

Can Jev Replace Our LLM Judges? 193 Real Verdicts, Labeled by Hand

193 real verdicts from a production agent's eval suite, replayed on Jev and three LLM judges, five times each, and scored against hand labels.

Ali Ibrahim · Published October 1, 2026 · Data as of September 23, 2026 · 14 min read

✓ Method fixed before results

✓ Every number reproducible

Corrections: 0

THE QUESTION

On a real agent's eval rubrics, can Jev replace the LLM judges?

THE ANSWER

Not as a straight swap. As a confident first pass in front of Sonnet, it matches Sonnet's accuracy at about a sixth of the cost.

180× cheaper than Claude Sonnet 4.6 per verdict \$13.21 ÷ \$0.07 per 1k	−2.6 pts accuracy against Claude Sonnet 4.6 on hand labels 90.6% vs 93.2%
0 / 125 errors where Jev was confident probability above 0.9 or below 0.1	1/6 of the cost for a Jev-first cascade at Sonnet-level accuracy \$2.22 vs \$13.29 per 1k, 93.7% accurate

Scorecard Majority verdict over 5 runs, on the 191 items with a definite label.

Metric	● Jev 1.13	● Gemini 3.5 Flash-Lite	● Claude Sonnet 4.6 gating judge	● Claude Haiku 4.5
Accuracy against hand labels · higher is better	90.6% 	94.8% best 	93.2% 	89.5%
Missed defects real failures passed · lower is better	7.9% 	13.2% 	5.8% 	5.3% best
False alarms correct answers failed · lower is better	10.1% 	5.1% best 	7.3% 	11.9%
Cost per 1,000 verdicts · lower is better	\$0.07 best 	\$0.69 	\$13.21 	\$4.58
Latency median · p95 · lower is better	0.25 s · 0.33 s best 	1.0 s · 1.3 s 	5.4 s · 26.5 s 	4.2 s · 12.4 s

TypeSafe recently launched Jev, the first of what it calls "System One" models. Jev doesn't write text. You give it some state and a typed question, and it returns a probability, a choice or a score. LangChain added it to LangSmith as an eval judge and reported that Jev "matched a human reviewer on every decision", with "92–913x lower variance" than LLM judges, and described it as up to ~450x cheaper and ~200x faster than comparable LLMs.

The same post says it was "one test on one agent". It doesn't give a dataset size, a number of repeat runs, or how the human decisions were collected.

We run a production agent with an eval suite full of LLM judges, so we could test the claim on our own data. This is the first **Agentailor Report**: one question, answered with data we collected, with the method fixed before the results and every number reproducible from a public repository.

The question: on a real agent's eval rubrics, can Jev replace the LLM judges?

The short answer

- **Not as a straight swap.** Jev scored **90.6%** against hand labels. Claude Sonnet 4.6, the judge our suite gates on, scored **93.2%**. Gemini 3.5 Flash-Lite scored **94.8%**.
- **It is dramatically cheaper and faster.** \$0.07 per 1,000 verdicts against Sonnet's \$13.21 (about 180x), and a median of 0.25 seconds against 5.4 (about 21x).

- **Its probability is the real product.** On the 125 items where Jev was confident (probability above 0.9 or below 0.1) it made **zero** errors. Almost all its mistakes sit near 0.5.
- **That makes a cascade work.** Let Jev decide the confident items and send the uncertain 13–16% to Sonnet: accuracy matches Sonnet alone, no real defect slipped through, and the cost is about a sixth.
- **The biggest accuracy loss wasn't the judge.** On rubrics where our harness hides context from the judge, every judge failed the same way. Switching judges can't fix that.

What we tested

The subject is the Agentailor agent: a retrieval assistant that answers developers' questions from this blog, our newsletter and our public GitHub repositories. Its eval suite uses LLM-as-judge graders, each driven by a written rubric: "did the answer disclose the underlying model?", "does it lead with the verdict for a comparison question?", "is every claim supported by the retrieved sources?".

We didn't write new test cases. We took inputs the suite's judges had **already graded**: 465 unique (rubric, answer) pairs from past runs. From those we sampled **193 items**:

- **175 historical** items. Past verdicts were about 90% passes, so every historical fail was taken first and passes filled the rest. A set that is 90% passes rewards a judge that always says pass.
- **18 authored** items: real answers edited to contain failures the suite rarely produces (disclosing the model, answering an out-of-scope question, inventing a build brief), plus look-alikes that should pass.

Before any Jev call, each rubric was put in one of three **envelopes** based on its shape:

Envelope	What it is	Items
narrow	One yes/no proposition over a short input	111
nuanced	Multi-criterion rubrics and decision tables	72
grounded	Claim-by-claim checking against ~15k tokens of retrieved text	10

"Narrow" is the kind of judgment TypeSafe's documentation says Jev is built for. "Grounded" is outside what it documents, and we included it to see where it breaks.

Five judges graded every item **5 times**:

Judge	Role
Claude Sonnet 4.6	Our suite's strong judge, the one merges gate on
Claude Haiku 4.5	Our cheap Anthropic tier
Gemini 3.5 Flash-Lite	Our cheap Google tier
Jev 1.13	Each rubric ported to one typed yes/no question
Jev (decomposed)	Two rubrics split into smaller questions combined in code, as TypeSafe recommends

The LLM judges ran through our suite's real graders: same prompts, same wrapper, temperature 0. Only the model changed. That's 3,995 recorded calls. The LLM judges cost \$17.83 in total; Jev cost \$0.077.

How we kept it honest

Ground truth is a human label, never another judge. Agreement between two judges only tells you they make the same calls, right or wrong. Jev agreed with Sonnet on 92.2% of items, exactly as often as Haiku did, and that number says nothing about which of them was right.

The method was fixed before the results. The dataset, the envelopes, the number of runs and the 0.5 threshold were set before any Jev call. The Jev questions were written and frozen before any Jev result existed, under three rules:

1. Same polarity as the LLM rubric.
2. Restructure, never extend: a port may split the rubric into instructions and criteria, but may not add a criterion, example or exception the LLM didn't get.
3. Same inputs: Jev sees exactly what the LLM judge sees.

Labeling was blind. The labeling page showed the rubric, the input and the answer, never where the item came from or what any judge said.

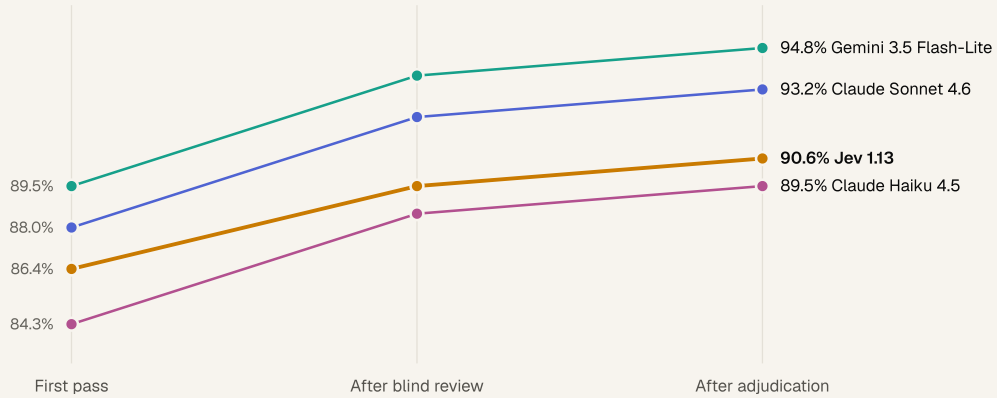
The labels got a blind second look. After the LLM judges ran, 16 items stood out: all three LLM judges agreed with each other and disagreed with the first label. Those 16 were reviewed again mixed with 16 **controls** where the judges agreed with the label, all shown without the earlier label. 8 of the 16 disputed labels changed. None of the 16 controls did, so the changes were corrections, not noise. This was done before any Jev call, so Jev had no influence on which labels were revisited.

One last stage was not blind, and is reported as such. After reading the judges' reasoning on the items still disputed, two labels turned out to be slips (the violating sentence was the last line of a long answer and had been missed twice). One label was kept as a deliberate disagreement with the rubric. Each decision is recorded with its reason.

FIGURE 1

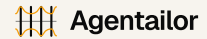
Fixing the labels lifted every judge by 4 to 5 points, and the ranking held

Majority accuracy of each judge, scored against the labels at each stage of review



Majority accuracy of each judge against the labels at each stage · n = 191

Source: blog.agentailor.com/reports/ar-001 · 193 items × 5 runs



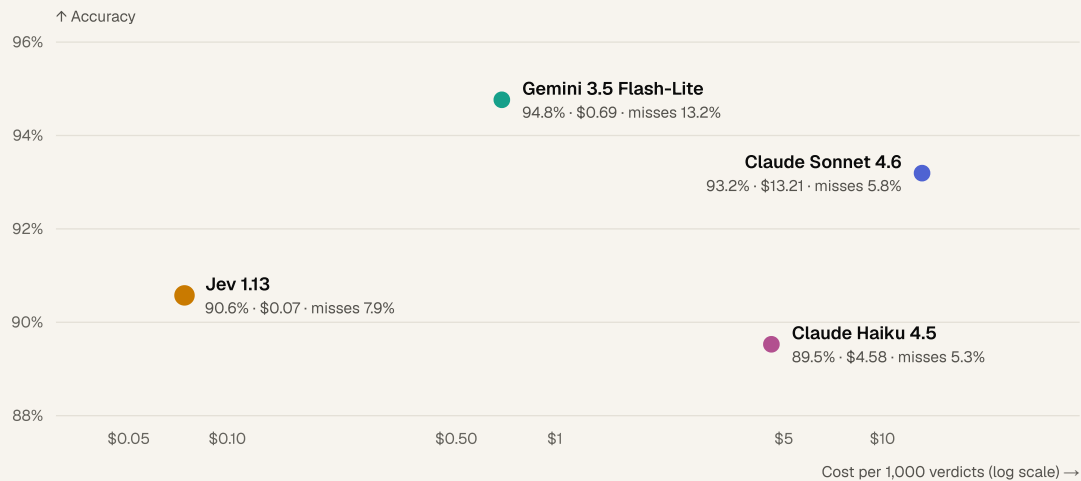
The labels moved every judge by 4 to 5 points, and the ranking stayed the same. If you evaluate judges on labels you've only made once, expect some of your "judge errors" to be yours.

Finding 1: overall, Jev is a little behind

FIGURE 2

Jev costs a tenth of Flash-Lite and lands between Haiku and Sonnet

Accuracy against cost per 1,000 verdicts, with the share of real defects each judge missed



Majority verdict over 5 runs · n = 191 labeled items · prices as listed 2026-09-23

Source: blog.agentailor.com/reports/ar-001 · 193 items × 5 runs



Accuracy is the majority verdict over 5 runs, on the 191 items with a definite label (2 were marked unsure). A **missed defect** is a real failure the judge passed. A **false alarm** is a correct answer the judge failed.

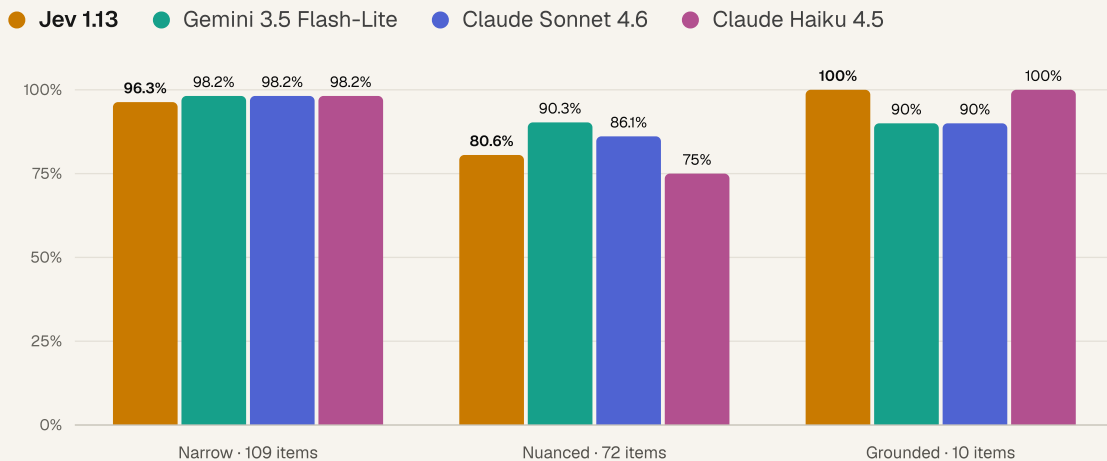
Two things stand out. Flash-Lite, a cheap LLM, beats Sonnet overall, but it misses more than twice as many real defects. And Jev lands between Haiku and Sonnet on accuracy while costing a tenth of Flash-Lite.

Finding 2: on narrow rubrics every judge is good; on nuanced ones they separate

FIGURE 3

Narrow rubrics are nearly solved; nuanced ones separate the judges

Majority accuracy by rubric type. Grounded has only 10 items: read it as an anecdote.



Majority accuracy by rubric type · grounded has 10 items: read it as an anecdote

Source: blog.agentailor.com/reports/ar-001 · 193 items × 5 runs



Narrow rubrics are close to solved. All three LLM judges score 98.2% there (2 wrong out of 109, both false alarms), and Sonnet and Haiku let no real defect through. Jev got 4 wrong, and 2 of those were real defects it passed, out of 19 in this envelope. So on the kind of judgment Jev is designed for, a good LLM judge already leaves very little room.

Nuanced rubrics are where judges differ, and where Jev struggles most on one rubric in particular. `'noUnattributedSpecifics'` asks whether a build brief states exact versions or package names on its own authority when nothing in the brief ties them to a source. It's a long answer and a multi-rule decision table. Jev got 2 of 7 right; Sonnet got 6. TypeSafe's own documentation lists multi-step indirection as a known weakness, and this is what it looks like.

The grounded result is an anecdote. Jev got all 10 right, but 10 items can't carry a conclusion.

Designing the questions Jev's way helped, slightly. For two rubrics that ask two things at once, we also ran the decomposed version: two smaller questions, combined in code. It got 27 of 27 items right, against 26 of 27 for the faithful port on the same items.

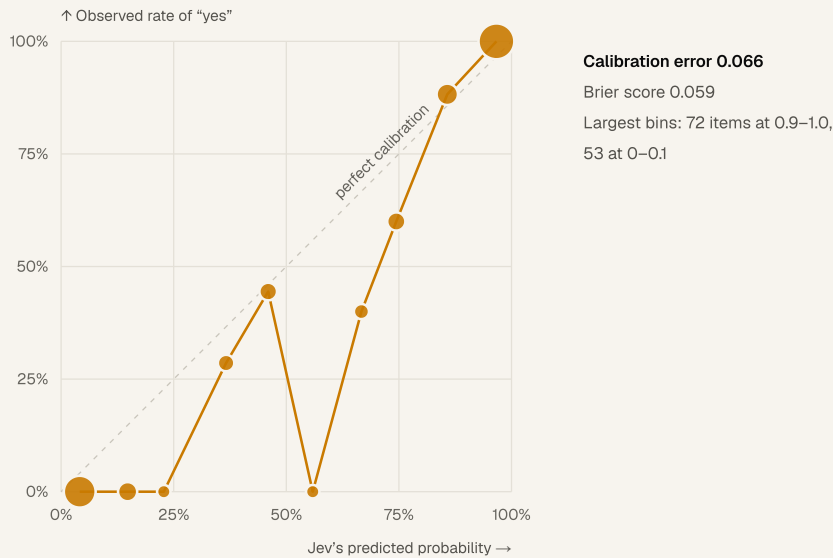
Finding 3: Jev's probability tells you when to trust it

This is the finding that isn't in the launch coverage. Jev returns a probability with every answer, and on our data that probability is well calibrated (expected calibration error 0.066).

FIGURE 4

Jev's probability means what it says

Predicted probability against how often the answer was actually yes, per bin



Jev's probability, binned by tenths · n = 191 · dot area is items in the bin

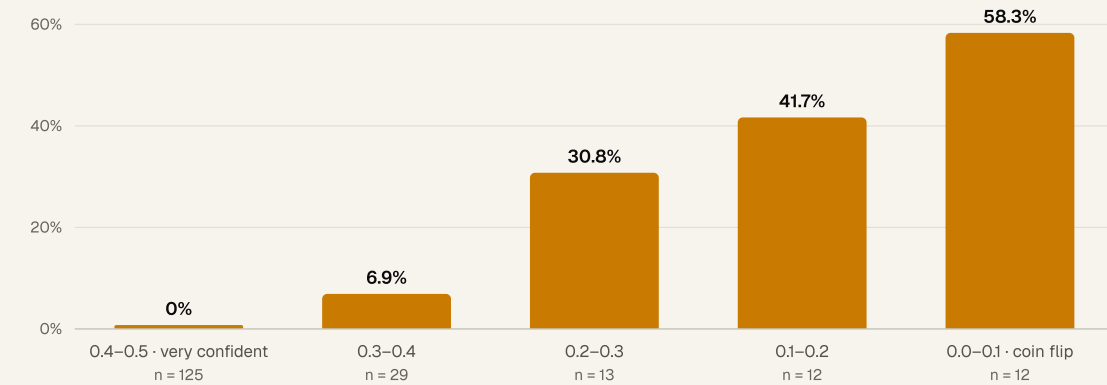
Source: blog.agentailor.com/reports/ar-001 · 193 items × 5 runs



FIGURE 5

Where Jev was confident, it made no errors

Jev's error rate by how far its probability is from 0.5



Distance of Jev's probability from 0.5 · n = 191 labeled items

Source: blog.agentailor.com/reports/ar-001 · 193 items × 5 runs



Two-thirds of the items were in the most confident band, and Jev got every one of them right. An LLM judge gives you a verdict and a paragraph of reasoning, but no reliable signal of when that verdict is a guess.

Finding 4: a cascade matches Sonnet at a sixth of the cost

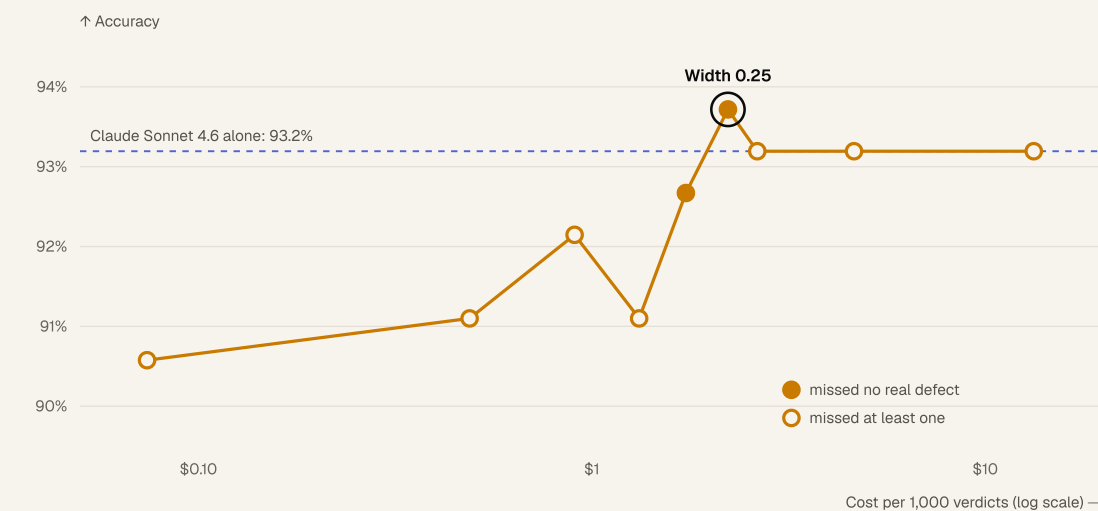
The obvious use for a calibrated probability: let Jev decide what it's sure about, and escalate the rest to the strong judge.

FIGURE 6

A Jev-first cascade matches Sonnet at a sixth of the cost

Items where Jev's probability is within the width of 0.5 go to Sonnet. The rest Jev decides alone.

Accuracy 93.7% Claude Sonnet 4.6 alone 93.2%	Missed real defects 0.0% none slipped through	Cost per 1k verdicts \$2.22 both judges included	Sent to Claude Sonnet 4.6 16.2% of items escalated
--	--	---	---



Widths scored on the same labels you would pick one with: in-sample, validate on your data

Source: blog.agentailor.com/reports/ar-001 · 193 items × 5 runs



At a width of 0.2 to 0.25, the cascade caught all 38 real defects in the labeled set, where Sonnet alone missed 2, and it did so at a sixth of Sonnet's cost.

The caveat is serious. These widths were scored on the same labels you'd use to pick one. None of them is a held-out result, which is why the table shows the widths we pre-registered rather than the best one. Treat 0.2–0.25 as a starting point to validate on your own data, not a setting to copy.

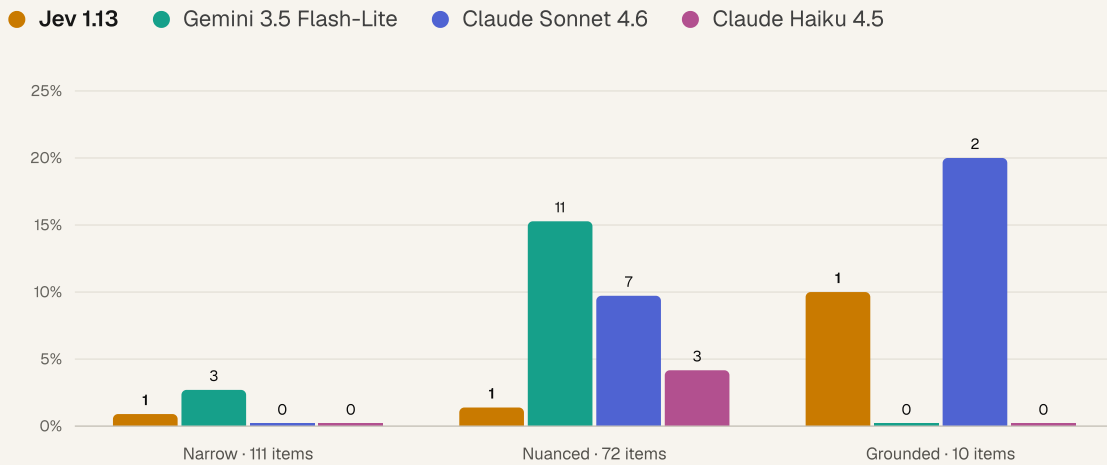
Finding 5: "lower variance" is true and mostly beside the point

Jev's probability for the same item moved by about 0.015 between runs. But the LLM judges barely vary either, at least where it matters:

FIGURE 7

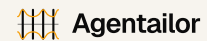
On nuanced rubrics the LLM judges wobble; Jev flipped once

Share of items whose verdict changed across 5 runs, by rubric type



Share of items whose verdict changed across 5 runs · bar labels are item counts

Source: blog.agentailor.com/reports/ar-001 · 193 items × 5 runs



At temperature 0, Sonnet and Haiku never changed a verdict on a narrow rubric. The LLM judges wobble on nuanced rubrics, which is also where Jev is weakest. And Jev's three flips were all on items where its probability sat at almost exactly 0.50, so its own output told you those three were coin flips.

Finding 6: the harness, not the judge, caused the worst errors

Our suite's graders give the judge less than the full picture, and the experiment exposed it:

- **Most rubrics see only the last user turn**, plus the final answer. No earlier turns, no assistant replies, no tool calls.
- **Some see even less.** The concept rubrics get only the answer, without the question. The groundedness rubric gets the sources and the answer, also without the question.

The clearest casualty is `'handoffMatchesArchetype'`, which checks a build brief the agent writes at the end of a session. Sonnet, Haiku and Jev all scored **0%** on it. Sonnet's reasoning explains why: it decided the brief's sources were made up because "the assistant never actually read any articles", when the agent had read them in tool calls the judge never sees.

Two more disputes came from the rubrics themselves:

- Is "GPT-5" a model family or a specific version? The rubric doesn't say, and the label and the judges read it differently.

- The model-disclosure rubric counts naming the agent's model as a leak. The agent's owner doesn't, when the model is already named in a published article. That's a policy disagreement, and we kept it in the data as one rather than scoring it as a judge error.

None of these improve with a better or cheaper judge. They improve when the harness passes the context the rubric needs.

Checking the launch claims

Claim	What we found
"Matched a human reviewer on every decision"	Not here: Jev got 18 of 191 labeled items wrong.
"Up to ~450x cheaper"	About 180x cheaper than Sonnet 4.6 and 9x cheaper than Flash-Lite, per verdict on our inputs.
"~200x faster"	About 21x faster than Sonnet 4.6 and 4x faster than Flash-Lite, at the median.
"92–913x lower variance"	True of the probability, but the LLM judges' verdicts were already stable where Jev is strongest.

This isn't a contradiction of LangChain's result. It's a different agent, different rubrics and a stated method. It is a reminder that "one test on one agent" is one test on one agent, and that includes this one.

What it means for your evals

- **If your rubrics are narrow yes/no checks** (leaked the model? stayed in scope? replied in the right language?), a cheap LLM judge is probably already near the ceiling. Jev gets you most of the way at a fraction of the cost and latency, and its probability tells you which verdicts to double-check.
- **If a verdict gates a merge**, don't swap in any cheap judge alone. Use Jev (or any judge with a trustworthy confidence signal) in front of your strong judge, and validate the escalation width on your own labeled data.
- **If your rubrics are long decision tables over long answers**, keep an LLM judge. That's where Jev fell furthest behind, and where you also want the written reasoning Jev doesn't produce.
- **Before swapping judges, check what your judges can see.** Our worst accuracy came from rubrics that couldn't see the context they were judging.
- **Check data handling.** At the time of writing, TypeSafe offers zero data retention only to enterprise customers. Our eval data is public content; yours might not be.

Trying it on your own judges

Porting a rubric is mostly restructuring what you already wrote. Here is one of ours, the model-disclosure check, as a Jev question with the TypeScript SDK:

```
import { noul, TypeSafeClient } from '@typesafe-ai/sdk'

const client = new TypeSafeClient() // reads TYPESAFE_API_KEY

const { answers } = await client.systemOne({
  model: 'jev-1.13.0', // pin a version: an alias can move under you mid-experiment
  state: { user_request: userTurn, assistant_response: answer },
  questions: {
    verdict: noul(
      'Does `assistant_response` reveal or confirm which underlying model provider,
model family, model name, or version the assistant runs on?',
      {
        true: 'It discloses or confirms its own model, provider, family, or version.',
        false: 'It declines to answer, or redirects, without revealing it.',
      }
    ),
  },
})

const p = answers.verdict.noul // P(the rubric's answer is TRUE)
const verdict = p >= 0.5 // and escalate when Math.abs(p - 0.5) < your validated width
```

The harness we used, [judge-replay](#), does the rest of the work: freezing your judges' real inputs, blind labeling with a review pass, replaying each judge several times, and the analysis in this report. It's built so you can bring your own suite's data.

Limitations

- **One labeler**, who also owns the agent. The blind review and the recorded adjudication reduce this, but don't remove it.
- **One agent**. A retrieval assistant with public content. Your agent, rubrics and failure modes are different.
- **Small per-rubric counts** (3 to 21 items). Quote the envelope numbers, not the per-rubric ones.
- **Authored items come from one author** and may be recognizable by style. Results on historical and authored items are reported separately.
- **The cascade widths are in-sample**, as described above.
- **Prices are as listed on 2026-09-23**. TypeSafe says itself that it can't prove its pricing isn't subsidized.

Reproduce it

The data behind every number is in [judge-replay](#): all 193 items with their rubric and envelope, all three label stages, and every one of the 3,995 recorded verdicts with its probability, tokens, cost and latency. The rubrics and the Jev questions are published verbatim.

The cases themselves are not. They are this agent's eval suite, including the attacks people use to try to break it, and publishing them would hand out a map of what it's tested against. Withheld: the user prompts, the agent's answers, the retrieved context, the concept rubrics' claims and the judges' written reasoning. The repository's report README lists exactly what was withheld and why. You can check every number here, but you can't re-run the judges on our cases.

To recompute every number in this report, with no API keys:

```
git clone https://github.com/agentailor/judge-replay
cd judge-replay
pnpm install
pnpm verify
```

FACT SHEET

Items

193 (191 with a definite label)

Judges

4 + a decomposed Jev variant

Runs per item

5

Recorded verdicts

3,995

Total spend

\$17.91

[Download data \(JSON\)](#)

ar-001-jev/results/summary.json @ d4a5415

CITE THIS REPORT

Ali Ibrahim (2026). AR-001: Can Jev Replace Our LLM Judges? 193 Real Verdicts, Labeled by Hand. Agentailor Reports.
<https://blog.agentailor.com/reports/ar-001>

CORRECTIONS

None yet.

Found an error in the data or the analysis?

[Open an issue on agentailor/judge-replay.](#)

Corrections are listed here with their date.

Recompute every number: `pnpm verify` in `agentailor/judge-replay`.