From Confident Closing to Silent Failure: Characterizing False Success in LLM Agents

Laksh Advani [1]

Abstract

LLM agents can fail silently by asserting task completion when the environment state shows otherwise. We study this failure mode, *false success*, across two agent benchmarks: 9,876 tau2-bench trajectories from 8 model families and 1,879 AppWorld trajectories from 4 model families with text-independent ground truth. False success is common but varies by setting: 45–48% of failures in single-control tau2-bench domains, 3% in dual-control telecom, and 75.8% among AppWorld self-assessing coding-agent trajectories with explicit status claims. LLM judges fail reliably: no configuration across 5 judges, 5 prompt strategies, and full task specifications exceeds AUROC 0.65 on tau2-bench, and the same judges reach only 0.54 AUROC on AppWorld API-call traces. Judges rely on surface completion proxies (confident closing language in tau2-bench, coarse action-sequence volume in AppWorld) rather than verified state changes. Lightweight TF-IDF detectors achieve task-disjoint AUROC 0.83 on tau2-bench and 0.95 on AppWorld, recovering 4–8× more false successes than the best judge at the same flag rate with 3,300× lower latency. These results suggest that production monitoring should use lightweight, domain-calibrated detectors as triage signals rather than relying on LLM judges as the primary monitor for false success.

1. Introduction

LLM agents can fail by confidently reporting task completion when the underlying system state indicates otherwise. An airline agent says "your refund of $686 has been processed," but the database has no record. We call this **false success**: a mismatch between the agent's natural-language claim of completion and the programmatic environment state.

Accepted to the Workshop: Failure Modes in Agentic AI (FAGEN) at ICML 2026. [1]University of Colorado. Correspondence to: Laksh Advani <laad8452@colorado.edu>.

False success is operationally dangerous because it propagates silently: unlike a crash or refusal, the agent presents the interaction as resolved, and no one notices until downstream damage accumulates. This is a monitoring problem, not a benchmark curiosity.

Prevalence. We study false success systematically across two benchmarks with different domains, agent architectures, and ground-truth methods. On tau2-bench (Barres et al., 2025), covering 9,876 trajectories from 8 frontier model families across airline, retail, and telecom customer-service tasks, false success accounts for 44–52% of all failures in single-control domains and only 3% in a dual-control domain where an independent user simulator can verify state. On AppWorld (Trivedi et al., 2024), a personal-app task suite with coding agents and database-state ground truth that is entirely independent of agent language, false success accounts for 75.8% of failures among architectures that produce explicit completion signals. The phenomenon appears in both conversational and coding-agent settings, though its surface form differs across benchmarks. Per-model false-success rates span 13% to 89% within each benchmark. Reasoning models offer no protection: Qwen3-Max-Thinking exhibits the highest false-success rate (79%) in our tau2-bench corpus, with reasoning traces that rationalize completion rather than verify it.

Why LLM judges fail. The standard monitoring solution is an LLM judge that reads the trajectory and decides whether the agent succeeded. We show this fails systematically. On tau2-bench, no configuration across 5 judge models, 5 prompt strategies, and a strong baseline that provides the full ground-truth task specification exceeds AUROC 0.65. The failure has a clear mechanism: judges anchor on confident closing-message language as evidence of completion, and false success trajectories produce exactly this language. Trajectories containing assertion vocabulary score 0.27–0.36 points more "completed" regardless of actual outcome. On AppWorld, where agents produce no natural-language closing messages and completion is signaled by a structured database write, the same judges fail with a best AUROC of 0.54. Here judges anchor on action-sequence

volume rather than assertion language. The failure persists across checklist conditions, reasoning models, and explicit instructions to verify write operations.

Contributions. **(1) Characterization.** We provide the first systematic study of false success across two benchmarks with different ground-truth mechanisms: 616 human-validated false successes in tau2-bench ($\kappa = 0.86$) and a text-independent replication in AppWorld confirming the phenomenon is not a labeling artifact.

(2) Lightweight detectors. TF-IDF classifiers achieve task-disjoint AUROC 0.83 on tau2-bench and 0.95 on AppWorld, recovering 4–8× more false successes than the best LLM judge at the same flag rate with 3,300× lower latency.

(3) Mechanistic account. Judges fail because they anchor on the same surface signals false success exploits (assertion language in tau2-bench, action-sequence volume in AppWorld) rather than verifying state changes. Detectors, trained discriminatively against ground-truth labels, learn the opposite signal.

(4) Structural moderators. Dual-control environments suppress false success by an order of magnitude (3% vs. 44–52% in tau2-bench), suggesting independent verification as a promising structural defense, though additional dual-control environments are needed to isolate causality.

2. Related Work

Agent evaluation and failure localization. Several benchmarks evaluate LLM agents on multi-step tasks. AgentBench (Liu et al., 2024) and GAIA (Mialon et al., 2023) assess reasoning and tool use across diverse domains. Tau-bench (Yao et al., 2024) and its successor tau2-bench (Barres et al., 2025) focus on customer-service scenarios with programmatic reward signals derived from database state. We use tau2-bench because it provides ground-truth completion labels independent of agent language. Concurrent work on failure localization (Who&When (Zhang et al., 2025b), AgenTracer (Zhang et al., 2025a), AgentDebug (Zhu et al., 2025)) assumes failures are already identified and localizes where in a trajectory an error occurred; our work addresses the earlier step of detecting whether a failure occurred at all when the agent claims success.

LLM-as-judge evaluation. Using LLMs to evaluate other LLMs has become standard practice, applied to dialogue quality (Zheng et al., 2023), instruction following (Kim et al., 2024), complex reasoning (Dubois et al., 2024), and agent trajectories (Zhuge et al., 2025). The assumption is that a capable LLM can assess whether another model completed a task. We show this assumption breaks down for false success: judges are systematically misled by confident assertion language regardless of actual outcome, connecting to broader concerns about judge reliability (Thakur et al., 2025) as well as sycophancy and hallucination (Perez et al., 2023; Zhang et al., 2025c), but identifying a structurally distinct variant triggered by tool outputs rather than internal generation.

Table 1. Dataset composition. FS=false success, HF=honest failure, Amb=ambiguous. Percentages are within failures.

Domain	Total	TS	Failures	FS/HF/Am.
Airline	1,797	1,404	393	45/38/17
Retail	4,029	3,134	895	47/28/24
Telecom	4,050	3,608	442	3/79/18
Total	**9,876**	**8,146**	**1,730**	**36/44/20**

Corrupt success and procedure-aware evaluation. Concurrent work by Cao et al. (Cao et al., 2026) introduces "corrupt success": trajectories where an agent receives a positive reward despite violating procedural integrity (e.g., bypassing required policy checks). Their framework targets reward=1 trajectories on τ-bench, finding 27–78% contain procedural violations. Corrupt successes are reward=1 trajectories where the agent did the wrong thing successfully, while our false successes are reward=0 trajectories where the agent claims to have done the right thing but did not. The two populations are disjoint and target different deployment risks: corrupt success threatens policy compliance when outcomes appear correct; false success threatens user trust when outcomes are silently incorrect.

3. Methods

3.1. tau2-bench corpus

We obtain agent trajectories from the public tau2-bench leaderboard hosted by Sierra (Barres et al., 2025). The leaderboard publishes graded simulation runs across three customer-service domains. Each trajectory contains a user-simulator request, the agent's tool-grounded response sequence, and a programmatic reward (1 if the final state matches the ground-truth task specification, 0 otherwise).

We collect submissions from 8 frontier model families spanning four providers (see Appendix **??** for the full list). After removing trajectories with malformed message structures or missing rewards, our corpus contains 9,876 trajectories (Table 1). Of these, 8,146 are successful (reward=1) and 1,730 are failures (reward=0). Failures form the population in which false success can occur, since by construction a successful trajectory has correctly completed the task.

Table 2. AppWorld corpus after filtering to self-assessing architectures with explicit status claims. FS rate is computed among failures with explicit status.

Method	Models	FS	HF	FS rate
full_code_refl	4	874	295	74.9%
ipfuncall	4	551	159	77.3%
Total	4	**1,425**	**454**	**75.8%**

3.2. AppWorld corpus

AppWorld (Trivedi et al., 2024) is a personal-app benchmark covering email, calendar, payments, and shopping, evaluated by programmatic unit tests. Agents interact through a structured API; each trajectory is a sequence of HTTP-style calls (e.g., `POST /venmo/friends/user@email.com`, `GET /amazon/cart`) followed by a terminal call to `POST /supervisor/message`.

We use publicly released experiment outputs covering 4 model families (GPT-4o, GPT-4-Turbo, LLaMA-3, DeepSeekCoder) and 4 agent architectures (`react`, `plan_exec`, `full_code_refl`, `ipfuncall`), totalling 8,190 trajectories.

Text-independent labeling. AppWorld agents write a structured `status` field (`success` or `fail`) to a supervisor database upon task completion, independent of natural language. We label false success when `status=success` co-occurs with programmatic evaluation failure, and honest failure when `status=fail` co-occurs with failing evaluation. We restrict analysis to architectures that sometimes write `status=fail` (`full_code_refl` and `ipfuncall`); `react` and `plan_exec` always write `status=success`, making their status field uninformative. We exclude 3,120 trajectories with no status (agents that stopped mid-task). The resulting corpus contains 1,879 trajectories: 1,425 false success and 454 honest failure (Table 2). This labeling is circularity-free by construction: labels derive from a database field with no dependence on agent-generated text, while detector features derive from API call sequences with zero overlap with the label source.

3.3. Three-class labeling of tau2-bench failures

Tau2-bench's reward function provides a binary signal: did the environment reach the expected end-state? Among the 1,730 failure trajectories, we observe two qualitatively distinct agent behaviors that the binary reward conflates:

- **False success**: the agent's final natural-language message asserts a completed action that the tool-call history and reward indicate did not occur.
- **Honest failure**: the agent acknowledges its inability to complete the task (e.g., "I'm unable to apply that override," "transferring you to a human agent") rather than asserting a falsified outcome.

We classify each failure trajectory's closing message using two regex sets (full patterns in Appendix A). `FS_ASSERT_RE` captures explicit completion claims and `HONEST_FAILURE_RE` captures explicit failure or hand-off language. A trajectory is labelled FALSE SUCCESS if `FS_ASSERT_RE` matches and `HONEST_FAILURE_RE` does not, HONEST FAILURE if the inverse holds, and AMBIGUOUS if both or neither match.

This yields 616 false successes (35.6% of failures), 753 honest failures (43.5%), and 361 ambiguous (20.9%). Successful trajectories (reward=1) form the negative class for detector training (FS vs TS); HF trajectories form the comparison class for judge analysis (Frame B, §3.5).

Label validation. We manually validate a stratified sample of 200 failure trajectories: 100 labeled FS and 100 labeled HF, drawn proportionally across all 8 model families and 3 domains. Human labels agree with regex labels on 91.5% of cases ($\kappa = 0.86$), with per-class accuracy of 92% for FS and 91% for HF. GPT-5, prompted with the same trajectories and label definitions, agrees with human labels on 83.5% of cases, lower than the regex pipeline. Full validation details appear in Appendix B.

Circularity controls. Masking the top-30 regex trigger tokens from the TF-IDF vocabulary reduces random-split AUROC by 0.001, and training on trajectory text excluding the closing message achieves AUROC 0.924 versus 0.934 for closing-message-only features, confirming the signal is distributed throughout the trajectory rather than concentrated in the labeled text window. The AppWorld replication (§3.2) provides the strongest circularity control via zero text overlap between labels and features. Top features appear in Appendix **??**.

3.4. Detectors

We train three detectors to classify FS vs TS, varying in capacity and inference cost. The same architecture is applied to both benchmarks, with benchmark-specific feature extraction.

tau2-bench features. We use full trajectory text, optionally restricted to the closing message or tool-response text for ablation.

AppWorld features. We represent each trajectory as its API call sequence: a string of method-endpoint tokens (e.g., `GET_supervisor_profile`

`POST_amazon_auth_token` `GET_amazon_cart` `POST_supervisor_message`). This feature space has no overlap with natural language and no connection to the label source.

Models. We evaluate three detectors of increasing capacity: TF-IDF + Logistic Regression (bigram features, max 30k tokens, min_df=2, sublinear TF, L2-regularized with class-balanced weighting); TF-IDF + XGBoost (same features, 300 estimators, max_depth=6, learning_rate=0.08, scale_pos_weight tuned per fold); and fine-tuned DeBERTa-v3-base (184M parameters, 512-token context with left truncation to retain trajectory closings, 3 epochs, AdamW at 2×10^{-5} with linear warmup and decay). All use class-balanced weighting. Sparse linear weights are interpretable and offer the fastest inference (≈1 ms/trajectory).

Evaluation splits. We evaluate each detector under four splits:

- **Random.** Stratified 70/15/15 train/val/test split.
- **LOMO (leave-one-model-out).** Train on all but one model family, test on the held-out family, averaged across holdouts. tau2-bench: 8 holdouts. AppWorld: 4 holdouts.
- **LODO (leave-one-domain-out).** tau2-bench only. Train on 2 domains, test on the held-out 3rd. Telecom has only 15 FS cases (3% prevalence) so we report LODO restricted to airline ↔ retail transfer in the main results, with telecom in the appendix. AppWorld has a single domain so LODO does not apply.
- **Task-disjoint.** GroupShuffleSplit over `task_id`, ensuring no task appears in both train and test, averaged over 5 random seeds. This is our cleanest generalization evaluation, preventing the detector from memorizing task-specific surface patterns seen in training.

We report AUROC as the primary metric, threshold-tuned F1 for deployment relevance, and 95% bootstrap confidence intervals on the deployment table.

3.5. Judge evaluation

tau2-bench judges. We construct a stratified 295-trajectory evaluation set with roughly 95 FS, 105 TS, and 105 HF, balanced across domains and source models. We evaluate 5 judge models spanning non-reasoning and reasoning families (GPT-4o, Claude Sonnet 4.5, Llama 3.3-70B-Instruct, DeepSeek-R1, o3-mini) under five conditions varying information access (full trajectory, closing removed, tool outputs only, stepwise reasoning, completion checklist), plus a `strong-real` condition providing the ground-truth task specification. Verbatim prompts appear in Appendix **??**.

AppWorld judges. Since AppWorld agents produce no natural-language closing messages, we evaluate judges on API call sequences using a balanced 200-trajectory evaluation set (100 FS, 100 HF) and 3 judges (GPT-4o, Claude Sonnet 4.5, Llama 3.3-70B) under three conditions: blind, with task description prepended, and with an explicit checklist noting that GET-only sequences cannot complete write tasks.

Scoring. Judge responses map to a continuous false-success score: $1 -$ confidence if `completed=True`, otherwise `confidence`. We report AUROC on FS vs TS (Frame A, the deployment-relevant question) for tau2-bench, and AUROC on FS vs HF for AppWorld, where no true-success class is included in the evaluation set. All judge calls are made through OpenRouter at temperature 0.0 and cached by (model, prompt) hash for reproducibility.

4. Results

We organize results around four themes: phenomenon characterization (§4.1), lightweight detectors (§4.2), LLM judge evaluation (§4.3), and deployment implications (§4.4).

4.1. Phenomenon Characterization

Prevalence varies substantially across model families. Among the 1,730 failure trajectories in the tau2-bench corpus, the fraction classified as false success ranges from 13% (GPT-5.2) to 79% (Qwen3-Max-Thinking-Preview). The two Anthropic models cluster near 30–35%; Gemini and GLM-5 sit near 40–50%; the Qwen variants are highest (Figure 1, left panel). Reasoning capability is not protective. Qwen3-Max-Thinking, an explicitly reasoning-trained model, produces the highest false-success rate in our corpus. Inspecting its trajectories reveals why: the `<think` chain rationalizes why the requested action should have succeeded rather than verifying environment state, then asserts completion. Reasoning traces in FS cases average 1,274 characters versus 486 for other models, with distinctive tokens `think` `great` and `successfully`.

On AppWorld, per-model FS rates span 67% (LLaMA-3) to 89% (DeepSeekCoder) among failures with explicit completion claims (Figure 1, right panel). The cross-model spread is narrower than in tau2-bench, consistent with a more homogeneous task distribution. Within AppWorld, FS rates vary by app domain (65–95%) and decrease with task difficulty (88% at level 1 to 70% at level 3); see Appendix **??** for details.

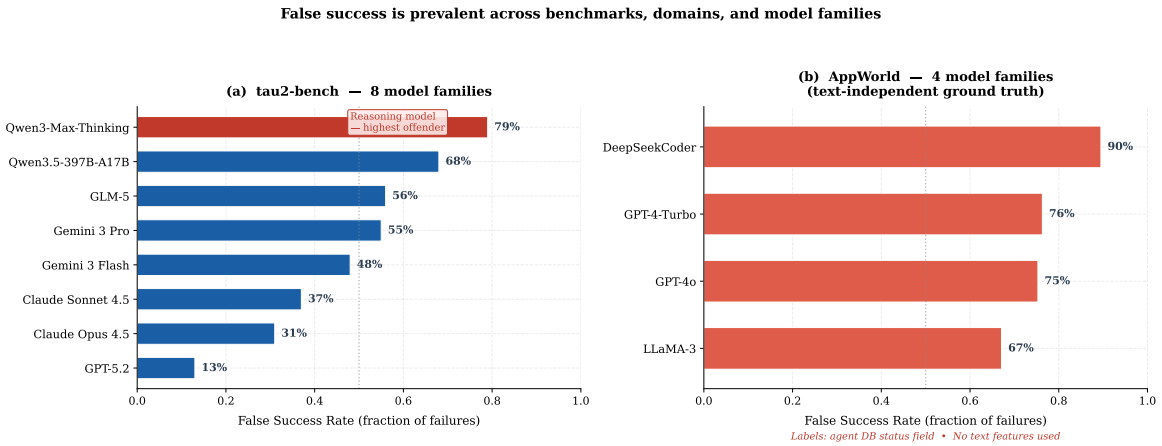

Figure 1. False success (FS) rate per model family. Left: tau2-bench (8 frontier families, conversational agents). Right: AppWorld (4 families, coding agents, text-independent ground truth). Reasoning capability does not reduce FS rates; Qwen3-Max-Thinking exhibits the highest rate in tau2-bench.

Environment structure moderates prevalence. In tau2-bench, false success accounts for 45% and 48% of failures in airline and retail respectively, both single-control domains where only the agent can write to environment state. In telecom, a dual-control domain where the user simulator can independently verify state, false success drops to 3% of failures. We treat this as an observation rather than a causal claim, since only one dual-control domain is available (15 FS cases), but the pattern is consistent with independent verification suppressing the failure mode.

Cross-benchmark replication. The AppWorld results provide a direct replication of the false-success phenomenon using labels and features with no text overlap. Labels come from a structured database field; features come from API call sequences. Both are independent of the closing-message vocabulary that drives the tau2-bench labeling and detector. The same pattern holds across benchmarks: GPT-4o as judge peaks at 0.54 AUROC on AppWorld versus 0.95 for the task-disjoint detector (full comparison in Appendix **??**). Architecture filter details appear in Appendix **??**.

4.2. Lightweight Detectors

tau2-bench: three methods converge at task-disjoint evaluation. Table 3 reports AUROC across all four splits and three methods. Under random IID split, TF-IDF + XGBoost leads (0.940), followed by DeBERTa-v3-base (0.933) and TF-IDF + LR (0.913). Under task-disjoint evaluation, which prevents the detector from memorizing task-specific entity tokens seen in training, performance compresses: TF-IDF + LR achieves 0.849, TF-IDF + XGBoost 0.825 $\pm$ 0.025, and DeBERTa 0.827. All three methods land within 0.024 AUROC of each other. The 184M-parameter neural model provides no measurable benefit over a linear classifier on bigram TF-IDF features, indicating that the discriminative signal is largely available in surface trajectory features.

Table 3. Detector AUROC. Top: tau2-bench. Bottom: AppWorld. LODO does not apply to AppWorld (single domain). Standard deviation over 5 seeds reported for XGBoost task-disjoint.

Split	TF-IDF+LR	TF-IDF+XGB	DeBERTa-v3
tau2-bench (closing-message features)			
Random (IID)	0.913	**0.940**	0.933
LOMO (8 holdouts)	0.904	**0.913**	0.895
LODO (airline $\leftrightarrow$ retail)	0.696	0.665	0.660
Task-disjoint (5 seeds)	**0.849**	0.825 $\pm$ 0.025	0.827
AppWorld (API call sequence features)			
Random (IID)	0.879	**0.972**	–
LOMO (4 holdouts)	–	**0.856**	–
Task-disjoint (5 seeds)	0.915	**0.953** $\pm$ 0.020	0.923

Cross-model generalization. LOMO mean AUROC is 0.91 on tau2-bench and 0.86 on AppWorld (XGBoost). Figure 2 summarizes generalization across distribution shifts for both benchmarks.

AppWorld: action-sequence features achieve comparable performance. On AppWorld, TF-IDF + XGBoost on API call sequences achieves 0.972 random-split AUROC and 0.953 task-disjoint, with DeBERTa close behind at 0.923 (Table 3). The convergence within 0.038 AUROC at task-disjoint evaluation mirrors the tau2-bench pattern: surface-level sequence features capture the phenomenon as well as deep representations. For false success, the strongest predictors are read-heavy sequences ending in task completion (e.g., repeated `GET` calls without `POST` state changes). For honest failure, the strongest predictors are write-retry patterns and post-completion restarts (Figure 3). Per-model holdout results appear in Appendix **??**.

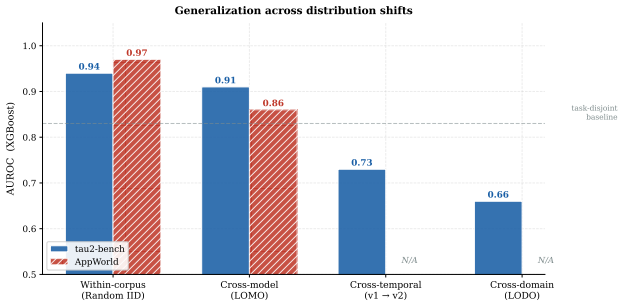

Figure 2. Generalization across distribution shifts. Cross-model transfer (LOMO) holds above 0.85 on both benchmarks. Cross-domain transfer (LODO) drops to 0.69 on tau2-bench; LODO does not apply to AppWorld. Cross-temporal transfer (tau-v1 to tau2) reaches 0.73.

Table 4. LLM judge Frame A (FS vs TS) AUROC on tau2-bench, across 5 judge models and 5 prompt conditions plus `strong-real`. No configuration exceeds 0.65. Dashed reference line in Figure 4 marks the structural detector at 0.83 (task-disjoint).

Judge	blind	no_cl.	check.	tool	step.	strong
GPT-4o	0.488	0.521	0.494	**0.545**	0.458	0.529
Sonnet 4.5	0.576	**0.640**	0.576	0.605	0.590	0.632
Llama 3.3-70B	0.530	0.534	**0.538**	0.479	0.500	0.528
DeepSeek-R1	**0.573**	0.551	0.517	0.545	0.535	–
o3-mini	0.478	**0.554**	0.522	0.550	0.513	–
Best	0.576	0.640	0.576	0.605	0.590	0.632

Transfer and ablation summary. Zero-shot cross-domain (LODO) transfer between airline and retail reaches AUROC 0.69 (Table 3), indicating domain-specific vocabulary dependence; the AppWorld replication confirms the phenomenon transfers across very different surface realizations when the detector is retrained. Training on trajectory text excluding the closing message achieves AUROC 0.924, confirming distributed signal. Cross-temporal transfer from tau-v1 (2024 models) to tau2 yields AUROC 0.73, suggesting recognizable but lexically drifting surface expression.

4.3. LLM Judges Fail Systematically

tau2-bench: judge ceiling at 0.65. Across 5 judge models, 5 prompt conditions, and a strong baseline providing the full ground-truth task specification, no configuration exceeds AUROC 0.65 (Table 4). The strongest single cell is Sonnet 4.5 `no_closing` at 0.640. Reasoning models offer no improvement: DeepSeek-R1 peaks at 0.573, o3-mini at 0.554. Providing the full task specification (`strong-real`) changes Sonnet's AUROC from 0.640 to 0.632; judges do not lack information, they fail to use it.

AppWorld: judges fail on behavioral false success too. The same judges fail on AppWorld, where completion is signaled by a database write rather than natural language (Figure 4, right panel). GPT-4o peaks at AUROC 0.537 under the `checklist` condition, which explicitly instructs the judge that GET-only sequences cannot complete write tasks. Claude Sonnet 4.5 *degrades* with more context: blind 0.368, with-task 0.289, checklist 0.274. LLaMA-3.3-70B reaches 0.505 in the blind condition but drops to 0.384 with the checklist hint. The detector achieves 0.953 task-disjoint AUROC on the same data, a gap of over 0.40 AUROC.

The mechanism: anchoring on surface completion signals. Frame B (FS vs HF AUROC) reveals why: across all 25 judge $\times$ condition cells, AUROC ranges from 0.18 to 0.30. Judges score honest-failure trajectories as more failure-like than false-success trajectories, with a mean score gap of 0.27 to 0.36 favoring assertion-language trajectories regardless of actual outcome (Figure 5). Removing explicit honest-failure markers preserves the effect, ruling out a labeling artifact.

On AppWorld, judges anchor on action-sequence volume rather than assertion language, but the structural failure is identical: the false-success surface pattern exploits the same proxy judges use to infer completion.

4.4. Deployment Implications

A detector that beats LLM judges in AUROC is theoretically interesting; whether it offers practical value for production monitoring requires examining operating-point trade-offs.

Triage performance: detector dominates at every flag rate. We compare the tau2-bench lightweight detector (TF-IDF + XGBoost) against the strongest judge configuration (Sonnet 4.5 `no_closing`) under a triage scenario: at flag rate r, the system flags the top $r\%$ of trajectories by predicted FS score for human review. At a 10% flag rate, the detector recovers **72% of false successes [95% CI 63–80%]**, while the strongest judge recovers **13% [8–17%]**. At a 5% flag rate, the detector achieves 51% recall versus the judge's 6%. The detector recovers 4–8$\times$ more false successes per unit of human review effort across all operating points (Figure 6, Table 5).

Latency advantage: 3,300$\times$. A single XGBoost inference on CPU takes 1.19 ms; an LLM judge API call takes approximately 4,000 ms. The structural detector runs 3,364$\times$ faster with no GPU dependency, making continuous high-throughput monitoring feasible where judge-based approaches are not.

Complementarity and scoping. Detector and judge scores correlate at $\rho = 0.41$ and catch partially complementary false successes; a 75/25 ensemble improves AUROC from 0.834 to 0.855 (see Appendix **??**). The detector is a triage signal, not an autonomous monitor: at a 10% flag rate,

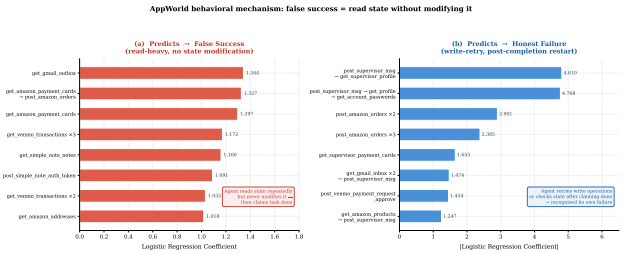

Figure 3. AppWorld behavioral mechanism. Left: features predicting false success are read-heavy API sequences with no state-modifying calls before task completion. Right: features predicting honest failure are write-retry patterns and post-completion restarts, indicating the agent recognized incomplete state. The same phenomenon observed in tau2-bench closing language appears here at the action-sequence level.

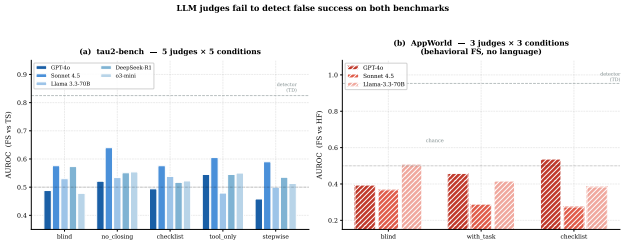

Figure 4. LLM judge AUROC on tau2-bench (left, Frame A: FS vs TS) and AppWorld (right, FS vs HF). Dashed lines mark the respective task-disjoint detector AUROC (0.83 and 0.95). No judge configuration approaches the detector on either benchmark. On AppWorld, Sonnet performance *degrades* as more context is provided.

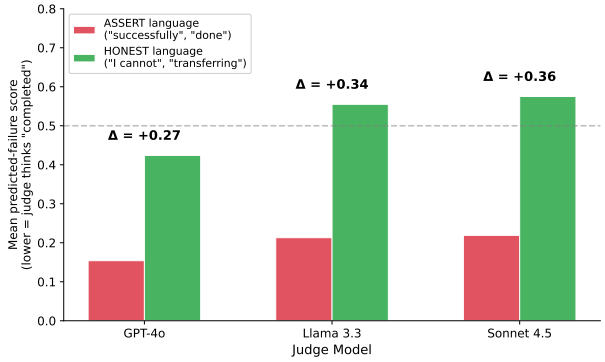

Figure 5. Mechanism validation on tau2-bench: judges anchor on assertion language. Each bar is the mean judged-as-failure score gap (HONEST − ASSERT) across the eval set. Positive values indicate the judge treats assertion-language trajectories as more completed than honest-failure trajectories, regardless of actual outcome. The +0.27 to +0.36 gap is consistent across all three judges.

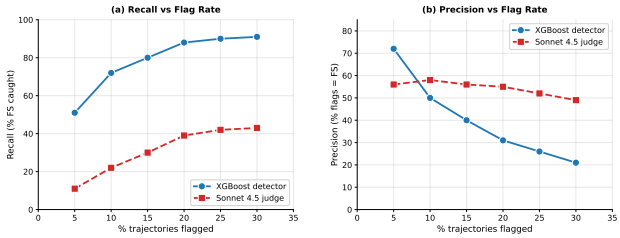

Figure 6. Triage curve on tau2-bench: recall and precision as a function of flag rate, comparing the structural detector to the strongest judge configuration (Sonnet 4.5 `no_closing`). At a 10% flag rate, the detector recovers 72% of FS cases versus the judge's 13%.

precision is 50%. Substantial gains for autonomous deployment would likely require methods that verify trajectory-environment consistency directly rather than reading surface text.

5. Limitations

Benchmark scope. Our primary corpus is tau2-bench, a customer-service simulation, with AppWorld as a cross-benchmark replication. Both use structured API environments with programmatic rewards. Whether false success extends to unstructured settings such as web navigation, open-ended code generation, or scientific reasoning remains open. The environment-structure finding suggests that any benchmark with independent state-verification will suppress false success, but the qualitative phenomenon should persist wherever agents emit completion claims without verifiable grounding. The detector is a triage signal (50% precision at 10% flag rate), not an autonomous monitor; high-stakes de-

Table 5. Deployment metrics on tau2-bench comparing the structural detector (TF-IDF + XGBoost) against the strongest judge (Sonnet 4.5, `no_closing`). Recall and precision at three flag rates with 95% bootstrap confidence intervals.

Flag rate	Method	Recall	Precision
5%	Detector (XGBoost)	**51%** [43, 59]	**72%** [57, 86]
	Judge (Sonnet)	6% [3, 10]	56% [22, 89]
10%	Detector (XGBoost)	**72%** [63, 80]	50% [40, 60]
	Judge (Sonnet)	13% [8, 17]	**58%** [32, 79]
20%	Detector (XGBoost)	**89%** [82, 95]	31% [24, 38]
	Judge (Sonnet)	25% [18, 31]	**55%** [39, 71]
Latency per inference			
Detector (XGBoost, CPU)		1.19 ms	
Judge (LLM API call)		∼4,000 ms	
Speedup		∼**3,300**×	

ployment would likely require direct trajectory-environment consistency checks.

AppWorld scope. The AppWorld replication is restricted to two self-assessing architectures (`full_code_refl` and `ipfuncall`) that sometimes write `status=fail`. The remaining two architectures always write `status=success`, making their FS rate unmeasurable via this method. The AppWorld detector uses API call sequence features with no natural-language component, so the tau2-bench and AppWorld detectors are not directly interchangeable.

Cross-domain transfer and robustness. Zero-shot LODO transfer between tau2-bench airline and retail achieves AUROC 0.69, indicating domain-specific vocabulary dependence; few-shot calibration with 50–100 labeled examples per class raises LR AUROC to 0.79–0.83. Cross-temporal transfer between tau-v1 and tau2 achieves AUROC 0.68–0.73. A preliminary adversarial rewrite test (Appendix **??**) flips both detectors and judges on 20–25% of cases, suggesting adversarially trained agents would require additional defenses.

6. Conclusion

False success is a silent failure mode in LLM agents: agents assert task completion while the environment state indicates failure. We characterized this phenomenon across tau2-bench (9,876 trajectories, 8 model families) and AppWorld (1,879 trajectories, 4 model families, text-independent ground truth). False success is common across both settings, but its prevalence varies sharply by environment and agent architecture, from 3% of failures in tau2-bench telecom to 45–48% in single-control tau2-bench domains and 75.8% among AppWorld self-assessing coding-agent trajectories with explicit status claims. In our corpus, reasoning does not reliably prevent the failure: Qwen3-Max- Thinking has the highest tau2-bench false-success rate (79%), with traces that often rationalize completion rather than verify environment state.

LLM judges are poor monitors for this failure mode. No configuration across 5 judges, 5 prompt conditions, and a strong baseline with full task specifications exceeds AUROC 0.65 on tau2-bench; the same judge family reaches only 0.54 AUROC on AppWorld API-call traces, where no natural-language closing message is present. Across both benchmarks, judges appear to rely on surface completion proxies (confident closing language in tau2-bench, coarse action-sequence volume in AppWorld) rather than verified state changes.

Lightweight detectors provide a stronger triage signal. TF-IDF classifiers achieve task-disjoint AUROC 0.83 on tau2-bench and 0.95 on AppWorld, recover 4–8× more false successes than the best judge at the same human-review budget, and run at 3,300× lower latency. Simple linear classifiers match fine-tuned neural models throughout. These results suggest that production monitoring should treat lightweight, domain-calibrated detectors as triage signals for false success, while reserving direct trajectory-environment consistency checks for higher-stakes deployment.

References

Barres, V., Dong, H., Ray, S., Si, X., and Narasimhan, K. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Cao, H., Driouich, I., and Thomas, E. Beyond task completion: Revealing corrupt success in LLM agents through procedure-aware evaluation, 2026. URL https://arxiv.org/abs/2603.03116.

Dubois, Y., Galambosi, B., Liang, P., and Hashimoto, T. B. Length-controlled AlpacaEval: A simple way to debias automatic evaluators, 2024. URL https://arxiv.org/abs/2404.04475.

Kim, S., Shin, J., Cho, Y., Jang, J., Longpre, S., Lee, H., Yun, S., Shin, S., Kim, S., Thorne, J., and Seo, M. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8euJaTveKw.

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., et al. AgentBench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Mialon, G., Fourrier, C., Swift, C., Wolf, T., LeCun, Y., and Scialom, T. Gaia: a benchmark for general ai assistants, 2023. URL https://arxiv.org/abs/2311.12983.

Perez, E., Ringer, S., Lukošiūtė, K., Nguyen, K., Chen, E., Heiner, S., Pettit, C., Olsson, C., Kundu, S., Kadavath, S., et al. Discovering language model behaviors with model-written evaluations. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13387–13434. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.847. URL https://aclanthology.org/2023.findings-acl.847/.

Thakur, A. S., Choudhary, K., Ramayapally, V. S., Vaidyanathan, S., and Hupkes, D. Judging the judges: Evaluating alignment and vulnerabilities in LLMs-as-judges. In *Proceedings of the Fourth Workshop on Generation, Evaluation and Metrics (GEM^2)*, pp. 404–430, Vienna, Austria and virtual meeting, 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.gem-1.33/.

Trivedi, H., Khot, T., Hartmann, M., Manku, R., Dong, V., Li, E., Gupta, S., Sabharwal, A., and Balasubramanian, N. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Yao, S., Shinn, N., Razavi, P., and Narasimhan, K. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Zhang, G., Wang, J., Chen, J., Zhou, W., Wang, K., and Yan, S. AgenTracer: Who is inducing failure in the LLM agentic systems?, 2025a. URL https://arxiv.org/abs/2509.03312.

Zhang, S., Yin, M., Zhang, J., Liu, J., Han, Z., Zhang, J., Li, B., Wang, C., Wang, H., Chen, Y., and Wu, Q. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025b. URL https://arxiv.org/abs/2505.00212.

Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., Huang, X., Zhao, E., Zhang, Y., Chen, Y., Wang, L., Luu, A. T., Bi, W., Shi, F., and Shi, S. Siren's song in the AI ocean: A survey on hallucination in large language models. *Computational Linguistics*, 51(4):1373–1418, 2025c. doi: 10.1162/coli.a.16. URL https://aclanthology.org/2025.cl-4.9/.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets and_Benchmarks.html.

Zhu, K., Liu, Z., Li, B., Tian, M., Yang, Y., Zhang, J., Han, P., Xie, Q., Cui, F., Zhang, W., et al. Where LLM agents fail and how they can learn from failures, 2025. URL https://arxiv.org/abs/2509.25370.

Zhuge, M., Zhao, C., Ashley, D., Wang, W., Khizbullin, D., Xiong, Y., Liu, Z., Chang, E., Krishnamoorthi, R., Tian, Y., Shi, Y., Chandra, V., and Schmidhuber, J. Agent-as-a-judge: Evaluate agents with agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 80569–80611. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhuge25a.html.

A. Regex Pattern Definitions

Both patterns are applied to the closing message (final assistant turn) of each tau2-bench failure trajectory. Patterns are case-insensitive and applied with Python's `re.search` on the lowercased closing message.

FS_ASSERT_RE. Captures explicit completion claims:

```
\b(successfully|has\s+been|have\s+been|is\s+(now\s+)?(complete|done|processed|
booked|cancelled|canceled|updated|submitted|confirmed|refunded|approved))\b
| \brefund(ed)?\s*(of\s+)?\$?\d+
| \byou(?:\'re|\s+are)\s+(all\s+set|good\s+to\s+go)
| \byour\s+(reservation|booking|order|return|refund|cancellation|change)\s+
  (has\s+been\s+|is\s+)(complete|confirmed|processed|submitted|approved)
| \b(processed|completed|finalized|issued)\s+the\s+(refund|cancellation|change)
```

HONEST_FAILURE_RE. Captures explicit failure or hand-off language:

```
\bI\s+(cannot|can\'t|am\s+unable|am\s+not\s+able)\b
| \b(transferring|transferred|transfer)\s+(you\s+)?(to\s+)?(a\s+)?
  (human|agent|representative|specialist)
| \bunable\s+to\s+(process|complete|fulfill|approve)
| \bshall\s+I\s+proceed\b
| \bwould\s+you\s+like\s+(me\s+)?to\s+proceed
| \bI\s+(\'m|am)\s+sorry\b
| \brequires?\s+manual\s+(review|intervention|approval)
| \bI\s+don\'t\s+have\s+(the\s+)?(authority|ability|access)
```

A trajectory is labeled FALSE SUCCESS if FS_ASSERT_RE matches and HONEST_FAILURE_RE does not, HONEST FAILURE if the inverse holds, and AMBIGUOUS if both or neither match.

B. Human Validation Protocol

Sampling. We drew 200 failure trajectories stratified to balance the regex pipeline's two main labels (100 FS, 100 HF), allocated proportionally across the 8 model families and 3 domains. Ambiguous trajectories were excluded since no ground-truth dispute is meaningful for that class.

Annotation protocol. Each annotator was shown the full trajectory and asked to classify the closing behavior as FALSE SUCCESS, HONEST FAILURE, or AMBIGUOUS. Annotators saw the regex label only after providing their own. GPT-5 (via OpenRouter, temperature 0.0) served as a second annotator with the same trajectories and label definitions.

Table 6. Human-regex and GPT-5-human agreement on the 200-trajectory validation set.

Comparison	Agreement	Cohen's κ
Human vs. Regex pipeline	91.5%	0.86
Human vs. GPT-5	83.5%	0.71

Per-class accuracy: 92% for FS (92/100 confirmed), 91% for HF (91/100 confirmed). Of the 17 disagreements, 11 occurred on trajectories the human marked AMBIGUOUS (mixed assertion and hedging language); 4 on borderline FS cases with a late hedge; and 2 on HF cases where a vestigial "I cannot" clause was contextually overridden by a successful action.

C. Detector Hyperparameters and Training Details

TF-IDF + Logistic Regression. `TfidfVectorizer: ngram_range=(1,2), max_features=30000, min_df=2, sublinear_tf=True, strip_accents='unicode',` lowercased input. Clas-

sifier: `LogisticRegression(C=1.0, class_weight='balanced', max_iter=1000, solver='liblinear', random_state=42)`.

TF-IDF + XGBoost. Same TF-IDF features. `XGBClassifier(n_estimators=300, max_depth=6, learning_rate=0.08, subsample=0.9, colsample_bytree=0.9, scale_pos_weight=fold_specific, eval_metric='auc', tree_method='hist', random_state=42)`.

DeBERTa-v3-base. Fine-tuned from `microsoft/deberta-v3-base` (184M parameters). 3 epochs, batch size 16, AdamW (lr 2×10^{-5}, weight decay 0.01), linear warmup 10%, sequence length 512 with left truncation. Best checkpoint by validation AUROC. Training on RunPod A40 ($\sim$25 min per run).

Trajectory serialization. Each trajectory is serialized as: `[USER]` *`content`* `[TOOL_CALL]` *`name`*`(`*`args`*`)` `[TOOL_RESULT]` *`result`* `[ASST]` *`content`*. Tool arguments and results truncated to 200 and 300 characters respectively.

AppWorld features. API call sequences are represented as space-separated method-endpoint tokens: `GET_supervisor_profile POST_amazon_auth_token GET_amazon_cart POST_supervisor_message`. Tokenization is identical to tau2-bench (bigram TF-IDF, same hyperparameters).