No textual match to a pushed memory brief in agent replies

Luiz Antonio Busnello

Version 1.0 · doi:10.5281/zenodo.24017459

Version 1.0 (2026-11-06), doi:10.5281/zenodo.24017459. Not peer reviewed. A data note, dated: every measurement below covers 2026-09-21 to 2026-10-23 on one deployment. Every result with a decimal, a percentage or a thousands separator has an entry in `PROVENANCE.json`, in the code-and-data archive of this record.

Abstract

A memory system for a group of software agents pushed a short brief of ten memory items into each agent's system prompt on every turn. We asked whether the agents' replies show any trace of what the brief contained. Two textual measurements found none, under detectors whose reach we state; a third, behavioural measurement was inconclusive because it failed its own positive control. Across 1,524 assistant messages from six agents, none of 12,782 exposed (message, brief item) pairs carried a strict textual match of the item (the fuzzy detector, reported for the first run, found none in 9,398 pairs), while a control built from items that entered the brief later matched 57 times. A second measurement, on 574 messages, found zero traces both in the served brief and in a recency-ordered brief that was logged but never delivered, and an injection control showed that the strict detector fires on 95.7% to 100.0% of injected item summaries. A behavioural probe that planted a salient fact in the brief and watched the agent's first action failed its positive control (0 of 10 decisive scenarios against a gate of 7), so its main phase was not run and it supports no conclusion about the first action. We state the result narrowly: on the pairs we could evaluate, under these detectors, we found no textual match of the pushed brief in the replies. This is not evidence that the agents do not use memory. On the pull side, no search in the memory server's log was attributed to an agent, and one agent given a task that required the corpus did search it.

1. Setting

The memory system (nox-mem) serves a brief: ten items chosen by salience, refreshed by a scheduled job, written to a file per agent, and appended to the agent's system prompt on every turn. The agents are long-running assistants that act on schedules and on messages. The brief is the only channel by which memory reaches them without the agent asking; the other channel is a search tool the agent may call.

The question this note answers is narrow: when the brief is in the prompt, do the agent's replies show it? Which brief was in the prompt is reconstructed from the brief log: for each message, the latest brief written for that agent before the message. The note does not check, message by message, that the rendered prompt contained it.

2. Measurement 1: textual trace in replies

Source: `audits/2026-10-14-brief-usage.md`, script `audits/2026-10-14-brief-usage.py`.

Unit and arms. One assistant message, with the items of the brief that was in the prompt when the message was written (the exposed arm). Two controls use the same agent's briefs at other moments of the window: items that had left the brief before the message (past), and items that entered it later (future). The future control is the check that the detector can see overlap at all, because a conversation can push its own material into a later brief. Items whose text is no longer in the database are excluded, because there is nothing to compare, and so are messages with no earlier brief for their agent. The first run reports 623 of 974 distinct brief items as still resolvable; the rerun does not report its coverage, so the share of delivered items that the exposed arm covers is not quantified.

Detectors. Strict: the item's one-line summary (at least 25 characters) verbatim, or a distinctive file stem on word boundaries. Fuzzy: at least 60% of the item's distinctive tokens. A first version matched stems as substrings and counted dictionary words, and produced eight strict "hits", all false on inspection; it is kept in the audit as a lesson and not used.

Result. Over 2026-09-21 to 2026-10-15, reading all three places where transcripts are kept: 1,524 messages and 12,782 exposed pairs, with zero strict hits; the rerun's table reports only the strict count for the exposed arm, and in the first run the exposed arm had zero strict and zero fuzzy hits in 9,398 pairs. The future control produced 57 fuzzy hits. In the first run, the hits of that control were read by hand: they were summaries derived from the same conversations, which entered the brief afterwards. The past control, reported in the audit for the first run, had no hits in 74,347 pairs. The future control shows that the detectors are not blind to overlap of that kind. It does not measure how often they would miss an agent reusing a brief item in other words, so the exposed zero is a count of matches under these detectors, not a measured absence of reuse.

The base was understated at first. The first run read one of three transcript roots and reported 1,120 messages. Transcripts are kept in two live directories and an archive, and one of the live directories is pruned on a seven-day clock, so a rerun of the same script later reaches fewer messages than it did. Rerun on one day over the same window, the script found 1,029 messages reading one root and 1,524 reading all three. The verdict did not change: the exposed arm stayed at zero strict hits while the future control rose from 20 to 57 fuzzy hits.

3. Measurement 2: served brief against a recency-ordered brief

Source: `audits/2026-10-21-r1-recency-relevance.md`, positive control `audits/2026-10-21-r1-positive-control.py`.

From 2026-10-14 the brief server also logged, on every refresh and without changing what the agent received, what a recency-ordered brief would have contained. Over 574 messages in the following week, the served arm and the recency arm were measured on the same message with the same detectors. Both had zero strict and zero fuzzy hits, and no message was discordant, so the paired comparison carried no information. The recency arm was never in any agent's prompt; its zero says nothing about how agents would respond to a recency-ordered brief. The served arm's zero is a second count of the kind in §2.

A positive control injected 300 items, drawn from the served and recency arms together, into message text and re-ran the strict detector. It fired on 95.7% of injected one-line summaries and on 100.0% of summaries with their file stem, and 483 of the 485 served items (99.6%) had a summary or stem long enough to be detectable at all. If an agent had copied an item's one-line summary, alone or with its file stem, the detector would almost always have seen it; the control tests neither a stem on its own nor partial quotation or paraphrase.

4. Measurement 3: the first action, behaviourally

Source: `audits/2026-10-15-7b-positive-control.md`, design `specs/2026-10-14-7b-brief-action-probe-design.md`.

Influence without textual trace is possible: an agent can be steered by a brief item and never reuse its wording. A behavioural probe tried to measure this. It froze 20 scheduled scenarios, 10 of them decisive, and ran each with one brief item replaced by a planted fact and without the replacement, in two replicas (80 runs), checking whether the agent's first message used the planted value. A decisive scenario counted as a hit only if both replicas used the planted value without also touching the value it replaced. By the rule fixed before the run, the main phase would run only with at least 7 hits in 10.

There were 0 hits in 10, so the main phase did not run, and the probe supports no conclusion about whether the brief changes the first action. What it recorded is descriptive. The planted value appeared in two decisive scenarios, in both replicas of one and in one replica of the other, and one of those replicas also touched the replaced value, so neither scenario met the rule. In 12 of the 40 planted-arm runs the first message only loaded tools and could not carry any value. Restricted after the run to first messages that were actions, the planted value appeared in 3 of 28 planted-arm runs and in none of 27 runs without the plant, and the replaced value appeared at similar rates in both (13 of 28 and 12 of 27). These counts neither show nor rule out an effect.

5. The pull side

Source: `audits/2026-10-21-r1-recency-relevance.md` (search telemetry) and `audits/2026-10-23-item7-consumidor-construido.md`.

The brief is the push channel. The pull channel is a search tool. In the census reported on 2026-10-21, covering searches since 2026-10-14, 337 of 382 searches recorded by the memory server were its own 30-minute canary probe, and none was attributed to an agent. On 2026-10-23, one agent was given a recurring task whose answer depended on the corpus (a weekly sweep for contract deadlines, where an exhaustive query finds the candidates and the agent checks whether each was later amended). The prompt did not name the search tool. In two production runs that day it searched 11 and 6 times. The searches are attributed to it by time window, because the server recorded their caller as unknown. One agent, one task and two runs show that an agent whose task needs the corpus can reach for the search tool; they do not explain why the agents had not searched before.

6. What this note claims, and what it does not

It claims: in the two textual measurements, on this deployment and in this window, on the pairs that could be evaluated, the strict detector found no textual match of the pushed brief in the agents' replies, nor did the fuzzy detector where its count is reported, and controls showed that the detectors fire on the forms of overlap they were tested on. It claims nothing about the first action, because the behavioural probe failed its positive control.

It does not claim that the agents do not use memory, that the brief has no effect on what they do, that a different brief would also go unused, or that the detectors would catch partial quotation or paraphrase.

7. Two things that transfer

Transcripts are pruned by a job the measurement does not control. The same script, run a week apart, read different data, because a cleanup job deletes old sessions daily and only an archive keeps them. The first count of the first measurement, 1,120 messages, held about three quarters of the 1,524 the archive-aware rerun found. The R1 script now reads all three roots by default and prints the roots and the directory count; the 7a rerun named them through its `PROJ_ROOTS` variable, whose default still lists two.

A null needs a positive control, and the control has to pass. Measurement 2 carries an injection control that passes, and Measurement 1 a future control that shows its detector can fire, so their zeros are counts under detectors shown to work on the forms they were tested on. Measurement 3 carried one that failed, so this note reports it as inconclusive.

8. Data availability

The transcripts and the brief items are private and are not released. The detectors, the positive controls and the audits that report each number are in the public repository; raw outputs that contain agent text stay on the host. Every result with a decimal, a percentage or a thousands separator has an entry in `PROVENANCE.json` naming its audit, a locator string that must occur in it, and the commit that holds it; counts below that and design constants are in the audits named in each section.