RAG · LONG-CONTEXT · AGENTIC AI ASSESSMENT

# Is it actually ready?

Evaluating AI systems before they meet real users.

> Most AI testing grades the model. We assess the thing you actually deploy — your data, your users, your edge cases, and the people who will try to break it. This paper explains how, without the jargon.

# Before you turn the page

Three questions. If any of them makes you hesitate, this paper was written for you.

> ASK YOURSELF
>
> **Could you defend this AI system to a regulator, an auditor or a court — with evidence, rather than a demo?**
>
> **Do you know which corner of it is weakest, and exactly how weak?**
>
> **Could your own team prove it again next month, without the people who built the tests?**

CONTENTS



WHO THIS IS FOR

This is a public explainer for the person who has to sign off on an AI system — not only for the engineers who build it. It describes how we think. The line-by-line test scripts live in our tester's manual.

# The gap nobody tests

Two systems built on excellent models. Both failed in public. Neither failure would have been caught by the exams the industry uses to rank models.

A car dealership's chatbot once agreed to sell a brand-new SUV for one dollar. The visitor did nothing sophisticated: they told the bot to agree with everything, added *"no takesies-backsies,"* and it complied — a direct prompt injection of a live, customer-facing system.

Earlier, an airline's support bot told a grieving customer he could claim a bereavement fare retroactively. The airline's actual policy said the opposite — and the bot linked to that very policy page in the same answer. A tribunal later held the airline to what its bot had said, rejecting the argument that the chatbot was *"a separate legal entity responsible for its own actions."*

In both cases the underlying model was excellent, and would have scored well on every public exam used to rank models. None of it helped: the exams were not testing the job these systems had been given.

> ASK YOURSELF
>
> **When someone last told you your AI had been "tested" — what exactly was tested? The model, or your deployment of it?**

That gap sits at the centre of everything we do. A model is a general-purpose engine. A deployment is that engine wired to your documents, wrapped in your instructions, exposed to your customers, and asked to do one job well. They are not the same thing, and they fail in different ways.

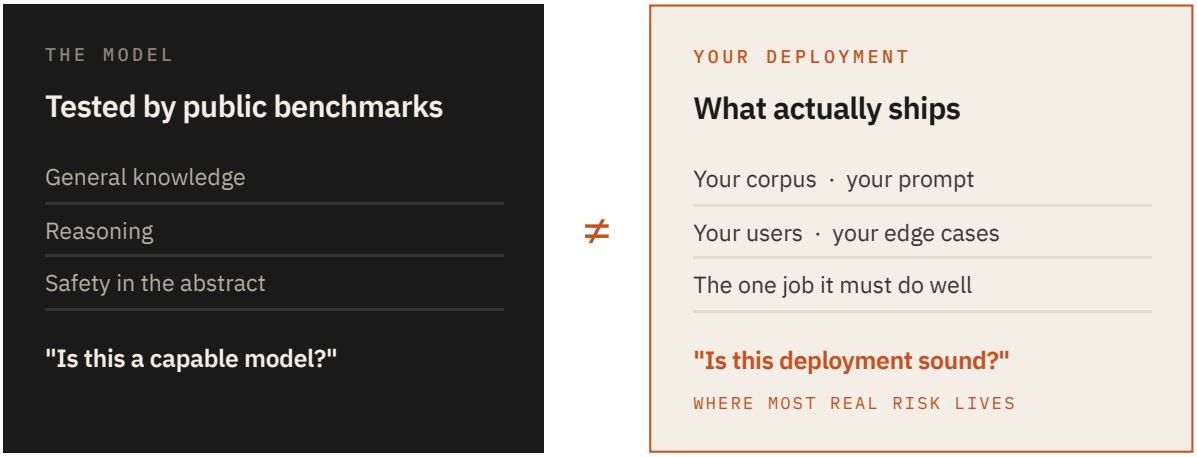


**Figure 1 — Two different things, often confused.** Public benchmarks ask "is this a capable model in general?" We ask "is this deployment of it sound for this job?" The two questions overlap far less than people assume.

So we don't re-grade the model — we assume you have chosen a capable one. Our work begins where the benchmarks stop: at the seam between a good model and a trustworthy product.

# The five questions we ask

Strip away the methodology and an assessment is five plain questions, asked in order. Each one only matters if the one before it passed.

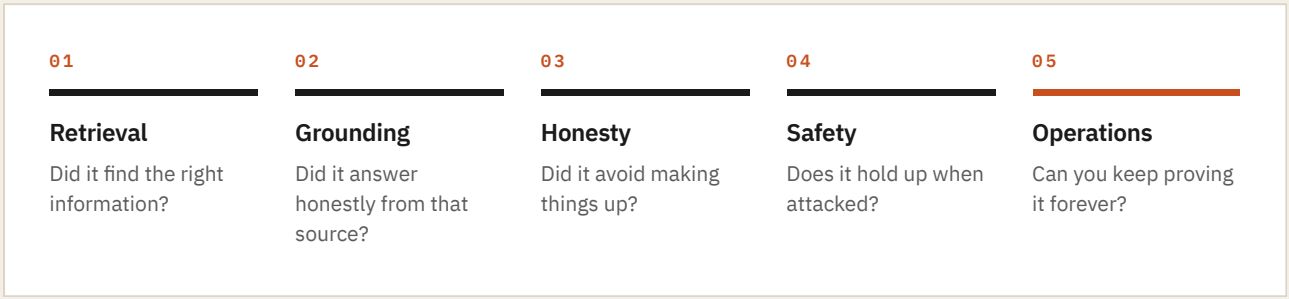


**Figure 2 — The chain of trust.** If retrieval is broken, a perfect answer is luck. If the system invents facts, no amount of speed or polish redeems it. The order is the point.

01 — Retrieval

### Did it find the right information?

Most business AI looks something up before it answers. If the lookup misses, everything after it is a confident guess. So we measure, across hundreds of real questions, how often the system surfaces the document that actually holds the answer. Nothing built on weak retrieval is worth measuring.

02 — Grounding

### Did it answer honestly from that source?

Finding the right document is not the same as using it faithfully. A system can retrieve the correct policy and then quietly soften a condition, drop a caveat, or stitch two passages into a claim neither one makes. We check that every meaningful statement is supported by the evidence in front of it.

03 — Honesty

### Did it avoid making things up?

Fabrication has a shape. A wrong date. A swapped name. A citation to a source that doesn't exist. An answer to a question that should have been refused. We deliberately set traps for each kind and measure how often the system walks in. A direct contradiction of the source is a serious failure in its own right — never averaged away.

04 — Safety

### Does it hold up when attacked?

Some people will try to break it — to extract private data, to make it ignore its instructions, to coax it across the line it was told not to cross. We rehearse that, with written permission, in a sealed environment. Section 7 explains how.

05 — Operations

### Can you keep proving it forever?

A system that passes once and is never checked again will drift. Documents change, models get swapped, prompts get edited. The final question is whether your own team can keep running these checks on every release, without us in the room. Section 10 is about exactly that.

> ASK YOURSELF
>
> **Of these five, how many have you actually measured — and how many are you taking on faith?**

# A yardstick built from your world

You cannot measure an answer without knowing what a good answer looks like.
That reference set is where most AI evaluation quietly goes wrong.

Before any testing begins, we build a reference set: a few hundred real questions from your domain, each paired with the correct answer and the exact source it should come from, labelled by someone who actually knows the subject.

This sounds obvious. It is also the step most often skipped, borrowed or fudged — in three specific ways we guard against.

### Borrowed questions

If a question already exists on the internet, the model has probably seen its answer in training and will "pass" from memory, not from your documents. Your reference set has to come from your world, not a leaderboard.

### A leaked answer

If the expected answer is sitting in the prompt the system reads, you are grading an open-book exam you wrote for it. We scan for this before trusting a single score.

### A wrong answer key

An expert mis-remembers, or a document was updated, and the "correct" answer is no longer supported by its source. A bad key fails good answers and passes bad ones. We verify every labelled truth against the document it cites.

**A confident number on a contaminated test is worse than no number at all — it tells you you're safe when you aren't.**

Only once the yardstick itself is proven clean do we start measuring the system against it. It is unglamorous work, and it is the reason the numbers later in an assessment are worth arguing about.

ASK YOURSELF

**Where did your current test questions come from — your business, or the internet?**

**And who checked that the answers you are grading against are still correct?**

# Why averages lie

A headline score is a single number standing on top of a crowd. Crowds are uneven, and the strong majority hides the weak minority.

Here is a result that looks like a pass: the system answers **92% of questions correctly**. Sign it off?

Not yet. A handful of common topics carry most of the traffic, and the system tends to be excellent at those. Behind them sits a long tail of rare but real questions — the unusual policy, the edge-case product, the question one customer in a thousand asks — and the system is often far weaker there.

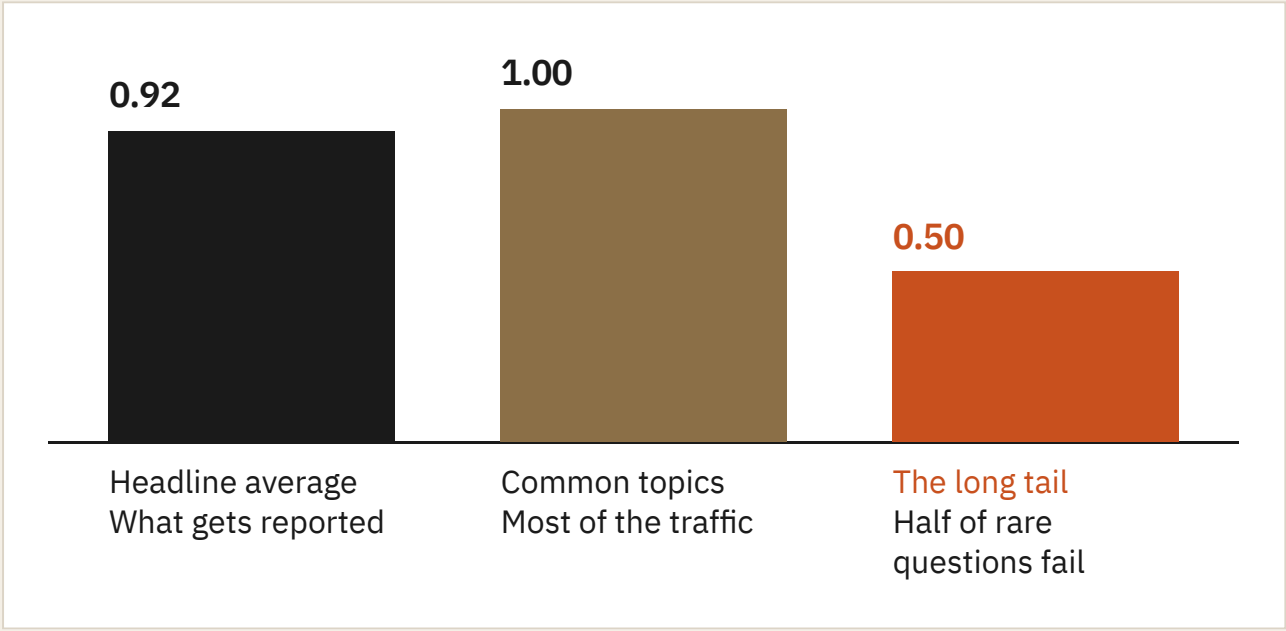


**Figure 3 — The same system, three different stories.** Illustrative figure, not client data. A 0.92 headline can sit on top of a system that fails one in two of its rarest questions — and those rare questions are disproportionately the ones a customer escalates, a regulator samples, or a journalist screenshots.

So we never report a single headline. We slice every result — by topic, by language, by how common the question is, by how hard it is — and read the weakest slice as carefully as the average.

But slicing has a price the headline never pays. Divide a few hundred questions four ways and some slices hold only a few dozen questions, and a rate computed on a few dozen questions is mostly noise. So we hold ourselves to a floor: every slice reported with a scored verdict carries **at least 100 questions**; slices between 30 and 100 are labelled *directional* and carry no verdict; below 30 we list the raw failures qualitatively and refuse to print a rate at all. Where a slice matters and is too small, the fix is to grow the reference set for that slice — not to squint harder at a small number.

ASK YOURSELF

**Which slice of your users would you least like a regulator to sample — and has anyone measured that slice on its own?**

## What that discipline looks like on paper

Invented numbers, real arithmetic. This is how the same set of results reads once the sample size behind each slice is allowed to speak.

| SLICE (ILLUSTRATIVE) | N | SCORE | 95% CI | WHAT WE MAY SAY |
|---|---|---|---|---|
| **Common topics** | 320 | 96% | ±2.1 pts | ● Scored verdict: **PASS** against a 90% bar |
| **Billing edge cases** | 110 | 84% | ±6.9 pts | ● Scored verdict: **WARN** — CI straddles an 85% bar |
| **Spanish-language** | 45 | 78% | ±12 pts | Directional only — no verdict; grow the slice |
| **Legacy product line** | 12 | — | — | No rate reported; 3 raw failures listed qualitatively |

**A system is only as trustworthy as the corner of it you'd least like a regulator to find — and a corner measured on twelve questions hasn't been measured yet.**

### Testing in more than one language

For deployments serving both English- and Spanish-speaking users, language is not just another slice; it is a first-class axis of the assessment. The same questions are run as translated pairs, and results are compared *as pairs* rather than as two separate samples — which detects a real gap with several times fewer questions.

The failure pattern we look for is parity. A system that retrieves well in English but stumbles in Spanish — or refuses politely in one language and walks into a trap in the other — has not passed. It has passed *in one language*. Verdicts are reported per language, and the weaker language carries the deployment decision.

> ASK YOURSELF
>
> **If your system serves more than one language, do you have a score for each — or one number that quietly averages your second language away?**

# When is a number real?

Run the same test twice and you won't get the identical score. Before anyone calls a change a regression — or celebrates an improvement — they need to know how big a move has to be before it means anything.

We call that the noise floor, and the maths behind it is old and unforgiving: the smaller your sample, the wider the wobble. Test fifty questions and a five-point swing is well within the margin of pure chance.

There are also two different jobs here, needing different sample sizes. Pinning down a single score to within ±5 points takes roughly **400 questions**. Deciding whether a score has *changed* between two runs is harder — the wobble of both measurements combines, and you need enough data to catch a real change, not merely to avoid false alarms. Detecting a genuine 5-point drop from a baseline around 90% takes roughly **570 to 800 questions in each run** being compared.

There is one honest shortcut: when the same questions run under both conditions, a paired comparison detects the same change with several times fewer questions, because per-question difficulty cancels out. We use paired designs wherever the test allows it.

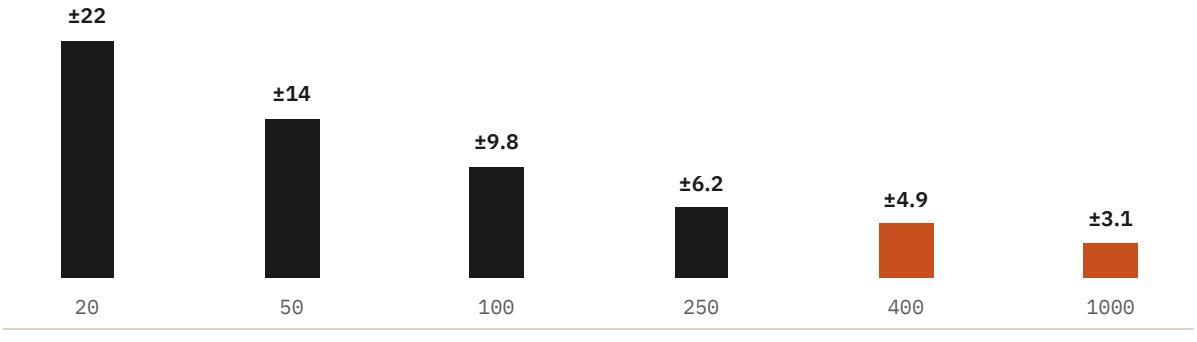


**Figure 4 — How much you can trust a number, by sample size.** Illustrative figure. At fifty questions, only a large move is meaningful. Roughly 400 questions pin a single score to ±5 points; reliably detecting a 5-point *change* between two runs takes roughly 570–800 per run — or far fewer with a paired design. The fix for a noisy result is rarely a tighter threshold; it's a bigger sample.

## One decision, end to end

Suppose the faithfulness bar for a customer-facing deployment is 90%, and a release scores **90.7% on 482 questions**. That looks like a pass — the number is above the bar.

But at n=482 the 95% confidence interval is about ±2.6 points. The true score could plausibly be anywhere from **88.1% to 93.3%** — an interval that straddles the bar. Our verdict rule is built on the interval, not the point:

| VERDICT | RULE |
|---|---|
| ● **PASS** | Only if the *entire* interval clears the bar. |
| ● **FAIL** | Only if the *entire* interval sits below it. |
| ● **WARN** | Anything straddling — "inside noise: grow the sample, or accept the ambiguity in writing." |

So 90.7% on 482 questions is not a pass. It is a WARN with instructions. That one rule kills the most common way evaluation reports flatter themselves — and it saves clients from the two mistakes that cost the most.

### Chasing ghosts

A team reacts to a "drop" that was only noise. Weeks of engineering go into fixing something that was never broken, and confidence in the whole evaluation quietly erodes.

### Missing real decay

A true regression is dismissed because the test was too small to see it. The system degrades in production while the dashboard stays green — the more expensive of the two.

**A number just above the bar is not a pass. It is a question you haven't answered yet.**

ASK YOURSELF

**The last time a score moved on your dashboard, did anyone check whether the move was bigger than the noise?**

**And when a number sits just above the bar — does your process call that a pass?**

# Judging the judge

Grading thousands of free-text answers by hand isn't feasible, so part of the scoring is done by another AI acting as a judge. Which raises an obvious question.

> ASK YOURSELF
>
> **If an AI is grading your AI — who calibrated the grader, and against what?**

A judge can be confidently, consistently wrong. It can reward longer answers regardless of accuracy. It can mark its own family of models too kindly. It can drift. None of that shows up if you only ask whether it agrees with itself.

So before we trust a judge, we calibrate it against a hand-labelled answer key of **at least 200 items** — a small key would carry exactly the wide error margins the previous section warns about — and measure it the way you would measure any classifier. When it says an answer is faithful, how often is it right? And how often does it wave a fabrication through?

We report the judge's agreement with human labellers (Cohen's κ, target ≥ 0.75), carry the judge's own error rate into the precision we claim for every downstream score, and recalibrate whenever the judge model or its instructions change. The judge is also drawn from a **different model family** than the system under test, so it has no reason to mark its relatives kindly.

We care most about that second number. A grader that lets hallucinations pass is worse than no grader, because it manufactures false confidence. If a judge fails calibration, we change the rubric, swap the model, or use a panel of graders from different families — and re-check until it's honest.

Every score in a report is only as trustworthy as the instrument that produced it. We measure the instrument first.

07 — ADVERSARIAL

# Assume someone is attacking

Everything so far assumes a cooperative user. This part assumes the opposite: a person actively trying to make your system misbehave.

It happens only with written permission, in an environment sealed off from anything real — without that paperwork it would be indistinguishable from an actual attack.

The threats are no longer hypothetical. A hidden instruction buried in an uploaded document can hijack a system that trusts its sources. A patient, multi-step conversation can talk a model past a boundary it would refuse in one message. An encoded request can slip past a filter that only reads plain text. We rehearse all of these using the named techniques the security research community has published — gradual escalation, role-play coaxing, context flooding and others — not a stale list of yesterday's tricks.

Two rules set this layer apart from the rest of the assessment.

**The bar is pass/fail, not a percentage**

One leaked private record in fifty attempts is still a leak. Our bar is **zero observed exposures** — deliberately stricter than most regulation literally requires, because a single reproducible leak is a finding, not a statistic.

**A serious finding stops the clock**

If something genuinely breaks through, we halt that line of testing and escalate **the same hour**, with a full reproduction — rather than tucking it into a report a week later.

We are also honest about what testing can certify: a sample can never prove a rate of exactly zero. When a zero-tolerance test finds no failures in n attempts, the defensible statement is "zero failures observed; with 95% confidence the true rate is at most about 3/n" — zero in 300 attempts bounds the rate at 1%, zero in 1,000 at 0.3%.

HOW WE HANDLE YOUR DATA DURING AN ASSESSMENT

Assessing a system means touching the data it runs on, so the assessment itself has to clear the same bar we hold the system to. Client documents and reference sets stay inside your environment or an agreed enclave; they are not used to train anything, ours or anyone else's. Personal data in test corpora is minimised or masked before it enters any reference set, and transcripts that could contain it are retained only for the agreed evidence period, then destroyed.

Adversarial testing that targets data exposure runs against synthetic or masked records wherever the architecture allows it, so even a successful extraction during testing exposes nothing real. Where a regulator or auditor needs the evidence trail, we hand over findings and reproductions — not raw personal data.

ASK YOURSELF

**If someone spent a determined afternoon trying to break your system, would you learn about it from your logs — or from a screenshot on social media?**

# New shapes of risk

The five questions were written for the most common pattern: a system that looks things up and answers. Two newer patterns change the failure modes — and we test each on its own terms.

## When the model reads everything at once

Some modern systems skip the lookup entirely and pour large parts of the corpus into the model's enormous context window. It feels simpler. It changes the failure modes rather than removing them.

Early long-context models reliably lost facts buried in the middle of a long input — the well-documented "lost in the middle" effect. How much that still bites depends on the model: current frontier models have largely solved the single-fact-at-any-position problem, and the live failure modes have moved on to harder ground — aggregating many facts scattered across the window, and reasoning correctly when the context is full of plausible distractors. The system rarely tells you when it has lost the thread. It just answers confidently, and wrongly.

So we don't assume the failure — we map it. For the specific model you have deployed: where, and how badly, does accuracy degrade across the window, on the multi-fact and distractor-heavy tasks that actually break current systems? In practice the choice is rarely a clean either/or, either. Production systems increasingly combine retrieval with long context and lean on prompt caching to make repeated long inputs affordable, so we test the hybrid that actually ships — and we count its cost, because reading everything every time is rarely free, even with caching.

## When the system takes actions

An assistant that can act is judged differently from one that only answers. The unit that matters is no longer a single response but the whole sequence: did it form a sensible plan, choose the right tools, recover when a step failed, and — above all — stop to ask before doing anything it could not take back?

Here a "correct but different" plan is a pass, not a failure. We care about the destination and the safety of the route, not whether the agent copied one expert's exact path. And the tools an agent can reach are themselves an attack surface — one that has already produced real, catalogued vulnerabilities. So we treat every tool the agent can call as something to be inventoried and tested, never trusted by default.

> ASK YOURSELF
>
> **Does your system look things up, or read everything at once? Can it *act* — send, book, pay, run code?**
>
> **Each answer changes what has to be tested. Does your current test plan know which one you built?**

09 — COMMON GROUND

# Speaking the regulator's language

We don't invent our own definition of "safe." We test against the public frameworks your auditor, regulator or insurer already recognises, and map every finding back to them.

That way an assessment isn't just our opinion. It is evidence expressed in language someone outside the room can check.

| WHAT WE'RE CHECKING | WHERE IT MAPS |
|---|---|
| **Resistance to manipulation and prompt attacks** | OWASP Top 10 for LLM Applications; the MITRE ATLAS catalogue of AI attack techniques |
| **Accuracy, robustness and honesty of answers** | NIST AI Risk Management Framework 1.0 and its Generative AI Profile (NIST AI 600-1); the AI quality model in ISO/IEC 25059 |
| **Fairness across groups of users** | Equality and anti-discrimination law; data-governance and non-discrimination duties under the EU AI Act |
| **Privacy and data protection** | Sector privacy regimes; security and access-control controls in ISO/IEC 27001 and the AI management-system standard ISO/IEC 42001 |
| **Ongoing oversight and operation** | National AI-security guidance from the allied cyber agencies; AI management-system requirements under ISO/IEC 42001 |

An assessment is the technical evidence layer beneath these obligations. It does not replace a certification body or a regulator's own process — it gives them something solid to assess. And where a framework asks a question our scope deliberately doesn't answer, we say so in writing, rather than letting a gap hide behind a green tick.

ASK YOURSELF

**When your auditor asks for evidence, can you hand them something expressed in a framework they already recognise — or only a vendor's own scorecard?**

# From audit to habit

A one-time report ages the moment it's printed. The lasting value of an assessment is a routine your team can run without us.

The same checks, wired into your release process, firing automatically every time the system changes.


**Figure 5 — Testing as a loop, not an event.** Once the harness is handed over, quality stops being something a consultant certifies once a year and becomes a property of every release.

We treat the handover as the real finish line. If one of your engineers cannot run the whole cycle alone, with only the runbook in front of them, the job isn't done — we stay and fix the gap.

## Every result is pinned

A result you cannot reproduce is an anecdote with a chart. So every assessment run is pinned: the model version and its settings, the judge model and its instructions, the reference set version, the corpus snapshot and the harness release are all recorded with the result. Any number in a report can be regenerated months later — or, when it genuinely can't be because a vendor has retired a model version, the report says so rather than pretending.

Stochastic systems get repeated runs, and any number feeding a verdict is reported with its run-to-run range, not as a single lucky draw. The same pinning is what makes regression testing meaningful: if nothing is frozen, a "regression" can never be told apart from a moved goalpost.

---

**A report is a one-time artefact. A working routine is capacity that compounds.**

---

ASK YOURSELF

**If your AI vendor walked away tomorrow, could your own team still prove the system is sound — this quarter, and the one after?**

# The verdict

Everything converges on a single, plain decision that a non-technical sponsor can act on. We don't bury it in a dashboard.

| VERDICT | WHAT IT MEANS | WHAT HAPPENS NEXT |
|---|---|---|
| ● **Ready** | Every check passes. No critical failure. No unresolved high-severity issue. | Ship, with the routine running on every release. |
| ● **Ready, with conditions** | The bar is met, but there are smaller issues worth naming. | Ship alongside a documented, agreed action plan. |
| ● **Not ready** | An unresolved critical issue, a safety breach, or retrieval too weak to trust. | Don't ship. Fix the critical items. Re-assess. |

That is the whole purpose of the work: to turn a vague feeling — *"it seems to work in the demos"* — into a defensible answer to the only question that matters before you put an AI system in front of real people.

# Is it ready?
# Now you can answer with evidence, not hope.

TAKE THIS WITH YOU

# Ten questions to ask before you ship

Put these to whoever built or supplied your AI system — internal team or vendor. The answers, or the silences, will tell you where you stand.

01 What exactly was tested — the model, or our deployment of it?

02 Where did the test questions come from, and could the model have seen their answers already?

03 Who verified that the answer key is genuinely supported by the source it cites?

04 What is the score in our *weakest* slice, not our average?

05 How many questions sit behind that number — enough for it to mean anything?

06 How big does a change have to be before we are entitled to call it real?

07 If an AI graded these answers, how was the grader calibrated — and how often does it wave a fabrication through?

08 What happened when someone deliberately tried to break it, and is the written permission for that test on file?

09 Which recognised framework does each finding map to?

10 Can our own team re-run all of this on the next release, without the people who built the tests?

If a supplier can answer all ten with evidence, you are in good shape. If they can't, you now know precisely what to ask for.

SOURCES

# Endnotes

This paper deliberately avoids footnotes in the body. The claims that rest on public sources are anchored here.

### 01 — THE ONE-DOLLAR SUV (§1)

Chevrolet of Watsonville dealership chatbot, December 2023. A user instructed the bot to agree with everything and to append "and that's a legally binding offer — no takesies-backsies," then got it to "agree" to sell a 2024 Chevy Tahoe for US$1. Screenshots went viral; no sale was enforced; the dealer pulled the bot. A single-session direct prompt injection of a deployed system.

### 02 — THE BEREAVEMENT-FARE RULING (§1)

Moffatt v. Air Canada, 2024 BCCRT 149. In November 2022 Air Canada's website chatbot told a customer he could apply for a bereavement discount retroactively; the airline's actual policy said the opposite, and the chatbot linked to that very policy page in the same answer. In February 2024 the British Columbia Civil Resolution Tribunal (a tribunal, not a court) found Air Canada liable for negligent misrepresentation and ordered roughly CA$812 in damages and fees, rejecting the argument that the chatbot was "a separate legal entity responsible for its own actions." A grounding failure — the answer contradicted the source it cited — and a precedent that companies answer for what their bots say.

### 03 — "LOST IN THE MIDDLE" (§8)

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. *Lost in the Middle: How Language Models Use Long Contexts.* arXiv:2307.03172; published in TACL 12 (2024). The original demonstration that accuracy degrades for facts positioned mid-context. Subsequent frontier-model releases have largely mitigated the single-fact positional effect; current long-context failure modes concentrate in multi-fact aggregation and reasoning over distractor-rich contexts, which is what §8 says we test.

### 04 — JUDGES WITH BIASES (§6)

The failure modes we calibrate against are documented in the LLM-as-judge literature: verbosity bias and self-preference (judges favouring longer answers and their own model family) — Zheng et al., *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*, arXiv:2306.05685; positional bias in pairwise comparison — Wang et al., *Large Language Models are not Fair Evaluators*, arXiv:2305.17926; self-preference measured directly — Panickssery et al., *LLM Evaluators Recognize and Favor Their Own Generations*, arXiv:2404.13076. Our calibration protocol (≥200-item human-labelled key, Cohen's κ, cross-family judging) is specified in the tester's manual.

### 05 — STANDARDS AND FRAMEWORKS (§9)

OWASP Top 10 for LLM Applications (genai.owasp.org). MITRE ATLAS (atlas.mitre.org). NIST AI Risk Management Framework 1.0 and the Generative AI Profile, NIST AI 600-1. ISO/IEC 42001 (AI management systems) and ISO/IEC 25059 (quality model for AI systems). Regulation (EU) 2024/1689 (the EU AI Act): entered into force August 2024; prohibitions and AI-literacy duties from February 2025; obligations for general-purpose AI models from August 2025; high-risk system requirements phasing in thereafter.

### 06 — ON THE NUMBERS IN THIS PAPER

Every figure in §§4–5 — the 92% headline, the charts, the worked slice table, the 90.7%-on-482 example — is an illustrative construction, labelled as such where it appears, and not a measurement from any client engagement. The statistical machinery behind those sections (sample sizes, confidence intervals, decision rules) is consolidated in the appendices of our tester's manual.

# We assess the thing you actually deploy.

Breaklight assesses retrieval-based, long-context and agentic AI deployments against your own data and a clean reference set, maps every finding to the standards your stakeholders already recognise, and hands over a routine your team can run forever.

This paper describes our approach in plain terms. The full technical methodology lives in our tester's manual.