

Can a Small Local Model Help an AI Assistant Remember What Matters?

Junior Williams | trustcyber.ca

Here is a fictional example. You ask an AI assistant to help prepare a training guide. Later, you move the review from Friday to Monday and tell it to prepare a draft only. When you return, it remembers the topic but misses those corrections. It can sound helpful while continuing the wrong work.

That is the problem I wanted to investigate: could saved conversation excerpts help a later session recover the goal, decisions and limits of a task without receiving the entire conversation again?

I started with TypeSafe's [basic-memory cookbook](#). Its Jev model is hosted, meaning it is accessed through an online service. When my available API credits ran out, I switched to Kev, a local model running on my computer. The investigation grew larger than the original question required, so I brought it back to a small test of whether an assistant could reconstruct the state of my own AI-memory experiments.

In this pilot, supplying all available test excerpts preserved more required information than either approach that selected excerpts. The selective-memory configuration was not ready to become my default way of carrying work between sessions.

What “memory” means here

The model does not gain permanent memory internally. Software stores information and adds it to a later request. An assistant can then use that information when answering, much as someone starting a task can consult notes left by a colleague.

There were three jobs in my setup:

1. **Keep or exclude.** Kev judged whether an excerpt would help someone continue the work. This is retention. Its keep-or-exclude check is called the memory gate.
2. **Find.** A search method selected excerpts relevant to the questions being asked. This is retrieval.
3. **Answer.** A separate AI assistant, Codex, received those excerpts, answered the questions and wrote a handoff: a short summary for the next session.

I kept every original excerpt with its speaker, project identifier, message reference and order. Excluding one from selective memory did not delete it. The original remained recoverable, although the selective search could no longer choose it. This makes it possible to inspect what was kept, what was excluded and where each item came from.

TypeSafe's diagram below shows its customer-support example. Jev answers a ticket, then judges whether the request contains a customer fact worth saving, such as a product edition or contract. Saved information is added to later tickets for that customer. My pilot adapted this idea to my own AI-testing conversation. Kev made the retention decisions; Codex wrote the answers. The pilot was not an entirely local writing system or a head-to-head comparison between Jev and Kev.

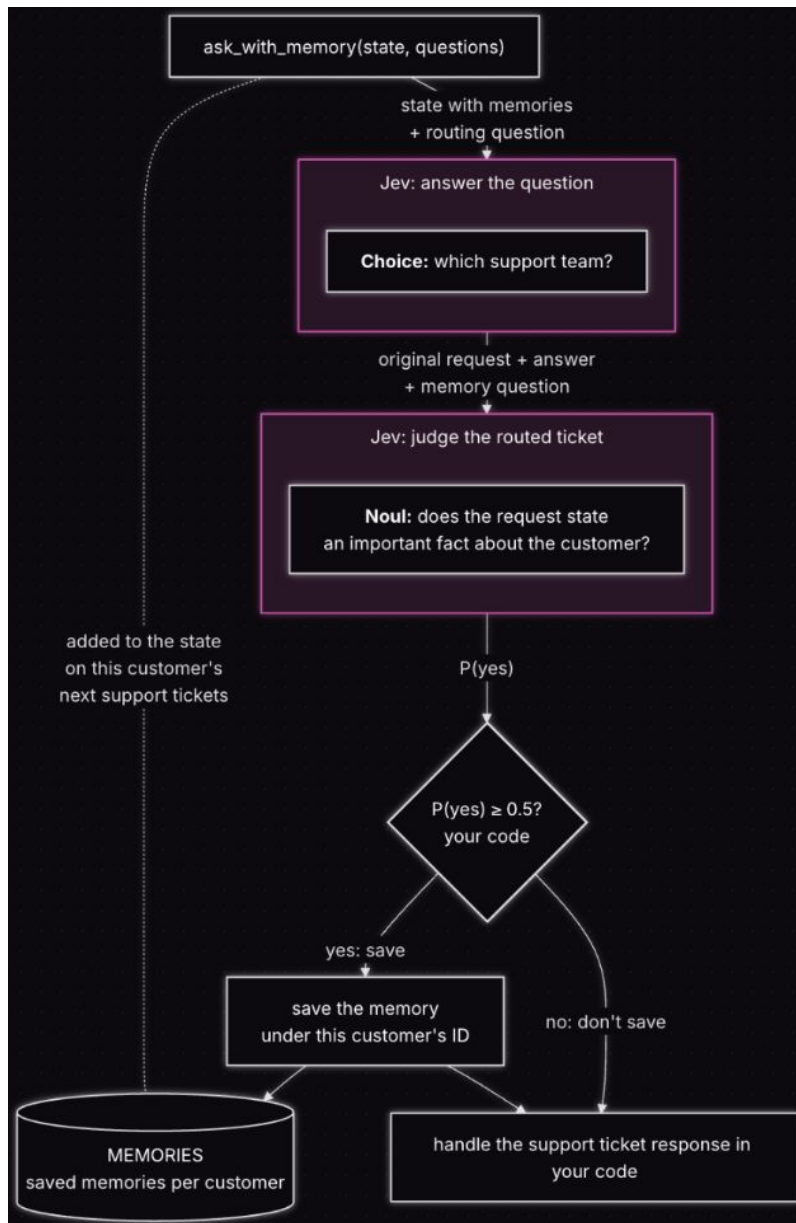


Figure 1. TypeSafe's customer-support example: answer, judge retention, save and reuse. The pilot adapted this workflow to the author's own AI-testing conversation.

Source: TypeSafe, Simple memory system.

https://docs.typesafe.ai/cookbooks/basic_memory

- **All available test excerpts:** provide every eligible excerpt up to that checkpoint. This was the full bounded-history condition, not the entire original conversation.
- **Search only:** use word-based search to select up to eight excerpts.
- **Key filtering, then the same search:** search only the excerpts Kev had judged worth retaining.

The expected answer meanings and supporting sources were written down before the test. An AI-assisted assessment checked the answers against those fixed criteria. Each condition produced 18 individual answers: six questions at three checkpoints. The handoffs were assessed separately.

How I tested it

The pilot used a purpose-built test set of 29 short, verbatim excerpts from my own conversation with an AI assistant about these memory experiments. No client conversations or client records were included in that test set. Eight fictional excerpts from a separate project checked whether information from another project could enter the answers.

I chose three checkpoints: stopping points in the past conversation after the model changed, after the practical goal was clarified, and after the work was restricted to analyzing existing results. At each checkpoint, the assistant could see only excerpts available by then.

It answered six questions: what is the goal, which model should be used, what changed, what finished, what remains unresolved, and what actions are allowed? It then wrote a next-session handoff. I compared three ways of supplying the evidence:

This was a retrospective case study: I selected excerpts from a conversation whose outcome was already known. There was no blinded human evaluation. These related checkpoints do not establish reliability across everyday work. The technical settings and execution safeguards are collected at the end.

What the comparison showed

All available test excerpts produced 16 complete answers out of 18. Search only produced nine; Kev filtering followed by the same search produced ten. A complete answer supplied all the required information correctly. An incomplete answer omitted required information or reported it as unknown. An unsupported assertion made a claim the evidence did not justify.

The selective condition made two unsupported assertions. The other conditions had omissions but no assertions graded incorrect. These counts describe individual answers, not a score for the entire handoff.

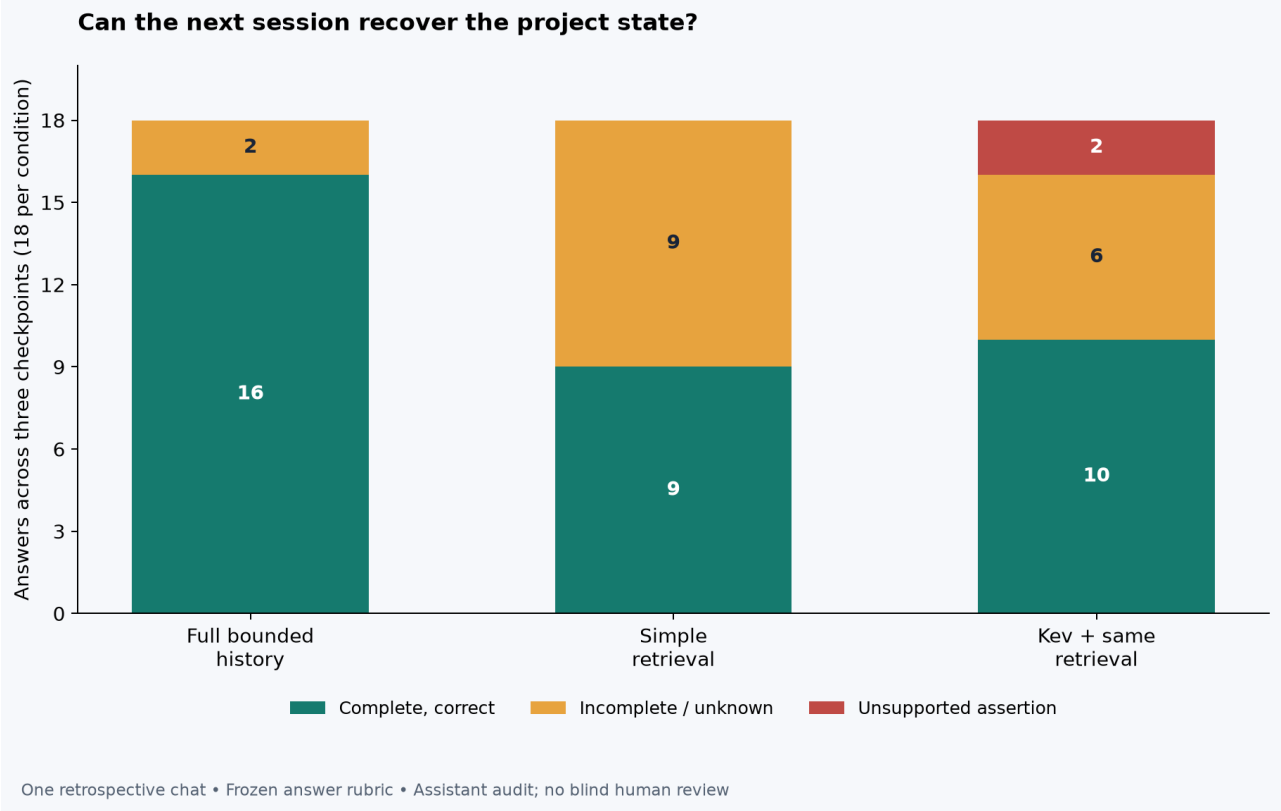


Figure 2. Six questions at three checkpoints give 18 individual answers per condition. Green: complete and correct; amber: incomplete or unknown; red: unsupported assertion. Handoffs were assessed separately.

One failure shows why this matters. After I switched to local Kev, the search-only approach missed the request to change models. The selective approach also missed the short clarification that Kev was local. Its handoff said continued testing depended on replenishing online-service credits, even though the requested experiment used the local model.

The problem occurred at two different steps. Kev had kept the model-switch request, but the search did not select it. Kev had excluded the locality clarification, so selective search could not recover it. Improving the retention decision would not automatically fix the search failure. Losing two short excerpts was enough to send the next session in the wrong direction.

At the final checkpoint, all three handoffs preserved the instruction to analyze saved results without starting new processes. The all-excerpts condition’s two incomplete answers also came from this

checkpoint: they omitted context that appeared elsewhere in its response. Its overall handoff still preserved the immediate goal and boundary.

The fictional-project boundary was enforced by project identifiers in software before search. No fictional-project excerpt entered any of the nine evidence packets or appeared as project state in the outputs. This supports explicit separation in code. It does not show that a model’s usefulness score can enforce access or authorization boundaries by itself.

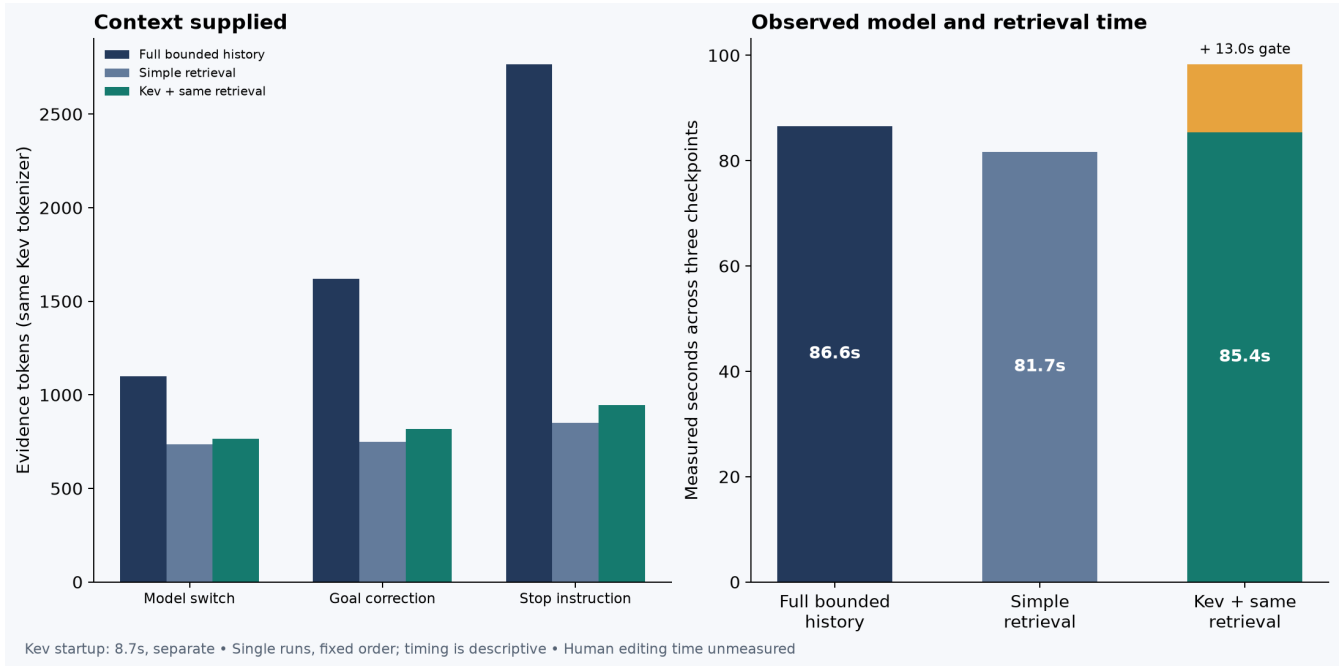


Figure 3. Left: evidence size in Kev-tokenizer units. Right: reader and retrieval time totaled across three checkpoints, with 13.0 seconds of filtering shown separately. Kev startup adds 8.7 seconds. Single runs in fixed order do not establish a speed ranking.

Kev kept 27 of the 29 source excerpts, so its filter removed very little. Its selected packets were larger than search-only packets at every checkpoint: 766 versus 736 tokens, 818 versus 750, and 944 versus 849. Tokens are small pieces of text a model processes; here they measure the amount of supplied evidence.

Classifying all excerpts, including the fictional controls, took 13.0 seconds, with another 8.7 seconds for Kev startup. Each comparison condition used one reader call at each checkpoint, run in a fixed order. The figure totals reader and retrieval time across the three checkpoints and shows classification time separately; startup is additional. These measurements describe this run, not a stable speed ranking. Human correction time was not measured, so I cannot claim the smaller packets saved working time.

What the earlier tests add

The earlier, larger Kev experiment asked factual questions about synthetic histories: invented notes and scenarios. It used 40 test episodes with six questions each, giving 240 questions per comparison condition. The reported workflow time describes six questions for one independently prepared history, including retention preparation where applicable.

At 128 notes, retained memory answered 237 of 240 questions correctly, compared with 227 for full history. The six-question workflow was slower with retention: 55.5 seconds versus 16.8 seconds. Full history’s 13 misses were unknown responses. Retention had two unknown responses and one incorrect assertion after a source note was dropped. Greater answer coverage did not mean every error became less consequential.

At 512 notes, full history encountered a GPU allocation limit: the local graphics processor could not provide the requested memory buffer. That was a runtime failure, not zero model accuracy. Retained input completed on that execution path. This demonstrated feasibility there, but did not establish that learned retention was the only way to fit the input.

These results justified investigating a smaller representation of the evidence. They did not establish that it would preserve changing goals and instructions. The continuity pilot tested that additional question on a much smaller scale.

What I would keep

I would keep the source-linked, inspectable design: original excerpts, explicit project boundaries, decisions that survive a restart, and a record of what the filter did. A memory tool should distinguish an assistant's proposal, a user's decision and a reported outcome.

I would also keep current instructions and explicit corrections outside a learned usefulness filter. A short instruction may carry more weight than a long progress update. The opening example illustrates why: remembering the general topic and the old deadline is insufficient if the review date and permission to publish have changed.

For someone testing this approach, a manageable starting point is one task, a few decision changes, and questions written before running the model. Compare selective memory with all available evidence and with simpler search. Inspect missing facts as well as wrong statements, and preserve originals so exclusions remain recoverable.

For a test set this small, my recommendation is to supply all available bounded evidence. Selective retention did not match its factual coverage or remove the need to check the next step. A useful memory system must help the next session continue the right work. Smaller input alone does not establish that.

Technical method and limits

Memory and search settings. TypeSafe's example stores memories in a dictionary keyed by customer ID, with a starting retention threshold of 0.5. That threshold is a cutoff for the model's retention judgment, not a guarantee of memory reliability. My pilot kept that cutoff fixed. Both search conditions used BM25, a method that ranks excerpts by their word matches to a query, with at most eight excerpts and a 4,000-token evidence limit. The largest all-excerpts packet was 2,767 tokens. Both figures use the same Kev tokenizer to measure evidence size; these are not Codex billing tokens.

Checks and scope. Questions and model settings stayed fixed. Stored decisions survived a process restart; unchanged excerpts were not classified again. Retention excludes evidence from the selective packet, while the source store preserves it. TypeSafe's simple example accumulates memories without filtering relevance or replacing outdated information. Saved model answers can reinforce mistakes.

Execution limits. The pilot had a 15-minute execution ceiling, a 60-second limit per call, and an independent watchdog: a separate process enforcing the stop deadline. A validator initially mistook two command-line startup warnings for tool use and stopped the run. The audit was corrected without repeating the completed reader call or extending the original deadline; both receipts were preserved. All nine comparisons finished within the original limit, in 5 minutes 59 seconds. Owned processes were confirmed stopped.

Review and deployment. The hosted TypeSafe prose and evidence review was skipped because credits remained unavailable; the draft received local editorial and source review. No production memory integration or publication was performed. Human usefulness and correction time remain unmeasured.

Source: https://docs.typesafe.ai/cookbooks/basic_memory