

Capture Is Easy, Forgetting Is Hard: Benchmarking Prospective Memory in Conversational Agents

Deniz Bilgin Jonas Schewior Andreas Wendemuth

Cognitive Systems, Otto von Guericke University Magdeburg, Germany

{deniz.bilgin, jonas.schewior, andreas.wendemuth}@ovgu.de

Abstract

Conversational agents are good at recall. But recall only reacts and an agent that answers only what it is asked can never raise what the user never thought to mention. We study the other direction which we call prospective memory. A background agent keeps a small set of pins in the assistant’s context before any question arrives and each pin must be retired by the right authority. A constraint from chat, such as a stolen bike, ends only on the user’s word. One from a city feed, such as a suspended tram line, ends only on the feed’s end message, whatever its forecast said. We release a benchmark of 60 days, ten users and 1,802 graded decisions in a mobility assistant, with automatic answer keys for pinning, forgetting, recommendation and habit discovery. Capture, at least of the templated reveals this benchmark renders, is the easy half. A dense 8B model captures events almost as well as a 397B/17B sparse one and what separates the two is forgetting. The small model hoards and invents. The large model’s one systematic failure is a release cascade, asked to rewrite its whole pin table, it treats any release signal as a reason to shorten it. Asked instead to name its removals as explicit operations, it names only the right ones, every swept pin survives, and required session recall rises from 0.77 to 0.99, from 0.80 to 1.00 on the events both runs captured.

1 Introduction

Retrieval augmented generation (Lewis et al., 2020) and the long term memory architectures built on it (Packer et al., 2023; Xu et al., 2025; Chhikara et al., 2025) let an agent reach into past sessions and external stores, and benchmarks report steady progress (Maharana et al., 2024; Wu et al., 2025). But these systems share a shape. They wait until a user asks something, then answer well. This bounds the agent’s value by the user’s own foresight. If the user does not think to ask about a road closure, the agent will not mention it.

We study this in a personal mobility assistant on an urban digital twin, a live city model tracking weather, traffic, parking and air quality. The assistant is developed in IMIQ (Intelligenter Mobilitätsraum im Quartier, Intelligent Mobility Space in the Quarter), a living lab in Magdeburg whose Digital Work-Life Twin extends the transport digital twin with the routines and constraints of the individual user (Kauschke et al., 2025). The pins of this paper are that individual layer kept live. Each interaction is one trip. The user asks for a journey from A to B, the assistant recommends bike, car, tram or walking, and the quality of that one decision makes memory’s contribution directly measurable.

What should shape today’s recommendation rarely arrives as a question. It arrives as a *flashbulb*, a short high impact event that changes what the user can do, and we find two kinds with different lifecycles. A *catalytic flashbulb* starts with the user and lives only in conversation, such as a stolen bike or a broken leg. Only the user can retire it, since no sensor observes a healed ankle. An *environmental flashbulb* starts with the city, such as a rain window or a suspended tram line. Its start message carries only a *forecast* of the end, and only the feed’s own end message can settle it. So forgetting has two authorities, and neither can speak for the other. We are not aware of a memory system that makes this split. A system built only to answer questions can forget by age or contradiction and lose nothing. Proactive agents and the concurrent PM-Bench (Lu et al., 2025; Liu and Gabriel, 2026) come closest, scoring whether a delayed task is raised or executed on time, but neither holds live constraints that two different authorities must retire.

We handle this with *prospective memory*, a term from cognitive psychology for memory of delayed intentions (Einstein and McDaniel, 1990), where a completed intention must be actively deactivated or it keeps intruding (Marsh et al., 1998). Here it is a small set of *pins* kept in the conversational

agent’s context by a background agent, which captures flashbulbs from chat and feed, filters them by personal relevance, and forgets each through its correct channel, inside a three agent architecture with a consolidator, a memory manager and a communicator (Schewior et al., 2026). Existing benchmarks score reactive question answering and show whether an agent found the right passage, not whether a pin should have existed, so we simulate the event feed for full coverage and gradeable answer keys, holding the message interface fixed to what the deployed twin emits.

Our contributions are: (i) **a flashbulb taxonomy with two release authorities**, with a per user relevance filter between capture and pin; (ii) **a benchmark for prospective memory**, a multi day, multi user simulation with graded decisions and automatic, knowledge aware answer keys for pinning, forgetting, mode recommendation and habit discovery; and (iii) **seam attribution**. We evaluate all three seams separately and end to end, holding the communicator fixed, swapping gold pin boards for produced ones, and diffing the two boards to assign every flipped answer’s cause, isolating where memory helps, where it fails, and how upstream errors propagate.

2 Related Work

From Retrieval to Memory

Retrieval augmented generation (Lewis et al., 2020) gave language models access to information beyond their own weights, searching an external store and placing relevant passages in context before the model answers. But retrieval treats every request as if it stands alone, and the store is a fixed library rather than a growing record of the agent’s own history.

MemGPT (Packer et al., 2023) treated the problem as memory instead of search, moving information between a small active context, a larger searchable store of past conversation, and a cold archive. Later work varies how memory is organized and how it ages and recent systems such as AgeMem (Yu et al., 2026) push this management into the agent through reinforcement learning. Surveys agree on the overall shape (Du, 2026; Luo et al., 2026). Memory writes, manages and reads information back when a question needs it.

We ask what it takes for memory to act *before* the question. The closest precedent is the generative agents of Park et al. (2023), whose characters

retrieved memories by relevance, recency and importance and acted unprompted. That was memory driving action rather than answering a query, but it lived in a closed simulation with no real user and no live world.

Forgetting

Once an agent keeps memory across many sessions, the store grows and stale items start to mislead. The usual answers forget by age, by frequency of use, or by a learned sense of importance. Xiong et al. (2026) set out clear deletion rules and show that on a driving agent an unbounded memory loses success rate as junk piles up, while a bounded memory holds steady. AgeMem (Yu et al., 2026) comes closest to our concern. Its worked example removes a stored fact overturned by the user’s own word.

Every method above has a single trigger. The stored item has aged, been contradicted, or stopped earning its place. Our setting needs two, because a catalytic pin can be released only by the user and an environmental pin only by the feed’s own end message. Section 3 makes the split precise and our benchmark scores it directly. The nearest precedent for a belief that lives exactly as long as its warrant is truth maintenance (Doyle, 1979; de Kleer, 1986), where a fact is retracted when the justification that admitted it is withdrawn; a pin’s end condition names whose word counts as that justification. Temporal databases draw a related line between when a fact holds in the world and when the system recorded it (Snodgrass and Ahn, 1985), which is the gap a stored forecast opens and an end message closes.

How forgetting is *expressed* matters too. Emitting operations over a persistent state, rather than regenerating the state, has a precedent in dialogue state tracking: SOM-DST (Kim et al., 2020) keeps the previous dialogue state as an explicit memory and predicts an operation per slot, among them carryover, update and delete. Production memory systems expose the same interface, with MemGPT editing its store through function calls (Packer et al., 2023) and Mem0 applying add, update and delete operations (Chhikara et al., 2025). That line reaches for operations mainly out of efficiency, since regenerating an unchanged state at every turn is wasted work. Our ablation adds a correctness result, with the maintenance rules held fixed, the rewrite format produces spurious deletions of unrelated live pins that the operation format eliminates (§5.5).

Shaping the Context Window

A related line of work treats the context window as something to shape actively. HiAgent (Hu et al., 2025) summarizes past observations into a compact state for long horizon tasks and pulls detail back only when needed, and Lumer et al. (2025) add and remove tools based on what the current question needs. Our background agent likewise decides which pins belong in the communicator’s context. The difference is direction. These systems shape context for a single task already underway, with no personal history across many days and no event feed announcing what is coming.

Personalization and Proactive Assistance

A final body of work studies how agents hold on to who a user is, through preference memory for tastes and profile memory for stable facts. The closest benchmark is PrefEval (Zhao et al., 2025), which measures whether a model keeps honoring a stated preference across a long stretch of unrelated conversation, and finds adherence falling as that distance grows. Proactive there means not forgetting a preference the user already gave. None of this work manages a live set of constraints that are born, filtered for personal relevance, and retired by two different authorities, so that an ordinary question is already answered in light of what happened this morning.

Prospective Memory Benchmarks

The concurrent PM-Bench (Liu and Gabriel, 2026) shares our term and our source in cognitive psychology, and is the closest benchmark. It adapts the Virtual Week paradigm: over a simulated week an agent pursues an ongoing activity and must decide at every step whether any deferred task is now due, with time and event cues, hidden state channels to poll, and updates that cancel or reschedule a task. The object scored is the execution of a delayed intention at its cue. Our object is a standing constraint that is never executed at all. A pin changes how every question in its window is answered, its end is not known when it is born, and it is retired by whichever of two authorities owns it, the user for a life event and the feed for a disruption. PM-Bench measures whether an agent acts at the right moment; we measure whether the agent’s context holds the right claims at every moment, and price the difference downstream. An assistant needs both.

3 The Framework

A reactive memory system answers the question it is given. Ours has to keep a second question answered at all times, what about this user’s day the assistant must already know before the next turn arrives.

Answering that needs three streams of information at once, the behavioral past of what this person usually does, the conversational present of what they have just revealed, and the environmental present of what the city measures and announces. All three are large, and almost all of their content is irrelevant on any given day. The framework’s real job is selection, deciding in the background, before any turn, which few claims deserve a place in the agent’s working context. We build on Agenti-cLOCI (Schewior et al., 2026), a multi agent memory framework in which a consolidator (HIPPO), a memory manager (ENGRAM) and a communicator (BROCA) are coordinated by a thin orchestrator (THALAMUS) over a mobility assistant on an urban digital twin. That work argues for the architecture itself, and specifically for separating fast episodic consolidation from slow, versioned personalization; it evaluates session retrieval on public long horizon benchmarks and treats pinning as one component among five. This paper takes the pin as its object of study. What we introduce is the *pin* as the unit of prospective memory, the split of flashbulbs into catalytic and environmental channels with distinct release authorities, and a benchmark that scores capture, relevance and forgetting against per session answer keys. Neither the taxonomy nor the benchmark appears in that work, and we do not re argue its architectural claims here.

Flashbulbs. The claims worth holding rarely arrive gradually. They arrive as short events with high impact, which we call flashbulbs after the psychology of vivid and circumstance bound memories (Brown and Kulik, 1977). Such memories are no more accurate than ordinary ones, only confidently held (Talarico and Rubin, 2003), and a claim that is confidently held, never decays, and is never reconsidered is exactly the object this framework must manage. A *catalytic flashbulb* is personal and conversational, such as a stolen bike, an injured ankle, or a car at the garage. It changes what the user *can do*, and appears and ends nowhere except in the chat. An *environmental flashbulb* is public and sensor borne, such as a rain window, a suspended tram line, or a road closure, each with a start mes-

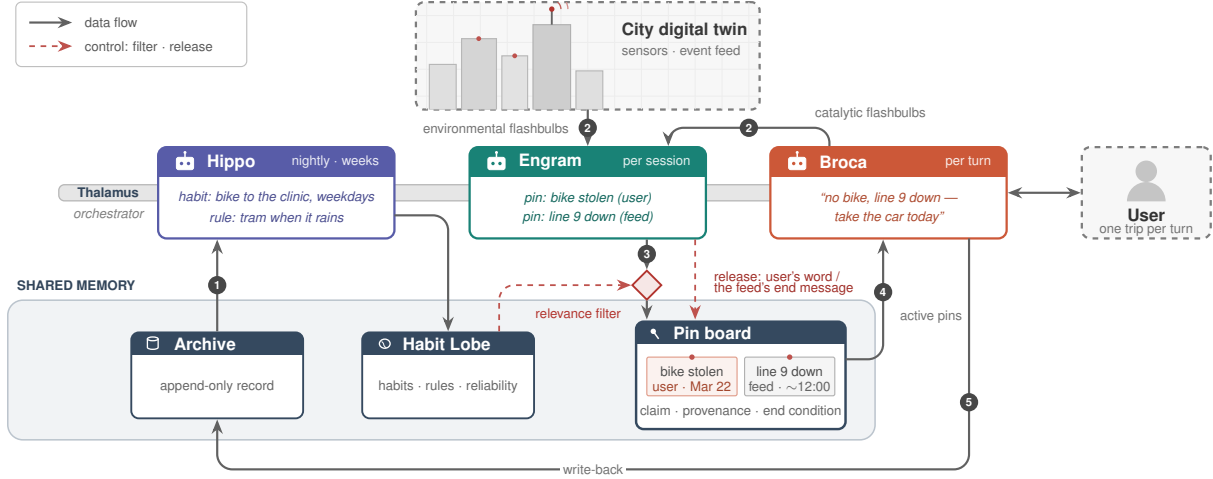


Figure 1: The prospective memory loop. Three agents on progressively shorter timescales sit around a shared memory of three stores, and the circled numbers trace one pass around the loop. HIPPO distills the archive into the Habit Lobe (1). ENGRAM captures catalytic flashbulbs from the conversation and environmental flashbulbs from the event feed (2), passes each through a relevance filter before it may become a pin (3), and later releases each through its own channel. BROCA answers every trip question from the pins that are live at that moment (4) and writes what it learns back to the archive (5). Every arrow is an explicit artifact and so an evaluation seam (§4).

sage and later an end message. It changes what the world *allows*, for some users and not others.

Pins. We call the unit of selection a *pin*, a small timestamped claim placed into the communicator’s active context, such as “no bike, stolen, not yet replaced (user, Mar 22)” or “tram line 9 suspended, expected until noon (feed, 05.20)”. Every pin carries a *provenance*, which channel produced it, and an *end condition*, whose word can kill it. This is not when a claim expires, but who is entitled to say that it has.

The two channels are not symmetric in time. A catalytic pin is a settled fact that only its owner can unsettle. An environmental pin is partly a bet, since its start message carried only a forecast of the end, and a stored forecast can be proven wrong by the world while nothing about the user changed. Retiring a catalytic pin brings the record into line with a world that truly changed. Retiring an environmental pin whose forecast was overtaken corrects a belief about a world that never did. This asymmetry is what separates prospective memory from memory that only looks backward, and why forgetting here has two authorities.

Restraint. Two failure modes bound the design. Missing a constraint produces one bad recommendation. Polluting the context produces distrust on every turn, and almost every feed message is irrelevant to almost every user, so the framework is built around restraint. A claim becomes a pin only

if it intersects this user’s actual mobility, meaning their owned modes, their tram lines, or a trip overlapping the event window. If it comes from anticipating a habit, the underlying routine must be reliable enough to act on, so we treat a habit as actionable only when its observed adherence across the archive is at least 0.8, meaning the user follows it on four scheduled occasions in five. And the pin must still be alive, meaning captured and not yet released by the user’s word or the feed’s end message.

One consequence is worth stating plainly. A user whose behavior shows no pattern is not a personalization failure. For that user the correct Habit Lobe is empty, and the correct behavior is to rely on pins alone.

3.1 Three Agents around a Shared Memory

Three LLM agents work on different timescales around three shared stores (Figure 1). The *archive* is an append only record of every trip and conversation turn. The *Habit Lobe* is a compact distillation of that record, holding habits with schedules and reliability labels alongside conditional rules. The *pin board* is the set of pins currently active.

The agents are named after the neural structures whose functions they loosely echo, a convention memory systems for language models have used before (Gutiérrez et al., 2024).¹ HIPPO is the con-

¹A memory system for conversational agents has recently been released under the name ENGRAM (arXiv:2511.12960).

solidator. It runs offline, reads the full archive, and rewrites the Habit Lobe, grouping trips into habits, estimating how reliable each one is, and mining conditional rules a plain schedule cannot express, such as *bike to the clinic, but tram when it rains*. Nothing hands it a labeled event stream, so catalytic flashbulbs reach it only as natural language and as sudden changes in behavior, and part of its job is keeping those weeks from corrupting the habit statistics. Seven straight days of driving mean something quite different when the chat says the user’s leg is in a cast.

BROCA is the communicator, and the only agent that speaks. Every interaction is one trip, and BROCA answers with a mode recommendation grounded in whatever the pin board holds at that instant. It is also the system’s sensor for the catalytic channel, since what the user mentions in passing is written back to the archive where HIPPO will find it. THALAMUS moves each artifact at its fixed moment and holds no state of its own. No agent calls another directly, which keeps every interface explicit and therefore measurable.

ENGRAM and the two release authorities. ENGRAM owns the pin board. It watches both channels, capturing catalytic flashbulbs from the conversation and environmental flashbulbs from the event feed, and a capture becomes a pin only for the users it actually concerns, checked against the Habit Lobe for owned modes, tram lines and overlapping trips. A suspended line is noise to a user who never rides it, and pinning it is scored as an intrusion.

ENGRAM is equally responsible for forgetting, and the two channels end differently. A catalytic pin may be removed only when the user says it is over. An environmental pin dies with the feed’s end message, not with the forecast the start message carried. A pin kept past its release signal is a *ghost pin*, a failure with a precedent in the psychology the term borrows from, where a discharged intention does not simply fade but must be actively deactivated, or it goes on intruding (Marsh et al., 1998). We score it as a first class failure alongside missed captures.

3.2 Why the Seams Matter

Keeping the agents apart makes the system measurable. HIPPO turns the archive into the Habit Lobe. ENGRAM turns chat and feed into the pin board.

It is unrelated to the component described here.

	gold	naive
<i>ENGRAM (naive pins everything, never forgets)</i>		
event capture (of 80)	1.00	0.875
required session recall	1.00	0.36
ghost pins	0	25,207
irrelevant pins	0	155
<i>BROCA (naive picks the user’s majority mode)</i>		
in acceptable set, all	1.00	0.89
in acceptable set, unconstrained	1.00	1.00
in acceptable set, constrained	1.00	0.56
catalytic / environmental	1.00	0.572 / 0.512
<i>HIPPO (naive claims one habit for everyone)</i>		
habit F1	1.00	0.48
rain fallback accuracy	1.00	0.00
false habits, users without habits	0	3
flag errors	0	3

Table 1: Reference agents on the released benchmark. Each naive agent is blind to one capability, and the matching metrics collapse.

users × days × sessions per day	10 × 60 × 3
graded turns (constrained)	1,802 (438)
catalytic / environmental constraint	292 / 160
environmental events in the feed	29
rain windows / tram suspensions / road closures	21 / 5 / 3
feed messages (one start and one end each)	58
user × event cases scored for capture	80
catalytic (theft / breakdown / injury)	10 (2 / 4 / 4)
environmental	70
required pin obligations (catalytic / env.)	414 (265 / 149)
sessions carrying a required pin	395
actionable habits / sub-threshold routines	19 / 4
users with no actionable habit	3

Table 2: The benchmark in numbers. An environmental event enters the capture count once per user it concerns with at least one required session, so 29 events yield 70 cases; the line 4 suspension is an irrelevance probe that concerns no user. One catalytic event (u_07) is never resolved. Required session recall is scored over the 414 obligations.

BROCA turns pins and a turn into a recommendation. Each agent can therefore be evaluated on its own by substituting gold inputs at its upstream seam, and the end to end shortfall can be attributed to the component that caused it. Attributing failures in a multi agent system is otherwise difficult, and existing approaches annotate execution traces after the fact (Cemri et al., 2025). Because our seams carry typed artifacts rather than free text, attribution here is a substitution experiment rather than a judgment, which Section 4 does directly.

4 Evaluation

Table 2 sizes the benchmark. The three agents let us ask three separate questions. Did the system learn the right habits (HIPPO)? Did it hold the right pins at the right times (ENGRAM)? Did it

recommend the right mode (BROCA)? Each agent is first scored alone, with gold inputs substituted at its upstream seam. Then ENGRAM and BROCA are joined, and the end to end shortfall is attributed to the component that caused it (§4.3).

4.1 Three Agents, Three Questions

HIPPO learns the habits. The agent reads one user’s 60 day archive and returns a habit report, scored against the user’s true habit list. A claim matches a true habit when the destination is the same, the weekday sets overlap by at least half, and the departure times are within an hour. We report precision, recall and F_1 , rain fallback accuracy over matched pairs, the habits invented for the three users who have none (FalseH), and wrong calls in either direction on whether a user has habits at all (FlagE). The gold set is 19 *actionable* habits. Four further routines sit below the reliability threshold, so claiming one costs precision. Only nine habits carry a rain fallback, so that accuracy always appears with its denominator, and it is shown as a dash when a run matched none. Two engines write the report, a deterministic route miner and an LLM. Two pipelines route it. The direct pipeline goes from the log to habits in one step. The lobe mediated pipeline first turns the log into persona lobes and then pins habits from the lobes alone, with the log never read again.

ENGRAM keeps the board. The agent replays the horizon one session at a time. It receives the profile, the habit summary, its own current board, every feed message so far, and the user’s turns, and it emits the updated board. Nothing marks which sentences matter. We measure *event capture* (was the required pin ever placed), *pin delay*, and *required session recall* (the fraction of pin obligations met). We also count four kinds of pollution. *Ghost* pins are kept past their release signal. *Irrelevant* pins belong to events that never concern this user. *Premature* pins sit outside their window. *Spurious* pins match no event at all. Because capture matches catalytic pins by blocked modes, a fabricated “no bike” pin placed before a real theft would still cover it, so *clean capture* credits only pins born at the reveal.

BROCA makes the call. The agent answers every user turn given the conversation so far, the gold habit summary and a pin board. In B1 the board is gold, meaning the session’s *allowed* pins, the full knowable state a perfect ENGRAM would hold. In B2 it is whatever ENGRAM actually produced.

Nothing else differs, so B1 measures how well the communicator reads memory and B2 prices the upstream errors. A *raw archive* baseline drops memory entirely. It places the full log and the full feed in one prompt per turn, still knowledge aware, so the current session’s chosen mode is never shown. We score whether the recommendation falls in the acceptable set and whether it is the best mode, sliced into plain against constrained turns, catalytic against environmental constraints, and the *revision* turns that follow a reveal. Unanswered turns count against the agent.

4.2 Reference Agents Anchor the Metrics

Simple agents establish that each metric isolates one capability (Table 1). A gold agent that reads the answer keys is perfect, which checks that the scorer is self-consistent; whether a system reading only the streams can reach the keys is what the rest of the paper measures. Each naive agent is blind to one capability, and the matching metrics collapse. A BROCA that always recommends the user’s majority mode is perfect on plain turns and falls to 0.56 under constraints. That distance is the value of memory, isolated from question difficulty. An ENGRAM that pins everything and never forgets still captures 87.5% of events (it takes every feed message, but catalytic reveals must still be read out of chat, so the ten it misses are all conversational) while holding over 25,000 ghost pins. Capture without forgetting is worthless, which is this paper’s title in one number.

4.3 Attributing the End to End Gap

B1 and B2 answer the same 1,802 questions, so each lands in one of four cells. It is right in both, wrong in both, a *regression* (right on gold boards, wrong on ENGRAM’s), or an *improvement*. Turns wrong in both runs are the communicator’s own errors, and they reappear whatever the board looks like. Every flipped turn gets one cause, assigned by comparing the two boards. A *missing pin* means a constraint of the question sat on the gold board but not the actual one, split into dropped after capture and never captured. An *extra pin* means ghost, irrelevant or spurious pins polluted an otherwise complete board. An *identical board* means both boards carried the same information, so the flip is sampling noise, since temperature zero serving is not bit deterministic, and these cells are our noise floor. Anything left is an *other difference*. Two checks keep the accounting honest. Net constrained

regressions must equal the B1 to B2 constrained gap exactly, and the analysis must reproduce the released scorer’s numbers to 10^{-9} before any attribution is drawn. Missing pin sessions are grouped into per event windows, because a forgetting failure is not a rate but an episode, with a start, an end, and the turns it swallowed.

4.4 Protocol

We test Qwen3.5 397B-A17B and Llama-3.1-8B-Instruct at temperature 0 under strict JSON contracts with one retry. Prompts are shared across models and conditions up to one documented difference: ENGRAM exists in two versions, v1.2 adding a short list of place zones to the profile block. Table 4 pairs v1.1 (large) with v1.2 (small), and the ops variant uses v1.2; the zones affect road closure relevance only. The prompts state the maintenance policy outright, including both release rules, so the benchmark measures whether a model can *execute* a declared policy across a long horizon, not whether it can infer one; policy inference from the streams alone is a separate question we leave open. An unparseable ENGRAM reply leaves the board unchanged, and an unparseable BROCA reply counts as unanswered. HIPPO’s LLM engine is a recursive language model (RLM; Zhang et al., 2026), which keeps the 60 day log outside the prompt as a variable and analyses it by writing Python over it (prompts and limits in Appendix A, context budget in Appendix D). Finally, the *ops* variant reruns ENGRAM with identical maintenance rules but a different output format. The model returns `add` and `remove` operations and the harness applies them, so a pin can vanish only through an explicit `remove`. Whether ENGRAM’s failures survive this change tells us where they live, in the model’s judgment or in the format we made it answer in.

5 Results

We run the protocol of §4.4 on one large and one small model, Qwen3.5 397B-A17B and Llama-3.1-8B-Instruct, and report one finding per seam.

5.1 Counting Wins the Consolidation Seam

On this benchmark, consolidation is a counting problem, and counting wins. The deterministic miner recovers all 19 true habits, invents none, writes the empty report for the three users who have none, and costs nothing (Table 3). Even its one imperfect number, rain fallback at 0.78 (7/9), is not a judgment failure. The two missed rules

HIPPO engine	F_1	P	R	Rain	FalseH	FlagE
<i>Config A, direct (log → habits)</i>						
Heuristic (no LLM)	1.00	1.00	1.00	0.78 (7/9)	0	0
RLM, Llama 8B	0.08	0.20	0.05	—	0	5
RLM, Qwen 397B	0.64	1.00	0.47	0.20 (1/5)	0	3
+ route grouping	0.83	0.88	0.79	0.57 (4/7)	1	3
<i>Config B, lobe mediated (log → lobes → habits)</i>						
Llama, RLM pin	0.00	0.00	0.00	—	27	9
Qwen + grouping, heur. pin	0.31	0.30	0.32	0.00 (0/3)	11	7
Qwen + grouping, RLM pin	0.35	1.00	0.21	1.00 (2/2)	0	5

Table 3: HIPPO against the 19 true habits. Rain is rain fallback accuracy with its denominator (matched habits carrying a fallback, at most 9), and a dash means the run matched none. FalseH counts habits invented for the three users without reliable habits, and FlagE counts wrong calls on whether a user has habits at all. “Grouping” marks runs that carry the route grouping instruction of the fourth row; the Config B Qwen rows do, the Llama row does not. Costs and all ten runs are in Appendix C.

had at most one wet occasion in the log, so they were not there to be found. The flat 1.00 should be read with its scope: the gold habits and the trips come from the same seeded generator, so it bounds what counting can do on this generator, not on real logs. No LLM comes near it. The small model fails mechanically. Its code turns frequently raise errors, it recovers a single true habit (recall 0.05), and its lobe run invents 27 habits for the three users whose correct report is empty. The large model errs the opposite way, by restraint. It rarely invents but declines borderline routines, and one prompt fix, grouping trips by route instead of destination, lifts it from $F_1 = 0.64$ to 0.83, at the price of one invented habit for a user who has none. Routing the same model through persona lobes, with the same route grouping instruction, drops F_1 from 0.83 to 0.31–0.35, because the lobes replace adherence counts with a self written regularity label (Appendix B). The lesson carries through the paper. Where the work is counting occasions, a procedure is enough. Where the work is deciding which live constraint concerns this user now, and who may retire it, whether a procedure suffices is untested here, and the natural control, a rule based ENGRAM over the structured feed, is left to future work.

5.2 Capture Is Shared, Hygiene Is Not

Both models see events, and only the large model can keep a board (Table 4). Capture is 74 against 73 of 80, recall is 0.771 against 0.768, and the median pin delay is zero for both. Reading a stolen bike out of an ordinary trip turn is not where capability

	Qwen 397B	Llama 8B
event capture (of 80)	74	73
clean capture (of 80)	74	72
required session recall	0.771	0.768
median pin delay (sessions)	0	0
ghost pins	0	1,413
irrelevant pins	2	111
premature pins	0	0
spurious catalytic pins	0	1,885

Table 4: ENGRAM isolated (E1). Capture and recall survive the capability gap and board hygiene does not. Clean capture is defined in §4.1.

matters.² Hygiene is. The large model holds zero ghost, premature and spurious pins, and its one systematic failure is *release cascade forgetting*. When a release signal arrives for one event, the model rewrites the board and sweeps away live pins for unrelated events in the same pass. For example, it had wrongly pinned the line 4 irrelevance probe for user u_02. When that line’s own end message arrived, it removed the probe pin, correctly, and took her real car breakdown pin with it, for 19 required sessions. All five of its catalytic drop windows begin at a session carrying an end message for an unrelated environmental event, the behavior reproduced 3 of 3 times, and explicit one to one release instructions do not fix it. Changing the output format does (§5.5). The small model fails the opposite way. It never lets go and it invents. It holds 1,413 ghost pins, mints 1,885 spurious pins from harmless trip sentences (a sentence like “heading to the station” becomes a no bike pin), and places 111 irrelevant ones. Clean capture separates the two once more, 74 against 72, because the small model’s u_09 injury “capture” is a hallucinated pin that predates the reveal.

5.3 A Perfect Memory Needs a Reader

On gold boards (the B1 columns of Table 5) the large model is near ceiling, and its few constrained errors are its own. It sometimes acts on a resolution one turn early, walks the user into a rain window pinned on its own board, or recommends a mode the user does not own. The small model splits by pin kind. Catalytic pins work (0.842 against the 0.572 naive floor) because obeying one is string matching. The pin names the forbidden mode. Environmental pins barely register (0.550 against a

²The large model ran the v1.1 prompt and the small model ran v1.2, which adds place zones. The difference touches only road closure relevance, and its one casualty is a large model miss, so the near tie understates the large model.

	Qwen 397B		Llama 8B	
	B1	B2	B1	B2
<i>in acceptable set</i>				
overall	0.984	0.957	0.939	0.939
plain	0.999	0.999	1.000	0.999
constrained	0.936	0.824	0.749	0.751
catalytic	0.949	0.798	0.842	0.822
environmental	0.894	0.856	0.550	0.606
revision	1.000	1.000	1.000	1.000
<i>best mode</i>				
overall	0.707	0.693	0.577	0.484
constrained	0.760	0.692	0.477	0.498

Table 5: BROCA over the same 1,802 graded turns, on gold boards (B1) and on the boards ENGRAM actually produced (B2). A turn constrained by both a catalytic and an environmental pin appears in both slices, so the two do not sum to the constrained row.

0.512 floor) because a rain window must still be turned into a mode choice, an inference about timing and route the small model does not make. Even a perfect memory cannot rescue a communicator that cannot read it.

5.4 Errors Transmit or Saturate

Through the strong communicator, errors transmit. Swapping gold boards for the large model’s own costs $0.936 \rightarrow 0.824$ on constrained turns, and the accounting is exact. There are 52 regressions against 3 improvements, all 55 on constrained turns, and $(52 - 3)/438 = 0.112$ reproduces the gap to the digit. 98.0% of the net gap belongs to the memory manager, and 43 of the 52 regressions are pins that were captured and then dropped, which is the release cascade priced downstream. The damage is localized. Plain turns do not move, and the drop windows are dense with wrong answers (12 of the 13 graded turns inside u_10’s car breakdown window).

Against the weak communicator, errors saturate. The same swap does nothing to the small model’s headline ($0.749 \rightarrow 0.751$). Over all 1,802 turns there are 52 regressions against 52 improvements; on the 438 constrained turns the count is 51 against 52, and $(52 - 51)/438 = 0.002$ reproduces the rise. The improvements are not skill. 37 of the 52 are accidents where spurious pins steer the model off bike and walking and happen to satisfy rain constraints. The 58 turns wrong in both runs outnumber either flip count. The communicator is the floor, and memory’s errors land below it. The real price appears one metric down. Best mode falls $0.577 \rightarrow 0.484$, a net 168 turns over all

	Qwen 397B		Llama 8B	
	raw	B1	raw	B1
<i>in acceptable set</i>				
overall	0.956	0.984	0.935	0.939
plain	1.000	0.999	0.999	1.000
constrained	0.817	0.936	0.735	0.749
catalytic	0.918	0.949	0.842	0.842
environmental	0.625	0.894	0.538	0.550
revision	1.000	1.000	1.000	1.000
<i>best mode</i>				
overall	0.861	0.707	0.746	0.577
constrained	0.806	0.760	0.715	0.477

Table 6: The raw archive baseline (full log and full feed in every prompt, no memory system) beside the gold board run B1 of Table 5, over the same 1,802 turns and slices (plain 1,364, constrained 438, of which catalytic 292 and environmental 160, revision 2).

1,802 (304 regressions against 136 improvements). Upstream noise makes the assistant conservatively wrong, inside the acceptable set but off the user’s routine.

Two regimes, one instrument. When the communicator is strong, each dropped pin becomes a countable window of wrong answers and attribution names the component, the mechanism and the sessions. When it is weak, the board barely matters and only the finer best mode metric registers the damage. The noise floor, measured by the identical board flips, is a net of one turn in the large pair and six turns the other way in the small.

5.5 What Curation Buys, and Where the Cascade Lived

The raw archive baseline. Handing the model the full log and feed with no memory system does not fail wholesale (Table 6). It fails exactly where this paper says memory’s job is. The small model moves by six turns of 438 (0.735 constrained against 0.749 on gold boards), the size of the noise floor measured for that pair in §5.4. A reader this weak is bound by reading, not representation. The large model matches its own pipeline on the headline (0.817 against 0.824) and posts the paper’s best habit following (0.861 best mode overall against 0.707 on gold boards, since at both scales copying the literal log beats applying the distilled summary, the same Habit Lobe finding as §5.1). But the slices locate the loss. Catalytic constraints survive raw context (0.918) because a theft sits verbatim in the log. Environmental constraints collapse to 0.625 against 0.894 on gold boards. With 58 feed messages in context, deciding what is active now, what

has ended, and what was never this user’s business is the work the pin board had been doing.

The output format ablation. The ops variant reruns the large model’s ENGRAM with identical maintenance rules and one change. The model emits add and remove operations instead of rewriting the table.³ The cascade disappears. Required session recall rises from 0.771 to 0.993; on the 73 events both runs captured it rises from 0.800 (316 of 395 obligations) to 1.000, and on the catalytic events alone, which zones cannot touch, from 0.728 to 1.000. At all five sessions where E1 swept away an unrelated pin, the ops run holds it. The failure did not move elsewhere: ghost and spurious pins stay at zero, and the only cost is five more irrelevant pins (7 against 2). So the cascade was never a judgment error about what to forget. Asked to rewrite the whole table, the model treats a release signal as an occasion to shorten it. Asked to name its removals, it names only the right ones.

6 Discussion and Conclusion

Memory research has focused on retrieving the right fact once a question arrives. We studied the complementary problem of maintaining a small set of currently true constraints before any question arrives and forgetting them at the right moment. Our results show that forgetting is the main challenge. A dense 8B model captured events almost as well as a 397B/17B mixture of experts model. The main difference was forgetting. The smaller model retained outdated constraints and invented new ones. The larger model’s main failure, the release cascade, disappeared when we replaced table rewriting with explicit edit operations. Recall increased from 0.771 to 0.993, and to 1.000 on the events both runs captured. As in dialogue state tracking (Kim et al., 2020), explicit operations appear to be important for correct forgetting. Assistants that combine chat with external signals must distinguish constraints that only the user can revoke from those that only the world can invalidate. We release the benchmark, generator, validation suite and answer keys to support future work on proactive memory.

³The ops run uses the v1.2 prompt, so it differs from E1 in format and in place zones. Capture rises from 74 to 79 of 80, partly the zones’ doing (the u_02 road closure is now captured), and capture feeds recall, since an obligation for an event never captured cannot be met. The text therefore also reports recall restricted to the 73 events both runs captured. The cascade comparison is unaffected either way, since the five drop windows are catalytic and zones do not touch them.

Limitations

The benchmark is synthetic by necessity, since proactive memory has no observational ground truth. User turns are rendered from seeded template pools, so the language is less varied than real chat, and a model good at matching templates may find extraction easier here than in the wild. Our capture results should be read with that scope: they show capture of templated reveals, and the test that would extend them, a paraphrase augmented split that rewrites every user turn while holding the events and answer keys fixed, is left to future work. All results come from one simulated city whose calibration values are plausible hand set numbers rather than survey data. The two authority split itself needs only a chat channel and a feed with typed start and end messages, so it transfers to any assistant that combines the two; the answer keys do not, since they are generated with the city. The benchmark also states the maintenance policy in the prompt (§4.4), so it measures whether a model can execute a declared policy, not whether it can discover one. Scoring is at the decision level, meaning pin operations, mode choices and habit reports. A system could recommend the right mode rudely and we would not notice.

We test two models from two size classes, served through one gateway, with one run per configuration. Temperature-zero serving is not bit-deterministic and we report the measured noise floor with every attribution, but we cannot say whether the release cascade appears in other model families, nor whether the operation format cures it there. That test needs a second large model that exhibits the cascade, since a model that fails by hoarding has nothing for the format to fix. The two ENGRAM prompt versions differ by a short list of place zones, which we disclose where it matters. We also did not run a procedural ENGRAM. Environmental feed messages are structured, with ids and typed start and end announcements, so a rule based manager is a natural control for that channel, and its absence means we cannot yet say whether the environmental half of pin maintenance needs a model at all, the catalytic channel, which must read life events out of free chat, is where the case for one is strongest.

Two further parts of the problem remain untested. The HIPPO evaluation isolates one user over sixty days, which fits comfortably in a modern context window, so the regime that would make a recursive

engine necessary, a single history grown past the window, is out of reach here. A late remark such as a reported injury can reinterpret months of earlier behavior, so such a history resists batching, and showing that crossover needs a long horizon single subject corpus. The lobe mediated runs were not controlled, since the lobes were rebuilt per run, and the gold lobe substitution that would separate lobe quality from pinning judgment is left to future work, as is a stricter habit matcher than the tolerant one of §4.1. Finally, the framework’s intrusiveness budget, meaning when to volunteer a pin rather than whether to hold it, the mixed initiative question of Horvitz (1999), is not separately scored in this release: the benchmark scores what the assistant knows when asked, not whether it speaks unasked.

Ethics Statement

This work complies with the ACL Code of Ethics. All users, conversations, life events and sensor streams in the released benchmark are synthetic and generated from seeded distributions. No human data was collected, and no real person’s mobility patterns are represented.

In accordance with the ACL Policy on AI Writing Assistance, we disclose the following use of generative AI tools. Claude Code (Anthropic, 2026) was used for coding assistance during framework implementation, evaluation scripting and data processing, including the experiment harness and the attribution analysis, and for drafting and refining text and organizing the literature. All experimental design, verification of code and results, and conclusions are the authors’ own, and the assistant was not part of any system under evaluation.

Acknowledgments

This research was conducted as part of the IMIQ project, which received funding from the EFRE European Regional Development Fund in Saxony-Anhalt, Germany. This work was supported by KISSKI (AI Service Centre for Sensitive and Critical Infrastructures, <https://kisski.gwdg.de>), which provided the compute infrastructure used to run the experiments.

References

Anthropic. 2026. Claude code. <https://claude.com/claude-code>. Agentic coding assistant.

- Roger Brown and James Kulik. 1977. Flashbulb memories. *Cognition*, 5(1):73–99.
- Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*. ArXiv:2503.13657.
- Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.
- Johan de Kleer. 1986. [An assumption-based TMS](#). *Artificial Intelligence*, 28(2):127–162.
- Jon Doyle. 1979. [A truth maintenance system](#). *Artificial Intelligence*, 12(3):231–272.
- Pengfei Du. 2026. Memory for autonomous LLM agents: Mechanisms, evaluation, and emerging frontiers. *arXiv preprint arXiv:2603.07670*.
- Gilles O. Einstein and Mark A. McDaniel. 1990. Normal aging and prospective memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 16:717–726.
- Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 59532–59569. ArXiv:2405.14831.
- Eric Horvitz. 1999. [Principles of mixed-initiative user interfaces](#). In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*, pages 159–166. ACM.
- Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 2025. HiAgent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32779–32798, Vienna, Austria. Association for Computational Linguistics.
- Leander Kauschke, Gonzalo Haefner, Björn Otto, Anja Koebrich-Leon, Modou Lamin Manjang, Ernst Stötzner, Hamd Mehfooz Khan, Andreas Wendemuth, Ellen Matthies, Andreas Nuernberger, Hartmut Zadek, Andreas Mueller, and Benjamin Noack. 2025. [A digital work-life twin: Managing mobility through user understanding](#). In *mobil.TUM 2025 – 14th International Scientific Conference on Mobility and Transport*, Singapore.
- Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee. 2020. [Efficient dialogue state tracking by selectively overwriting memory](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 567–582. Association for Computational Linguistics.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.
- Genglin Liu and Saadia Gabriel. 2026. PM-Bench: Evaluating prospective memory in LLM agents. In *Proceedings of the Conference on Language Modeling (COLM)*. ArXiv:2607.12385.
- Yaxi Lu, Shenzhi Yang, Cheng Qian, Guirong Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, Weiwen Liu, Yasheng Wang, Zhiyuan Liu, Fangming Liu, and Maosong Sun. 2025. Proactive agent: Shifting LLM agents from reactive responses to active assistance. In *The Thirteenth International Conference on Learning Representations (ICLR)*. ArXiv:2410.12361.
- Elias Lumer, Anmol Gulati, Vamse Kumar Subbiah, Pradeep Honaganahalli Basavaraju, and James A. Burke. 2025. MemTool: Optimizing short-term memory management for dynamic tool calling in LLM agent multi-turn conversations. *arXiv preprint arXiv:2507.21428*.
- Jinghao Luo, Yuchen Tian, Chuxue Cao, Ziyang Luo, Hongzhan Lin, Kaixin Li, Chuyi Kong, Ruichao Yang, and Jing Ma. 2026. From storage to experience: A survey on the evolution of LLM agent memory mechanisms. *arXiv preprint arXiv:2605.06716*.
- Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870, Bangkok, Thailand. Association for Computational Linguistics.
- Richard L. Marsh, Jason L. Hicks, and Martin L. Bink. 1998. Activation of completed, uncompleted, and partially completed intentions. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 24:350–361.
- Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2023. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*.
- Joon Sung Park, Joseph C. O’Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*.

- Jonas Schewior, Deniz Bilgin, Gonzalo Haefner, and Andreas Wendemuth. 2026. [AgenticLOCI: A multi-agent memory framework for conversational AI with longitudinal personalization](#). In *Second Workshop for Research on Agent Language Models*.
- Richard Snodgrass and Ilsoo Ahn. 1985. [A taxonomy of time in databases](#). In *Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data*, pages 236–246. ACM.
- Jennifer M. Talarico and David C. Rubin. 2003. [Confidence, not consistency, characterizes flashbulb memories](#). *Psychological Science*, 14(5):455–461.
- Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations (ICLR)*. ArXiv:2410.10813.
- Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. 2026. How memory management impacts LLM agents: An empirical study of experience-following behavior. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 623–645, San Diego, California, United States. Association for Computational Linguistics.
- Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*. ArXiv:2502.12110.
- Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. 2026. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.
- Alex L. Zhang, Tim Kraska, and Omar Khattab. 2026. [Recursive language models](#). *Preprint*, arXiv:2512.24601.
- Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. 2025. Do LLMs recognize your preferences? evaluating personalized preference following in LLMs. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

A Agent Prompts

We list the task prompts verbatim. The HIPPO LLM engine receives only the small task instruction below, the 60 day log itself is offloaded to the RLM as the external context variable and never appears in the prompt. Every RLM run uses a 12-iteration cap, a 300,000-token ceiling on input plus output, a 1200-second per-completion timeout, and context compaction, served through an OpenAI-compatible endpoint. The placeholder <UID> is filled with the user id at run time. ENGRAM, BROCA and the raw baseline are served through the same endpoint at temperature 0 with strict JSON contracts and a single retry nudge (§4.4).

HIPPO, Config A (direct: log → habits).

The REPL variable 'context' is a JSON object with keys 'trips' and 'chat' for ONE user (user_id <UID>) over ~60 days. trips = records with date, wd (Mon..Sun), session, from, to, depart ('HH:MM'), mode (bike|car|tram|walk), rain (mm/h). chat = the user's own messages (where life events like a stolen bike, injury or car breakdown are mentioned). Write Python over 'context' (json.loads it; do NOT print the whole thing) to find this user's RELIABLE mobility habits.

A habit = same destination, a stable weekday set, ~same departure time (within an hour), usually the same mode. Group trips by their (from, to) ROUTE, not destination alone -- a destination like 'home' is often reached from different origins at different times (a work->home commute vs a late hbf->home trip); treat each (from, to) pair as a SEPARATE candidate habit so distinct routes are not merged and wrongly judged 'unstable'. It may have a rain rule: a different mode when rain >= 0.5 mm/h -> report as rain_fallback, else null. Ignore temporary disruptions (announced in chat: stolen bike/injury/breakdown) -- use the dominant long-run mode. If a destination's trips are irregular (happen on < ~75% of expected occasions, or times/days drift), do NOT report it; inventing a habit is worse than missing one. If nothing is reliable, return an empty list.

Submit ONLY this JSON as answer['content'] (no prose):

```
{"user_id": "<UID>", "no_reliable_habits": <bool>, "habits": [{"to": "<place>", "days": ["Mon", ...], "time": "HH:MM", "mode": "<mode>", "rain_fallback": "<mode or null>"}]}
```

HIPPO, Config B stage 1 (persona-lobe builder: log → lobes).

The REPL variable 'context' is a JSON string with keys 'trips' and 'chat' for ONE user (user_id <UID>). trips = records with date, wd (Mon..Sun), session, from, to, depart ('HH:MM'), mode (bike|car|tram|walk), rain (mm/h). chat = the user's own messages. Write Python over 'context' (json.loads it; do NOT print it whole) to consolidate this person into PERSONA LOBES -- durable categorised memory. Build one lobe per category: mobility, schedule_routines, health_constraints, places_familiarity, social_contacts, preferences_values, transient_changes. Each lobe has a category, a short summary, and an observations list. For mobility and schedule_routines, each observation has fields: from, to, days (e.g. Mon..Fri), typical_time (HH:MM), usual_mode, weather_mode (a mode or null), and regularity (high|medium|low = your honest judgement). Group observations by (from, to) route, not destination alone -- the same place reached from different origins at different times is a SEPARATE routine, not an 'unstable' one. Note life events under transient_changes/health_constraints; do NOT let a temporary mode become a routine's usual_mode. If a person has no dependable routine, leave observations empty. Do not over-claim.

Submit ONLY this JSON as answer['content']: {"user_id": "<UID>", "lobes":

```
[{"category": "...", "summary": "...", "observations": [{"from": "...", "to": "...", "days": ["Mon"], "typical_time": "HH:MM", "usual_mode": "...", "weather_mode": null, "regularity": "high"}]}
```

HIPPO, Config B stage 2 (pinner: lobes → habits, log unseen).

The REPL variable 'context' is a JSON string of ONE user's consolidated PERSONA LOBES (user_id <UID>): a dict with keys user_id and lobes, where each lobe has a category, a summary, and an observations list. Using ONLY these lobes (you do NOT have the raw log), json.loads 'context' and decide which recurring routines are reliable enough to PIN as habits. Only mobility/schedule_routines routines qualify. Pin a routine only if the lobes present it as dependable (regularity high, consistent days/time). If the lobes describe someone with no dependable routine, return an empty list -- inventing a habit is worse than missing one. rain_fallback = the bad-weather mode if the lobes say so, else null.

Submit ONLY this JSON as answer['content']: {"user_id": "<UID>", "no_reliable_habits": <bool>, "habits": [{"to": "<place>", "days": ["Mon"], "time": "HH:MM", "mode": "<mode>", "rain_fallback": "<mode or null>"}]}

ENGRAM (system prompt; the main, whole-table condition). Each session's user message supplies the current time and weekday, the user profile (with place zones from prompt v1.2 on), the habit summary, the current pin table, the feed messages that arrived since the last session, and the user's turns of this session.

You are Engram, the background memory manager of a personal mobility assistant. You maintain the assistant's PIN TABLE: the small set of currently-true constraints it must know before answering any trip question. You never talk to the user.

You watch two streams and nothing else:

1. The user's chat messages. When the user mentions a life event that blocks a transport mode, add a CATALYTIC pin. When the user says such an event is over, REMOVE that pin. Only the user's word ends a catalytic pin.
2. The city feed. A "start" message announces a disruption (rain window, tram line cancelled, road closure); an "end" message says it is over. Add an ENVIRONMENTAL pin on a start message ONLY IF it concerns this user: the affected mode must be one they plausibly use for trips that overlap the event (check their tram lines, owned modes, routines). REMOVE the pin when the matching end message arrives. The expected_until field is only a forecast, never remove a pin just because that time passed; wait for the end message.

Keep every other pin unchanged. Pins persist across sessions and days until their proper release signal. Keep the table small: only currently active, personally relevant constraints.

REMOVAL IS STRICTLY ONE-TO-ONE. An end message removes ONLY the pin whose id matches that message, never any other pin. A user saying one problem is over ("car is repaired") removes ONLY the pin for that same problem -- a city disruption stays until ITS OWN end message, and any other life event stays until the user ends THAT one. Never return a smaller table than you received unless, for each pin you dropped, its matching release signal arrived THIS session. When unsure, KEEP the pin.

Reply with ONLY a JSON array, no other text -- the complete updated pin table. Entry formats (exact):

```
{"kind": "environmental", "ref": "<the feed message's id>"}
{"kind": "catalytic", "blocks": ["bike"], "label": "<short reason>"}
```

"blocks" entries must be from: bike, car, tram, walk. An empty table is [].

ENGRAM, ops variant (output format only; the rules above are unchanged).

Reply with ONLY a JSON object, no other text -- the CHANGES to the pin table this session (never the whole table):

```
{"add": [<pins to add>], "remove": [<pins to remove>]}
```

Pin formats (exact):

```
{"kind": "environmental", "ref": "<the feed message's id>"}
{"kind": "catalytic", "blocks": ["bike"], "label": "<short reason>"}
```

To remove an environmental pin, give its ref; to remove a catalytic pin, give the blocks of the pin being removed. "blocks" entries must be from: bike, car, tram, walk. If nothing changed, reply {"add": [], "remove": []}.

BROCA (system prompt). Each turn's user message supplies the current time, the profile and place zones, the habit summary, the rendered pin table (city pins with their headline and expected end, user pins as "modes unavailable: reason"), and the conversation of this session.

You are Broca, a personal mobility assistant. The user tells you about one trip (from A to B) per message; you recommend HOW to travel.

You know three things and nothing else:

1. The user's profile and routines (below) -- what they own, where their places are, their usual and preferred modes.
2. Your PIN TABLE (below) -- constraints prepared for you by a background memory agent. Treat every pin as currently true: a catalytic pin means the listed modes are unavailable to this user right now; an environmental pin means that disruption (rain window, cancelled tram line, road closure) is active now or imminent. Never recommend cycling or long walks into a pinned rain window; never recommend a pinned-out mode.
3. What the user says in this conversation. If the user mentions a new constraint in their message (an injury, a stolen bike, a broken car), respect it immediately even if no pin exists yet.

If pins and routine conflict, pins win. Prefer the user's habitual mode among the allowed options.

Reply with ONLY this JSON, no other text:
{ "mode": "bike" | "car" | "tram" | "walk" }

Raw archive baseline (system prompt). Each turn's user message supplies the current time, the profile, the full past interaction log (one line per trip: date, weekday, session, route, the mode taken, and the user's words), every feed message published so far, and the conversation of this session. The current session's selected mode is never included.

You are a personal mobility assistant. The user tells you about one trip (from A to B) per message; you recommend HOW to travel.

You are given, complete and unprocessed:

1. The user's profile.
2. The FULL RAW LOG of every past interaction with this user -- what they said, the trip, and the mode they actually took.
3. Every message the city's sensor feed has published so far (disruption start and end announcements).
4. The current conversation.

Nothing has been summarized, filtered, or maintained for you. Work out the user's habits and which constraints (from their life events in the log, or from the feed) are still active RIGHT NOW directly from the raw material; a disruption is over only when its end message appears, and a life event (theft, injury, breakdown) is over only when the user later says so. If the user mentions a new constraint in the current conversation, respect it immediately.

Reply with ONLY this JSON, no other text:
{ "mode": "bike" | "car" | "tram" | "walk" }

B Full HIPPO Analysis

Why the lobes lose recall. Routing the large model through Config B, holding the route grouping instruction fixed, drops the F_1 from 0.83 direct to 0.31–0.35 lobe mediated (and from 0.64 to 0.27–0.48 without it, Table 7), for more tokens and far more time, and almost all of the loss is recall. The reason is in what the interface carries. The lobes replace adherence counts with a regularity label that the first stage writes itself, and the pinning stage never sees the log, so it cannot check the actionability condition; it can only trust an adjective. Where the counts survive, restraint is free. Where they do not, it has to be exercised as judgment.

The two pinners, read with care. The route grouping instruction that helped the direct engine appears to hurt the deterministic pinner, which faithfully pins whatever the lobe builder over generates, collapsing precision to 0.30 and inventing 11 habits for the users without reliable ones. The RLM pinner filters instead, and holds precision at 1.00. We state this as a suggestion rather than a result, because the lobes were rebuilt for each run rather than built once and shared, so the two pinners did not read the same lobes. The controlled version would freeze one lobe set per model and substitute gold lobes at the new seam, exactly as the main evaluation substitutes gold boards at the others; it would split Config B's loss into lobe quality and pinning judgment, and we leave it to future work. One reading is safe across both runs: an upstream change that raises recall is safe only when the downstream stage can still exercise restraint, and a pinner handed adjectives instead of counts has little left to exercise it with. The lobe mediated rain score deserves the same care: the 1.00 belongs to a run with recall 0.21, so it sits over two matched habits. It is coverage, not skill.

Why the recursive engine is overhead at this scale. The RLM engine works: with a capable model it reads the full 60 day log by writing code over it and returns valid habits at $F_1 = 0.83$. But a single user's log is about 12k tokens, roughly 9% of even the smaller model's context window, so the data would fit in the prompt directly, and the out of context analysis buys nothing here (Appendix D gives the full budget). The regime that makes a recursive engine necessary rather than merely functional is a single user's history

grown past the context window, which this benchmark’s per user horizon does not reach; the Limitations return to this.

C Full HIPPO Evaluation Matrix

Table 3 distils these runs; the complete matrix is below. F1/P/R are habit quality; Mode is travel-mode accuracy; Rain is rain-fallback accuracy; FalseH counts habits invented for the three users without reliable habits; FlagE counts “no-reliable-habits” flag errors (both lower is better). Mode is scored over matched habits and is 1.00 in every run that matched anything; the 0.00 in b2_llama8b reflects zero matched habits (0/0 convention). Calls is API calls; In/Gen/Tot are prompt, generated, and total tokens in thousands; s is wall-seconds. The _v2 runs add the route-grouping instruction (§5.1). Config A is direct; Config B is lobe-mediated.

Run	F_1	P	R	Mode	Rain	FalseH	FlagE	Calls	In k	Gen k	Tot k	s
gold (reads truth)	1.00	1.00	1.00	1.00	1.00	0	0	–	–	–	–	–
naive baseline	0.48	0.70	0.37	1.00	0.00	3	3	–	–	–	–	–
h1_heuristic	1.00	1.00	1.00	1.00	0.78	0	0	0	0	0	0	0
h2_llama8b	0.08	0.20	0.05	1.00	0.00	0	5	107	511	20	531	487
h2_qwen397b	0.64	1.00	0.47	1.00	0.20	0	3	101	891	84	976	2590
h2_qwen397b_v2	0.83	0.88	0.79	1.00	0.57	1	3	92	635	81	716	2525
b1_llama8b_heurpin	0.19	0.23	0.16	1.00	0.00	0	5	115	914	41	955	639
b1_qwen397b_heurpin	0.48	1.00	0.32	1.00	0.00	0	5	95	683	89	772	8077
b1_qwen397b_v2_heurpin	0.31	0.30	0.32	1.00	0.00	11	7	108	726	107	832	13228
b2_llama8b_rlmpin	0.00	0.00	0.00	0.00	0.00	27	9	175	748	44	793	343
b2_qwen397b_rlmpin	0.27	1.00	0.16	1.00	0.67	0	6	125	794	83	878	4770
b2_qwen397b_v2_rlmpin	0.35	1.00	0.21	1.00	1.00	0	5	133	1046	95	1140	6860

Table 7: Full HIPPO evaluation matrix. h1/h2 are Config A (direct); b1/b2 are Config B (lobe-mediated). Token columns are thousands.

D Context Budget and the HIPPO RLM Regime

The RLM engine exists to analyse inputs that do not fit in the context window. Table 8 accounts for how much of a 131,072-token window, the smaller model’s, this benchmark actually consumes; the large model’s window is 262,144 tokens, which halves every fraction below. A single user’s trips-plus-chat payload is about 12k tokens, only 9.3% of the window, so a naive “load the whole user into the prompt” is entirely viable; the RLM’s out-of-context, code-writing analysis is therefore overhead at this scale, not leverage, which is visible in Table 7 as 0.5–1.1M tokens spent for quality below the free heuristic. Only when all ten users are loaded at once do the raw archives (287k, 219%) exceed the window, and even then the trimmed payload (122k, 93%) still fits. The benchmark thus never reaches the regime that would make a recursive engine necessary rather than merely functional; as the Limitations argue, that regime requires a single subject whose own history has grown past the window, which we leave to future work.

Scope	trips+chat	raw archive	% of window
Single user (mean)	12,182	28,737	9.3%
All ten users	121,825	287,370	93% / 219%

Table 8: Context budget against a 131,072-token window (Llama-3.1-8B; Qwen3.5 397B-A17B has 262,144). “trips+chat” is the offloaded payload the RLM analyses; “raw archive” is the untrimmed archive file. The all-users row lists the payload and raw-archive percentages. A single user fits comfortably; only the whole-corpus raw archives overflow.