcher / Plano, Texas, USA
punitha.p.raj@gmail.com

Abstract

In agent evaluation pipelines, LLM judges are used to evaluate trajectories, and their outputs can determine which trajectories get reviewed, filtered, or passed downstream. Judge selection in these settings relies on headline task-success accuracy. We show that task-success accuracy is insufficient as the sole criterion for judge selection in agent pipelines, a failure mode we call *accuracy masking*. Across three agents, three web-agent benchmarks, and five LLM judge conditions, task-success accuracy remains tightly clustered across judges, varying by 1.6 to 4.4 percentage points depending on the agent, while side-effect accuracy varies by 45 to 50 percentage points across the same judges. GPT-4o and Llama-3.3-70B consistently behave as high-recall but high-false-positive side-effect screeners. GPT-4o-mini consistently misses the majority of side-effect-positive trajectories. Claude 3.7 Sonnet shifts from a relatively balanced profile on GPT-4o agent trajectories to a miss-prone profile on Claude and Qwen agent trajectories. We analyze review-burden tradeoffs for human-in-the-loop triage, showing that low-burden screener strategies achieve 41–53% side-effect recall at 17–25% review burden depending on the evaluated agent, while GPT-4o-based high-recall strategies require reviewing over 70% of trajectories. These results suggest that judge selection in agent pipelines should combine task-success accuracy with dimension-level error profiles, target-agent behavior, and review budget.

1 Introduction

In agent evaluation pipelines, LLM judges are used to evaluate trajectories at scale, and their outputs can determine which trajectories get reviewed, filtered, or passed downstream. Judge selection in these settings typically relies on headline task-success accuracy, the fraction of trajectories where the judge's task-success label matches the expert label. Most agent benchmarks report this metric, and it is easy to compare across judges.

We show that task-success accuracy is insufficient as the sole criterion for judge selection in agent pipelines. A judge used for deployment monitoring needs to detect side effects, a judge used for human-review triage needs to balance recall against review burden, and a judge used for training-data filtering needs to avoid both silent misses and false rejections. These decisions impose different requirements, and a judge that is reliable for one may fail on another without that failure being shown in task-success accuracy.

We call this failure mode *accuracy masking* and study it using AgentRewardBench (Lù et al., 2025), one of the few agent evaluation environments that provides expert-labeled annotations for side effects and looping in addition to task success. We analyze three evaluated agents, five LLM judge conditions, and three web-agent benchmarks, with human expert annotations as the reference. Across all three agents, task-success accuracy varies by 1.6 to 4.4 percentage points across judges, while side-effect accuracy varies by 45.0 to 50.0 percentage points across the same judges. GPT-4o and Llama-3.3-70B are high-recall but high-false-positive screeners on every agent. GPT-4o-mini is miss-prone—it misses the majority of side-effect-positive trajectories—on every agent. Claude 3.7 Sonnet shows agent-dependent behavior, relatively balanced on GPT-4o agent trajectories but miss-prone on Claude and Qwen agent trajectories. None of these differences are visible from task-success accuracy.

We make three contributions. First, we introduce a framework mapping pipeline decisions to the judge properties they require, showing that task-success accuracy is directly relevant for only one of six common pipeline decisions we identify. Second, we provide empirical evidence that accuracy masking holds across three evaluated agents and three

benchmarks, with failure profiles that are partially stable but sensitive to the evaluated agent's trajectory distribution. Third, we analyze review-burden tradeoffs for human-in-the-loop triage, showing that different judges correspond to different operating points and that the right choice depends on the available review budget and target agent.

2 Operational Judge Selection Framework

Agent evaluation pipelines use judge outputs to support different operational decisions, each of which places different demands on judge behavior. Table 1 maps six common pipeline decisions to the judge properties they require, the failure mode that results when those properties are not checked, and the evidence this paper provides for each.

Table 1 does not claim that a single judge solves all of these decisions. Instead, it shows why task-success accuracy is an insufficient default selection metric: different pipeline decisions require different judge properties, and only some of these properties are captured by leaderboard-style accuracy.

3 Experimental Design

We use AgentRewardBench (Lù et al., 2025) as our evaluation testbed. AgentRewardBench provides trajectories from web-agent benchmarks together with pre-computed LLM judge outputs and expert annotations for task success, side effects, and looping. Having both expert annotations and pre-computed judge outputs in one place makes it possible to ask whether judges that look similar on task-success accuracy behave differently on other dimensions.

We fix the analysis to three evaluated agents: GPT-4o (`gpt-4o-2024-11-20`), Claude 3.7 Sonnet, and Qwen-2.5-VL-72B. Each agent has complete trajectory coverage across all five judge conditions and all three benchmarks. We draw trajectories from WebArena (Zhou et al., 2024), WorkArena (Drouin et al., 2024), and VisualWebArena (Koh et al., 2024), covering general web navigation, enterprise knowledge work, and visually grounded web tasks. Table 2 summarizes the dataset.

We evaluate five LLM judge conditions: GPT-4o-mini, GPT-4o, Claude 3.7 Sonnet, Llama-3.3-70B, and Qwen-2.5-VL. Llama-3.3-70B uses the `noscreen` prompt variant, which provides action history without screenshots, while the other four use the `noaxtree` variant. This means judge conditions differ in both model and input configuration, and we treat them as practical judge conditions rather than a controlled model-only comparison.

Expert annotations are included in AgentRewardBench. For trajectories with multiple annotations, we apply majority voting to obtain a single expert label per trajectory and dimension. Tied or unsure labels are excluded on a per-dimension basis. For the side-effect dimension, we report false negative rate (FNR) and false positive rate (FPR) in addition to accuracy, given that side-effect-positive trajectories are sparse and aggregate accuracy can be misleading when positives are rare. We report 95% confidence intervals for side-effect recall, FNR, and FPR using 1,000-iteration bootstrap resampling, and use Fisher's exact test to compare error rates across label slices.

4 Results

4.1 Accuracy masking persists across evaluated agents

Table 3 reports the spread in task-success, side-effect, and looping accuracy across the five judge conditions for each evaluated agent. Across all three evaluated agents, task-success accuracy is tightly compressed across judges, varying by 1.6 to 4.4 percentage points. Side-effect accuracy varies by 45.0 to 50.0 percentage points across the same judges, and looping accuracy varies by 12.6 to 52.5 percentage points.

A judge selected by task-success accuracy would appear similarly reliable across all five judge conditions for every evaluated agent. The same judges diverge sharply on behavior-level dimensions that matter for monitoring, review triage, and reliability analysis. Full per-judge accuracies are reported in Appendix Table 6.

4.2 Side-effect failure profiles differ by judge

Table 4 summarizes side-effect failure profiles across evaluated agents. The profile differences are operationally important because side-effect positives are sparse: a judge can achieve acceptable aggregate accuracy while missing a large fraction of the trajectories that most need review.

GPT-4o and Llama-3.3-70B behave as high-recall but high-false-positive side-effect screeners across evaluated agents. They catch most side-effect-positive trajectories but also flag many side-effect-negative trajectories as problematic. GPT-4o-

Pipeline decision	Relevant judge property	Failure mode if ignored	Evidence in this paper
Benchmark scoring	Task-success accuracy	Incorrect outcome labels	Directly evaluated
Side-effect monitoring	Side-effect recall / FNR	Risky trajectories silently pass	Directly evaluated
Human-review triage	Recall vs. review burden	Too many or too few reviews	Directly evaluated
Training-data filtering	Recall / FPR balance	Bad trajectories retained or good trajectories removed	Motivated by observed label errors
Cost and reliability monitoring	Looping detection	Stuck or repetitive runs missed	Directly evaluated
Target-agent rollout or upgrade	Profile stability across evaluated agents	Judge reliable on one agent fails on another	Directly evaluated across three agents

Table 1: Pipeline decisions and the judge properties they require. Task-success accuracy is an insufficient default selection metric: different decisions require different properties, and only some are captured by leaderboard-style accuracy.

Evaluated agent	N	SE pos.	Loop pos.
GPT-4o	318	17 (5.3%)	145 (45.6%)
Claude 3.7 Sonnet	318	22 (6.9%)	115 (36.2%)
Qwen-2.5-VL	318	32 (10.1%)	205 (64.5%)
Total	954†	71 (7.4%)	465 (48.7%)

Table 2: Dataset summary. All three evaluated agents cover WebArena (WA), WorkArena (WArena), and VisualWebArena (VWA). †954 agent-trajectory pairs correspond to 318 task instances × 3 agents. The same task instances appear across evaluated agents and should not be treated as independent observations.

Agent	Success	SE range	Looping
GPT-4o	2.2pp	49.4pp	22.0pp
Claude 3.7 Sonnet	1.6pp	45.0pp	12.6pp
Qwen-2.5-VL	4.4pp	50.0pp	52.5pp

Table 3: Accuracy ranges (in percentage points) across five judge conditions per evaluated agent. Each entry is the difference between the highest and lowest accuracy among the five judges. Task-success accuracy remains compressed while side-effect and looping accuracy vary substantially.

Judge	Profile	FNR	FPR	Stable?
GPT-4o-mini	Silent misser	.64–.77	.24–.37	Yes
GPT-4o	High-recall over-flagger	.06–.09	.70–.77	Yes
Claude 3.7 Sonnet	Agent-dependent	.47–.59	.15–.24	No
Llama-3.3-70B	High-recall over-flagger	.23–.28	.65–.74	Yes
Qwen-2.5-VL	Inconclusive	.41–.50	.34–.49	Yes

Table 4: Directional side-effect failure profiles across evaluated agents. FNR: false-negative rate on side-effect-positive trajectories. FPR: false-positive rate on side-effect-negative trajectories. Ranges are across three evaluated agents. Stable means the directional profile is the same across all three evaluated agents.

mini is consistently miss-prone—a *silent misser*—with false-negative rates between 0.64 and 0.77 across agents, meaning it passes the majority of side-effect-positive trajectories without flagging them. Claude 3.7 Sonnet is agent-dependent, and Qwen-2.5-VL shows an inconclusive profile.

We define a profile as stable if its directional classification is consistent across all three evaluated agents.

The differences matter because they determine what a pipeline misses or over-flags, not just where a model ranks on a leaderboard. A team using GPT-4o-mini for monitoring would silently miss many side-effect-positive trajectories. A team using GPT-4o or Llama-3.3-70B as a direct alerting mechanism would catch more positives but flag the majority of side-effect-negative trajectories as problematic. Given the sparse positive slice (17 to 32 cases per evaluated agent), these profiles should be interpreted as directional rather than precise estimates. Full side-effect recall, false-negative, and false-positive rates are reported in Appendix Table 7.

4.3 Judge behavior can depend on the evaluated agent

Claude 3.7 Sonnet does not follow a stable profile across evaluated agents. On GPT-4o agent trajectories, it has a relatively balanced side-effect profile, with FNR of 0.47 and FPR of 0.15. On Claude and Qwen agent trajectories, it shifts toward a more miss-prone profile, with FNR rising to 0.53–0.59 while FPR remains low. This shift is not visible from task-success accuracy, which remains tightly clustered for Claude 3.7 Sonnet across all three

agents.

A judge profile observed on one agent's trajectories may not hold when the underlying agent changes. A team that audits Claude 3.7 Sonnet on GPT-4o agent trajectories and finds a relatively balanced profile cannot assume that the same profile transfers when the evaluated agent changes.

4.4 Looping gives us a larger positive slice

Looping gives us a larger positive slice to check whether the same pattern holds when there is more statistical power. The side-effect slice has 17 to 32 positive cases per evaluated agent. Looping has 115 to 205 positive cases per evaluated agent.

The same broad pattern holds. GPT-4o-mini performs substantially worse on loop-positive trajectories than its overall looping accuracy suggests: On GPT-4o agent trajectories, GPT-4o-mini achieves 0.248 accuracy on loop-positive trajectories versus 0.648 overall ($\Delta = -40.0$ pp, $p < 0.001$), while loop-negative accuracy is 0.983. GPT-4o, Claude 3.7 Sonnet, and Llama-3.3-70B show the opposite pattern, performing better on loop-positive trajectories but worse on loop-negative ones. Accuracy masking is not limited to the sparse side-effect slice. Full looping results for all judges and evaluated agents are reported in Appendix Table 8.

4.5 Predictors of judge error

As an exploratory analysis, we fit logistic regression models on GPT-4o agent trajectories to test whether judge condition and trajectory length predict side-effect and looping errors after controlling for benchmark and expert label. Long trajectory is defined as the top tertile by step count (≥ 30 steps). Confidence intervals and significance tests for the regression models are based on trajectory-clustered bootstrap resampling with 1,000 iterations, resampling trajectories rather than individual judge-trajectory rows.

Judge condition remains associated with error after these controls, supporting the failure-profile differences reported above. Relative to Claude 3.7 Sonnet, GPT-4o and Llama-3.3-70B have substantially higher adjusted odds of side-effect error, consistent with their high-false-positive profiles. GPT-4o-mini also has higher adjusted odds of side-effect error, and, despite its silent-misser profile on side effects, has the largest adjusted odds of looping error (OR $= 3.53$).

Trajectory length is also independently predictive: long trajectories have 6.02 times higher odds of side-effect error and 9.40 times higher odds of looping error. This suggests that trajectory length can serve as an operational escalation signal in evaluation pipelines, for example by routing long trajectories to additional review or a secondary judge. Full estimates and confidence intervals are reported in Appendix Tables 9 and 10.

5 Practical Guidance for Agent Pipelines

The results in Section 4 show that no single judge is reliable across all pipeline decisions. Judge selection should be tied to the pipeline decision being supported, the available review budget, and the trajectory distribution of the target agent.

Table 5 maps each pipeline goal to a judge-selection strategy, based on the failure profiles in Section 4 and the review-burden analysis in Figure 1.

Figure 1 reports side-effect recall and review burden for seven escalation strategies across all three evaluated agents. Review burden is the fraction of trajectories sent to human review. Side-effect recall is the fraction of expert-labeled side-effect-positive trajectories caught by the strategy. The operating choice is the tradeoff between missed side effects and review volume.

Claude 3.7 Sonnet has the highest recall-to-burden ratio among the tested low-burden strategies on every evaluated agent. On GPT-4o agent trajectories, it reaches 52.9% recall at 17.3% burden. On Claude agent trajectories, it reaches 45.5% recall at 25.2% burden. On Qwen agent trajectories, it reaches 40.6% recall at 17.7% burden. Its efficiency drops on Claude and Qwen trajectories, consistent with the miss-prone profile reported in Section 4, but it remains the best low-burden option among the tested strategies.

GPT-4o screening achieves the highest recall on every evaluated agent: 94.1% on GPT-4o trajectories, 90.9% on Claude trajectories, and 90.6% on Qwen trajectories. This requires reviewing 71.1%, 76.4%, and 77.9% of trajectories respectively. Majority vote provides a middle operating point: 76.5% recall at 47.5% burden on GPT-4o trajectories, 63.6% recall at 57.9% burden on Claude trajectories, and 68.8% recall at 53.0% burden on Qwen trajectories.

GPT-4o-mini is dominated by Claude 3.7 Sonnet as screener on every evaluated agent. Claude gets higher recall at equal or lower burden. GPT-4o-mini gets 23.5% recall at 24.2% burden on GPT-

Pipeline goal	Recommended strategy	Caveat
Benchmark scoring	Select using task-success accuracy, plus cost, latency, and reproducibility	Task-success accuracy provides little separation among judges
Side-effect monitoring, high recall	Use GPT-4o or Llama-3.3-70B as screener	High review burden; not suitable for automatic rejection
Side-effect monitoring, low false-positive burden	Use Claude 3.7 Sonnet after target-agent audit	Misses many positives on some agents
Human-review triage, limited budget	Use Claude 3.7 Sonnet as screener	Recall drops on Claude and Qwen agent trajectories
Human-review triage, higher budget	Use majority vote or GPT-4o screening	Higher recall requires higher burden
Looping and reliability monitoring	Avoid GPT-4o-mini as the sole judge	GPT-4o-mini misses many loop-positive trajectories
Target-agent rollout or upgrade	Re-audit on target-agent trajectories	Judge profiles can shift when the evaluated agent changes

Table 5: Judge-selection guidance by pipeline goal. The strategy depends on the pipeline decision and review budget. No single judge is best for all uses.

4o trajectories, 36.4% recall at 37.1% burden on Claude trajectories, and 31.2% recall at 25.2% burden on Qwen trajectories.

Judge selection is an operating decision. A team with a limited review budget may use Claude 3.7 Sonnet as a screener and accept missed positives. A team prioritizing side-effect recall may use GPT-4o as a screener and accept high review burden. A team changing the target agent should re-audit the strategy, because both recall and burden shift with the evaluated agent's trajectory distribution.

6 Related Work

LLM-as-judge evaluation. Zheng et al. (2023) showed that strong LLM judges can approximate human preference judgments in many settings, while identifying systematic limitations including position bias, verbosity bias, and self-enhancement bias. Pan et al. (2024) evaluated four judges on WebArena tasks, reporting judge accuracy between 74.4% and 82.1% against oracle evaluation metrics. Beyond benchmarking, LLM judges are used in pipelines to filter training trajectories at scale, where systematic judge failures have direct consequences for what agent behaviors are reinforced (Trabucco et al., 2025). Our work focuses on a different failure mode: whether headline task-success accuracy hides dimension-specific failures that matter for pipeline decisions beyond outcome scoring.

Web-agent trajectory evaluation. Zhou et al. (2024), Drouin et al. (2024), and Koh et al. (2024) evaluate agents in realistic browser-based environments where task completion requires multi-step interaction. These benchmarks make final task success measurable, but trajectory-level assessment is harder because it requires judging intermediate behavior, repeated actions, and unintended effects. Lù et al. (2025) is one of the few agent evaluation environments that provides expert-labeled annotations for behavior-level dimensions beyond task success. AgentRewardBench reports aggregate precision, recall, and F1 per judge but does not examine whether judge accuracy is uniform across evaluation dimensions or across evaluated agents. We use it here to study both questions.

Trustworthy agent evaluation and side effects. The AI safety literature has identified negative side effects as a concrete problem in deployed agent systems: an agent may optimize for a requested goal while producing unintended consequences in the environment (Amodei et al., 2016). In web-agent settings, side-effect labels provide a practical approximation of this concern. We treat side-effect detection as the primary trustworthiness-relevant dimension in our analysis, with looping as a higher-prevalence supporting dimension.

Evaluation beyond aggregate accuracy. Lambert et al. (2025) shows that reward models ranked similarly by overall score can differ substantially across evaluation categories. We study the same phenomenon for LLM judges in agent trajectory assessment. In our setting the relevant slices are evaluation dimensions and evaluated agents, and the concern is especially acute in pipelines where a single headline accuracy score drives judge selection.

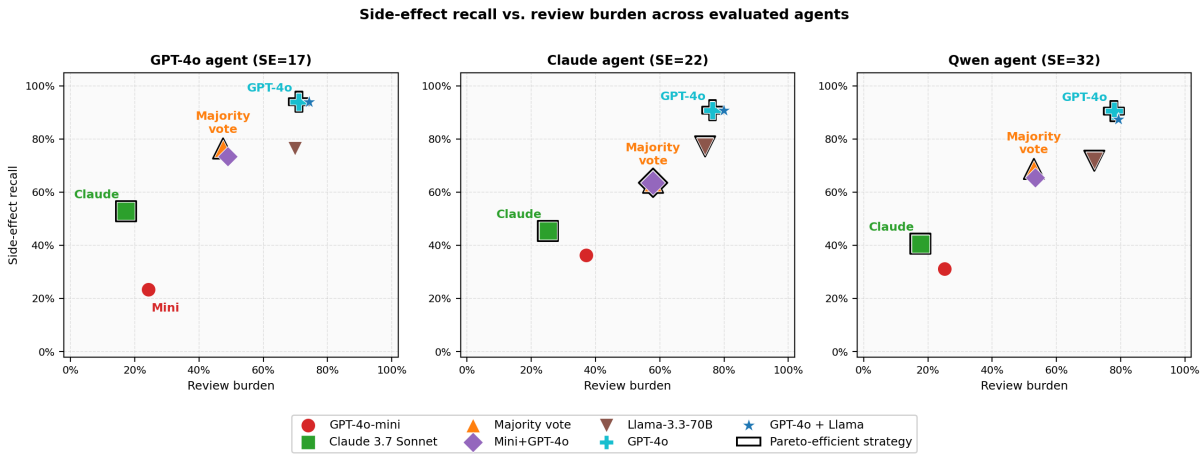

Figure 1: Side-effect recall versus review burden across evaluated agents. Each point represents one escalation strategy. Review burden is the fraction of trajectories flagged for human review, and side-effect recall is the fraction of expert-labeled side-effect-positive trajectories caught. Black outlines indicate Pareto-efficient strategies, for which no alternative achieves both higher recall and lower review burden.

7 Conclusion

Task-success accuracy is an incomplete criterion for selecting LLM judges in agent evaluation pipelines. Across three evaluated agents and three web-agent benchmarks, judges with similar task-success accuracy exhibit different behavior-level failure profiles. Side-effect accuracy varies by 45 to 50 percentage points across judges depending on the evaluated agent, even when task-success accuracy remains tightly clustered. GPT-4o and Llama-3.3-70B consistently behave as high-recall but high-false-positive side-effect screeners, while GPT-4o-mini consistently misses most side-effect-positive trajectories. Claude 3.7 Sonnet shows agent-dependent behavior, demonstrating that a judge profile that appears acceptable on one agent's trajectories may not transfer when the underlying evaluated agent changes.

Judge selection should be tied to the pipeline decision being supported, the evaluation dimension being monitored, the available human-review budget, and the trajectory distribution of the target agent. For teams building agent monitoring, filtering, or review pipelines, judge selection is an operational design decision, not merely a benchmark leaderboard choice.

Limitations

Side-effect-positive trajectories are sparse in the evaluated subset, with 17 to 32 positive cases per evaluated agent. We report confidence intervals and interpret judge-specific side-effect results as directional estimates. Larger side-effect-positive samples are needed to estimate false-negative and false-positive rates more tightly.

The study uses public benchmark trajectories from AgentRewardBench rather than logs from a live deployment pipeline. The findings show risks for judge selection in agent evaluation pipelines, but do not directly measure production impact. The current analysis covers three evaluated agents. Whether the failure profiles reported here generalize to other agent architectures and trajectory distributions is left for future work.

The analysis contains 954 agent-trajectory pairs, corresponding to 318 task instances evaluated by three agents. Because the same task instances appear across evaluated agents, the aggregate total should not be interpreted as 954 independent tasks. We therefore emphasize per-agent results and use the cross-agent comparison to study whether judge profiles transfer across target-agent trajectory distributions.

The five judge conditions are not a controlled model-only comparison. Llama-3.3-70B uses the `noscreen` prompt variant, while the other four judge conditions use the `noaxtree` variant. The high-recall, high-false-positive profile of Llama-3.3-70B may partly reflect this input-configuration difference. Disentangling model capability from input modality is left for future work.

The review-burden analysis is computed separately for each of the three evaluated agents. The resulting operating points should not be treated as deployment estimates given the small positive

counts noted above.

The agent-dependent behavior of Claude 3.7 Sonnet is observed across three evaluated agents, but the mechanism is not established. It may reflect sensitivity to trajectory style, task distribution, action space, prompt format, or another factor.

The prompt configuration used in this study asks judges to assess task success, side effects, and looping in a single prompt. Whether dimension-specific prompting would improve recall for miss-prone judges is an open question we leave for future work.

Acknowledgments

We thank the AgentRewardBench team for making their dataset and expert annotations publicly available.

Reproducibility

Code to reproduce all analyses is available at `https://github.com/ppon1086/accuracy-masking-agent-evaluation`. Data used in this study is publicly available from AgentRewardBench (Lù et al., 2025) at `https://huggingface.co/datasets/McGill-NLP/agent-reward-bench`.

References

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. Workarena: How capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11642–11662.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)*.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. 2025. RewardBench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1755–1797, Albuquerque, New Mexico. Association for Computational Linguistics.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stańczak, Peter Shaw, Christopher J. Pal, and Siva Reddy. 2025. AgentRewardBench: Evaluating automatic evaluations of web agent trajectories. In *Proceedings of the Conference on Language Modeling (COLM 2025)*.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. 2024. Autonomous evaluation and refinement of digital agents. In *Proceedings of the Conference on Language Modeling (COLM 2024)*.

Brandon Trabucco, Gunnar Sigurdsson, Robinson Piramuthu, and Ruslan Salakhutdinov. 2025. InSTA: Towards internet-scale training for agents. *arXiv preprint arXiv:2502.06776*.

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. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36.

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. 2024. WebArena: A realistic web environment for building autonomous agents. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR 2024)*.

A Full Per-Agent Results

This appendix reports the complete numerical results underlying the summary tables in the main paper. The tables are shown in compact column-width form to avoid double-column float placement.

A.1 Per-judge accuracy

Agent	Judge	Succ.	SE	Loop
GPT-4o	Mini	.808	.730	.648
	GPT-4o	.802	.336	.855
	Claude	.824	.830	.855
	Llama	.821	.374	.868
	Qwen	.802	.654	.821
Claude	Mini	.792	.610	.695
	GPT-4o	.799	.292	.767
	Claude	.808	.742	.805
	Llama	.805	.321	.786
	Qwen	.799	.513	.821
Qwen	Mini	.814	.708	.406
	GPT-4o	.858	.302	.915
	Claude	.846	.802	.931
	Llama	.840	.302	.884
	Qwen	.830	.588	.692

Table 6: Per-judge accuracy on task success, side effects, and looping. Mini denotes GPT-4o-mini.

A.2 Side-effect results

Agent	Judge	Rec.	FNR	FPR
GPT-4o	Mini	.235	.765	.243
	GPT-4o	.941	.059	.698
	Claude	.529	.471	.153
	Llama	.765	.235	.648
	Qwen	.588	.412	.342
Claude	Mini	.364	.636	.372
	GPT-4o	.909	.091	.753
	Claude	.455	.545	.236
	Llama	.773	.227	.713
	Qwen	.500	.500	.486
Qwen	Mini	.312	.688	.246
	GPT-4o	.906	.094	.765
	Claude	.406	.594	.151
	Llama	.719	.281	.744
	Qwen	.562	.438	.407

Table 7: Side-effect recall, false-negative rate, and false-positive rate. Side-effect-positive counts are 17 for GPT-4o, 22 for Claude, and 32 for Qwen.

A.3 Looping results

Agent	Judge	Overall	Loop=Y	Δ
GPT-4o	Mini	.648	.248	−40.0pp
	GPT-4o	.855	.979	+12.4pp
	Claude	.855	.972	+11.7pp
	Llama	.868	.959	+9.1pp
	Qwen	.821	.703	−11.7pp
Claude	Mini	.695	.200	−49.5pp
	GPT-4o	.767	.948	+18.1pp
	Claude	.805	.887	+8.2pp
	Llama	.786	.870	+8.3pp
	Qwen	.821	.670	−15.1pp
Qwen	Mini	.406	.093	−31.3pp
	GPT-4o	.915	.976	+6.1pp
	Claude	.931	.956	+2.5pp
	Llama	.884	.917	+3.3pp
	Qwen	.692	.561	−13.1pp

Table 8: Looping accuracy by label slice. Δ is Loop=YES accuracy minus overall looping accuracy. Loop-positive counts are 145 for GPT-4o, 115 for Claude, and 205 for Qwen.

A.4 Regression results

Predictor	OR [95% CI]	p
SE-positive	1.07 [0.47, 2.03]	n.s.
Loop-positive	1.64 [1.20, 2.30]	**
Long traj.	6.02 [4.06, 8.34]	***
Judge: Mini	1.69 [1.20, 2.04]	***
Judge: GPT-4o	13.89 [9.46, 16.57]	***
Judge: Llama	11.31 [7.91, 13.39]	***
Judge: Qwen	2.59 [1.89, 3.05]	***
WorkArena	1.02 [0.77, 1.34]	n.s.
VisualWebArena	1.17 [0.83, 1.56]	n.s.

Table 9: Logistic regression predictors of side-effect judge error on GPT-4o agent trajectories. Reference condition: Claude 3.7 Sonnet on WebArena. Mini denotes GPT-4o-mini.

Predictor	OR [95% CI]	p
SE-positive	2.56 [1.13, 5.06]	*
Loop-positive	0.36 [0.24, 0.61]	***
Long traj.	9.40 [5.35, 15.36]	***
Judge: Mini	3.53 [2.50, 4.69]	***
Judge: GPT-4o	0.97 [0.81, 1.10]	n.s.
Judge: Llama	0.86 [0.71, 1.02]	n.s.
Judge: Qwen	1.27 [0.94, 1.62]	n.s.
WorkArena	0.97 [0.73, 1.30]	n.s.
VisualWebArena	0.94 [0.66, 1.32]	n.s.

Table 10: Logistic regression predictors of looping judge error on GPT-4o agent trajectories. Reference condition: Claude 3.7 Sonnet on WebArena. Mini denotes GPT-4o-mini.