Two reports land on your desk. Both were written by AI. Both look clean, sourced, and confident. One of them has a number in it that is wrong.

Which one?

You cannot tell by looking, and that is the whole problem. When an AI does the math and gets it wrong, the wrong answer looks exactly like a right one: a plausible figure in a confident sentence. No red flag, no error, nothing torn. Just a false number wearing the same clothes as all the true ones.

A spreadsheet with a broken formula does not do that. It hands you #REF!. It fails loudly, on purpose. The error announces itself.

That gap, between a failure that hides and a failure that announces itself, is the subject of this paper. The teams who trust their AI reporting are not the ones with the best model. They are the ones who moved the math somewhere that fails loudly, and left the model to do the one thing it is actually good at: the words.

So here is the claim, and it will not flatter the models. You cannot make AI-generated reporting trustworthy by choosing a smarter one. Capability and trustworthiness are different axes, and the most capable model is often the least trustworthy. Even the most cautious model still fails too often on real work to carry a report by itself. So you stop shopping for a model you can trust, and you build a workflow that is trustworthy around the one you have. Compute the number with something deterministic. Let the model narrate words it did not invent. Tag every claim back to its source. Stand a human at the few decisions that carry weight. Compute, then narrate, with a wall between them.

> ***The one rule.*** *Never let the model produce a number. Compute it with code, a query, or a governed metric layer, where a wrong answer fails loudly. Let the AI write the sentences*

> *around numbers it did not calculate. Put a direct source link and a status on every claim it makes. Then route only the high-stakes outputs to a human. The model writes. It does not count.*

## A wrong number that looks right

I am not going to quote you a scary hallucination rate from 2024. Those studies tested GPT-4 and Llama 2, models nobody runs anymore, and leaning on them today would be its own small dishonesty, the exact kind this paper is about. So here is the current picture, mid-2026.

The good news first. On easy, short, well-grounded tasks, the best models now hallucinate under 2% (Vectara Leaderboard, May 2026). A real four-fold improvement in two years.

Now the rest. On long documents, every frontier model still fabricates more than 10% of the time (a 172-billion-token study across dozens of open-weight models, March 2026). On open-domain factual questions, the rates run from 36% to 85% depending on the model (AA-Omniscience benchmark, via DeepLearning.AI, May 2026). And citations, the thing we add precisely to make a report trustworthy, still do not hold. In one study, 57% of them did not actually support the claim they were attached to (arXiv, December 2024), and the problem is confirmed alive in current commercial tools (April 2026). The model writes the claim first, then finds a citation that looks close enough. A footnoted AI report can pass a glance and still be wrong.

Sit with the stakes. A weekly business review can carry 200 figures. On the long, realistic documents that still fail more than 10% of the time, that is twenty or more quiet errors, each indistinguishable from the truth, going to whoever reads the review. For a team whose reports reach leadership, that is not a tail risk. It is Tuesday.

## Why a smarter model won't fix it

The instinct is to wait for the next model, the one that finally gets it right. The 2026 data says that instinct will cost you.

Take one comparison. On a 2026 open-domain benchmark, the highest-accuracy model was also the highest-hallucinating, because it answers confidently when it should abstain, while a more cautious model hallucinated less than half as often (AA-Omniscience, via DeepLearning.AI, May 2026). Some of that is just the refusal knob: a model that guesses more gets more right and more wrong. That is the point. "Most capable" and "most trustworthy" are set by different knobs, and turning up capability does not turn up trust. And note where it leaves you. Even the cautious model in that test is wrong more than a third of the time on open recall. Picking it does not save you. It loses slower.

Then there is the pattern that survives every model generation. Long documents break faithfulness. Numbers stay the danger zone, because a language model predicts a plausible next token, and a plausible number is not a correct one. Even handing the model a calculator does not settle it: if it formalizes the problem wrong, the code runs the wrong calculation with total confidence (May 2026). The failure is not the model being weak. It is asking a thing that generates plausible language to also be a source of truth. Those are two different jobs.

So a smarter model is not the fix, any more than a faster car is the fix for not knowing where you are going. The fix is to stop asking the model to be trustworthy, and to build a process that is trustworthy around it.

## The framework: Compute, then Narrate

The whole idea fits in three words, and the order matters. Compute the number first, with something that is not a language model. Then let the language model narrate. The wall between the two is the principle. The five moves below are how you build it.

None of these five moves is new on its own. Teams already move math to tools, tag claims, gate humans, keep regression sets. What is missing in practice is the absolute and the order. The model never computes a number. Narration only ever wraps numbers it did not make. The four supports exist to enforce that one rule. The contribution is not a new technique. It is refusing the comfortable middle where "the model usually gets the math right" is treated as good enough.

**1. Compute it deterministically.** The number comes from code, a SQL query, or a governed metric layer. Never from the model. This is not a style preference, it is measured. One April 2026 benchmark found the same models answering the same questions scored 84 to 90% accuracy when they wrote their own SQL, and 98 to 100% when the math was done by a deterministic semantic layer and the model only picked the metric (dbt Labs). It is an open benchmark you can run yourself. The reason is the asymmetry from the opening: when the model does the math, a wrong answer looks plausible; when the deterministic layer does it, a wrong answer looks like an error. Deterministic does not mean infallible, though. It means a different kind of failure. Broken plumbing throws an error instead of a confident wrong number, but the logic itself can still be wrong. The difference is that it is logic you can read, test, and version, which a number born inside the model is not. The scalable version of "do the math in a spreadsheet" is a metric layer or a script, so this holds for automated, recurring reports, not just a person checking by hand.

**2. Narrate, do not calculate.** Now the model earns its keep. It writes the summary, the context, the so-what, over numbers it did not invent. It is a copy machine for

figures that already exist, not a calculator. It is genuinely good at this, and this is where the speed comes from.

**3. Tag every claim.** Each sentence the model writes carries two things: a direct link to the source, and a status. Verified, not yet checked, or unverifiable. At volume this is automated. Research systems already walk each claim back to its source and label it supported, not fully supported, or inconclusive (Microsoft's VeriTrail, 2025, is the clearest example; the specific tools will turn over, the move is what lasts). The human does not check every claim. The human adjudicates the ones the system flags. "Trust me, it is sourced" stops being a phrase and becomes a status you can see. And treat any causal claim, the "revenue grew because of strong demand" kind, as unverifiable by default, because the number can be exact while the reason is invented.

**4. Gate the humans where it matters.** Not a human on everything, which only moves the bottleneck. A human on the outputs that carry weight: a number above a dollar threshold, a metric moving more than it should, a new and unfamiliar data source, anything feeding a high-stakes decision. Sample a small slice of the rest. The 2026 teams who do this well keep human intervention between 2 and 10% of outputs, and they confirm the specific flagged claim rather than waving through the whole document. Fast enough that nobody skips it, focused enough that the human actually reads the thing in question.

**5. Keep a golden set.** A small library of outputs you know are correct, re-run on every model change and every prompt change. It catches the silent drift nothing else will: a new model that quietly starts getting one thing wrong, a slip that never trips a human gate because it never looks alarming. It is your smoke alarm. It is also how you catch the one error the deterministic layer cannot, which comes next.

Let me show you one I built. I made a generator that turns a fulfillment-automation data room into board-ready briefs, decks, and white papers, on a single rule: the model never computes a number. A figure like fleet units-per-labor-hour starts in a CSV.

Plain code reads the rows, averages them, reconciles the result against an Excel financial model, and logs every computed number to the console as the build runs. The model only writes the sentences around those figures. Every chart traces back to the row it came from, and there is no model call anywhere in the pipeline, so it runs fully offline and the operational data never leaves the machine.

I will be straight about the edge I am still closing. The computed numbers, the tiles and the charts, are bound to the data. The prose is not yet. A figure typed into a sentence by hand can still drift from the chart beside it, and binding every sentence to that same computed value, which is move three above, is the version I am building next. That is the honest distance between "the model never invents a number," which is true today, and "every number in the document is provably tied to its source," which is close but not done. Naming that gap is the discipline the rest of this paper is about.

Compare either version to the path most teams are on: ask the model for the number, take the confident figure it returns, paste it into the deck. My system can still be wrong, but it is wrong in the data or a formula, where you can see it. The common path is wrong in a sentence, with a citation, in front of your CEO.

## What this does, and what it does not

I want to be precise about the promise, because overpromising is how trust gets lost in the first place.

Compute, then Narrate does not make trust vanish. It moves it from one place you cannot inspect, the number inside the model, to three you can: the narrative, the tag, and the gate. Each can still fail. The win is that each now fails somewhere a person can see and correct, which a number buried in the model never did.

It has one blind spot worth naming, because it is the failure this paper opened by denouncing. The wall moves the math out of the model. It does not move out the judgment of how a metric is defined. A semantic layer computes consistently against a definition a human chose, and if that definition is wrong, it will be wrong exactly, forever, with a source link attached. A wrong "churn" formula does not throw an error. It returns a confident, traceable, wrong number. The golden set and the human gate are how you catch a wrong definition, the one error the deterministic layer will never flag for you.

So the honest line to give a leader is narrow and strong. The numbers are exact and sourced. The judgment stays with a person: not on every routine line, which is sampled, but on every call that carries weight. The machine writes. A human still owns the decisions that matter. That is the same owner the sister paper to this one insists on, the named person behind every decision.

## What it's worth

Three parts to the case, and I will say which are hard and which are soft.

The hard part is the accuracy delta, and it is measured. Moving the math from the model to a deterministic layer bought 8 to 16 points of accuracy on the same models in that 2026 benchmark. Not a projection. An open test you can run.

The soft part is the cost of a bad number, and I will not hand you a fake figure for it. The shape is plain enough. One wrong number in a board deck can move a decision, blow a quarter, or cost the thing that is hardest to rebuild, which is leadership's belief that your reports are true. The workflow above is cheap against any of those. The asymmetry does the arguing.

The return that is easy to miss is the biggest one. Compute, then Narrate is more than a safety measure. It is the thing that lets you automate reporting at all. Most teams are not choosing between full automation and full hand-work. They are shipping half-automated reports and quietly hoping. That hope is what this replaces. Without the wall, you cannot trust an automated report, so you keep hoping. With it, you ship the weekly and the monthly at the speed the model gives you, and you trust what ships. Speed and quality stop being a trade.

## Where to start

You can begin this week, with what you have.

1. **Find where the model touches your numbers.** Walk your current AI reporting and mark every place a figure is produced or transformed by the model itself. That list is your risk.
2. **Move the math out.** Push each calculation into a query, a script, or a metric layer, and let the model reference the result instead of producing it. The first pass is cheap. A fully governed metric layer is real work, quarters not days, so start with the figures that reach leadership and grow from there.
3. **Demand a link and a status.** Make it a rule: every AI-written claim carries a source link and a verification status. A claim that cannot get one does not ship.
4. **Set your triggers.** Write down what sends an output to a human, a dollar line, a sharp move, a new data source, so it is explicit and not left to mood.
5. **Build the golden set.** Save ten outputs you know cold, and re-run them whenever the model or the prompt changes.

None of this needs a platform or a new team to start. It needs deciding that the model is not allowed to be the source of truth, and then arranging the work so it cannot be.

## What to watch

Do not trust the model's confidence. It is close to the worst signal you have, because the model is most fluent exactly when it is inventing. Confidence is a writing style, not a measure of truth.

Watch who checks the checker. If you use one AI to judge another's output, do not use the same model family, because it tends to agree with itself, and a judge can be talked into a wrong verdict more easily than you would like. The human still adjudicates the cases that matter.

Automate the tagging, not the judgment. The system can flag what is unsupported. A person decides what to do about it, and that person needs the standing to hold a number back.

Keep an eye on the rules. Hallucination is now a named risk in the US AI framework, and the EU's content-disclosure deadlines arrive in late 2026. The traceability you build for trust is becoming the traceability you build for compliance.

## The model writes, a person owns

There is an old discipline buried in all of this, and it predates AI. You do not trust a number because someone confident said it. You trust it because you can follow it back to where it came from. AI did not change that rule. It just made it easy to forget, because the confident voice now comes from a machine that never tires and never blinks.

So keep the jobs apart. A deterministic system owns the math, where a wrong answer fails loudly. The model owns the words, where it is genuinely good. And a person

owns the judgment, because that was never the machine's to take. Compute, then narrate. The order is the point.

---

## Sources


- Vectara Hallucination Leaderboard (checked May 2026). https://github.com/vectara/hallucination-leaderboard
- "How Much Do LLMs Hallucinate in Document Q&A?" 172B-token study (arXiv 2603.08274, March 2026). https://arxiv.org/html/2603.08274v1
- AA-Omniscience benchmark, via DeepLearning.AI The Batch #351 (May 2026). https://www.deeplearning.ai/the-batch/issue-351
- "Correctness is not Faithfulness in RAG Attributions" (arXiv 2412.18004, December 2024; published ICTIR 2025). https://arxiv.org/pdf/2412.18004
- "Detecting and Correcting Reference Hallucinations in Commercial LLMs" (arXiv 2604.03173, April 2026). https://arxiv.org/pdf/2604.03173
- dbt Labs, "Semantic Layer vs. Text-to-SQL: 2026 Benchmark Update" (April 2026). https://docs.getdbt.com/blog/semantic-layer-vs-text-to-sql-2026
- Microsoft Research, "VeriTrail" (Aug 2025). https://www.microsoft.com/en-us/research/blog/veritrail-detecting-hallucination-and-tracing-provenance-in-multi-step-ai-workflows/
- "Reasoning, Code, or Both?" math-variation study (arXiv 2605.26414, May 2026). https://arxiv.org/html/2605.26414v1
- Improvado, "Human-in-the-Loop AI" HITL triggers (June 2026). https://improvado.io/blog/human-in-the-loop-ai
- NIST AI 600-1, Generative AI Profile (July 2024). https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-generative-artificial-intelligence