Long-Term Memory for a Conversational Assistant

A product and architecture paper on how ATAPIC remembers

Enahoro Omoarukhe

Founder & Lead AI Engineer, ATAPIC

TECHNICAL WHITE PAPER • v1.0 • JUNE 2026

ABSTRACT

A conversational assistant with no memory starts cold every time. It cannot honor a preference you stated last week, cannot recall the goal you set last quarter, and forces you to re-explain your business in every session. The obvious fix, writing down whatever the user says, fails in the opposite direction: it fills the store with raw messages and one-off requests, so the assistant is now confidently personalized on noise. This paper describes a memory layer built to avoid both failures. Every conversational turn is read by a small language model that decides whether anything durable was actually said, distills it into a clean normalized fact, and deduplicates it against what is already known before saving. Memories are retained for a year, indexed as embeddings, and recalled two ways: the most recent facts are always in context, and older facts are surfaced by topical similarity to the question being asked, so the assistant can draw on something it learned a year ago the moment it becomes relevant. The user can see, edit, forget, import, and export everything, the same controls leading assistants now offer. We define the memory object, explain each stage of the write and read paths in depth, situate the design in the research lineage from retrieval-augmented generation to vector search, and argue why distillation, not transcription, is the right basis for machine memory.

1. Motivation: an assistant that forgets is an assistant you re-explain

The value of a conversational assistant compounds when it remembers. The first time you tell it you sell to mid-market RevOps leaders, that you prefer short and casual outreach, or that a particular account is already a customer, that should be the last time you have to. A session without memory throws all of it away at the door. Every new conversation starts from zero, and the user pays the tax of re-explaining themselves forever.

The naive fix makes things worse. If the assistant simply records what the user says whenever a message looks like a preference, it captures the literal words of one-off tasks, questions, and fix-this requests, not durable facts. The store fills with noise: "find me ten of these," "the images are broken, please fix," "show me my last meeting." None of that is worth remembering, yet all of it gets injected back into future conversations as if it were a stable truth about the user. The assistant is now personalized, confidently, on garbage.

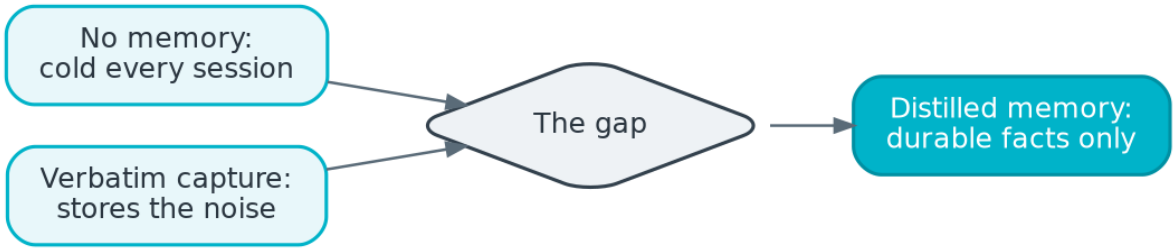

Figure 1. Good memory sits between two failures: an assistant that forgets everything and one that records the noise. The aim is durable facts only.

Good memory sits between the two failures. Remember the few things that will still be true and useful months from now, write them down as clean facts rather than raw messages, and make them easy to recall exactly when they matter. The rest of this paper is about how to do that well.

> Memory is not a transcript. It is the small set of durable facts worth carrying forward.

2. What the memory layer is

The memory layer is the system that decides what to remember about a user, stores it durably, and brings the right pieces back into context at the right moment. It is built from four capabilities that work together.

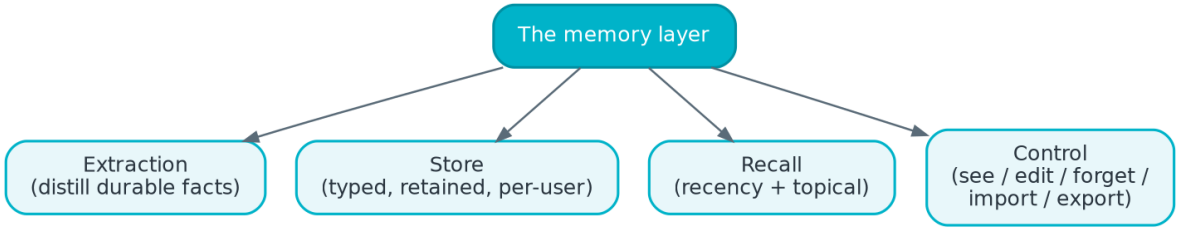

Figure 2. The memory layer is four capabilities around one store: extraction, storage, recall, and user control.

In brief: **extraction** turns a raw conversation into a clean fact only when there is one worth keeping; the **store** holds those facts, typed and time-bounded, scoped to their owner; **recall** brings memories back into a live conversation, both the most recent and the most relevant; and **control** gives the user full ownership of the record. Sections 4 through 8 take each apart in depth.

3. Why this matters

Making memory a deliberate, curated layer rather than a transcript pays off differently for each audience.

- **For operators and end users:** the assistant feels like it knows you. You state a preference once and it holds. The conversation stays personal across months without you managing anything, and when you want to, you can see and correct exactly what it believes.
- **For enterprise buyers:** memory is bounded and governed. It is scoped per user, retained on a defined horizon, and fully inspectable and erasable, which is what a serious data-handling posture requires.
- **For investors:** personalization is a durable moat. An assistant that accumulates an accurate, private model of each user gets more useful the longer it is used, and that switching cost compounds.
- **For builders:** the abstraction is small and safe. Memory is a set of distilled facts plus a recall step in front of the existing context pipeline. There is no second brain to keep in sync, and the risky part, deciding what to keep, is delegated to a model with a tightly scoped instruction.

In one line: **an assistant should remember the few things that matter, not transcribe everything you say.**

4. What gets remembered

Not everything in a conversation is worth keeping, and the things worth keeping are not all the same kind of thing. The memory layer records four types of durable fact.

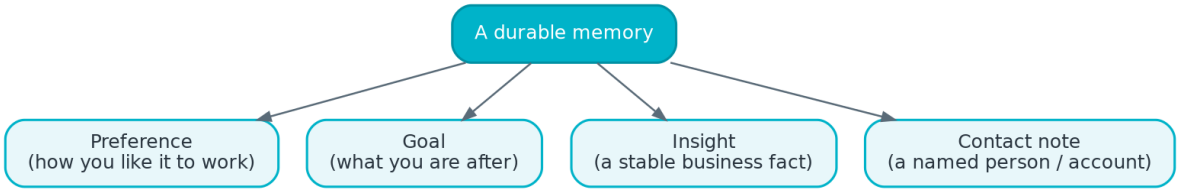

Figure 3. Four kinds of durable fact are worth keeping, preferences, goals, insights, and contact notes. Everything else is ignored.

- **Preferences** capture how the user likes the assistant to work: tone, format, channels, meeting style, and explicit do and do-not rules.
- **Goals** capture what the user is trying to achieve: target markets, ideal customer, territories, quotas, and growth plans.
- **Insights** capture stable facts about the user's business, product, pricing, or team, and about what works for them.
- **Contact notes** capture durable facts about a specifically named person, account, or deal, the kind of detail a good account manager would never forget.

The discipline is in what is excluded. One-off task requests, questions, requests to fix or build something, search queries, and small talk are explicitly not memories, no matter how they are phrased. Most conversational turns produce nothing at all, and that is the correct outcome.

5. Extraction: distillation, not transcription

The central decision in the memory layer is how a turn becomes a memory. The wrong approach, and the common one, is to match the message against a list of trigger phrases and save it verbatim when one hits. That is fast and free, but it is the source of every junk memory: it cannot tell a durable preference from a passing request that happens to contain the same words, and even when it is right about the category it stores the user's raw sentence instead of the fact inside it.

This layer uses a small, inexpensive language model as a judgment step instead. After each turn, and entirely out of band so the user never waits for it, the model is shown the exchange and the user's existing memories and asked a precise question: is there anything here that will still be true and useful months from now, and if so, what is it, stated as a clean fact?

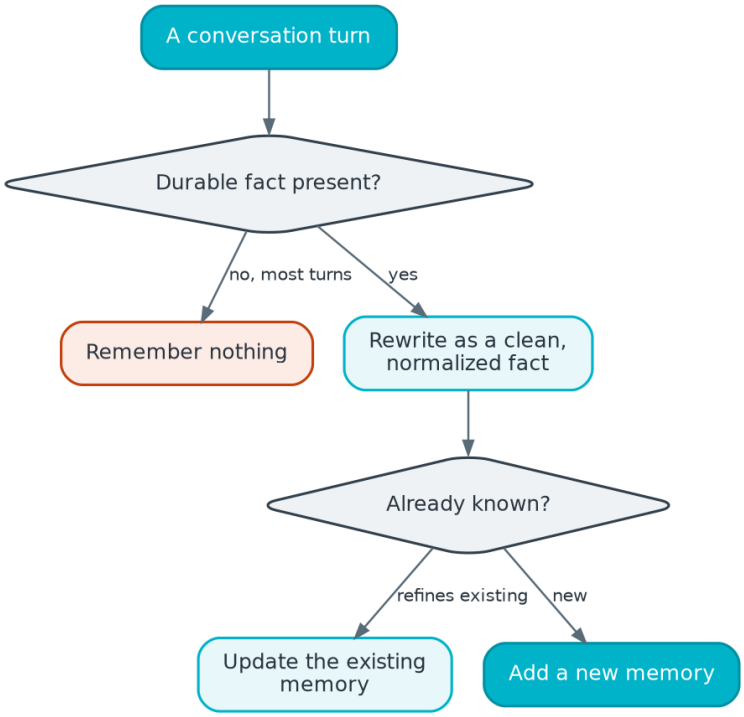

Figure 4. Distillation. After each turn a small model decides whether a durable fact was present, normalizes it, and either updates an existing memory or adds a new one. Most turns produce nothing.

Three properties make this work where transcription fails.

- **It distills.** The output is a short, normalized statement of fact written so it reads like a note the assistant keeps about you, not the words you typed and not a clinical record that begins with "the user." "Planning a newsletter campaign that drives prospects to the landing page" is a memory. The paragraph that produced it is not.
- **It abstains.** The instruction is explicit that most turns contain nothing worth keeping, and that when in doubt it should record nothing. A memory layer that saves rarely and accurately is far more valuable than one that saves often and noisily.
- **It deduplicates.** Because the model sees the existing memories, it can recognize when a new fact refines or contradicts one already stored and update it in place rather than adding a near-duplicate. The store stays small and current instead of accumulating slight variations of the same fact.

A light pre-filter skips the obviously empty turns, a greeting or a one-word acknowledgment, so the judgment step is spent only where a fact might plausibly exist. And because memory must never degrade the conversation, the entire process is best-effort: if it fails for any reason, the chat is wholly unaffected.

6. Storage: typed, time-bounded, and owned

A memory is stored as a typed fact belonging to exactly one user, with the context of where it came from and a clear lifetime. Three choices in the storage model matter.

- **Per-user scoping.** Every memory belongs to the individual who created it. The assistant recalls one person's memories and never another's, which is both a correctness property and a privacy property.

- **A retention horizon.** Memories are kept for a year by default, then expire. This is deliberate. The point of the layer is long-term recall, so the horizon is long, but it is finite, so stale facts do not accumulate forever. The horizon is a single configurable value rather than something hard-coded into the logic.
- **Soft deletion.** Forgetting a memory deactivates it rather than destroying the row immediately, which keeps deletion reversible in the near term and keeps the recall index consistent. From the user's point of view the memory is gone, and it never re-enters a conversation.

The result is a store that is durable enough for year-long recall, bounded enough not to rot, and structured enough that the user can reason about it a category at a time.

7. Recall: recency and relevance together

Storing a fact is only half the system. The harder half is bringing the right facts back into a live conversation, and there are two different notions of "right." The most recent things you told the assistant are usually relevant, and a fact from a year ago is relevant precisely when the current question is about it. The layer serves both.

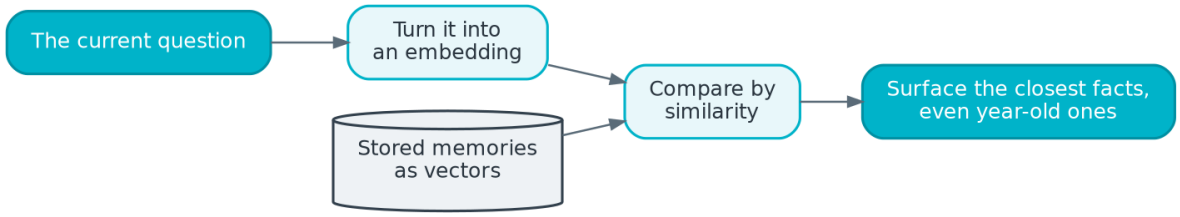

Figure 5. Topical recall. The question is embedded and compared against stored memory vectors, so a relevant fact surfaces no matter how old it is or how differently it was worded.

Topical recall is the part that makes "remember a year ago" real. Every memory is turned into an embedding, a numerical representation of its meaning, by a dedicated background process shortly after it is saved. When the user asks something, the question is embedded too, and the memories whose meaning is closest to it are pulled back, regardless of how old they are. This is semantic, not keyword based: a question about "reaching out to founders in the southeast" can surface a months-old memory about a South Carolina focus even though they share no words. Because the index is built incrementally in the background, recall stays fast and never blocks the conversation.

Recency recall is the complement. The most recently learned facts are always made available to the assistant, so the conversation reflects what you told it lately without depending on the question phrasing to retrieve it.

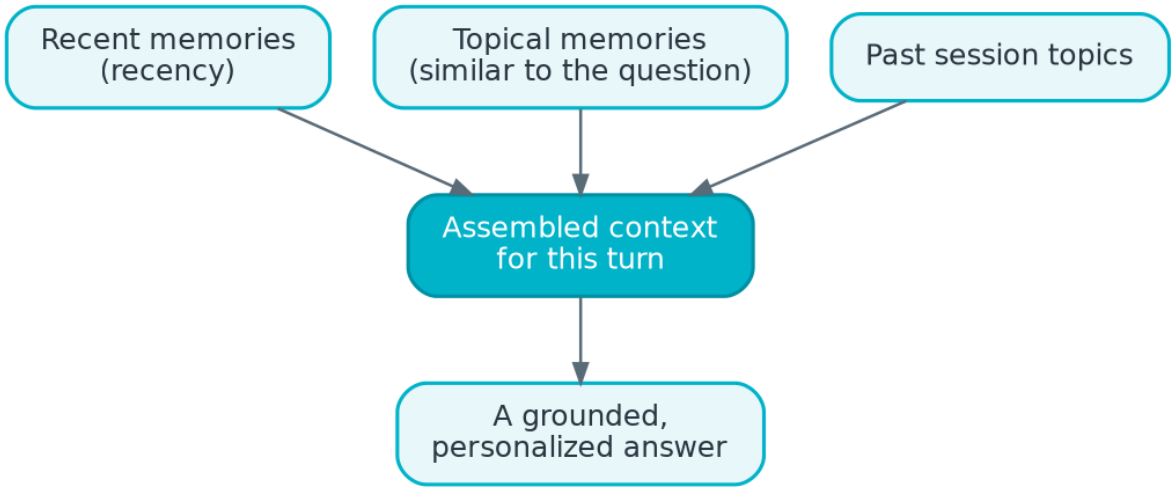

Figure 6. Context assembly. Recent facts, topically relevant facts, and past session topics combine into the context for a grounded, personalized answer.

Combined, the two give the assistant both a short-term sense of where the conversation has been lately and a long-term ability to reach back for the one old fact that matters now. Past session topics are folded in as well, so the assistant has a sense of the arc of the relationship and not just isolated facts.

8. Control: the user owns the memory

Memory the user cannot see is memory the user cannot trust. The layer is paired with a full management surface, reachable both from account settings and from inside the assistant itself, that gives the user complete authority over what is remembered.

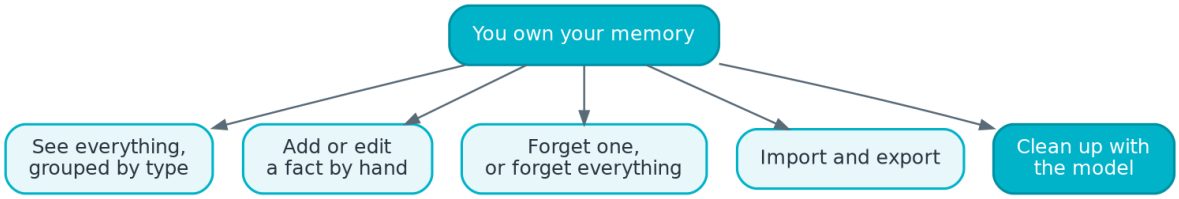

Figure 7. Control. The user sees, edits, forgets, imports, exports, and can ask the model to clean up the whole store; edits keep the recall index in step.

- **See and curate.** Every memory is listed, grouped by type, so the user can read exactly what the assistant believes and correct it. Editing a fact or forgetting it takes immediate effect, and the recall index is kept in step so a forgotten fact never resurfaces.
- **Portability.** Memories can be exported as a portable file and imported back, whether from this assistant or pasted from another, so the user's accumulated context is theirs to move, not locked in.
- **Clean up.** Because earlier, cruder capture can leave low-quality entries behind, the user can ask the model to re-read the whole store in one action, drop the one-off requests and duplicates, and rewrite what remains as clean facts. The operation is conservative: if the model returns nothing usable it leaves the store untouched rather than risk erasing it.

These are the same affordances that leading consumer assistants have converged on, for the same reason. Trust in a memory feature comes from visibility and control, not from asking the user to take it on faith.

9. Architecture: a write path and a read path around one store

The whole layer can be understood as two flows that meet at a single store. The write path turns conversation into curated memory. The read path turns a new question into grounded context. Neither path is ever on the critical latency of the user's reply.

The write path: a turn finishes, the distillation step decides out of band whether a durable fact was present, and if so the fact is normalized, deduplicated, and saved. A background process later turns the new fact into an embedding for topical recall.

The read path: a new question arrives, the most recent memories are gathered by recency, the most relevant ones are gathered by similarity to the question, past session topics are added, and the assembled context is handed to the assistant exactly as it would be for any other turn. There is no separate inference engine for memory; recall feeds the same context pipeline the assistant already uses, scoped to the owner and their organization.

This is the same architectural principle that governs the rest of the platform. Memory does not fork the runtime or duplicate behavior. It is a curation step on the way in and a retrieval step on the way out, wrapped around one durable, owned store.

10. The research behind distilled, retrieved memory

The design rests on a few well-studied ideas, combined deliberately.

- **Retrieval-augmented generation.** A large body of work shows that grounding a language model in a small set of retrieved, relevant facts produces better and more faithful output than relying on the model's parameters alone. Recall is exactly this: the memories pulled into context are the retrieved grounding for a personalized answer.
- **Semantic search over embeddings.** Representing text as vectors and comparing them by similarity is the mature way to find meaning rather than matching words. Using it for memory is what lets an old, differently-worded fact surface for the right new question.
- **Summarization over transcription.** Research on long-term conversational memory consistently finds that distilled, salient facts beat raw logs. Transcripts are long, redundant, and mostly irrelevant later; a few well-chosen facts are cheap to store, cheap to retrieve, and far more useful. Distillation is the application of that finding at the moment of capture.
- **Human memory as consolidation.** Biological memory does not store a recording of experience; it consolidates experience into durable traces and reconstructs them on demand. A system that distills facts on the way in and retrieves them by relevance on the way out is a faithful, if simplified, version of the same shape.

The contribution is not any single one of these. It is putting the judgment step, the model that decides what deserves to be a memory, at the front of the pipeline, so that everything downstream operates on signal instead of noise.

11. Privacy and governance

A system that remembers people has to be built for control from the start. Memory here is scoped to the individual user and never crosses between users or organizations. Retention is bounded by a defined horizon rather than kept forever. The user can inspect the entire store, correct any part of it, and erase it in full at any time, and erasure keeps the recall index consistent so nothing forgotten can leak back into a later conversation. The instructions that drive the distillation and clean-up steps are held as private server-side assets, not exposed to the browser. The governing idea is that accumulated personal context is the user's property, held on their behalf under their control, not an opaque profile assembled about them.

12. By the numbers

The operating parameters that bound the memory layer today. These are design parameters, not measured telemetry. Runtime metrics are being collected in the development environment and will be added in a future revision.

Parameter	Value
Memory types	Preference, goal, insight, contact note
Capture method	A small language model distills each turn; most turns produce nothing
Capture cost to the user	None; extraction runs out of band, never blocking the reply
Deduplication	New facts that refine an existing memory update it in place
Retention horizon	One year by default, configurable, then expiry
Deletion model	Soft deactivation; recall index kept consistent
Recall, recent	The most recent facts are always available to the assistant
Recall, topical	Semantic similarity surfaces relevant facts of any age
Indexing	Embeddings built incrementally in the background, never on the reply path
User control	See, add, edit, forget, forget all, import, export, model-assisted clean-up
Privacy scope	Per user; never shared across users or organizations

13. Relationship to Plays and autonomous agents

Memory is the connective tissue under the rest of the platform. Plays and autonomous agents both execute on the same assistant runtime, and that runtime carries memory with it. A Play that prospects "like we did last time" can mean it, because the context it rides already recalls last time. An autonomous agent working a standing goal stays aligned with the user's stated preferences because those preferences are in the context it runs against. Memory is not a sibling feature to these; it is the substrate that makes their personalization real.

	Without the memory layer	With the memory layer
A new chat	Starts cold, re-explained each time	Opens already knowing your preferences and goals
A scheduled Play	Runs the literal prompt	Runs against your accumulated context
An autonomous agent	Pursues the goal in isolation	Pursues it within what is known about you
A fact from a year ago	Gone	Surfaced when the question makes it relevant

14. Why it is valuable

- **Continuity.** State a thing once and it holds. The relationship with the assistant accumulates instead of resetting.
- **Signal over noise.** Because capture distills and abstains, what the assistant knows about you is accurate and small, not a junk drawer it mistakes for truth.
- **Relevance across time.** Topical recall makes the depth of a year of history usable, surfacing the one old fact that matters now instead of burying it.
- **Trust through control.** Full visibility, correction, portability, and erasure turn a feature that could feel intrusive into one the user owns.

15. Roadmap

The memory layer ships deliberately. Current work hardens distillation quality, the recall blend, and the user controls. Near-term directions include richer recall tuning, explicit memory seeding so a workflow can run "using what we learned last time," deeper consolidation that periodically merges related facts into cleaner summaries, and organization-aware memory for facts that a team should share rather than an individual. Shared memory remains gated behind the same per-user and per-organization scoping that governs everything else, so breadth never comes at the cost of control.

Closing

The thesis is simple. An assistant earns trust by remembering the right things and proving it knows only what you would expect. By distilling each conversation into the few facts worth keeping, storing them as the user's own property on a bounded horizon, and recalling them both by recency and by relevance so even a year-old fact returns at the right moment, ATAPIC turns a stream of disposable conversations into a durable, personal, and controllable model of the person it works for.