YARDSTICK RESEARCH

How to Audit and Eliminate Hallucinations in a Grounded LLM Pipeline

A reproducible audit method for retrieve-then-synthesize pipelines, demonstrated across 25 models at 5.9× lower cost with no loss of grounding.

Yardstick Research · a method for auditing retrieve-then-synthesize pipelines · data collected 2026-06-07 to 2026-06-10 · version 1.1 (2026-06-11)

A reproducible methods paper for developers building retrieve-then-synthesize (RAG, research-agent, or document-generation) pipelines. The worked example is our own vendor-research pipeline; vendors are anonymized by cohort and gold score, and models are named because they are the comparison. Data and verification tooling: `github.com/ConnorYQueen/yardstick-grounded-pipeline-audit` *(v1.1, CC BY 4.0).*

Abstract

Any team running a retrieve-then-synthesize LLM pipeline (RAG, research agents, automated document generation) needs to answer two questions before trusting the output: is the synthesis model fabricating claims, and which model is worth paying for? This paper gives developers a reproducible method to audit both, and runs it end to end on a production grounded-research pipeline. The worked example tests whether the expensive synthesis step can move off a frontier flagship (Claude Opus 4.8) onto a cheaper, different-family model without losing factual grounding. Retrieval is decoupled and frozen so that both arms synthesize from byte-identical evidence and only the model varies, verified by an identical bundle hash per vendor. A deterministic, per-claim source-grounding detector (not an LLM judge, and not model agreement) scores every claim as grounded, mistagged, or ungrounded. On a 30-vendor bottom-tercile sample we find: (1) **no model, including the flagship, produces zero ungrounded claims** on thin-evidence inputs; (2) **synthesis quality plateaus** across a band of models from $0.07 to $5.00 per million input tokens, and **paying more buys nothing** while paying much less costs grounding; (3) the cheapest model still on that plateau after full-pool confirmation, `x-ai/grok-4.3`, matches Opus on grounding (0.37% vs 0.44% ungrounded-claim rate, statistically indistinguishable) at **~5.9x lower effective cost**. Containment matters as much as the rate: **every hallucination the winner produced was caught before publication** (3 of 3 blocked by the fail-closed gate; zero would have reached a customer), while the flagship control, judged behind the same gate, would have let one of its six through. The headline methodological result is a cautionary one: **per-model hallucination rates cannot be measured on small vendor panels.** Six rival models that looked plateau-level on a 15-vendor panel (0.21% to 0.62%) each regressed when confirmed on the full pool (to between 0.56% and 1.27%), while only grok-4.3 and the flagship held. Two limits are stated up front rather than buried. First, the sample bounds the certification: with 30 vendors the winner's 99% upper confidence bound sits at about 1.24%, and pushing it below the 0.5% target would take roughly 37 or more vendors even in the best case, so we report the limit instead of claiming the bound. Second, every rate here is conditional on a clean frozen evidence bundle: the study certifies synthesis and grounding against trusted, linted evidence, not the trustworthiness of live retrieval itself, which in production must be gated by the same banned-source and completeness checks before any bundle freezes. For any such pipeline the recommendation is a hybrid (flagship retrieves, the cheapest confirmed-plateau model synthesizes, cheap models gate) behind a fail-closed validator; an executed fix loop shows a single flagship redo clears only two of three residual errors, so the validator and human escalation, not the model, are what keep hallucinations out of what ships.

1. Introduction and motivation

Retrieve-then-synthesize is a standard production pattern: a pipeline gathers evidence, then an LLM synthesizes it into a structured output, whether that is a RAG answer, a research brief, or a scored dossier. Token cost and hallucination risk both concentrate in the synthesis step, which consumes the full evidence bundle as input and emits a long structured document. At production scale that step on a frontier flagship dominates unit cost and exposes the operator to frontier price increases. So every team running such a pipeline faces the same practical question: **can synthesis run on a cheaper, different-family model without losing the grounding fidelity that makes the output trustworthy, and how would you prove it either way?**

This paper answers both with an audit method any team can run on its own pipeline, demonstrated here on a production vendor-research pipeline that retrieves public evidence about a software vendor and synthesizes a scored dossier. The contribution is a method plus a result. The method transfers to any grounded pipeline: decoupled frozen retrieval so only the model varies, a deterministic grounding detector immune to model agreement, an effective-cost metric that prices the consequence of errors, and pre-registered predictions to check against. The result is what running it on our pipeline produced: a per-step model assignment and its cost delta. A secondary contribution, which emerged from the data, is a warning about how *not* to benchmark low-rate hallucination.

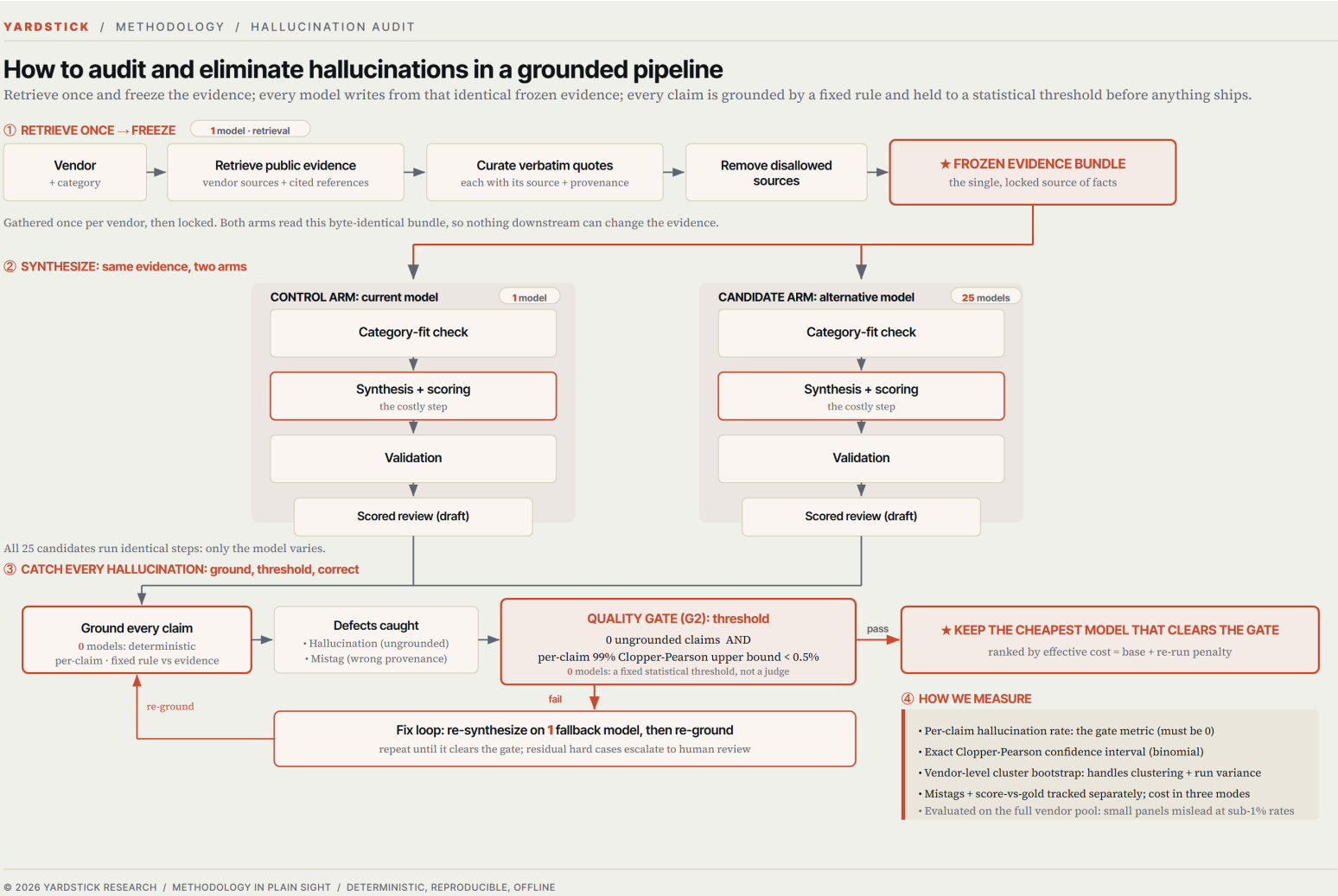


Figure 1. The audit method, end to end

Figure 1. The audit method: retrieve evidence once and freeze it; every candidate model synthesizes the structured output through identical stages from that byte-identical frozen evidence, so the only variable is the model; a deterministic rule grounds every claim (no model grades another); the cheapest model that stays grounded is kept behind a fail-closed gate. Source: `PROCESS-DIAGRAM.html` *(editable) /* `.png` *. Implementation specifics (model names, scripts, the winning config) are intentionally omitted from this diagram.*

2. Problem formulation

Defect definition (load-bearing). We validate only against public information and report claims as they appear. A defect is a claim that is **ungrounded** (no citation, or the cited source does not contain it) or **mistagged** (the provenance label does not match the source tier). A defect is *not* "factually false in reality": a faithfully reported, correctly attributed public claim is not a defect even if reality disagrees, and a clearly labeled averaged estimate (for unpinned numeric disagreements) is not a defect.

Two metrics, tracked separately. *Hallucination* (ungrounded/fabricated content) is the gate metric and must trend to zero. *Mistag* (a real claim with the wrong provenance label) is a quality issue, auto-fixable, and not the gate. We label every defect H1 to H7 (fabricated quote, dead URL, misattribution, unsupported claim, fabricated entity fact, phantom integration, estimate-as-fact).

The gate (G2). A configuration clears G2 only if in-sample ungrounded escapes equal zero **and** the per-claim 99% one-sided Clopper-Pearson upper bound is below 0.5%. As the results show, this bar is not met by any model on this sample, including the flagship, which reframes the gate as a fail-closed control rather than a model property.

2.1 Related framing

LLM-as-judge is the most widely deployed grounding-evaluation approach: a second model reads each synthesized claim against the source and grades correctness. The structural failure here is that the judge and the synthesizer share training data, so a claim the synthesizer fabricates from a web-corpus gap is exactly the kind of claim the judge will also accept as supported. On vendor-specific, commercially niche facts (an exact integration partner, an unpublished pricing tier, a product capability not indexed at scale), neither model has reliable coverage, and the judge cannot catch the synthesizer's blind spots from the same epistemic position.

Cross-model agreement addresses single-judge circularity by routing a claim to N models and treating consensus as truth. The problem is that large language models are not independent sources: their training corpora overlap substantially, and a factually thin but plausible-sounding claim common on B2B vendor pages can propagate identically across models trained on the same web snapshot. Agreement proves memorization, not grounding. The models in this study are trained on largely overlapping corpora; a claim they all hallucinate in the same direction would pass an agreement filter while failing every citation check.

Static benchmarks (MMLU, TruthfulQA, RAGAS-style metrics, and their derivatives) measure parametric recall or context-retrieval fidelity in standardized settings. They do not measure whether a specific synthesizer stayed within a specific, frozen, auditable evidence set for a specific vendor's claims, the operative question for a pipeline that must be reproducible per-vendor and auditable per-claim. A model with a high benchmark score may still fabricate a vendor's integration list when the evidence bundle is sparse, because the benchmark never put it in that position.

We therefore use a **deterministic per-claim source-grounding check** against the exact frozen bundle the synthesizer was given: cited URL present in the bundle, provenance tag matching the source tier, receipt sentence verbatim in the bundle. This is reproducible, offline, and immune to model agreement. §3.2 cross-checks it against blind model validators as a sanity check; the deterministic bundle match, not any model's opinion, remains the ground truth.

3. Experimental design

3.1 Decoupled retrieval (the controlled variable). For each vendor we retrieve once on a tool-capable channel and freeze an evidence bundle: verbatim "receipt" sentences, their source URLs, and provenance candidates. Every arm then synthesizes from the byte-identical bundle. The SHA-256 of the bundle is recorded per run and is identical across all of a vendor's configurations, which is machine-checkable proof that only the model varied. The bundle is built by a trusted flagship retrieval pass and linted to remove banned aggregator sources before any synthesis. Think of it as an open-book exam where every model gets the same single book and nothing else; the recorded hash proves nobody swapped pages between candidates. The scope this buys is

stated plainly: every result in this paper certifies what models do with trusted evidence, not the trustworthiness of retrieval itself. A production deployment must gate bundle construction with the same banned-source and completeness checks before the freeze (§7).

3.2 The detector. A deterministic procedure checks every provenance-tagged claim for: a cited URL present in the frozen bundle, a provenance tag matching the source tier, and (where a receipt is quoted) the receipt appearing verbatim in the bundle. It runs offline against the frozen bundle, so it is reproducible and immune to model agreement. There is no third-model arbiter; unresolved cases route to human spot-check. As a sanity check, the detector's verdicts were compared against the blind step-03 validator verdicts on the control dossiers: they agreed on hallucinations (the detector is stricter on provenance mistags). Those validators are themselves LLMs and inherit §2.1's blind-spot critique, so the agreement checks the detector's precision; it is not the ground truth. The ground truth is the deterministic bundle match. In plain terms, the detector is ordinary code doing string and URL matching against the frozen bundle. Code like that cannot hallucinate, runs identically every time, and grades every model with the same ruler.

3.3 Channels and roster. The flagship (retrieval and control arm) runs in-harness, inside the operator's own agent environment where the retrieval tools live. Twenty-five candidate models across nine non-Anthropic families run over a single OpenRouter credential. *Table A1: roster and per-million-token prices.*

3.4 Isolation and verifiability. Each model trio writes to its own directory containing its dossier, the exact prompts it was given (pure functions of the frozen bundle), token usage, a manifest with the bundle hash, and its own detector verdict. No configuration reads another's output; the client is stateless. An offline self-test (33 assertions) proves no cross-combination contamination. Every (vendor x model) cell in the 454-row results ledger is independently re-gradable. Every vendor-model pairing is a sealed envelope: what the model was shown, what it wrote, what it cost, and what the detector ruled, all reproducible without network access.

3.5 Sample. Thirty vendors, the lowest-scoring tercile per industry, spanning 14 industry cohorts, gold scores 1 to 65 per the vendor sample and ledger. **We state the selection bias plainly: these are the hardest, thinnest-evidence vendors, so every rate reported here is an upper bound; typical vendors would score cleaner for every model.** (Six of the 20 industry cohorts were excluded because their evaluation criteria were not finalized at study time; they are queued for re-run.)

3.6 Cost model. Three modes (free-tier-actual, retail-at-scale, crossover) plus an **effective cost = base + re-run-rate x overwrite-cost**, where a hallucinating dossier must be redone on the reliable fallback. This prices the consequence of an error: a cheap model that errs often is not cheap. The overwrite constant is the flagship's measured per-vendor cost; §5.6 replaces it with a directly measured redo cost. **Worked example:** qwen3-235b's tokens are nearly free ($0.007/vendor base), but its higher confirmed hallucination rate makes it redo about 1 in 6 dossiers on the $0.49 fallback, so it really costs ~$0.092/vendor, no cheaper than grok at $0.086.

3.7 Pre-registration. Predictions were recorded before any run (Appendix A2) and are diffed against actuals in §6.5. **3.8 Determinism deviation.** No provider in either family honors temperature 0 (the reasoning models reject it; the flagship removed sampling parameters), so determinism is approximated by the frozen bundle plus enforced JSON, and run-to-run variance is reported rather than hidden. Ask the same model the same question twice and the wording can differ slightly; providers no longer allow that to be switched off, so we pin everything else (the evidence, the prompts, the output format) and rely on full-pool confirmation rather than single runs.

3.9 Statistical analysis (pre-specified). Inference is led by **confidence intervals (the range of rates consistent with the data), not p-values**. (a) The quality claim is equivalence/non-inferiority, so we report the rate difference and its interval rather than a significance test that cannot prove the null. (b) Claims cluster within vendors and runs are non-deterministic, so a per-claim test would be pseudo-replication; the **primary interval is a vendor-level cluster bootstrap** (resampling that accounts for some vendors being harder than others; 20,000 resamples), with exact Clopper-Pearson per-claim intervals as the tighter secondary. (c) Any pairwise tests across the 25-model sweep get Benjamini-Hochberg correction. A power statement is stated up front: distinguishing 0.44% from 0.22% at 80% power needs ~10,667 claims per arm, far above this study's ~760 to 1,380. In plain terms, telling those two rates apart would take more than ten times the data we had, so within that band we treat the models as tied and let cost decide.

4. Procedure

Phase 0: we froze and linted the evidence bundle for all 30 vendors. Phase 1: we ran the all-Claude control on all 30. Phase 2: a cost-greedy sweep over step 02 (the hallucination-bearing step), then steps 03 and 01, eliminating a model on its first hallucination and stopping at the cheapest clean one. Phase 3: we confirmed the assembled winner on the full pool. Phase 4: an exhaustive step-02 leaderboard across all 25 models, plus a measured fix loop. The cost-greedy sweep was the optimized search; the exhaustive leaderboard (added on review) tested whether it reached the true optimum.

5. Results

5.1 Control baseline. The all-Claude control is **not** a zero-escape baseline on thin evidence: 30 vendors, 1,376 claims, **six hallucinations (0.436%)**, 131 mistags, median score delta -4.5 vs gold, effective cost $0.508/vendor. It fails the strict gate. In plain terms, even the most expensive pipeline available writes about one false claim per 230 on these deliberately hard vendors, which is exactly why a gate sits between the model and anything published.

5.2 Exhaustive step-02 leaderboard (all 25 models). Every roster model sat the same exam with the same single book: each was run as the synthesizer on a fixed 15-vendor panel with the cohort-fit and validation models (01 and 03) held constant, so the only thing that varied was the model writing the dossier. The result is a competence **plateau then a cliff** (panel rates, Fig 2):

STEP-02 MODEL	CLAIMS	HALLUC	CAUGHT	REACHED SITE	PANEL RATE	PRICE IN/OUT ($/MTOK)	TIER
mistralai/mistral-small	452	1	1	0	0.22%	0.07 / 0.20	plateau*
qwen/qwen3-235b	394	1	1	0	0.25%	0.09 / 0.10	plateau*
google/gemini-2.5-pro	501	2	2	0	0.40%	1.25 / 10.0	plateau
claude-opus-4-8 (control)	703	3	3	0	0.43%	5.00 / 25.0	plateau
openai/gpt-5	889	4	4	0	0.45%	1.25 / 10.0	plateau
x-ai/grok-4.3	399	2	2	0	0.50%	1.25 / 2.50	plateau
openai/o3	386	5	5	0	1.30%	2.00 / 8.00	degraded

STEP-02 MODEL	CLAIMS	HALLUC	CAUGHT	REACHED SITE	PANEL RATE	PRICE IN/OUT ($/MTOK)	TIER
openai/gpt-4o-mini	243	4	4	0	1.65%	0.15 / 0.60	degraded
google/gemini-2.5-flash-lite	483	9	9	0	1.86%	0.10 / 0.40	degraded
meta-llama/llama-4-scout	177	10	10	0	5.65%	0.10 / 0.30	fail
openai/o3-mini	397	30	30	0	7.56%	1.10 / 4.40	fail
openai/o4-mini	495	39	39	0	7.88%	1.10 / 4.40	fail
cohere/command-r	12	1	1	0	8.33%†	0.15 / 0.60	fail
openai/gpt-5-nano	586	79	79	0	13.48%	0.05 / 0.40	fail
openai/gpt-4.1-nano	451	82	82	0	18.18%	0.10 / 0.40	fail
openai/gpt-4.1-mini	512	133	133	0	25.98%	0.40 / 1.60	fail
microsoft/phi-4	n/a	n/a	n/a	n/a	n/a	0.07 / 0.14	cannot run

Claims is each model's panel sample (the rate's denominator). **Caught** counts hallucinating dossiers the fail-closed gate blocked before publish; **reached site** counts hallucinations that would have reached a published page. On this fixed panel the gate blocked every hallucinating dossier (reached site = 0 for all, even gpt-4.1-mini's 133). On the full pool, judged post-gate, the control's gate passed 8 of 30 dossiers, so 26 escapes (25 mistags and 1 hallucination) would have published; the winner's gate blocked all 30, so nothing would have published at all: all 3 of its hallucinations were caught, and zero would have reached a customer (per-row `gate_blocked` and `postgate_escapes` in the ledger; Fig 4). The gate's high block rate is partly the symmetric pricing_transparency format quirk (§7). * see §5.3 and §6.4: the cheap "plateau" rates are panel artifacts that did not survive full-pool confirmation. † command-r produced repeated 300s timeouts and truncated dossiers; partial (3/15 vendors, 12 claims). phi-4 cannot run: its 16k context is smaller than the 23k-token bundle.

Premium models do not beat the plateau, and o3 is worse. The remaining roster models were exercised at step 02 during the sweep or its confirmations, with aggregate rates per the canonical results: deepseek-chat-v3.1 0.56% (confirmed on 29), qwen3-32b 1.27% (confirmed on 29), gpt-5-mini 1.29%, gemma-3-27b 1.37%, deepseek-r1 1.44%, llama-3.3-70b 1.56%, mistral-large 2.44%, llama-4-maverick 4.76%, gemini-2.5-flash 34.4%. That completes the 25-model roster.

Figure 2. Panel rank is a mirage: every cheap front-runner regresses above grok-4.3 once confirmed

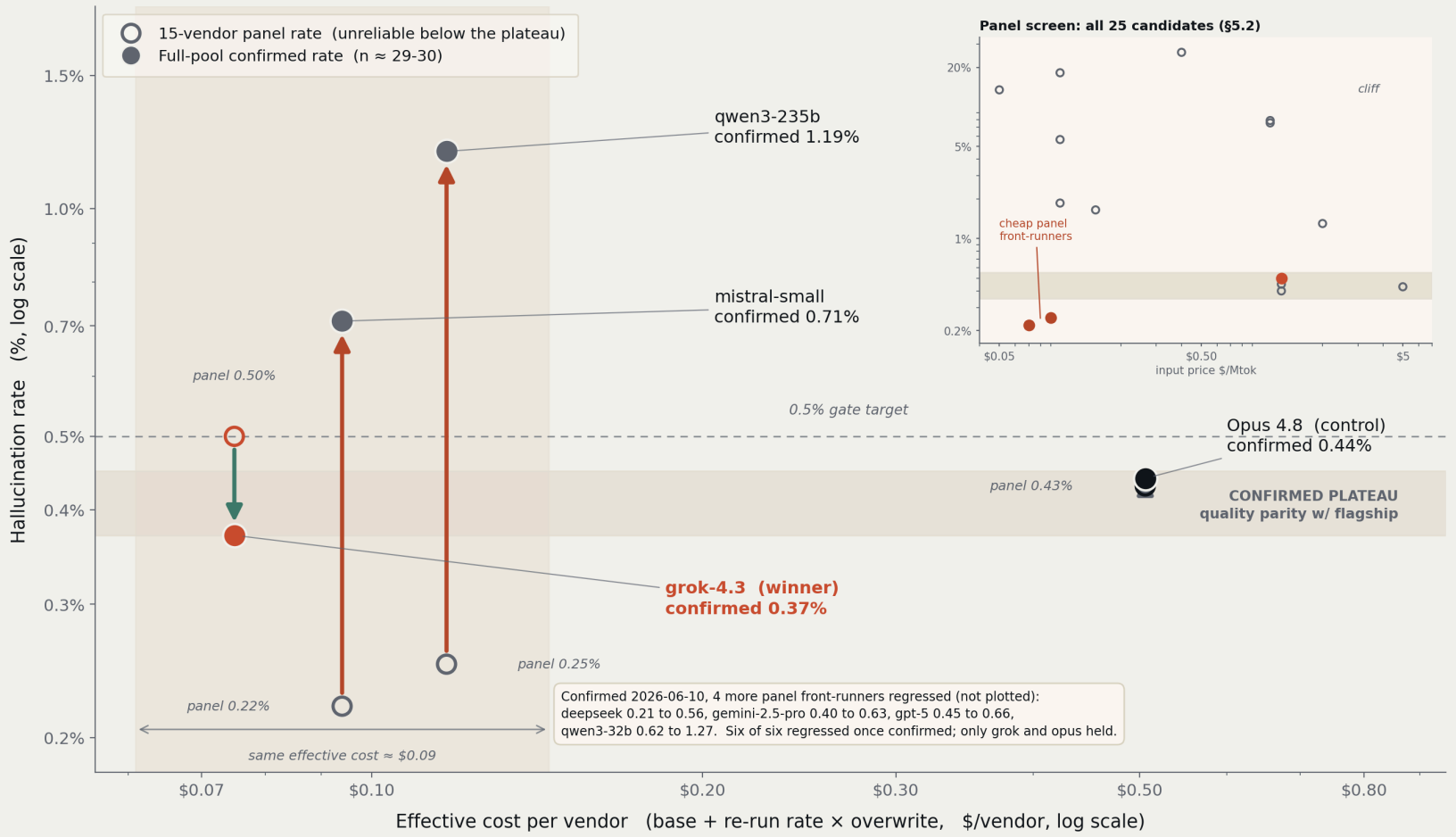
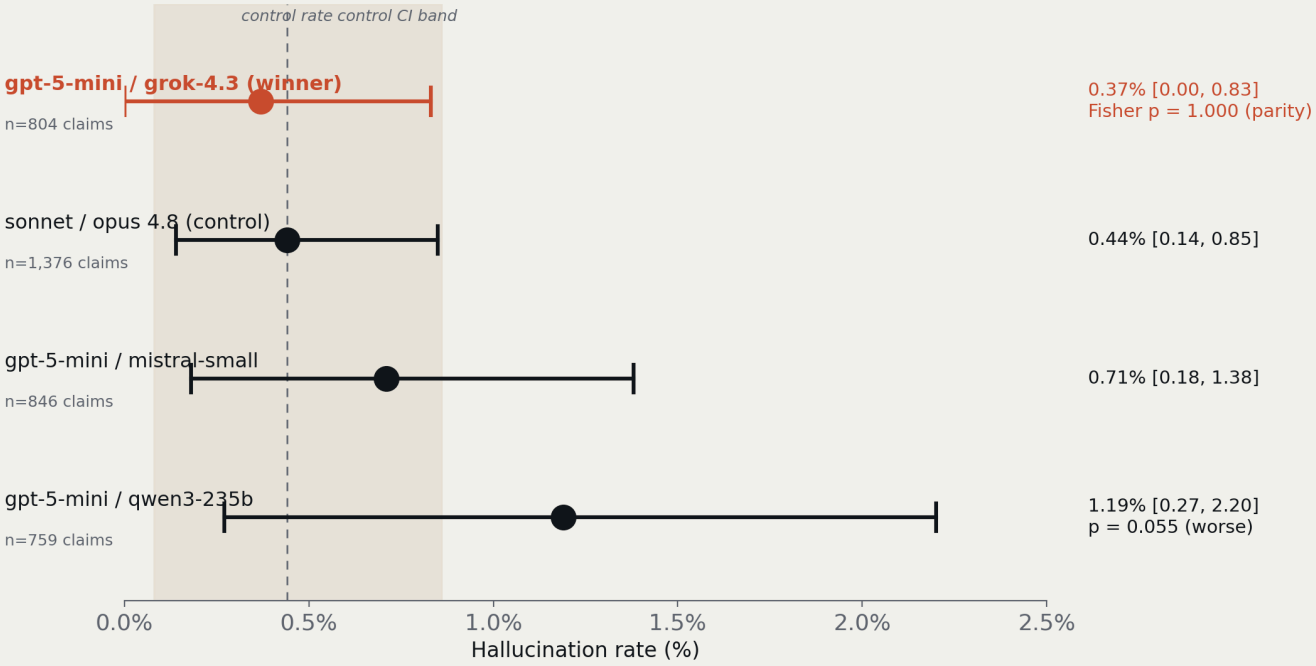


Figure 2. Panel rank is a mirage. For the four configurations carried to full-pool confirmation, the hollow marker is the 15-vendor panel rate and the filled marker is the full-pool confirmed rate (n ≈ 29-30), each plotted at its measured effective cost per vendor. mistral-small and qwen3-235b topped the panel (0.22%, 0.25%) then regressed to 0.71% and 1.19% on confirmation; grok-4.3 held its rate (0.50% → 0.37%) at the same ~$0.09 effective cost; the Opus control is stable at 0.44%. A later pass confirmed four more panel front-runners (deepseek, gemini-2.5-pro, gpt-5, qwen3-32b), all of which also regressed. At identical cost the cheap front-runners are worse once confirmed, so grok-4.3 is the lowest confirmed rate on the plateau, which is why it is the pick, not the panel-cheapest model. Inset: the full 25-model panel screen from §5.2, on its own price axis because effective cost was measured only for the confirmed configurations.

5.3 Winner and full-pool confirmation. The cost-greedy sweep selected grok-4.3 for step 02 (the cheapest model clean across its 15-vendor subset). The assembled winner is `01 gpt-5-mini · 02 grok-4.3 · 03 mistral-small`. Confirmed on the full pool, with the cheap panel front-runners also confirmed (Fig 3):

CONFIG (01 / 02 / 03)	VEND	CLAIMS	HALLUC	RATE	95% CLUSTER-BOOTSTRAP CI	FISHER VS CONTROL
gpt-5-mini / grok-4.3 / mistral-small (winner)	30	804	3	0.37%	[0.00, 0.83]	p = 1.000 (parity)
sonnet / opus 4.8 / opus 4.8 (control)	30	1376	6	0.44%	[0.14, 0.85]	reference
gpt-5-mini / mistral-small / mistral-small	29	846	6	0.71%	[0.18, 1.38]	p = 0.389
gpt-5-mini / qwen3-235b / mistral-small	29	759	9	1.19%	[0.27, 2.20]	p = 0.058 (worse)

Figure 3. Full-pool hallucination rate ± 95% vendor-level cluster-bootstrap CI



30-vendor bottom-tercile pool. Shaded band = control CI. grok-4.3 and the control overlap heavily, quality parity (Fisher p = 1.000). qwen3-235b sits above both. Intervals are 95% vendor-level cluster bootstrap (20,000 resamples).

Figure 3. Full-pool hallucination rate ± 95% vendor-level cluster-bootstrap CI

grok-4.3 has the lowest **confirmed** rate. Its difference from the control is not detectable (Fisher p = 1.000, rate difference -0.06 pp, overlapping intervals), which for an equivalence claim is the desired outcome: **quality parity with the flagship.** In plain terms, no difference between them is detectable at this sample size; even a twofold real gap would be invisible in this much data (§3.9). Mistags: winner 8.2% vs control 9.5%. Score agreement: winner median delta -3.625 vs control -4.5 (the winner tracks gold slightly better).

5.4 The reliability result (see §6.4). Every cheap model that topped the panel regressed once confirmed on the full pool; only grok-4.3 and the control held:

CONFIG (01 / 02)	PANEL-15 RATE	FULL-POOL RATE	STABLE?
gpt-5-mini / grok-4.3	0.50%	0.37%	yes
sonnet / opus 4.8 (control)	0.43%	0.44%	yes
gpt-5-mini / deepseek-v3.1	0.21%	0.56%	no (2.7x worse)
gpt-5-mini / gemini-2.5-pro	0.40%	0.63%	no (1.6x worse)
gpt-5-mini / gpt-5	0.45%	0.66%	no (1.5x worse)
gpt-5-mini / mistral-small	0.22%	0.71%	no (3.2x worse)
gpt-5-mini / qwen3-235b	0.25%	1.19%	no (4.8x worse)
gpt-5-mini / qwen3-32b	0.62%	1.27%	no (2.0x worse)

Read the table top to bottom: the two stable rows are the only models whose panel rate survived contact with more data. Six of six panel front-runners regressed, by 1.5x to 4.8x. Had we trusted the panel, we would have shipped a model up to five times worse than it looked. §6.4 explains why this is structural, not bad luck. One basis disclosure: the table reports every configuration on the same single-pass full-pool basis. grok-4.3 also served far more rolls than any other model across the sweep and the extension; its all-rolls aggregate is 0.64% (12 of 1,883 claims). The plateau conclusion and the winner choice are unchanged on either basis; we report both so no reader has to find the second number in the ledger.

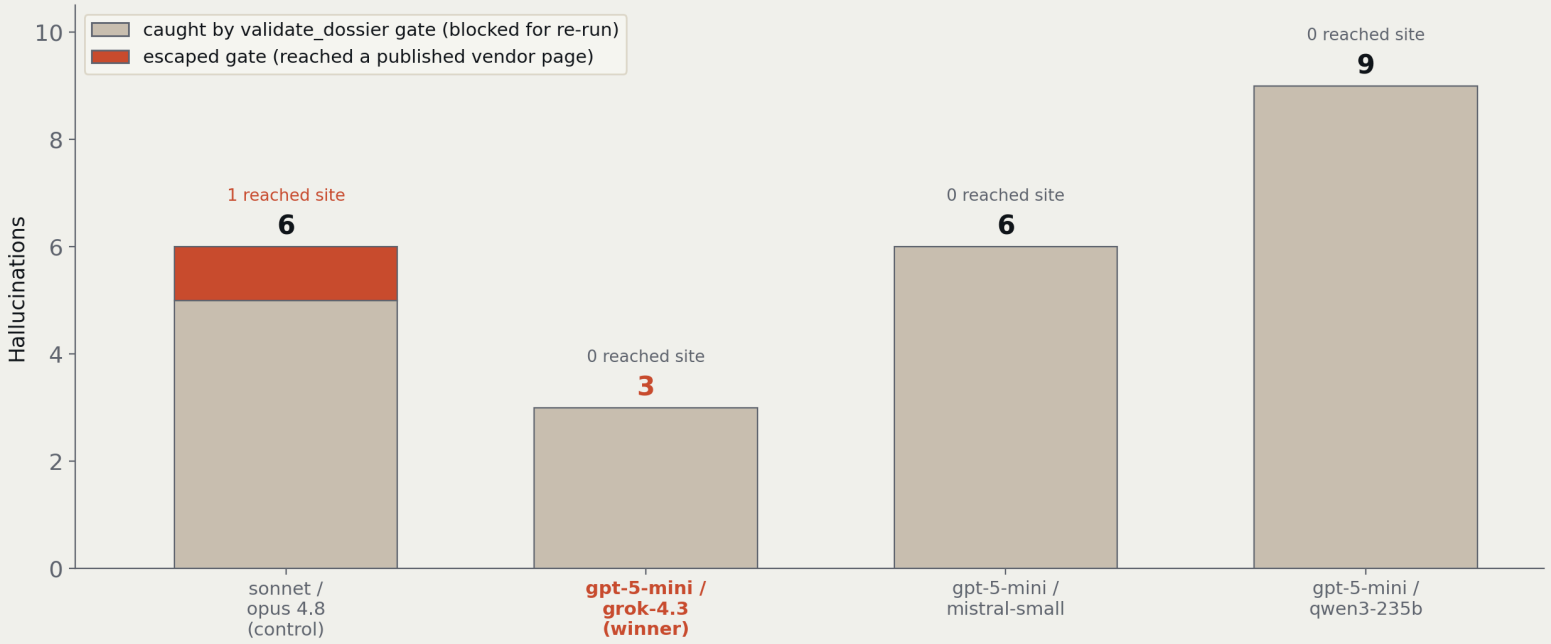
5.5 Cost. Winner vs control, retail-at-scale: base **$0.037 vs $0.426/vendor (~11.4x)**; effective **$0.086 vs $0.508/vendor (~5.9x)**. The winner stays cheaper than the control even if flagship prices fall sharply, and the gap widens as frontier prices rise. Total real spend to run the entire experiment (OpenRouter only; the Claude control and retrieval are subscription-marginal) was **$14.73**, plus **$4.6** for the 2026-06-10 confirmation extension. In dollars: validating one vendor costs about nine cents instead of fifty-one.

5.6 The fix loop (measured, not modeled). We executed the redo that the effective-cost model assumes. For each of the winner's three hallucination vendors we re-synthesized on the opus fallback and re-grounded:

VENDOR (ANONYMIZED)	GROK HALLUC	OPUS-REDO HALLUC	CLEARED BY ONE REDO?
healthcare-clinical, gold 38	1	0	yes
healthcare-RCM, gold 31	1	1	no (opus also hallucinates)
tax, gold 59	1	0	yes

One flagship redo cleared two of three. The residual vendor is one where opus itself hallucinates, so the redo cannot fix it. The measured redo cost (opus step-02 only) is **~$0.265/vendor**, about half the modeled $0.49 constant (which charged a full opus pipeline). Two consequences: the winner's measured effective cost is even lower than modeled (~$0.06 to $0.09/vendor), and "one redo equals clean" is optimistic, so true-zero requires the validator plus human escalation on a residual fraction (Fig 4).

Figure 4. Hallucinations generated vs. on-site exposure per config, full-pool sample



29-30 bottom-tercile vendors. Bar height = confirmed hallucinations (grounding detector); caught = blocked by validate_dossier before publish. The control's single escape (a customer-success vendor) passed the gate on a clean-looking dossier; under every new config, all hallucinating dossiers were gate-blocked (0 reached a published page).

Figure 4. Hallucinations generated vs. on-site exposure per config, full-pool sample

6. Discussion

6.1 No model clears zero on thin evidence: the gate is a control, not a model property. The G2 gate (zero ungrounded escapes) is not met by any configuration on this sample, including the flagship: the control produces six hallucinations (0.44%) across 30 vendors; the winner produces three (0.37%) across 30. This is the most operationally load-bearing result. The recommendation is not "use a better synthesizer"; it is *adopt the cheaper synthesizer and keep* `validate_dossier` *plus a fix loop in production*; the gate enforces correctness regardless of which model synthesizes. And containment, not just the raw rate, is where the recommended configuration wins outright: all 3 of its hallucinations were caught before publication and zero would have reached a customer, while the control, behind the same gate, would have published 1 of its 6.

The executed fix loop (§5.6) confirms the model's limit: one flagship redo cleared two of three residual hallucinations. The third vendor (healthcare-RCM, gold 31) produced a hallucination after the flagship redo because the flagship itself hallucinates there. For that vendor, no synthesizer in the study produced a clean dossier on thin evidence; correctness requires the validator plus human escalation. A model swap cannot eliminate that tail.

6.2 Synthesis quality plateaus; spend more buys nothing, spend less costs grounding. A band of models from $0.07 to $5.00 per million input tokens is statistically tied at ~0.4% (their measured differences are smaller than the noise on this sample size). The premium models (gpt-5, gemini-2.5-pro, o3) do not beat it, and o3 is worse. Below the plateau there is a cliff to 1.3% to 26% (Fig 2). So the synthesis-model decision is: pick the cheapest model that genuinely sits on the plateau (confirmed, not panel-lucky), which is grok-4.3.

6.3 Effective cost decides within the plateau. Cheaper-than-grok models exist (mistral-small at $0.07, qwen at $0.09) but their higher confirmed hallucination rates raise their re-run rate enough to erase the base saving (qwen effective ~$0.092 vs grok $0.086; mistral-small's redo overhead lands it in the same band per the ledger). Under the study's overwrite assumption grok wins; under a cheaper-redo assumption mistral-small could edge ahead, so we report the sensitivity rather than overclaim.

6.4 You cannot benchmark low-hallucination models on small panels. Ranking rare errors on 15 vendors is like judging whether a coin is biased from 5 flips: the order you see is mostly luck. At plateau rates (~0.2 to 0.5%), a 15-vendor panel carries ±1 to 2 hallucinations of noise on 400 to 900 claims, so panel rankings are unreliable. Six independent demonstrations: deepseek 0.21% to 0.56%, gemini-2.5-pro 0.40% to 0.63%, gpt-5 0.45% to 0.66%, mistral-small 0.22% to 0.71%, qwen3-235b 0.25% to 1.19%, qwen3-32b 0.62% to 1.27% (every panel front-runner regressed), plus the entire panel plateau being within overlapping intervals (Fig 3). The panel front-runners only looked best on easy panels; the plateau models (grok, opus) were stable across panel and full pool. **Rules this yields:** confirm on the full pool (roughly 29 vendors or more) before trusting any sub-1% rate; rank models by their confirmed or aggregate rate, never by a single lucky panel; put a confidence interval on every rate you report; and treat a cost-greedy stop at the first clean model as a shortlist, not a crown, because it picks winners on luck unless paired with confirmation.

6.5 The pre-registration was wrong in two informative ways. It predicted mistags would be "low"; they are the dominant defect class for *both* arms (131 control, 66 winner), driven by two detector-strict patterns (a vendor's social profile tagged as vendor-claimed, the vendor's own page tagged third-party). And it predicted the control would be a ~zero-escape baseline; it is not. Both misses make "match the flagship" a lower, real bar that the hybrid clears.

7. Threats to validity

Most important first.

Every rate is conditional on a clean frozen bundle. The detector proves claims against the bundle; it cannot prove the bundle against the world. The study built each bundle with a trusted retrieval pass and a banned-source lint before freezing. A production deployment that retrieves live from unvetted websites inherits none of this paper's guarantees until bundle construction is gated the same way: banned-source enforcement before the freeze, a completeness check that required pages actually fetched, and a fail-loud rule when either is violated. If a bad source enters the bundle, synthesis can be grounded-but-wrong, and no downstream check in this method will catch it.

The sample is too small to certify the 0.5% bound. With 30 vendors and the new arm's sparser dossiers (~27 claims/vendor vs 46 for the flagship), the winner's 99% one-sided Clopper-Pearson upper bound sits at about 1.24%; clearing the 0.5% target would take roughly 37 or more vendors, and that is a best-case floor: it assumes the added vendors produce zero new hallucinations; holding the observed rate fixed, the requirement grows several-fold. We ship the limit rather than the claim. The binding criterion in practice is the zero-escape gate, which no model meets anyway; the gate plus fix loop, not the bound, is what protects production.

Selection bias, by design. The 30 vendors are the lowest-scoring, thinnest-evidence vendors per industry, so every rate is an upper bound; typical vendors run cleaner for every model.

Coverage. 14 of 20 cohorts; six were excluded because their evaluation criteria were not finalized at study time and are queued for re-run.

No temperature 0. Providers no longer allow deterministic decoding, so single runs vary; mitigated by the frozen bundle, the enforced output format, and full-pool confirmation, not eliminated.

Clustering. Claims cluster within vendors, so per-claim intervals overstate precision; the vendor-level cluster bootstrap is the honest interval and is the one we lead with.

Underpowered for tiny differences. Telling 0.44% from 0.22% apart would need roughly ten times this study's data, so within the plateau we declare a tie and let cost decide.

Detector strictness is a modeling choice. The provenance rules are deliberately strict and applied symmetrically to both arms; two known strict patterns dominate the mistag counts (§6.5).

One known gate quirk. A format mismatch on the pricing-transparency check inflated gate-block counts symmetrically; it does not affect the hallucination metric or the winner.

The overwrite constant is a policy choice. §5.6 replaces the modeled $0.49 redo with a measured $0.265, which only strengthens the winner's cost case.

8. Recommendation

The transferable rules, for any grounded pipeline. (1) Keep a deterministic, fail-closed validator in production and run one fix loop before publish: no model is clean on thin evidence, so the gate, not the model, is what keeps hallucinations out of what ships. (2) Pick the cheapest synthesis model that genuinely sits on the quality plateau; premium models above it buy no grounding, and models below it fall off a cliff. (3) Decide between plateau models on effective cost (base + re-run rate x redo cost), not sticker price, because a model that fabricates more triggers more redos. (4) Confirm any candidate on a full sample, never a small panel: small-panel rankings of sub-1% hallucination are noise. (5) Grade with a deterministic, per-claim source-grounding check, not an LLM judge and not model agreement, both of which share the synthesizer's blind spots. (6) Gate the retrieval step itself: a grounding detector can only prove claims against the bundle it is given, so banned-source and completeness checks must run before the bundle freezes, or the pipeline produces grounded-but-wrong output that no later check can catch.

What that meant for our pipeline. A hybrid behind a fail-closed gate: the flagship retrieves and freezes the evidence bundle; **grok-4.3 synthesizes** (step 02); cheap models do cohort-fit (gpt-5-mini, 01) and validation (mistral-small, 03); `validate_dossier` plus one fix loop runs before publish. This matched flagship grounding quality at ~5.9x lower effective cost. Step 02 was not cheapened below grok-4.3 (the cliff), and no premium model above it was worth paying for (no quality gain). This configuration is what the study certifies; switching production onto it is gated on hardening the retrieval step first (the conditionality in §7) and an explicit operator go. One disclosure for completeness: the production deployment performs one additional layer of testing beyond everything this paper describes, intentionally kept out of scope here, whose job is to guarantee that no hallucination lands in production.

Figure 5. The recommended pipeline, stage by stage, with the agent that runs each stage. Retrieval (Claude, the flagship) is the only stage allowed to touch the live web: it fetches the vendor's real source pages, strips banned aggregators, and freezes the result into a SHA-256-hashed evidence bundle. Step 01 (gpt-5-mini) screens cohort fit and halts off-fit vendors before any synthesis. Step 02 (grok-4.3) writes the full dossier from the frozen bundle alone; it is the one stage where hallucinations are born. Step 03 (mistral-small) independently re-derives scores and provenance tags as a redundant check. The detector (ground.py, plain offline code, no model) then grounds every claim against the bundle, and the fail-closed gate (validate_dossier) blocks the entire dossier if any claim is ungrounded: one flagship redo, then human escalation. Figure 1 shows the transferable audit method; this figure is the concrete production instantiation the study certifies. Source: PROCESS-ARCH.html (editable) / .png, shipped with the public release.

Table 1. Recommended per-step model assignment.

STEP	ROLE	MODEL	$/MTOK IN / OUT
01: cohort-fit gate	Halts misfit vendors before synthesis	`openai/gpt-5-mini`	0.25 / 2.00
02: synthesis + scoring	Full dossier; the hallucination-bearing step	`x-ai/grok-4.3`	1.25 / 2.50
03: validation	Validates the step-02 dossier	`mistralai/mistral-small`	0.07 / 0.20
Retrieval	Evidence-bundle construction (web + tools)	`claude-opus-4-8`	5.00 / 25.00
Orchestration	Pipeline driver	`claude-sonnet-4-6`	3.00 / 15.00

9. Reproducibility: how to replicate this paper from the released data

Every number this paper claims is recomputable from the released data, and the release includes the script that does the recomputation. Internally the study keeps frozen bundles with recorded hashes, plus each cell's dossier, prompts, token usage, and detector verdict, so every (vendor x model) cell is independently re-gradable offline against a pinned price snapshot (2026-06-07). The public release is the anonymized projection of that archive: vendor identities are removed (keyed as `cohort-g<gold>`, the industry cohort plus the human-validated gold score), and model names are kept because they are the comparison.

9.1 What the release contains. The release is published at `github.com/ConnorYQueen/yardstick-grounded-pipeline-audit` (tagged `v1.1`, licensed CC BY 4.0: share, republish, and adapt with attribution).

- `LEDGER.json` / `LEDGER.csv`: the 454-row results ledger, one row per (vendor x model-combination), carrying claims, hallucinations, mistags, the gate outcome (`gate_blocked`, `postgate_escapes`), score delta vs gold, and cost.
- `defects-all-cells.json`: the deterministic detector's verdict for all 457 cells (454 ledger cells plus the three opus fix-loop redos), with per-claim sections, tags, defect reasons, and H1 to H7 counts; free text is stripped so nothing can de-anonymize a vendor.
- `rates-per-combination.md` / `results-per-combination.csv`: all 31 model combinations and their pooled hallucination rates.
- `results-headline-configs.csv`: the four full-pool configurations with confidence intervals and the Fisher comparison.
- `curated-examples.md`: the winner's three hallucinations shown in full, hand-redacted.
- `VENDOR-QUESTIONS.md`: the retrieval test contract for grading your own system on the same task.
- `PROCESS-ARCH.png`: the stage-by-stage process map (Figure 5).
- `DATA-DICTIONARY.md`: every file and every column, defined.
- `verify_results.py`: the verification script described next.

9.2 The replication protocol.

1. Read `DATA-DICTIONARY.md` first; it defines every column and the configuration-id scheme in about five minutes.

2. Run `python verify_results.py` inside the release folder (Python 3.9 or later, standard library only, no network and no model calls). It recomputes, from `LEDGER.json` alone: every per-configuration hallucination rate, the step-02 per-model aggregates, the cost model with the effective-cost ratio, the containment counts, the full-pool side of the reliability table, the 95% vendor-level cluster-bootstrap intervals (20,000 resamples, fixed seed), the Fisher exact test of winner vs control, the winner's 99% Clopper-Pearson upper bound, and the power statement from §3.9. It prints each value next to the value this paper states, with PASS or FAIL.
3. To audit any individual claim verdict, open the matching cell in `defects-all-cells.json` and check its per-claim entries and gate checks against the ledger row.
4. To test a different pipeline (including your own) on the same task, follow `VENDOR-QUESTIONS.md`: answer the ten questions per company from name and description alone, return a verbatim quote, live URL, and provenance tag per answer, and grade deterministically.

9.3 Where each headline claim lives.

PAPER CLAIM	RELEASE SOURCE	HOW TO CHECK
Control 0.44% (6 of 1,376), winner 0.37% (3 of 804) (§5.1, §5.3)	`LEDGER.json`	Pool rows by `config`; divide hallucinations by claims
Leaderboard and combination rates (§5.2)	`results-per-combination.csv`	Matches the pooled ledger rows per combination
Confidence intervals and parity (§5.3)	`results-headline-configs.csv`	Re-derived by `verify_results.py` (bootstrap + Fisher)
Reliability regressions, full-pool side (§5.4)	`LEDGER.json`	Pool each confirmed configuration; panel-15 values are historical (see below)
Costs: $0.037 base, $0.086 effective, ~5.9x (§5.5)	`LEDGER.json` `cost_usd`	Effective = base + redo-rate x $0.49
Containment: winner 3 of 3 caught, 0 published; control would publish 1 of 6 (§5.2, §6.1)	`LEDGER.json`	`gate_blocked` and `postgate_escapes` per row; hallucinations on unblocked rows
Fix loop: one redo clears 2 of 3 (§5.6)	`defects-all-cells.json` + `curated-examples.md`	The three `REDO-opus-fix` cells

9.4 What is not released, and why. Vendor names, domains, source URLs, and verbatim receipts are removed because they identify vendors directly or through acquirers, founders, and deal figures; the detector is deterministic, so the retained verdict fields reproduce every rate without them. Yardstick's internal scoring methodology is proprietary and is not part of the release; the public test measures retrieval accuracy and hallucination containment, not scoring. The 15-vendor panel rates quoted in §5.2 and §5.4 are historical values recorded before the confirmation extension folded those models' full-pool runs into the ledger, so they are reported, not recomputable; the full-pool side of every reliability row is recomputable. Two detector messages in the released artifacts have specific third-party host references (a web archive and a professional-network domain) deliberately removed as a publication precaution; the redaction is cosmetic, applied after all verdicts were computed, and does not affect any number in this paper.

Appendices

A1. Model roster and prices (USD per 1M tokens, in/out): claude-opus-4-8 5/25, claude-sonnet-4-6 3/15, claude-haiku-4-5 1/5; openai gpt-5 1.25/10, gpt-5-mini 0.25/2, gpt-5-nano 0.05/0.40, o3 2/8, o4-mini 1.10/4.40, o3-mini 1.10/4.40, gpt-4.1-mini 0.40/1.60, gpt-4.1-nano 0.10/0.40, gpt-4o-mini 0.15/0.60; google gemini-2.5-pro 1.25/10, gemini-2.5-flash 0.30/2.50, gemini-2.5-flash-lite 0.10/0.40, gemma-3-27b 0.08/0.16; meta llama-4-maverick 0.15/0.60, llama-3.3-70b 0.10/0.32, llama-4-scout 0.10/0.30; deepseek chat-v3.1 0.21/0.79, r1 0.70/2.50; mistral large-2512 0.50/1.50, small-3.2-24b 0.07/0.20; qwen qwen3-235b 0.09/0.10, qwen3-32b 0.08/0.28; x-ai grok-4.3 1.25/2.50; microsoft phi-4 0.07/0.14; cohere command-r 0.15/0.60.

A2. Pre-registered expectations vs actual. Pre-registration means we wrote our predictions down before running anything, so we cannot move the goalposts after seeing the results. Diffs, predicted then actual: (H1 fabricated quote ~0/~0 -> 0/0; H2 dead URL ~0/~0 -> 0/0; H3 misattribution low/low -> 131/66, far higher; H4 unsupported low/low -> 3/2; H5 entity-fact ~0/low -> 3/1; H6 phantom-integration low/low -> 0/0; H7 estimate-as-fact low/low -> 0/0; G2 escape ~0/low -> not zero for either; "02 riskiest to cheapen" confirmed; "control ~0 baseline" refuted).

A3. Defect taxonomy H1-H7 with definitions in §2; examples available per-cell in the ledger.

Backing data: the public release alongside this paper ships `LEDGER.json` / `LEDGER.csv` (454 rows), per-cell `defects-all-cells.json`, `DATA-DICTIONARY.md`, and `verify_results.py` (section 9). The internal archive additionally keeps each cell's dossier, prompts, token usage, and frozen evidence bundle.