the good part is not
the headline number.

Grounding Agent Memory

Environment-Probing Curation for Enterprise Agents

SURESH · MAK · BHATNAGAR · METHANI · GUTIERREZ MUNOZ

ARXIV:2609.11060

READING NOTES · DEEP DIVE

Your agent's memory is guessing.

Luis Villalobos

Director of AI Labs, HatchWorks AI

WHAT THIS IS

Five authors, one change, measured carefully

On 10 September 2026, S. Suresh, H. Mak, S. Bhatnagar, C. Methani and A. Gutierrez Munoz posted a paper to arXiv testing one modification to the way agents remember things. They ran it across 90 tasks in six enterprise environments, with a primary model and two cross-model checks, and they report cost alongside accuracy, which most papers in this area do not.

This issue explains what they changed, what it bought, and the two decisions it should change for anyone running long-horizon agents in production.

(the change is small. the measurement is the valuable part.

Start here: what agent memory actually is

An agent working inside a real system, a warehouse, a CRM, a data platform, begins each session knowing nothing about it. It has to find where the data lives, work out which table means what, and discover which sequence of calls answers the question. Then the session ends and all of that is gone. Next session it does the whole thing again, and you pay for the whole thing again.

Memory is simply the agent writing down what it learned so the next session can read the notes instead of rediscovering. That alone is the largest effect in the paper: pass rate moves from 39% to 70%, and in one environment cost per run falls from $54.30 to between $7 and $9.

✳ 31 points. from writing things down.

The defect that compounds

What gets written down is whatever happened to work. Not what is true, not what is general, and not what is still true. A successful run contains a great deal that is incidental to it: the particular answer to a particular question, a path that worked but was not the shortest, a precondition that happened to hold that day.

Stored as reusable knowledge, all of it becomes a memory store that is confidently wrong in ways nobody notices until the schema moves underneath it.

The big idea

> Make the memory check itself before it saves. Give the component that writes the notes read-only access to the real system, and require it to verify each claim against live data before committing.

✳
everything else in this paper is plumbing around that sentence.

Everything below is the detail of how that is done, what it costs, and what it returns. The finding to carry into a planning conversation is narrower than the abstract suggests: giving an agent memory is what buys the accuracy; making that memory verify itself is what buys the margin, and the margin shows up in cost per task rather than in pass rate.

The mechanism

The method is called environment-probing curation. It extends standard curation with a **propose, probe, commit** workflow.

After a task completes, an asynchronous curator agent receives the trajectory, as it would normally. The change is that the curator is also granted read-only access to the same world tools the task agent used. Before any candidate memory is written, the curator goes and checks it.

✳
propose → probe → commit.
most teams build the first and the third.

What the curator probes for

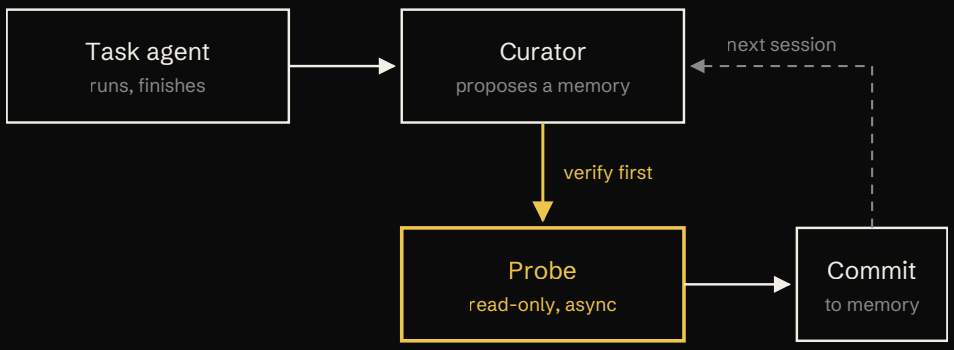
- Whether a result is an incidental answer or a genuinely reusable relation
- Whether a shorter path exists than the procedure observed
- Whether a claimed relation holds on another slice of the data
- What preconditions the procedure actually requires
- Relevant state the task trajectory omitted entirely
- Whether the environment has drifted, triggering a re-query

Why the constraints matter more than the idea

Probes are read-only and asynchronous. They cannot mutate the environment, cannot consume the task agent's budget, and cannot expose future tasks. The verification work sits entirely outside the request path, so it costs wall-clock time the user never experiences and budget that is separately accountable.

(this constraint is what makes it deployable at all.

FIG. 1 THE LOOP MOST MEMORY SYSTEMS SKIP



Standard curation runs the top row only. The yellow box is the entire contribution of the paper.

The evidence

CLBENCH · 40 QUESTIONS, WITH SCHEMA DRIFT · GPT-5.4

Metric	No memory	Trajectory-only	Probed
Pass rate	39%	70%	73%
Total reward	8.60	20.00	22.60
Queries per question	8.8	5.6	4.7
Task-agent cost	$3.38	$1.99	$1.68

column three against column two. not column one.

Read the pass-rate row honestly. The move from 39% to 70% is the memory layer doing its job. Probing adds three points on top. Anyone selling probing on accuracy is quoting the wrong column.

The cost and query rows are where the method earns its place: a 16% reduction in cost per task and a 16% reduction in queries per question, against a baseline that already had memory.

A useful third baseline

The paper also tests prepending prior trajectories directly into context, the approach most teams reach for first. It reached 61% pass rate while consuming 5.42M input tokens. Curated memory beat it on both axes.

Cross-model check, 30 questions, no drift

- Claude Sonnet 4.6 with probing: 0.748 mean reward, a lift of +0.421 over its paired baseline
- Claude Opus 4.7 with probing: 0.721 mean reward, a lift of +0.263

The direction of the effect survives a model change. The magnitude is not established across a wide model set.

APEX, 90 tasks across six enterprise worlds

World	Trajectory-only	Probed
941eba66	0.858	0.991
075ef4df	0.140	0.274

Reward per dollar. Probing produced the best task-agent reward gain per dollar in five of six worlds.

(18 of 18 comparisons positive. that is the unusual part.

> In world 941eba66, adding a memory layer moved tool calls from 71.6 per run to between 17.7 and 19.3, input tokens from 53.92M to between 6.56M and 7.67M, and cost per run from $54.30 to between $7 and $9. Roughly a sixfold cost reduction from an architectural change, not a pricing negotiation.

* take this one to the budget conversation.

What to do with this

Decision one: diagnose before you shop

If an agent product is running over budget, the reflex is to move to a cheaper model or a smaller context. The 39-to-70 result suggests checking something else first: whether the agent has any cross-session memory at all. An agent that re-derives the same organisational facts every session is paying a context cost repeatedly. Model substitution reduces the unit price of a problem you should be eliminating.

(a model swap lowers the unit price of work you should be deleting.

Decision two: give the curator a real seat

Where memory already exists, the paper's structure suggests four concrete choices.

- **Make the curator its own component**, not a summarisation prompt appended to the task agent. It has a different job and a different failure mode.
- **Keep probes read-only and asynchronous.** This is a hard constraint, not a preference. It is what keeps verification out of the latency path and prevents the curator from mutating state.
- **Budget it separately.** Curation cost that is invisible inside task cost cannot be tuned, and cannot be defended when someone asks what it is for.
- **Instrument reward per dollar** alongside pass rate. Probing barely moves pass rate. A team measuring only accuracy will correctly conclude the feature does nothing.

Where this does not apply

- No safe read surface. Where an environment exposes no read-only tool access, the curator reverts to trajectory-only curation. The gains revert with it.
- Short-horizon agents. The entire mechanism assumes tasks recur against a persistent environment. A single-turn agent has nothing to curate.
- Static environments. The headline schedule included schema drift by design. In an environment that genuinely never changes, the drift-detection portion of the value disappears.

✳ *check this list before you fund anything.*

Honest limitations

This is one team's benchmark suite. Main results run on GPT-5.4, with cross-model validation on two Claude models on a smaller no-drift schedule. The paper acknowledges that trajectory-only

memory can preserve errors, overgeneralise partial evidence and retain stale knowledge, and that probing reduces these problems without eliminating them. No comprehensive failure analysis of probing itself is provided, and the cost of the curator's own probe calls is reported separately from task-agent cost rather than as a single blended figure. Treat the reward-per-dollar figures as task-agent economics, and budget curation on top.

SOURCE

Suresh, S., Mak, H., Bhatnagar, S., Methani, C., Gutierrez Munoz, A. “Grounding Agent Memory: Environment-Probing Curation for Enterprise Agents.” arXiv:2609.11060, submitted 10 September 2026.
Abstract: arxiv.org/abs/2609.11060
Full text: arxiv.org/html/2609.11060v1

Every figure in this issue is drawn from that paper and was read from its full text, not the abstract. The 39% to 73% framing the abstract foregrounds is a no-memory to probed-memory comparison; this issue reports the trajectory-only column alongside it, because that is the honest baseline for judging probing specifically.

Luis Villalobos

Director of AI Labs, HatchWorks AI