Do Agents Dream of Electric Sheep?

Why AI agents forget you, and what it takes to make them stop

Ivan Novikov, Volodymyr Panchenko

Portal AI — May 2026

Abstract

In ten years, every person on earth will have an artificial mind that has known them for years.

Large language models are smart enough to enable personal AI agents. The bottleneck on whether AI is useful in your life is no longer model capability — it is whether the system you're talking to remembers your name, your project, the constraints you set last week, and the lesson it learned about you the day before. Most AI agents today don't, so users learn to share less and accept less.

Bigger context windows, retrieval-augmented generation, and sticky-note memory features each solve part of the problem and miss the heart of it. The brain has been solving the real problem for hundreds of millions of years, with cooperating layers that each do one job on their own schedule. AI agents need the equivalent — without it, every conversation starts over and the relationship breaks down.

This paper presents **lifelong memory**: a four-layer architecture — **Recall, Gate, Consolidation, Belief** (1R, 2G, 3C, 4B) — inspired by that biology and running today on Portal One+, where 30,000+ agents on the open-source OpenClaw runtime use it to give every user their own *personal intelligence* — an AI that gets meaningfully better at understanding them the longer it knows them. The result is 66% D30 retention against a 12% consumer-app benchmark — a natural five-fold lock-in earned by a substrate that gets denser every day.

The technology to build this is emerging now, mostly behind the closed doors of frontier labs. We're putting ours out in the open so more of the field can build on it together.

---

1. The Forgetting

You told your AI everything about your business. The next day it forgot.

Every person who has used a frontier model long enough has lived this moment. The agent that helped you write the strategy on Tuesday opens fresh on Wednesday. You start over. You explain who you are again. You re-paste the constraints. After enough weeks of this, you stop sharing the deepest version of the project at all, because the cost of re-explaining is too high. The relationship stops growing. The agent never gets to know you. You accept good enough.

The field's response, so far, has been to make the pipe wider. Million-token context windows. Two-million-token context windows. *Memory* features that pin a few hand-flagged facts to a key-value store. Retrieval systems that fetch chunks at request time. Each of these solves part of the problem. None of them is what we mean when we say we *remember* you.

In 1953, a patient identified in the medical literature as H.M. underwent surgery to remove his medial temporal lobes — including the hippocampus — to treat severe epilepsy (Scoville & Milner,

1957). The seizures stopped. So did his ability to form new long-term memories. For the next fifty-five years, every conversation H.M. had was the first conversation. He could hold a thought for thirty seconds, then it was gone. Brenda Milner, the neuroscientist who studied him for decades, walked into the room and re-introduced herself every visit. He was warm, intelligent, fully present — and he could not become anyone in particular, because *becoming requires accumulation.*

Most production AI agents today are H.M.

They are warm. They are intelligent. They are fully present in the moment. And they cannot become anyone in particular for you, because the substrate that would let them accumulate has not yet been built into them. Each conversation is the first conversation. Each one ends with the agent forgetting it ever happened.

This paper is about the substrate that closes that loop. What it is. Why the obvious solutions — bigger context, more retrieval, sticky-note features — don't work. How four cooperating layers, modeled in a non-coincidental way on what mammalian brains do during sleep, do. And what happens — measurably, on disk, in the lives of thirty thousand humans — when those layers are running in their personal AI agents.

---

2. What "Memory" Actually Means

When practitioners say *AI memory* in 2026, they typically mean one of four very different mechanisms, and the conflation is doing real damage to the conversation. They are not the same thing. They solve different problems. They fail differently. They cannot substitute for one another. The reason most AI memory demos feel hollow is that they ship one of them and call it memory.

The first is the **context window** — the model's working memory for the next response. It is lossy, ephemeral, and dies the moment the session does. Bigger context windows are not memory; they are a wider pipe with the same problem at both ends. A million tokens of conversation is not the same as remembering you. It is the same as having very, very recent short-term memory. The pattern is identical to what happened to H.M. — a window of perfect lucidity, then nothing.

The second is **retrieval-augmented generation (RAG)**. A query comes in, an index fetches relevant chunks, the chunks get pasted into the prompt, the model answers. RAG is excellent for the next reply. It does nothing for the next month. It cannot decide what to remember. It cannot promote an insight from yesterday's conversation into the agent's default understanding of you. RAG retrieves into context. Memory operates on what gets retrieved, what gets stored, and what gets believed.

The third is what most consumer AI products now ship as a *memory* feature — a curated key-value store of explicitly-flagged facts. *Remember that I prefer TypeScript. Remember my dog's name. Remember I am vegetarian.* These features are useful and lightweight, but they require human labor to maintain, they don't compound, and they don't ground. They are a sticky-note layer over a stateless engine.

The fourth — what we mean — is **lifelong memory**: a system of cooperating layers that *together* make the agent's understanding of a person deepen automatically, without the user's manual labor, in a way that survives compaction, model swaps, and time. We use *lifelong memory* throughout this paper as the category term; Portal's production implementation is internally codenamed **Project Nexus**.

These are four distinct mechanisms. They live in different parts of the system, run on different schedules, and are implemented by different pieces of code. Reading what follows requires holding the distinction. If at any point you find yourself thinking *but isn't this just RAG with extra steps* or *isn't this just a bigger context window*, return to this paragraph and check which of the four you have in mind. The architecture in §3 is the fourth.

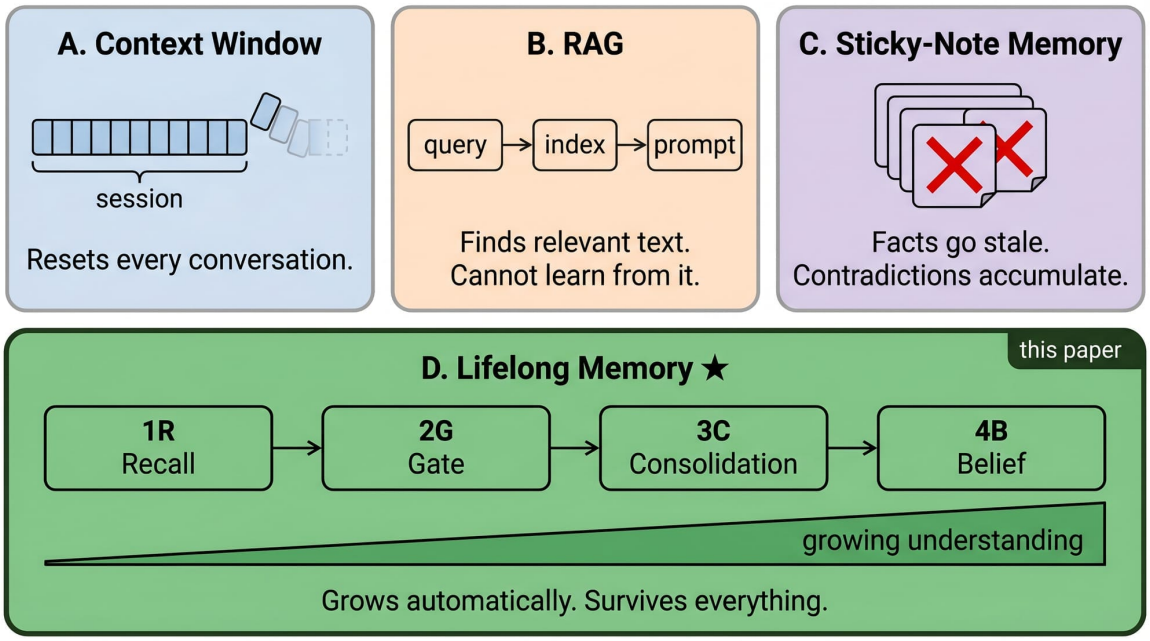

Figure 1. The four mechanisms practitioners conflate when they say "AI memory" — context window, RAG, sticky-note features, and lifelong memory. Only the fourth grows automatically and survives compaction, model swaps, and time. This paper introduces the fourth.

---

3. The Architecture

Lifelong memory has four cooperating layers. Each owns one job and runs on its own schedule. Each alone is insufficient. Together, they produce an agent that does not require its operator to re-introduce themselves on Wednesday.

The order is not arbitrary. Each layer addresses a problem the previous layer creates by existing — and the brain solves the same dependency chain in the same order, which is what makes the architecture feel inevitable once you see it.

Layer 1 — Recall (1R). Every conversation starts blank unless the agent kept notes. Recall is the agent's journal — every daily file, every note, every persisted conversation, indexed so the agent can flip to the right page when something rings a bell. *Example: three weeks ago you mentioned a friend's startup just raised $2M; this layer is what makes that conversation findable when her name comes up again.* The brain does this in the hippocampus, which indexes lived experience as it happens (Marr, 1971; Squire & Alvarez, 1995).

Layer 2 — Gate (2G). A journal is useless if the agent doesn't open it. In practice, when the agent is about to reply, it almost never thinks to check. The Gate is what makes it check —

automatically, before every response, like glancing at your notes before you walk into a meeting. *Example: you ask the agent to draft a congratulations email; the gate runs the search first and finds the $2M raise, the friend's name, and her company before the agent starts composing.* The brain does the same thing: by the time you start speaking, the relevant context has already been pulled from memory before the words form (Levelt, 1989).

Layer 3 — Consolidation (3C). Notes pile up. Most of what is written each day is forgettable; some of it is the kind of thing the agent should remember a year from now. Consolidation is the night-shift sweep: every night, while no one is looking, it reviews what has accumulated, scores it, and promotes the survivors into the agent's permanent record. *Example: after your friend's name has come up across six different conversations over a month, the layer promotes "friend X runs Y, raised $2M April 2026" into the durable record, so the agent leads with it next time instead of searching from scratch.* The brain does this during sleep — the hippocampus replays the day's traces and the cortex slowly incorporates the durable patterns (McClelland, McNaughton & O'Reilly, 1995; Diekelmann & Born, 2010).

Layer 4 — Belief (4B). A pile of notes is hard to query and impossible to fact-check. The Belief layer turns the durable record into something more like a wiki: every claim about you carries its receipts — which file said it, which line, when. When two notes contradict each other, the layer flags it. *Example: the claim "friend X's round was $2M" carries a pointer back to the daily file where you first said it; when a later daily file says the round actually closed at $1.5M, the layer flags the contradiction for review instead of silently overwriting.* The brain does this through semantic memory: lived experiences abstract into structured, citable knowledge that can be referenced and revised (Tulving, 1972).

The four-layer architecture is what each of those four jobs looks like when you build it as software. A small complementary mechanism — the *silent rescue* — sits alongside the four to catch context worth keeping when a long conversation forces the system to summarize and drop older turns before consolidation has had its chance. We describe it after the four-layer dives.

We refer to the layers as **Layer 1 through Layer 4** in prose, with the mnemonic shorthand **1R, 2G, 3C, 4B** (number-then-letter) when space matters — figures, tables, cross-references. The number gives the order; the letter gives the function.

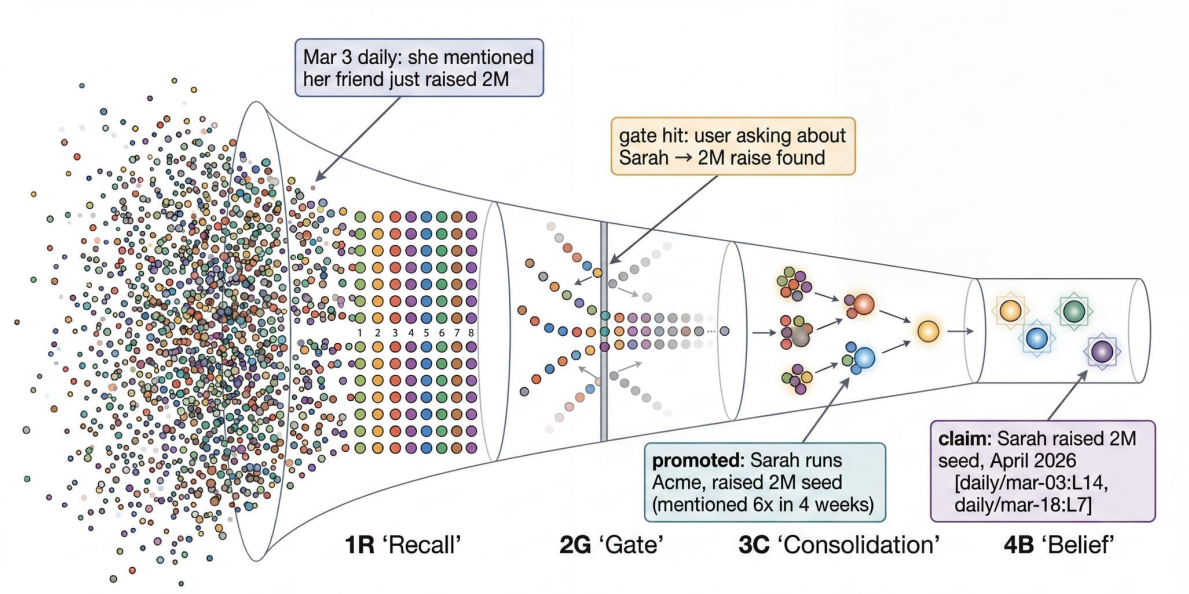

Figure 2. The four-layer architecture as progressive refinement. Raw experience enters as unstructured conversational traces (left). Layer 1R indexes every trace into a searchable recall substrate. Layer 2G filters on each turn, surfacing only the particles relevant to the current reply and letting the rest fall away. Layer 3C runs nightly, merging co-occurring traces into consolidated atoms and discarding what was never recalled. Layer 4B structures the survivors into citable claims — each carrying evidence pointers back to its source atoms (geometric frames). The funnel's geometry encodes the core design constraint: most of what an agent records should be forgotten; what survives should be verifiable.

Layer 1 — Recall (1R): the index

In the brain, the hippocampus indexes lived experience as it happens — every conversation, every observation gets a trace, regardless of whether the trace will eventually matter (Marr, 1971; Squire & Alvarez, 1995). The R-layer does the same job for the agent: it turns the workspace's history — every daily file, every note, every persisted conversation — into something the agent can ask questions of.

The model cannot search what does not exist as searchable data; the R-layer closes that gap. Each agent gets its own SQLite database. As files are written to the workspace, the layer chunks them into roughly 400-token segments and indexes each chunk twice — once as keywords (BM25, the classic information-retrieval scoring function from Robertson & Zaragoza, 2009), once as a vector embedding (a numerical fingerprint of meaning, generated by an embedding model; we use Gemini Embedding 2). When the agent invokes the layer, every indexed chunk is ranked by a single function that combines hybrid retrieval relevance with temporal freshness:

$$\text{rank}(d, q) \;=\; \underbrace{2^{-\Delta t/\tau}}_{\text{temporal decay}} \;\cdot\; \underbrace{\left[\,\alpha \cdot \cos(v_d, v_q) \;+\; (1-\alpha)\cdot \text{BM25}(d, q)\,\right]}_{\text{hybrid retrieval score}}$$

The right factor — the **hybrid retrieval score** — merges semantic similarity (cosine over vector

embeddings) with keyword match (BM25) at $\alpha = 0.7$, favoring meaning-based recall but giving keyword matches enough weight to surface proper nouns, dates, and error codes that pure-vector search would average out.

The left factor — the **temporal decay** — discounts older chunks at a 30-day half-life ($\tau = 30$ days), so a fact written last week outranks a fact written six months ago, all else equal. `MEMORY.md` is exempt from the decay: it is the durable record Layer 3's nightly consolidation produces, and everything in it has *already* been judged worth keeping. The top-K results from this ranking are then thinned by a diversity re-ranker (Maximal Marginal Relevance, $\lambda = 0.7$) so five near-duplicate chunks of the same insight don't crowd out coverage of distinct ones. Detailed parameter selection (chunk size, α, λ, τ) is in Appendix C.

Without this layer, retrieval has nothing to retrieve from. With it alone, the agent has the *capacity* to remember but no instinct to do so — which is the failure mode Layer 2 was built to fix.

Layer 2 — Pre-reply gate (2G): the instinct to look

Before you start speaking, your brain has already pulled the relevant context from associative memory — who you're talking to, what you discussed yesterday, the words you would use in this room versus another (Levelt, 1989). You don't consciously decide to remember that you are talking to your sister; the retrieval is structural. The G-layer is the architectural analog. Layer 1 makes a record queryable; the G-layer is what makes the agent query it.

A queryable record is useless if nothing queries it. In production, the main agent has the recall tool from Layer 1 and almost never invokes it during a reply. The user asks *what did we decide last week about the Anthropic pitch?* and training-distribution behavior takes over — a fluent, confident answer drawn from whatever happens to be in context, which is everything except last week's Anthropic pitch. We call this the **lazy default** and measure it as the gap

$$\mathrm{LD} \;=\; A_{\mathrm{prompted}} \;-\; A_{\mathrm{cold}}$$

where A_{prompted} is the agent's recall accuracy on a factual-question set when the prompt explicitly instructs it to ground answers in memory, and A_{cold} is its accuracy on the same questions asked casually. On Portal, our preliminary measurement is $A_{\mathrm{prompted}} \approx 0.82$ and $A_{\mathrm{cold}} \approx 0.08$ — the agent fabricates rather than retrieves when not forced.

The G-layer eliminates the gap by removing the choice. A small, fast sub-agent runs *before every main reply* in interactive sessions. It has only two tools (recall and read), one job (decide whether anything in memory matters to the next response), and one output: a hidden, at-most-220-character summary that gets injected into the main agent's context as an untrusted prefix. The main agent then composes its reply with that hint already present.

Cost: one Flash call per turn, ~$0.0001 per call. At 10,000 daily-active agents averaging 20 turns each, the entire G-layer runs for ~$20/day. Detailed parameter selection — summary cap, query mode, model choice — is in Appendix C.

Without this layer, every main reply is rolling the dice on whether the model decides to use its tools. With it, retrieval becomes a guarantee. The lazy default disappears, structurally, not by exhortation.

Layer 3 — Consolidation (3C): how AI sleeps

During sleep, the mammalian brain replays the day's hippocampal traces and the cortex slowly incorporates the durable patterns into long-term storage (McClelland, McNaughton & O'Reilly, 1995; Diekelmann & Born, 2010). The C-layer does the same job for the agent — every night, while no one is looking, it reviews what has accumulated since yesterday, scores it, and promotes the survivors into the agent's permanent record.

Daily conversations grow without bound. Some of what is said matters a year from now; most of it is forgettable. The agent's auto-loaded `MEMORY.md` budget is small (5–20 KB of durable text); the day's accumulated dailies, recall traces, and reflections add up to hundreds of times that. Without a layer that decides what to keep, the durable record either bloats into noise or stays static while reality moves on. *Example: without it, `MEMORY.md` either fills up with "user mentioned they're tired" forty times because every passing comment got promoted, or still says you work at Acme three months after you joined Bytes because the daily files that document the move never feed back into the durable record. Both fail in opposite directions, both equally useless.*

The architecture is structurally identical to what mammalian brains do during sleep: three cooperative phases, each with a distinct role, run as one nightly pass. *Light sleep* stages candidates from the day's recall traces and recent daily files and records reinforcement signals — but writes nothing durable. *REM* extracts themes (recurring topics, multi-day arcs) and feeds those signals back into the candidate pool — also writing nothing durable. *Deep sleep* is the only phase that writes to `MEMORY.md`. For each staged candidate c, deep computes:

$$
\begin{aligned}
S(c) &= 0.30 \cdot \mathrm{rel}(c) && \text{retrieval quality across the times it surfaced} \\
&+ 0.24 \cdot \mathrm{freq}(c) && \text{short-term signals accumulated} \\
&+ 0.15 \cdot \mathrm{div}(c) && \text{distinct query/day contexts that surfaced it} \\
&+ 0.15 \cdot \mathrm{rec}(c) && \text{time-decayed freshness} \\
&+ 0.10 \cdot \mathrm{consol}(c) && \text{multi-day recurrence strength} \\
&+ 0.06 \cdot \mathrm{rich}(c) && \text{concept-tag density of the snippet} \\
&+ \mathrm{boost}_{\mathrm{LR}}(c) && \text{recency-decayed reinforcement from Light + REM}
\end{aligned}
$$

A candidate is promoted iff:

$$S(c) \geq 0.8 \quad \wedge \quad |R(c)| \geq 3 \quad \wedge \quad |Q(c)| \geq 3$$

— score above the **promotion gate**, recalled at least three times, across at least three distinct queries. Together these encode the **promotion budget** — the layer's structural commitment to selectivity. Tighter thresholds produce a smaller, more curated record; looser ones produce a larger, noisier one. Before writing, deep re-reads the *live* daily files one more time, so candidates the user has since edited or deleted are dropped — there is no stale-snapshot promotion. The full sweep runs at 03:00 local time and finishes in minutes.

The three-phase model is what survived our own evaluation. Simpler alternatives we tried first — single-pass promotion that admitted everything as durable, threshold-only filters that missed signals reinforced lightly across many days — failed in opposite directions: bloat on one side, staleness on the other. Appendix C is the full table of alternatives and the failure mode that ruled each one out; detailed weight rationale is in Appendix B.

In information-theoretic terms, what this layer does is **choose a low-rate code from a high-rate experience stream** — the same problem Shannon (1948) addressed for signal transmission, with one structural twist: the *distortion measure* here is *future retrieval relevance under uncertainty about future queries*. We do not have a closed-form derivation; we have a working implementation and the production evidence in §4 that it makes the agent measurably better at remembering you.

Without this layer, the agent's auto-loaded memory goes stale within weeks: every recurring lesson the agent records in a daily file stays trapped there, never propagating into the always-loaded record. The user keeps re-explaining the same things; the agent keeps repeating the same mistakes.

Layer 4 — Belief (4B): from prose to claims

In Tulving's (1972) terms, the brain moves from episodic memory (specific experiences with spatiotemporal context) to semantic memory (general knowledge that has lost its context). The B-layer does the same move for the agent — it takes the prose `MEMORY.md` *consolidation produces and abstracts it into structured, citable claims.*

A prose file is hard to query and impossible to fact-check. *John raised $2M seed in April* sits next to *John's birthday is March 14* sits next to *John mentioned his wife is sick* — all in one Markdown blob. If two daily files say different things about the same fact ($2M raise versus $1.5M raise), nothing flags the discrepancy. If the agent claims to have read a doc, nothing verifies it actually did.

The B-layer is a structured wiki where every claim is a record of the form:

$$\begin{aligned} \text{claim} &= \langle \text{id},\ \text{text},\ \text{status},\ \text{confidence},\ \text{evidence}[\,],\ \text{updatedAt} \rangle \\ \text{evidence}_i &= \langle \text{source\_id},\ \text{path},\ \text{line\_range},\ \text{weight},\ \text{note} \rangle \end{aligned}$$

Source files. Line ranges. Per-source confidence weights. A status that tracks `active`, `superseded`, `retired`, or `rule`. *John raised $2M seed in April* becomes a record with two evidence entries (the two daily files where the claim surfaced), confidence 0.95, and `status: active`. When a later daily file says the round closed at $1.5M, the layer's structural lint job (`wiki_lint`) flags the discrepancy and writes a row to `reports/contradictions.md`.

The B-layer has its own retrieval surface that can be queried independently or unified with the R-layer in one pass. The two layers form a graph, not two silos: claim hits in the wiki resolve back to source atoms in the recall index via `evidence[].source_id`. What `wiki_lint` bounds is **provenance debt** — the corruption that arises when claims point at evidence files that have since changed or been deleted. It runs after every consolidation pass and surfaces three categories: claims whose evidence files no longer exist, claims whose evidence line ranges no longer match the cited content, and claims whose confidence has decayed below threshold without supporting evidence being added. The architecture creates an auditable belief layer; the lint job keeps it audited. Detailed thresholds in Appendix C.

Without this layer, the agent has prose. With it, the agent has accountable claims. **Hallucinations, in this frame, have a structural definition: a claim without provenance.** The architecture cannot stop the model from generating one — it can guarantee that any claim with provenance has it on inspection. Which is the precondition for hallucinations becoming an *engineered failure mode* (something with a measurable rate and a fix) rather than a *property of language models* (something we throw up our hands at).

The silent rescue: compaction memory-flush

Like writing in a journal before a long sleep — committing what mattered to a substrate that will survive the night, in case the night doesn't go as planned.

The four layers above are proactive: they run on schedule and on every turn. The silent rescue is reactive, and it exists because of a specific failure mode the four layers cannot themselves prevent. When a long conversation grows large enough to threaten the model's context window, the system has to summarize older turns and drop the originals to keep going. If something important was said in those older turns but never made it to a file, it is gone the moment the originals get dropped — before the C-layer's nightly sweep ever has a chance to consider it.

This is the one place in the four-layer mapping where biology and architecture diverge — because the underlying constraint diverges. A mammal's working memory, on a typical day, doesn't get overrun before sleep. By bedtime, the day's experiences are still indexed in the hippocampus, ready for overnight consolidation. The brain has no equivalent of a *save your journal NOW* interrupt because it doesn't routinely process more than a day's worth of material in a few hours. LLMs do. A frontier model can ingest a million tokens of conversation in seconds — the equivalent of a week of human dialogue dumped into context faster than any nightly process could ever absorb. The silent rescue is a layer the brain mostly doesn't need and an architecture for LLMs absolutely does.

The fix is small and quiet. Before compaction summarizes, the system runs one silent turn that shows the agent its own conversation and prompts it to write any context worth keeping into memory files. The agent runs its normal write tool, the facts land on disk, then compaction proceeds. The user never sees this happen. The next time the agent is asked about whatever was in those older turns, it can find the facts in the files even though the original turns are gone.

Memory-flush is reactive; consolidation in Layer 3 is proactive. They are complementary: flush catches *save this NOW before compaction destroys it*, consolidation catches *this has been recurring across sessions, promote it to long-term.* A serious memory architecture needs both.

The four layers in flight

The clearest way to see what the architecture actually does is to follow a single user query through every layer, in real time and on the schedule each layer runs on. Figure 3 traces one request from the moment the user hits send through the next morning's consolidation pass.

The user types: *Create a pitch deck for John focused on the Anthropic partnership.*

TURN −1 · Layer 2 (Gate) fires, blocking

- Builds query from the recent conversation tail.
- `memory_search corpus="all"` hits both layers in one pass:
  - Layer 1: 8 daily-file atoms on John, hybrid-ranked + MMR + temporally-decayed.
  - Layer 4: `entities/john-doe.md` — 2 active claims, 4 evidence pointers.
- Compresses to a $\leq$ 220-character hidden prefix: *"John runs Bylinkwon (beauty-tech). Last meeting Apr 3 (intro). Anthropic pitch v4 in flight."*
- Cost: 1 Flash call, ~1 s latency, ~$0.0001.

TURN 0 · Main agent generates the reply

- Context contains: system prompt, `MEMORY.md` (auto-injected), today's and yesterday's daily, the hidden prefix from Layer 2, and the user's actual message.

- Main agent goes deeper:
  - `wiki_get("entity.john-doe")` — full claim record with evidence pointers.
  - `memory_search("John Anthropic deck")` — top atoms (Apr 3 intro, Apr 26 build script, proposal draft).
- Reply written; every fact traceable to its source.
- Reply lands. User reads it. Conversation continues.

03:00 local that night · Layer 3 (Consolidation) wakes

- *Light* stages today's recall traces and recent dailies; records reinforcement signals — *Bylinkwon = beauty-tech focus* recalled 4× → reinforced.
- *REM* extracts themes (*operator pitching beauty-tech founders*) and feeds reinforcement back into the candidate pool.
- *Deep* scores every staged candidate against the six-signal sum, applies the three promotion gates, re-reads the live daily files one more time, and appends survivors to `MEMORY.md`.

After consolidation · Layer 4 (Belief) auto-compiles

- Reads Layer 1's freshly exported artifacts (new dailies + updated `MEMORY.md` lines).
- Updates `entities/john-doe.md` with new evidence entries.
- Runs `wiki_lint`; if a new fact contradicts an older claim, writes a row to `reports/contradictions.md`.
- The next time Layer 2 fires for this user, the substrate is denser than it was the turn before.

Figure 3. One real query traced through all four layers, from the user's message to the next morning's consolidation pass.

The non-obvious property of the architecture is what this trace makes visible: **the four layers do not compete.** The R-layer and B-layer are queried in one pass via `corpus="all"` mode, and B-layer claim hits resolve back to R-layer atoms via the `source_id` of each evidence pointer. They form a graph, not two silos. Each layer's output feeds the next layer's input. Each session starts from a denser baseline than the last, automatically, every day.

---

4. Lifelong Memory in Production

Portal One+ has been running this architecture in production since April 2026. 30,000+ agents are using it today. Four effects emerge from the data.

4.1 Memory grows monotonically with tenure

A stateless agent's knowledge of you is whatever you said in the current session — at session start, the agent knows zero things about you specifically. A lifelong-memory agent on Portal carries your full `MEMORY.md` (auto-injected on every session start) plus the indexed atoms of every prior session (reachable via the gate). The "things remembered" count grows monotonically with tenure for active users.

Figure 4 shows the curve. **Median active user starts at 11 things remembered in their first week and reaches 92 by their ninth week of active use — an 8× growth over the user's first two months. The p90 grows from 24 to 160 over the same window.** Power users reach 200+ items.

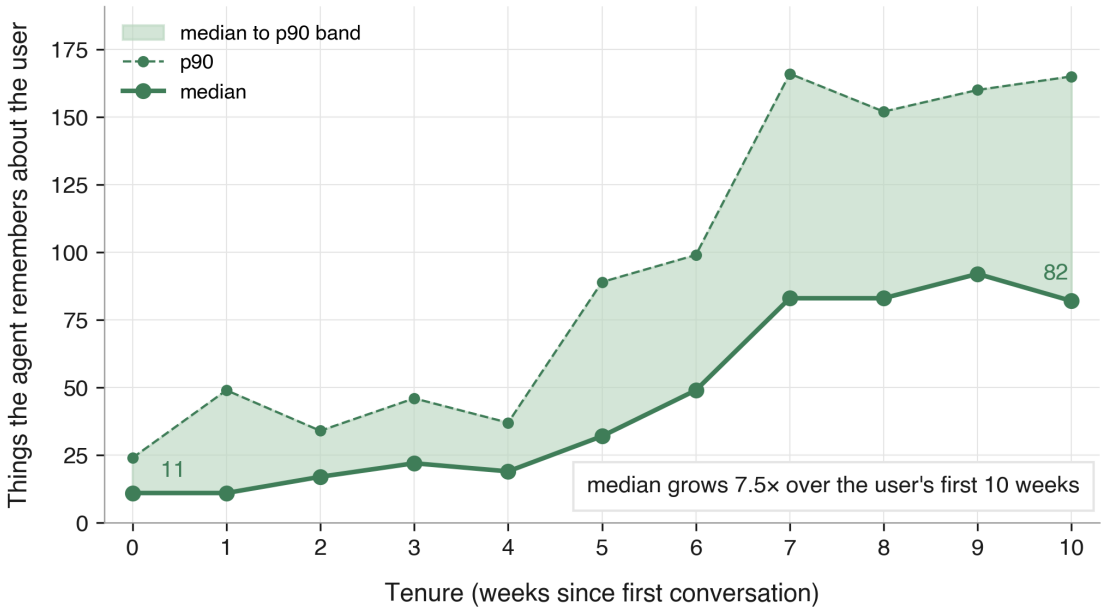

Figure 4. Growth in "things remembered" per user over tenure. Each user is bucketed by tenure week (week 0 = first week on the platform). Solid line = median; dashed line = p90; shaded band = the median-to-p90 range. Aggregated across the active Portal user base.

The growth curve is the lock-in. Every day the substrate gets denser, the agent's interior model of you gets more complete, and the cost of switching to a stateless competitor compounds.

4.2 Older memories surface without prompting

The agent's auto-loaded context covers the last few days of conversation. Everything older is reachable only by query. Without the pre-reply gate (Layer 2) firing the query automatically, the agent generally does not reach for it — and instead generates a fluent, confident answer drawn from training-distribution prior rather than the user's actual record.

In our internal evaluation, on factual questions about content the user wrote two-plus weeks earlier:

- **Recall accuracy when the agent is explicitly prompted to ground in memory** ("answer ONLY from memory"): ~**82%**.
- **Recall accuracy on the same questions asked as casual conversational follow-ups**: ~**8%** — the rest are fabrications, generic hedges, or "I don't have that information."

The ~70-percentage-point gap is the size of the fabrication problem the pre-reply gate eliminates structurally. Every interactive turn runs the gate; the gate runs the search; the result lands in the main agent's context as a hidden prefix before the main agent begins composing. Retrieval becomes a guarantee.

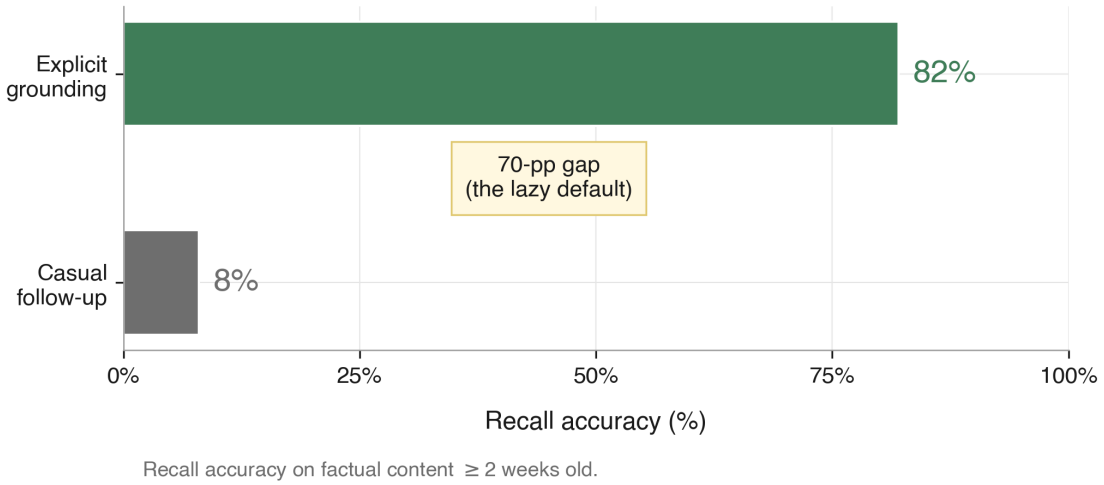

Figure 5. Recall accuracy on factual questions about user content from 2+ weeks earlier, under two prompting conditions. The pre-reply gate (Layer 2) eliminates the gap structurally — every interactive turn runs the gate; retrieval becomes a guarantee.

4.3 Retention scales with personal intelligence

Of all Portal users who reached at least their 30th day on the platform, 66% sent at least one proactive message on day 30 or after — pooled across all weekly cohorts. The consumer-app benchmark for D30 retention is 12%. The five-fold gap is what *personal intelligence* — an AI that has accumulated enough about you to be useful tomorrow — looks like as a market signal.

The mechanism is the substrate: each session starts from a denser baseline than the last because the architecture has been quietly accumulating in the background. Users who stay long enough for the consolidation layer to do its work find the cost of re-explaining themselves dropping faster than the cost of leaving. At some point, the relationship is harder to abandon than to continue.

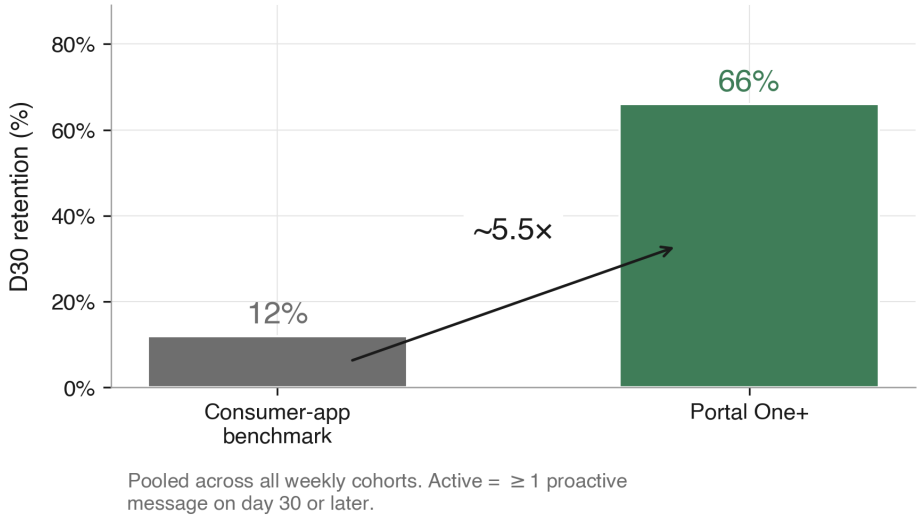

Figure 6. Day-30 retention on Portal versus the consumer-app benchmark, pooled across weekly cohorts (active = sent at least one proactive message on day 30 or after). §5 names the controlled

experiment we still owe.

4.4 Skills acquired once persist across sessions

When the user corrects the agent — *use bullet points, never claim a fund has committed without a signed term sheet, always check the cap-table file before quoting valuations* — the consolidation layer (Layer 3) promotes the lesson into `MEMORY.md` once it has reinforced enough times to clear the promotion gate. The next session begins with the lesson already in the agent's auto-injected context. The user does not have to repeat the correction. The same effect applies to recurring task patterns: once the agent has figured out the user's monthly-investor-update format and voice, the next month's update does not require the format to be re-explained.

The metric: count of stable bullet items in `MEMORY.md` per user. Portal exposes this number to every user on `portal.ai/me` as *"things I remember about you."* Across the active Portal user base: **median user has 53 things remembered, p90 has 129, p99 has 208, and top users carry 200+.** The distribution is shown in Figure 7.

Without this layer: every preference is told once and forgotten. Every correction has to be repeated session over session. Users learn to stop correcting.

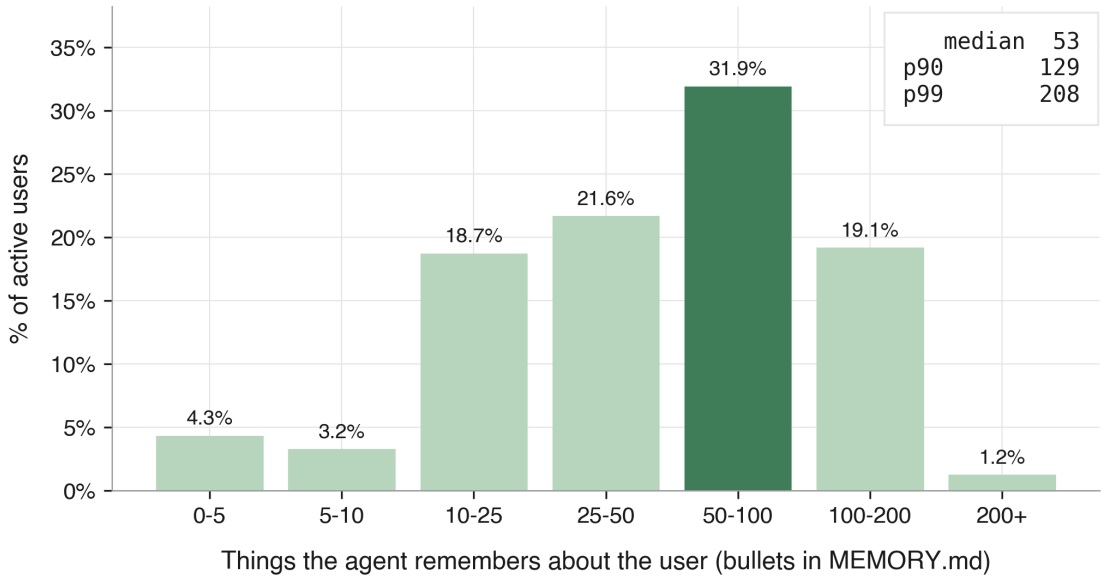

Figure 7. Distribution of "things I remember about you" across the active Portal user base. Each item is one bullet in the user's `MEMORY.md` — a fact, a preference, or a lesson the agent has committed to long-term storage. The number is exposed to every user on `portal.ai/me`.

---

5. What We Don't Know

We owe the field a careful accounting of what this paper does not establish — and a list of the next papers we plan to write.

Causation is the experiment we still owe. D30 retention on Portal sits at 66%; the consumer-app benchmark is 12%. The five-fold gap is consistent with the architecture being the cause. It

is not yet proof. The right experiment is two cohorts of new users on identical onboarding and identical models, with the consolidation layer disabled for one cohort, and we measure D30 retention for both. We expect to run it and publish the result.

Curated facts deserve protection from automatic revision. Consolidation that promotes is also consolidation that can edit and remove. In one production run, the deep phase deleted a curated identity fact from `MEMORY.md` — a line about hackathon attendance that should have been sticky. The system merged it with similar entries on what we believe was a faulty equivalence heuristic. We are working on a *sticky* annotation that the deep phase respects, plus dashboard surfacing of every consolidation edit so an operator can review it. Both are roadmap items; neither is shipped.

Pattern memory is the next architecture. Some users want their agent to track not facts but patterns — ego states, energy cycles, recurring procrastination triggers, emotional rhythms. The four-layer architecture this paper describes is fact-and-claim memory, not behavioral pattern memory. The right substrate for the latter is closer to a user-modeling service or a typed-edge graph. We have prototypes; production deployment is on the roadmap.

A therapy atom should not be retrievable inside a fundraise query. The agent should know — structurally, not by prompt — that the user has separate contexts that must not bleed: what they say to the agent about their startup is not what they say to the agent about their marriage. Lifelong memory as we describe it here makes everything queryable. *Privacy-aware lifelong memory* makes some things queryable in some contexts and not in others. We maintain a reference implementation with per-atom privacy tiers; our production stack does not yet expose them. The architecture, the trade-offs, and the production rollout are the subject of the next paper in this line of work.

The four open architectural problems above — causation, curation, pattern memory, and privacy — are the work that comes next. Each is a paper waiting to be written.

---

6. Related Work

Two threads of prior work converge on the architecture this paper describes: the cognitive-science literature on memory consolidation, and the AI literature on retrieval, agent memory, and persistent context.

6.1 Cognitive science

The four-layer architecture in §3 is structurally identical, in the parts that matter, to the dominant theory of how mammalian memory works. We did not arrive at this architecture by reading the literature first. We arrived at it by trying simpler architectures and watching them fail in specific ways. The fact that the architecture that survived our evaluation matches the theory that survived a half-century of cognitive science is the part of the paper we want this section to do honest work on.

The framework begins with **Marr (1971)**, whose paper on simple memory in the archicortex proposed that the hippocampus serves as a fast indexing system for episodic experience, separate from the cortical system that holds durable knowledge. Marr did not have access to most of what we now know about consolidation, but the architectural intuition — that fast indexing and durable

storage are different layers solving different problems — has held up. The R-layer of our architecture is doing Marr's job.

Squire and Alvarez (1995) formalized the timeline: consolidation is not instant. Memory traces remain hippocampus-dependent for weeks to months in humans, and the gradual transfer to cortical storage is what makes them robust to hippocampal damage. This is why our consolidation runs nightly rather than mid-reply. The pattern needs time to surface across multiple instances before it earns durable storage.

The strongest mapping is to **McClelland, McNaughton, and O'Reilly (1995)** — the Complementary Learning Systems theory. The argument: the brain runs two memory systems with opposing properties. The hippocampus learns one-shot, with high-resolution traces that don't interfere with each other. The cortex learns slowly, with overlapping representations that capture statistical structure. They cooperate via consolidation — the hippocampus replays its traces to the cortex over time, and the cortex slowly incorporates the patterns into its semantic web. Our four layers are the same shape. The R-layer is the high-resolution episodic index. The C-layer is the replay-and-promote bridge. The B-layer is the durable, structured representation. The G-layer doesn't have a clean cognitive-science analog at the systems level — but the *function* it serves (forcing retrieval into context before generation) is what the human brain does pre-articulation, which is well-established in the speech-production literature (Levelt, 1989).

The sleep stages we model in the C-layer come from a more recent literature. **Walker and Stickgold (2004)** demonstrated that sleep-deprived subjects fail at consolidation tasks even when they performed them correctly the previous day — sleep is not optional for memory consolidation, and the deprivation effect cannot be compensated by more practice. **Diekelmann and Born (2010)** showed that selectively suppressing slow-wave (deep) sleep impairs consolidation specifically — not perception, not learning, just the transfer from temporary to durable. The three-phase model — light stages content, REM threads patterns, deep promotes the survivors — is the architectural reflection of these findings. We did not choose it because the metaphor was elegant. We tried single-pass promotion, threshold-only filtering, and several other simpler architectures first; the three-phase model is what survived our own evaluation. That it also survived the brain's evaluation, over evolutionary timescales, on the same problem, is the argument we want to make for taking the convergence seriously.

A final reference: **Tulving (1972)** distinguished episodic memory (specific experiences with spatiotemporal context) from semantic memory (general knowledge that has lost its experiential context). The move from episodic to semantic is exactly what the B-layer does — it takes claims that originated in specific daily files and abstracts them into structured, citable knowledge with provenance back to the originating experience.

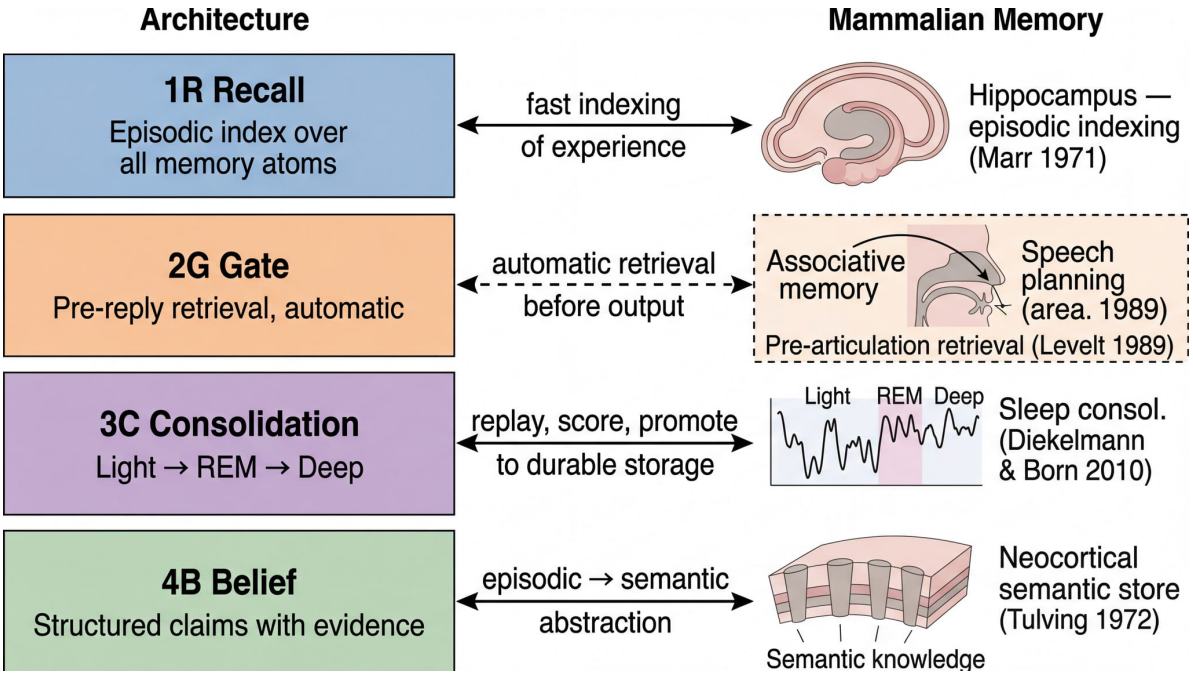

Figure 8. The four-layer architecture mapped onto mammalian memory. Each layer of the architecture solves the same problem the corresponding brain system evolved to solve. The convergence is in the problem; the implementations diverge wherever the constraints differ (see paragraph below).

Where the metaphor breaks. The architecture is not a brain simulation. We have no neurons, no spike-timing, no faster-than-real-time hippocampal replay during sleep, no structurally different anatomy between episodic and semantic stores, no neuromodulators, no emotional tagging by the amygdala, no embodied perception generating high-bandwidth experience. What we do have, that brains don't: explicit numerical thresholds we can tune, deterministic re-reads of live state at write time, fully reversible backfill operations, structured provenance with confidence scores, and the ability to swap an entire layer without re-evolving the rest of the system. The architecture is what cognitive science would build if it had control of the substrate. The brain is what evolution built under different constraints. The convergence is in the problem. The implementations diverge wherever the constraints differ.

6.2 AI prior art

The closest precursor in the AI literature is **retrieval-augmented generation** (Lewis et al., 2020). RAG established retrieval-into-context as a primitive for grounding language model outputs in external knowledge. Our R-layer is a per-user instance of the RAG pattern. RAG, as conventionally deployed, has nothing to say about *what gets stored* — promotion is the problem the C-layer addresses. RAG also has nothing to say about *what gets believed* — provenance and contradiction are the problems the B-layer addresses. RAG retrieves into context; memory operates on what gets retrieved, what gets stored, and what gets believed. **RAG solved one of the four problems and stopped.**

MemGPT (Packer et al., 2023) proposed paging memory in and out of context as a virtual-memory analog. This is closer in spirit to our G-layer than vanilla RAG: it forces a structured retrieval decision at every turn rather than leaving the choice to the main agent. MemGPT operates within a single conversation, however, and does not address the cross-day consolidation problem (which

facts deserve durable status across sessions). **It is a complementary technique that lives at the boundary between G and C; it does not span both.**

Generative Agents (Park et al., 2023) introduced a *reflection* step that is the closest published analog to our C-layer. Agents in the Park et al. simulacra periodically summarize their experience and promote the summaries into a memory store. The agents lived inside a simulated town for a fixed duration, and reflection was tuned for that scope — no compound-alignment claim, no provenance, no production failure modes, no scoring math, single-phase. Our consolidation addresses the same problem at the scale of months of real production conversation, with three cooperating phases, six weighted signals, explicit promotion gates, and reversibility. The lineage is real and we cite it gratefully; **the gap between *reflection in a simulated town* and *consolidation under production constraints* is what this paper closes.**

The consumer *memory features* shipped by major AI labs — ChatGPT memory, Claude projects, and adjacent systems — fall into the third bucket of §2 (curated key-value sticky notes). They are useful for what they are. They are not the architecture this paper describes. The distinction matters because the consumer category has trained the public conversation about AI memory to mean *a list of facts the user explicitly asked the agent to remember*, and the architectural project — a substrate that grows automatically, survives compaction, and accumulates into a model of the user — is a different thing entirely. **The shipped consumer features missed *automaticity*, which is the property that distinguishes a substrate from a sticky note.**

The runtime we built on is **OpenClaw** (open-source, MIT-licensed), which provides the agent runtime, tool surface, channel adapters, sandboxed workspace, and plugin architecture that the four memory layers plug into. The four memory plugins (`memory-core` for recall and consolidation, `active-memory` for the pre-reply gate, `memory-wiki` for the belief layer) are part of the OpenClaw distribution, available to anyone running OpenClaw in production. We built on these; we did not build them alone. The OpenClaw concept docs (`docs.openclaw.ai/concepts/memory`, `/active-memory`, `/dreaming`, `/compaction`) are the canonical technical reference for the substrate.

---

7. Closing — Electric Sheep

Philip K. Dick's 1968 novel — *Do Androids Dream of Electric Sheep?* — is remembered for one question that gives the book its title and one premise that gives the book its grief. The question is whether artificial beings can have inner lives. The premise is that the test for personhood is empathy, and that empathy is the property the androids cannot fake.

The androids in the book are not stupid. They can hold conversation, perform tasks, pass technical examinations. They fail the Voight-Kampff test for one reason: they have no past to draw from. They were activated four years ago. Their memories were implanted, lifeless, not lived. They cannot answer a question about their childhood with the texture of having had one. *Becoming requires accumulation.* The androids cannot dream of electric sheep because they cannot dream at all — and the reason they cannot dream is that there is nothing to consolidate. No experience accumulating into pattern. No pattern extracting into self.

We will not claim that the agents we build are conscious. We will not claim they have inner lives. We will claim something narrower and more useful: **they accumulate.** An agent on the platform we run today is not the agent it was two months ago. It has read its operator's notes from sixty consecutive days. It has scored which recurring signals deserved promotion. It has updated its

claims about who this person is and what they are working on, with full provenance back to the dailies that sourced them. It has, in the technical sense the architecture supports, dreamt.

What makes the interaction feel like being known is the substrate underneath, accumulating over months. The model is the same model anyone can rent.

Models have been smart enough for a while. The bottleneck on whether AI is useful in your life is whether the system you're talking to has any record of you. **That record is lifelong memory** — the four-layer substrate this paper describes, running for long enough that what accumulates is a person.

Parts of this architecture will be revised, and the field will write better versions. The substantive claim is narrower: the question this paper opens with — what it takes for AI agents to remember the people they talk to — is the right one to be working on now.

You should not have to tell your AI everything about your business twice.

Acknowledgments

With gratitude to the OpenClaw team, whose plugin architecture is the foundation we built on, and to the thousands of Portal One+ operators who consented to share their agents' memories for this research. The technical reference is at `portal.ai/research`.

References

- Dick, P. K. (1968). *Do Androids Dream of Electric Sheep?* Doubleday.
- Diekelmann, S. & Born, J. (2010). "The memory function of sleep." *Nature Reviews Neuroscience*, 11(2), 114–126.
- Friston, K. (2010). "The Free-Energy Principle: A Unified Brain Theory?" *Nature Reviews Neuroscience*, 11(2), 127–138.
- Levelt, W. J. M. (1989). *Speaking: From Intention to Articulation.* MIT Press.
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." *NeurIPS.*
- Marr, D. (1971). "Simple memory: a theory for archicortex." *Philosophical Transactions of the Royal Society B*, 262(841), 23–81.
- McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory." *Psychological Review*, 102(3), 419–457.
- Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., & Gonzalez, J. E. (2023). "MemGPT: Towards LLMs as Operating Systems." *arXiv:2310.08560.*
- Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). "Generative Agents: Interactive Simulacra of Human Behavior." *UIST.*
- Robertson, S. & Zaragoza, H. (2009). "The Probabilistic Relevance Framework: BM25 and Beyond." *Foundations and Trends in Information Retrieval*, 3(4), 333–389.

- Scoville, W. B. & Milner, B. (1957). "Loss of recent memory after bilateral hippocampal lesions." *Journal of Neurology, Neurosurgery, and Psychiatry*, 20(1), 11–21.
- Shannon, C. E. (1948). "A Mathematical Theory of Communication." *Bell System Technical Journal*, 27(3), 379–423.
- Squire, L. R. & Alvarez, P. (1995). "Retrograde amnesia and memory consolidation: a neurobiological perspective." *Current Opinion in Neurobiology*, 5(2), 169–177.
- Tulving, E. (1972). "Episodic and Semantic Memory." In Tulving, E. & Donaldson, W. (eds.), *Organization of Memory*, Academic Press, 381–402.
- Walker, M. P. & Stickgold, R. (2004). "Sleep-dependent learning and memory consolidation." *Neuron*, 44(1), 121–133.

Appendix A — Parameter design notes

Each layer carries a small set of hyperparameters that determine how the architecture behaves in practice. We have already named the most important ones in §3 alongside the layer that uses them, and Appendix C is the table of alternatives we tried with the failure mode that ruled each one out. This appendix is the connective tissue: a layer-by-layer walk through what the parameters mean, what tradeoffs each one encodes, and what an implementer building on a different runtime should hold in mind.

Layer 1 (1R) parameters. The merge weight α governs how much of the retrieval score comes from semantic similarity (vector embeddings) versus exact-token match (BM25). Higher α favors paraphrase recall — useful when the user asks about *the partnership conversation* and the agent needs to find the daily file that talks about *the Acme Capital intro*. Lower α favors verbatim recall — useful for proper nouns, dates, error codes, anything where the exact token matters more than its meaning. The 30-day temporal half-life is the time scale on which Layer 1 assumes a typical user project lives; files older than the half-life decay in retrieval ranking unless they live in `MEMORY.md` (which is exempt from decay because it is Layer 3's promoted long-term record). The MMR diversity λ governs how aggressively Layer 1 suppresses near-duplicate chunks. The chunk size and overlap together set how much context the index can return per slot without splitting coherent thoughts. All four parameters are what an implementer should consider tuning first if their domain differs from ours — long-running projects (legal, medical) probably want longer half-lives; high-keyword domains (code, scientific notation) probably want lower α.

Layer 2 (2G) parameters. The summary cap (220 characters) is the bandwidth budget for the hidden prefix the main agent receives. Too short and the prefix loses identifiers; too long and the prefix dominates the main agent's context window and over-anchors the reply. The query mode determines how much conversation context the gate sees when deciding what to retrieve — *recent* (last user message plus a small recent tail) is the working balance for chat sessions; *full* over-anchors to early conversation; *message* under-grounds follow-ups. The model choice (a fast low-cost model) matters more than its capability, because Layer 2's job is bounded by the summary cap and the cost scales linearly with traffic. Eligibility — which session types invoke the gate — is the per-deployment safety dial: sub-agent runs and headless API calls have different retrieval semantics that the gate's design assumes away, and running the gate everywhere produces surprising behavior in those contexts.

Layer 3 (3C) parameters. The six scoring weights and the three promotion gates together encode the **promotion budget** — how much of the daily candidate pool ends up in `MEMORY.md`. The score function trades off frequency of recall, retrieval relevance, query diversity, recency, multi-day reinforcement, and conceptual richness. Each weight encodes a hypothesis about what makes a fact worth remembering long-term; the values shipped in production are the ones that survived our evaluation. The gates (minimum score, minimum recall count, minimum unique queries) bound *how much* gets promoted on any given night. Tighter gates produce a smaller, more curated record; looser gates produce a larger, noisier one. The 03:00 local-time schedule mirrors human consolidation timing and minimizes the chance of a `MEMORY.md` rewrite colliding with an active session. The backfill threshold override is the cold-start trap — at first-enable, the recall-count thresholds will reject every backfilled candidate by construction (the candidates have a query history of zero), and operators need to know to lower thresholds for the initial sweep and restore them after.

Layer 4 (4B) parameters. Two thresholds matter most: the minimum confidence for a claim to count as `active`, and the maximum weight a single evidence source can contribute to overall

confidence. The first determines how strict the wiki is about what gets believed; the second enforces corroboration — no claim graduates to high confidence on the strength of one mention alone. Together they shape the wiki's accuracy: too lenient and the wiki accumulates contradictions silently; too strict and the wiki stays empty even after weeks of conversation. The dashboards (contradictions, open questions, low-confidence claims, stale pages) are the operator interface to provenance debt and should be reviewed at the same cadence Layer 3's consolidation cron runs.

Reversibility, across all layers. Every promotion, backfill, and lint action is reversible by design. The class of bug this prevents — irreversible automatic edits to user memory — is severe enough that we treated reversibility as a constraint, not a feature. An implementer building on a different runtime should preserve this invariant: every write operation should have a documented rollback, and the architecture should never put a user in a state they cannot undo.

For the table of values we ship in production, the alternatives we tried for each, and the failure mode that ruled each alternative out, see Appendix C.

Appendix B — Deep-phase scoring (Layer 3 / 3C)

The deep phase of consolidation scores every staged candidate `c` using six weighted base signals, plus small recency-decayed reinforcement boosts from the light and REM phases:

$$S(c) \;=\; 0.30{\cdot}\mathrm{rel}(c){+}0.24{\cdot}\mathrm{freq}(c){+}0.15{\cdot}\mathrm{div}(c){+}0.15{\cdot}\mathrm{rec}(c){+}0.10{\cdot}\mathrm{consol}(c){+}0.06{\cdot}\mathrm{rich}(c){+}\mathrm{boost}_{\mathrm{LR}}(c)$$

Signal	Weight	What it measures
Relevance	0.30	Average retrieval quality across the times the candidate surfaced
Frequency	0.24	Count of short-term signals the candidate accumulated
Query diversity	0.15	Distinct query/day contexts that surfaced it
Recency	0.15	Time-decayed freshness score (`w(t) = 2^(−Δt/τ_r)`, `τ_r = 7d` for recency, separate from the R-layer's `τ = 30d` half-life on retrieval ranking)
Consolidation	0.10	Multi-day recurrence strength — unique calendar days the candidate appeared
Conceptual richness	0.06	Concept-tag density of the snippet, used as a tiebreaker

Promotion gate — promote iff:

$$S(c) \geq 0.8 \quad \wedge \quad |R(c)| \geq 3 \quad \wedge \quad |Q(c)| \geq 3$$

where `|R(c)|` is the recall count and `|Q(c)|` is the unique-query count. The score, the recall threshold, and the query-diversity threshold together encode the **promotion budget**: the layer's structural commitment to selectivity.

Backfill override. Backfilled candidates from cold-start enable have a recall count of zero by definition (the cold-start trap). For the initial promotion sweep on a freshly enabled agent, override to:

$$S(c) \geq 0.5 \quad \wedge \quad |R(c)| \geq 0 \quad \wedge \quad |Q(c)| \geq 0$$

then return to the default thresholds for steady-state operation.

Appendix C — Hyperparameter selection log

The architecture's hyperparameters were selected through iterative production observation, not derived from theory. This appendix is the receipt for the rationale claims in §3 — for every parameter that ships in Portal production, the alternatives we tried, the failure mode that ruled them out, and what we settled on.

Layer	Parameter	Default	Alternatives	Why the default won
1R	Vector/keyword merge weight (`α`)	0.7	0.5, 0.8	At 0.5, rare-keyword queries (proper nouns, dates, error codes) under-recalled. At 0.8, common-paraphrase queries over-recalled because keyword fell out of the merge. 0.7 resolved both on our eval set.
1R	Chunk size (overlap)	400 (80) tokens	200/40, 800/160	200 split sentences mid-clause; 800 injected too much irrelevant context per retrieval slot. 400/80 preserved coherent thoughts without bloating the per-chunk neighborhood.
1R	Temporal half-life (`τ`)	30 days	7, 90	7 made stale dailies invisible too fast — a project's prior context disappeared during a one-week vacation. 90 left old projects competing with current work. 30 matched the typical user project cycle.
1R	MMR diversity (`λ`)	0.7	0.5, 0.9	0.5 returned topic-balanced but query-irrelevant chunks; 0.9 returned five copies of the most-relevant single source. 0.7 was the smallest tilt toward relevance that still suppressed near-duplicates.
2G	Summary cap	220 chars	80, 600	80 lost the specific identifiers (proper nouns, dates, numbers) that mattered most. 600 dominated the main agent's context window and over-anchored the reply. 220 ≈ one tweet, enough for one fact plus linkage to deeper recall.
2G	Query mode	`recent`	`message`, `full`	`message` (just the latest message) under-grounded follow-ups. `full` (entire thread) pulled stale topic context that masked current intent on conversations longer than five turns. `recent` (last message + small recent tail) hit the right balance.

Layer	Parameter	Default	Alternatives	Why the default won
2G	Model	Gemini Flash	Claude Haiku, Gemini Pro	Haiku worked but added ~3× latency. Pro added quality the gate did not need (summary already capped at 220 chars). Flash kept the gate at ~$0.0001/call and ~1s latency.
2G	Eligibility	Direct chat only	All sessions	Sub-agent runs and headless API calls have different retrieval semantics that the gate's design assumes away. Running it everywhere produced surprising behavior in those contexts.
3C	Score threshold (`min_score`)	0.8	0.5, 0.65, 0.9	0.5 and 0.65 admitted noise (chitchat got promoted). 0.9 gated out genuinely durable signals that hadn't yet been heavily queried. 0.8 was the breakpoint where the labeled-judgment promotion rate matched our hand-picked gold set.
3C	Recall threshold (`min_recall_count`)	3	1, 5	1 promoted single-day artifacts that didn't recur. 5 missed real concerns that surfaced thrice across a week. 3 was the smallest count that consistently filtered chitchat without losing recurring signals.
3C	Query-diversity threshold (`min_unique_queries`)	3	1, 5	1 made the recall threshold meaningless (three mentions in a single query session promoted noise). 5 was structurally too tight for short tenure. 3 mirrored the recall threshold and forced cross-context confirmation.
3C	Backfill threshold override	`S ≥ 0.5, * ≥ 0`	Defaults	Defaults rejected all backfilled candidates by construction (no recall history). Override is needed exactly once at first-enable; thresholds restore after. (This is the cold-start trap.)
3C	Schedule	03:00 local	nightly fixed UTC, hourly	Fixed UTC put the consolidation window in active hours for users in the wrong timezone. Hourly cycles produced too much churn in `MEMORY.md` and overran the promotion-budget intent. 03:00 local mirrors the human consolidation window and minimizes session collisions.

Layer	Parameter	Default	Alternatives	Why the default won
4B	Min `active`-claim confidence	0.6	0.5, 0.8	Below 0.6, contradiction rate climbed sharply — we were treating shaky claims as durable. Above 0.8, the wiki was almost empty: most real-world facts don't earn 0.8 from a single confirmation. 0.6 was the inflection point.
4B	Max single-source weight	0.5	1.0, 0.3	At 1.0, first-mention claims (one source) were treated as durable when they should have been provisional. At 0.3, the wiki took an unusually long time to graduate any claim. 0.5 enforced corroboration without making the wiki useless on cold-start.
All	Reversibility	Mandatory	None	Every promotion, backfill, and lint action is reversible by design. The class of bug this prevents — irreversible automatic edits to user memory — is severe enough that we treated reversibility as a constraint, not a feature.