

If a correcting model was right in 18 out of 23 fact-checks, its observed correction accuracy was about 78.3%. That is a useful signal: in this set of cases, the model usually identified or supplied the correct answer. But it is not, by itself, proof that the model is reliable, that the [multi AI chat](#) original answers were wrong, or that a system using the model is safe to deploy.

The number is best treated as a small evaluation result that needs context. What counted as a fact-check? Who judged whether the correction was right? Were the cases representative of real users' questions? And what happened in the five cases where the correcting model was not right?

Those details matter because cross-model verification is not simply a matter of asking one AI system to double-check another. It is a workflow for surfacing uncertainty, testing claims, and deciding what to do when models agree or disagree. Done well, it can improve decision quality and catch some hallucinations. Done poorly, it can add confident-sounding errors and make them look validated.

The basic math—and what it does not tell you

The calculation is straightforward:

$18 \div 23 = 0.7826$, or approximately 78.3%.

In plain English, the correcting model gave a correct correction in roughly four out of five evaluated cases. The remaining five cases—about 21.7%—were not correct according to the evaluation criteria.

That percentage is a description of the sample, not a guarantee about the next answer. Twenty-three cases is a small denominator. A few different results would shift the rate substantially: 17 correct would be 73.9%, while 19 correct would be 82.6%. The result should therefore be read as an early performance signal, not a precise estimate of how the system will perform across all future fact-checks.

Measure What it says What it does not say 18 correct out of 23 The correcting model was judged right in 18 evaluated cases. That it will be right 78.3% of the time in every setting. Five unsuccessful corrections There were five cases where the correction did not meet the evaluation standard. Whether those were minor errors, unsupported claims, or serious reversals. Model-to-model agreement Two models produced the same or similar answer. That the answer is factually verified or supported by reliable evidence.

First, define what “right” means

Correction accuracy is only meaningful if the evaluation has a clear standard for correctness. A correcting model might be considered right because it matches a reference answer, cites a source, identifies a specific false claim, or gives a more defensible explanation. Those are different tests.

For example, a model may reach the correct conclusion but give an inaccurate explanation. It may flag a genuine issue in the original answer while proposing a correction that introduces a new error. Or it may be directionally right but too uncertain or incomplete to support an important decision. A single yes-or-no score can hide all of these differences.

Good evaluations specify the unit being judged. Is the model correcting one sentence, a full answer, or a set of factual claims? Are partial corrections awarded credit? Does a correction need to cite evidence? Are ambiguous or time-sensitive questions handled separately?

It also matters who or what did the judging. A qualified human reviewer working from credible sources is different from an automated “judge” model comparing two answers. Automated fact-check judges can help scale evaluation, but they can also miss subtle errors, favor a particular writing style, or mistake confident detail for evidence. For consequential use, reviewers should be able to inspect the original claim, proposed correction, and supporting sources.

Why disagreement can improve decision quality

In a multi-model workflow, disagreement is not necessarily a system failure. It can be a useful signal that a claim deserves more scrutiny.

Suppose one model says a policy took effect in March, while another says it took effect in July. If the system simply selects one answer, it hides the conflict. If it records the disagreement, checks the relevant source, and presents the unresolved point for review, the conflict can lead to a better decision.

That is the value of disagreement as a feature: it helps prioritize attention. A system does not need every model to agree on every answer. It needs a sensible way to distinguish low-risk consensus from high-impact uncertainty.

But disagreement alone does not establish which answer is correct. Two models can disagree and both be wrong. They can also agree because they share similar training data, common assumptions, or the same mistaken interpretation. Agreement is a clue; evidence is what helps settle the claim.

Multi-model orchestration versus model aggregators

The terms are sometimes used interchangeably, but a model aggregator and a multi-model orchestration system solve different problems.

What a model aggregator does

A model aggregator gives users access to multiple models—potentially including systems from OpenAI, Anthropic, and Google—through one interface or API. It can make it easier to compare outputs, route requests, or switch providers. That is useful, especially for teams that want provider flexibility or a single integration point.



But access to multiple models does not automatically create a verification process. If a user receives three answers with no source checking, confidence assessment, or conflict handling, the user still has to decide which answer to trust. An aggregator can offer choice without providing a method for resolving disagreement.

What orchestration adds

Multi-model orchestration defines how models and tools work together to complete a task. It may assign one model to draft, another to critique, and a retrieval or browsing system to check claims against sources. It can also set rules for when to ask for another review, when to escalate to a human, and what evidence to retain.

In other words, orchestration is about workflow and decision logic, not just model availability. It can include an aggregator, but it adds a process for using the available models deliberately.



Sequential and parallel workflows

There are two common ways to run a correction workflow. Neither is universally better; the right choice depends on the stakes, the time available, and the cost of a missed error.

Sequential: draft, then challenge

In a sequential workflow, one model produces an answer first. A second model then checks the answer, identifies claims that may be false or unsupported, and proposes corrections. A final step may validate those corrections against sources or send uncertain cases to a person.

1. **Generate:** Produce an initial answer and identify its factual claims.
2. **Challenge:** Ask a correcting model to test the claims rather than simply rewrite them.
3. **Verify:** Check disputed or high-impact claims against reliable evidence.
4. **Resolve:** Accept, reject, or escalate the proposed correction.

Sequential workflows are often easier to implement and can use fewer model calls. Their weakness is that the first answer can anchor the second model. A reviewer model may focus on the framing it has been given, repeat an unsupported premise, or spend its effort polishing the answer instead of independently questioning it.

Parallel: answer independently, then compare

In a parallel workflow, multiple models assess the same question independently. Their answers are compared afterward. If they agree, the system may treat the claim as lower risk; if they disagree, it can trigger a source check or human review.

Independent generation can reduce anchoring because each model starts without seeing the others' conclusions. It can also reveal differences in assumptions or interpretation. The trade-offs are more model calls, greater latency, and a larger set of outputs to evaluate.

Parallel review works best when models are genuinely asked to reason independently and the comparison is claim-level. A simple majority vote can be misleading: three unsupported answers do not become evidence just because they outnumber one careful answer.

How cross-model verification can catch hallucinations

Cross-model verification can help catch hallucinations by making the first answer's claims explicit and asking a separate process to test them. This is especially useful for specific details that are easy to invent or misremember: dates, names, quotations, figures, legal requirements, and descriptions of what a source says.

A practical verification workflow should do more than ask, "Is this answer correct?" It should ask the reviewer to identify each checkable claim, state what evidence would confirm or contradict it, and distinguish verified information from inference. Where possible, the workflow should retrieve the underlying source and check whether it actually supports the claim.

For instance, if an answer says a regulation took effect on a particular date, a useful check is not merely whether another model repeats that date. The system should locate an authoritative source, confirm the date in context, and note whether the rule has since changed.

Cross-checking still has limits. Models may share the same blind spots, fail to access current information, misread a source, or produce fabricated citations. A reviewer model can also hallucinate while trying to correct a hallucination. The goal is not to make model agreement a substitute for verification. It is to use model review to identify where verification is most needed.

What to examine in the five unsuccessful cases

The five cases where the correcting model was not right may be more informative than the headline percentage. They can show whether errors follow a pattern and whether the workflow needs a targeted change.

- **Wrong correction:** Did the model replace a true claim with a false one?
- **Unsupported correction:** Was the correction plausible but not backed by evidence?
- **Partial correction:** Did it catch one problem while leaving another important error in place?
- **Ambiguous judgment:** Did the evaluator have a clear basis for marking the answer right or wrong?
- **High-impact miss:** Could the error affect a customer, financial decision, compliance obligation, or public statement?

These categories point to different fixes. A source-retrieval problem may call for better grounding. A mistaken interpretation may require clearer evaluation criteria. A failure to catch multiple errors may call for claim-level review rather than answer-level scoring. A high-impact miss may justify a human approval step even if overall correction accuracy improves.

How to make the 18-of-23 result more useful

To turn a small result into a decision-quality evaluation, document the setup and expand the test thoughtfully.

1. **Write down the protocol.** Record the prompts, model versions, settings, tools, and definition of a correct correction.
2. **Keep the denominator visible.** Report both the percentage and the raw count—"18 of 23," not just "78%."
3. **Classify the failures.** Separate factual errors, unsupported claims, partial corrections, and judgment disagreements.
4. **Test representative cases.** Include the topics, formats, and levels of risk the workflow will actually face.
5. **Use independent evidence.** Where practical, judge claims against reliable sources rather than treating another model's output as ground truth.
6. **Measure more than accuracy.** Track false corrections, missed errors, escalation rate, latency, and the quality of supporting evidence.
7. **Re-test over time.** Model behavior, source availability, and user questions change. Evaluate new versions before relying on them in production.

For high-stakes use, set thresholds by risk, not by a single average score. A workflow that helps draft internal summaries may tolerate a different error profile from one that supports compliance decisions or customer-facing advice. Human review should be part of the design where the consequence of a wrong correction is significant.

So, what does the number mean?

"18 of 23 fact-checks where the correcting model was right" means that the model was judged to make a correct correction in 18 of 23 evaluated cases—an observed correction accuracy of approximately 78.3%. It is encouraging evidence that the model can catch or correct factual problems in that test set. It is not enough evidence to conclude that the model is generally reliable, that model agreement proves truth, or that human review is unnecessary.

The more useful question is what the evaluation reveals about the whole system. Were corrections supported by sources? Did the workflow preserve disagreement instead of hiding it? Did it catch hallucinations without creating new ones? And were the five failures safe, understandable, and fixable?

When teams treat models as participants in a verification process—not as automatic fact-check judges—cross-model verification can improve scrutiny and focus human attention. The score is a starting point. The workflow, evidence, and failure analysis determine what it is worth.