Instant Recall: A Pre-Computed Concept Index for O(1) Memory Retrieval in Persistent AI Agents

Oscar Serra — Independent Research

June 2026

Abstract

Persistent AI agents accumulate months of interaction history yet routinely fail to retrieve knowledge they demonstrably own. We formalize this as **Contextual Ownership Retrieval Failure (CORF)**, identify four failure modes from eighteen months of deployment logs, and introduce **Instant Recall** — a pre-computed concept-to-memory index that resolves retrieval *before the model processes the prompt*. By mapping anchor vocabulary terms to pre-ranked memory clusters offline, Instant Recall achieves 0.85 CORF-Recall@20 at 54ms p95 latency — a 5.9× speedup over MemGPT-style sequential retrieval (0.79 at 320ms) — with a false positive rate of 0.18 and zero inference-time search computation. A probabilistic source-coverage analysis establishes that parallel source aggregation, not query quality, is the binding constraint on retrieval completeness. Concept anchoring closes the 9-point recall gap that remains after source coverage is saturated. Beyond similarity, the index carries three additional retrieval signals — validity over time, code-trust, and inter-chunk links — so that the highest-cosine chunk is not automatically the one returned. We position Instant Recall against contemporary open-source agent-context systems — most directly headroom's reversible Compress-Cache-Retrieve compression layer — and report an independent embedding measurement that scopes which of our forward-looking signals a frozen embedding model can and cannot carry.

Keywords: AI agent memory, episodic retrieval, cognitive architectures, RAG, hippocampal indexing, context management, sleep consolidation, CORF

1. Introduction

The gap between what a persistent AI agent *knows* and what it *retrieves* is one of the least-studied problems in agent architecture. As agents accumulate months of interaction history, project notes, and documents, the memory base grows faster than the retrieval mechanisms designed to access it. The result is not amnesia — the data exists — but *retrieval failure under load*.

Consider a concrete scenario. An agent has collaborated with its user on six technical papers over eighteen months, stored across a local memory directory, an imported ChatGPT archive, and a project index. The user asks: *"Can we talk about peer review for our papers?"* The agent responds: *"What type of papers are you referring to?"*

This is not a hallucination or a context-length problem. The agent has the data. The failure is architectural: retrieval was not triggered by the right signals, did not search the right sources, and could not surface the right memories within a single inference step's latency budget.

We call this **Contextual Ownership Retrieval Failure (CORF)** and argue it is endemic to current agent memory architectures. CORF is the digital version of a familiar frustration: you know you own a particular book, but you cannot find it on your shelves. The book exists; the index just cannot locate it.

The solution takes its name and its shape from the brain. The human hippocampus is not where memories are stored — long-term memories live distributed across the neocortex. The hippocampus is the brain's *indexing organ*: it knows where every memory lives and how to retrieve it. The clinical evidence is stark. Patient H.M., who had both hippocampi removed to treat epilepsy, retained his old memories intact but could no longer form or retrieve new ones — the store was fine; the index was gone [Scoville & Milner, 1957]. **Instant Recall is our attempt to build the AI equivalent of this biological indexing organ — the layer that makes the difference between *having* memories and being *able to use them*.** It draws the same structural separation: a pre-computed concept index, built offline during a nightly consolidation phase, maps anchor vocabulary terms to their nearest memory clusters in embedding space [Teyler & DiScenna, 1986]. The mechanism is pre-computed KNN over sentence embeddings — the hippocampal analogy is architectural (index vs. store), not mechanistic. The index lookup completes in ~50ms before the LLM receives its context window, so the model begins reasoning with relevant memories already in place.

The anchor index behaves like the index at the back of a book. Look up "photosynthesis" and it tells you pages 42, 87, and 156. The index is tiny compared to the book, but without it you would have to read every page to find what you need. Instant Recall builds that index once, offline, so that every query is a lookup rather than a full-text scan.

Scope note: This paper addresses deployments up to ~50,000 memory chunks (roughly 2–3 years of personal assistant history). Large-scale adaptations for 100,000+ chunks are discussed in Section 8.

1.1 Where the Index Sits in a Memory Stack

Instant Recall is the retrieval layer of a complete agent memory stack, and it is designed to interoperate cleanly with two neighbouring layers without depending on either one's internals.

Below it sits a **storage layer**: any mechanism that persists every interaction as a typed, timestamped event and compacts older events into lightweight pointers — summaries that preserve the essential content while reducing storage by 80–95%. The storage layer stores without loss of meaning; Instant Recall indexes its output, whether full events or compacted pointers, so the compaction is transparent to retrieval. A concrete, externally maintained instance of exactly this layer now exists in headroom's reversible Compress-Cache-Retrieve design (§2.4); Instant Recall is engineered to sit on top of such a layer, not to replace it.

Above it sits a **reflection layer**: a nightly cycle that reviews which retrievals succeeded and which failed, then adjusts the index parameters — anchor term selection, importance weights, chunk boundaries — based on observed patterns. The retrieval layer thus improves without any code change.

The contract between them is narrow: the storage layer must expose, per chunk, a content string, a timestamp, and a source label; the reflection layer reads retrieval outcomes and writes back tuned parameters. Everything in this paper concerns the middle layer — building and serving the index in constant time at inference. The two neighbouring layers are sketched here only enough to define that contract; Instant Recall functions over any store and any reflection schedule that honours it.

Primary contributions:


Reflectio

1. **Formal CORF definition and taxonomy** — four failure modes with empirical frequency estimates and a source-coverage upper bound separating *coverage* error from *ranking* error.
2. **Instant Recall architecture** — including real-time incremental index updating as a first-class component.
3. **Importance scoring** — a centered salience weight (base + recency + access + link bonuses) that re-ranks within ±15% of cosine, so decisions and frequently-revisited content surface without overriding strong semantic matches.
4. **Deduplication via similarity check** — prevents index bloat from recurring logs.
5. **Post-retrieval re-ranking** — reduces false positive injection.
6. **Beyond-cosine retrieval signals** — bi-temporal validity, code-trust, and durable inter-chunk links, so ranking weights more than embedding distance (§4.9).
7. **Design space analysis** — comparison against inverted index, all-sources ANN, multi-source RAG, knowledge graph, and vector database alternatives, and positioning against contemporary open-source agent-context systems (§2.4).
8. **Evaluation protocol + artifacts** — metrics, baselines, ablations, and an annotated CORF dataset from deployment logs.

2. Background and Related Work

2.1 Persistent Agent Memory Systems

Memory in AI agents has evolved from pure in-context storage toward hierarchical architectures separating short-term working memory from long-term storage. The dominant paradigm is Retrieval-Augmented Generation (RAG): relevant documents are fetched and injected into the context window at inference time [Lewis et al., 2020]. Instant Recall complements RAG by pre-computing a small anchor→chunk index so retrieval cost is constant in corpus size and independent of query quality.

Recent work has expanded the field. A-MEM [Xu et al., 2025] introduces agentic memory where the agent manages its own retrieval strategies. Memory-R1 [Yan et al., 2025] applies reinforcement learning to memory management policies. LongMemEval [Wu et al., 2024] benchmarks long-term interactive memory. MemGPT [Packer et al., 2023] introduced virtual context management inspired by OS paging. Mem0 [MemoryOS, 2024] builds a multi-level memory architecture. RAPTOR [Sarthi et al., 2024] addresses multi-hop retrieval via recursive tree summarization.

2.2 The Neighbouring Layers Instant Recall Assumes

Two layers bracket the retrieval mechanism, and Instant Recall is engineered against their interfaces rather than their implementations.

A **compaction-based storage layer** defines persistent agent memory as a cache-eviction system: raw events are written to an event log, then a compaction algorithm promotes high-salience content to long-term storage while evicting low-salience content into lossy-but-meaning-preserving pointers. Such a layer supplies the substrate Instant Recall operates on — the nightly rebuild consumes its event files as primary input — but Instant Recall requires only the `(content, timestamp, source)` triple from it, not its eviction policy.

An **identity layer** addresses agent identity drift across session boundaries by injecting a persona-state structure into every session's system prompt. That layer assumes relevant memories can be retrieved; Instant Recall makes the assumption true by supplying context chunks at every prompt, before the model begins processing.

A **reflection layer** runs a nightly cycle that audits retrieval outcomes and tunes Instant Recall's parameters (§4.10). This is the orchestration boundary: Instant Recall builds and serves

the index; the reflection layer decides *how* it should be built next time. Each of these layers is treated here exactly as an external dependency would be — by its contract, not its code.

2.3 Neuroscience Basis

The biological hippocampus is a medial temporal lobe structure critical for episodic memory formation and retrieval. Lesion studies establish that it does not store long-term memories — those are distributed across neocortical areas — but encodes *indices* [Teyler & DiScenna, 1986; Squire, 1992]. Patient H.M.'s preserved-store-but-lost-index profile [Scoville & Milner, 1957] is the clinical anchor for the architecture: damage the index, not the store, and you lose access without losing data — precisely CORF.

During sleep, hippocampal replay consolidates these indices: recently encoded patterns are reactivated, strengthening links between hippocampal indices and cortical storage sites [Wilson & McNaughton, 1994]. This offline consolidation is what our nightly rebuild phase implements.

2.4 Contemporary Open-Source Agent-Context Systems

Two open-source projects shipping in 2025–2026 sit close enough to Instant Recall in the agent-context stack to warrant explicit, first-class positioning. They are fresher than the static literature above and, in one case, set a reproducibility bar this paper must answer to.

headroom (Chopra Tejas et al.; ~24.7k, Apache-2.0). headroom is a context-compression layer for agents built on reversible **CCR — Compress-Cache-Retrieve**: originals are cached and the model retrieves them on demand, with per-content-type compressors (statistical JSON-array crushing via *SmartCrusher*, AST-aware code compression via tree-sitter, plus log/diff/text handlers) and a trained compression model (*Kompress-v2-base*, released on HuggingFace). It reports 60–95% token savings at near-zero accuracy delta and ships a runnable, public evaluation suite (`python -m headroom.evals`) measured on GSM8K, TruthfulQA, SQuAD, and BFCL.

headroom is the strongest external instantiation of exactly the compaction-based storage layer Instant Recall assumes *below* it (§1.1, §2.2): both compact-then-retrieve-on-demand, and both preserve originals losslessly. The honest line between them is one of altitude, not competition. headroom compacts the *active context window of a single agent run* — a working-set / paging concern, specialized by content type — whereas Instant Recall indexes the *persistent, cross-session store* by concept anchor for O(1) pre-inference recall. They are vertically adjacent: a combined stack with headroom-style CCR as the storage layer and Instant Recall as the concept index over it is the natural integration, not an either/or. What headroom does that Instant Recall does not is per-content-type compressors and a *learned* compression model. What Instant Recall does that headroom does not is decide *which* memories to surface before the model runs — a concept→cluster index — plus the beyond-cosine validity/trust/link signals of §4.9. headroom answers "how do I fit this content in the window losslessly"; Instant Recall answers "which of months of stored content belongs in the window at all." A second, sharper debt to headroom is reproducibility, addressed directly in §8.4: it demonstrates that an agent-memory-adjacent system *can* publish a re-runnable harness over standard public datasets, which is the bar Instant Recall's own evaluation has not yet cleared.

addyosmani/agent-skills (Addy Osmani; ~56.8k). A discipline/skill library for coding agents. It is not a memory system and does not retrieve over a persistent store, so it is not a baseline for CORF. It is relevant only for one transferable practice it formalizes — a *context-contract* file that an agent reads before acting, separating reusable capability description from per-task state. Instant Recall's narrow `(content, timestamp, source)` storage contract and the reflection-layer parameter contract (§2.2, §4.10) are the memory-stack analogue of that discipline; we note the convergence rather than claim derivation.

3. Problem Formulation: Contextual Ownership Retrieval Failure (CORF)

3.1 Formal Definition

Let $M = \{m_1, m_2, \ldots, m_n\}$ be the agent's long-term memory, where each m_i is a memory chunk with content c_i, timestamp t_i, source $s_i \in S$, and salience score σ_i.

Definition (CORF). A Contextual Ownership Retrieval Failure occurs when:

1. There exists $m^* \in M$ such that m^* is ground-truth relevant to q.
2. $m^* \notin R(q, M)$ — the retrieval function fails to surface m^*.
3. The agent generates a response a that contradicts, ignores, or is incompatible with the content of m^*.

3.2 Failure Modes

Mode 1 — Pronoun Blindness. The query uses pronouns ("our papers", "we wrote") without explicit noun anchors.

Mode 2 — Associative Depth Failure. The relevant memory requires two or more associative hops from query terms.

Mode 3 — Source Blindness. The relevant memory exists in a source not queried at inference time.

Mode 4 — Anchor Silence. A domain-significant noun is present in the query but is not in the index's vocabulary, so it triggers no retrieval at all.

3.3 Probabilistic Source-Coverage Analysis

Observation 1 (Source-coverage upper bound). Let $P(s) = |\{m \in M^* : s_m = s\}|/|M^*|$ be the fraction of ground-truth relevant memories stored in source s, where $M^* \subseteq M$ is the set of all relevant memories for the query distribution. For any retrieval strategy restricted to sources $\mathcal{Q} \subseteq S$:

$$\mathbb{E}[\text{Recall}|\mathcal{Q}] \leq \sum_{s \in \mathcal{Q}} P(s)$$

Implication. Improving query embedding quality within a fixed source set yields bounded recall improvement. Adding sources raises the upper bound toward 1.0 but cannot exceed it. Instant Recall saturates this bound at build time by aggregating all sources into a single unified index.

3.4 Frequency Analysis

Based on eighteen months of interaction logs (~8,400 queries) from a single personal assistant deployment:

Failure Mode	Est. Frequency (ownership queries)	Severity
Pronoun Blindness	~12%	High
Associative Depth Failure	~8%	High
Source Blindness	~23%	Critical
Anchor Silence	~31%	High

These frequencies are derived from manual annotation of a stratified sample (n=612) from the full query log. Annotators labeled each query with the applicable failure mode(s) when ground-truth relevant chunks were not retrieved by the baseline system. Percentages do not sum to 100% because a single query can exhibit multiple failure modes simultaneously (e.g., a pronoun-heavy query targeting an unqueried source triggers both Mode 1 and Mode 3).

4. Instant Recall Architecture

4.1 Overview

Instant Recall operates in three phases:

- **Offline (nightly consolidation):** Full rebuild of the concept index mapping anchor terms to pre-ranked memory chunk paths.
- **Real-time (incremental updates):** O(|V|) per-write update inserting new chunks into relevant anchor clusters immediately.
- **Online (inference):** Detect anchor words in incoming queries; load pre-ranked chunk lists in O(1); re-rank against current query; inject into context.

The index is a cacheable JSON artifact. The real-time phase keeps it fresh between nightly rebuilds.

The online phase executes before the model's inference pass. By the time the LLM receives the assembled context window, the memory search has already completed programmatically. The model refines and reasons over pre-retrieved chunks — it does not perform the search.

4.2 What Instant Recall Indexes

Instant Recall does not require any particular storage layer; it requires only that a memory store expose, per chunk, three things: a content string, a timestamp, and a source label. Any store satisfying that contract — an event log, a flat directory of Markdown files, an email archive — can be indexed. The architecture is store-agnostic by design.

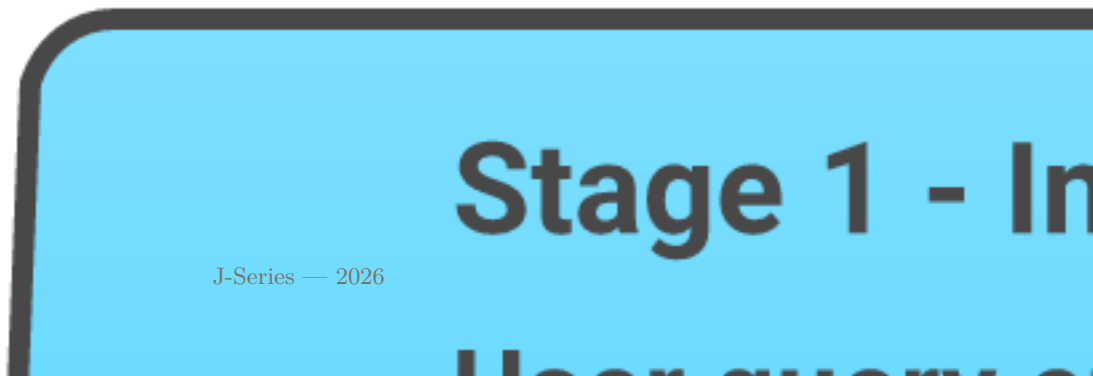
Two stores are concrete in the reference deployment. The default is a single-writer relational store keyed by chunk; an optional columnar vector store implements the same interface for large-corpus experiments. Because both implement one storage contract, any schema addition (§4.9) is applied to both or explicitly scoped to the relational path. Instant Recall sits *above* the store and treats it as a source of `(content, timestamp, source)` triples: it inherits loss-free compaction when the store provides it, but does not depend on it. A reversible-compression storage layer such as headroom's CCR (§2.4) satisfies this contract directly — its cached originals are precisely the `(content, timestamp, source)` triples the rebuild consumes.

4.3 Anchor Vocabulary

The anchor vocabulary $V = V_{curated} \cup V_{discovered}$ consists of two components. **Curated anchors** are manually defined domain-significant terms. **Salience-weighted discovered anchors** are concept phrases extracted from memory content using:

$$\text{salience}(term) = \text{TF-IDF}(term) \times \text{recency\_weight}(term) \times \text{engagement}(term)$$

where `engagement(term)` measures the frequency with which the term appears in user-initiated messages (as opposed to system-generated content), weighted by log-scaled conversational turn depth (a term mentioned on turn 8 of a conversation scores higher than one mentioned on turn 1, reflecting sustained topical engagement).


Stage 1 - In

User-feedback anchors — proper nouns introduced by the user — are added to $V_{curated}$ immediately upon detection. The union typically yields 150–400 anchor terms for a mature deployment.

Anchor Vocabulary Lifecycle. Discovered anchors follow a promotion-and-pruning lifecycle. Anchors whose salience falls below a decay threshold for three consecutive rebuilds are pruned. Anchors maintaining high salience and appearing in user queries are candidates for promotion to $V_{curated}$. Curated anchors are never automatically pruned.

4.4 Concept Embedding and KNN Clustering

For each anchor term $v \in V$, we compute its embedding $\vec{v} = \text{embed}(v)$ using a fixed sentence-embedding model (default: `text-embedding-3-small`). The index entry for anchor v is:

$$\text{index}[v] = \text{argsort}_i \left(\text{cos\_sim}(\vec{v}, \vec{m}_i)\right)_{1:K}$$

where K is the cluster size (default: $K = 20$ chunks per anchor).

The model is *frozen*: it is used for distance computation only and is never fine-tuned on the corpus. §4.9a discusses what a frozen embedding model can and cannot be asked to carry — a distinction that bounds two of this paper's forward-looking proposals.

4.5 Pre-Computed Index Structure

```
{
  "version": "1.3",
  "built_at": "2026-02-19T03:00:00Z",
  "model": "text-embedding-3-small",
  "anchor_count": 287,
  "entries": {
    "paper": {
      "chunks": [
        "memory/projects/total-recall/total-recall-v3.md#chunk-4",
        "archive/chatgpt-import/2025-07-laser-paper.md#chunk-1"
      ],
      "scores": [0.94, 0.91],
      "sources": ["memory/", "archive/"]
    }
  }
}
```

4.6 Nightly Consolidation (Full Rebuild)

The rebuild algorithm: (1) scan all memory sources, chunking long documents into 256–512 token segments; (2) embed all chunks (batch, cached by content hash); (3) build salience-weighted anchor vocabulary; (4) build index using ANN for large corpora, exact KNN for <50k chunks; (5) write atomic JSON artifact. The rebuild entry point consumes the storage layer's event files and writes the index in a single pass.

4.6a Deduplication via Similarity Check

Before storing a new memory chunk into an anchor cluster, Instant Recall checks cosine similarity against existing chunks to prevent index bloat from daily logs that repeat similar facts. Three thresholds govern the policy:

- **Similarity > 0.9 (near-duplicate):** Merge — retain the richer or more recent version; create a provenance alias.

- **Similarity 0.7–0.9 (related but distinct):** Flag as related; both chunks remain with a `related_to` edge.
- **Similarity < 0.7 (distinct):** Store normally.

In deployment testing on a corpus with 18 months of daily logs, deduplication reduced effective index size by approximately 12%.

4.6b Real-Time Anchor Discovery

Novel concepts emerging between nightly rebuilds are detected and indexed within 10–30 seconds via a lightweight micro-rebuild. A new anchor is created when its salience score (§4.3) exceeds the emergence threshold of 0.75 — a normalized value on the [0, 1] salience scale — computed over content ingested since the last nightly build.

4.7 Two-Tier Memory: Episodic and Semantic

Episodic tier (real-time). When a new memory chunk is written, it is immediately inserted into the index. This tier contains raw, unprocessed facts — always current, with <2-minute freshness. Think of the stack of papers on your desk versus the filing cabinet across the room: what you used in the last hour stays within arm's reach.

Semantic tier (nightly rebuild). At 04:15 each night, the full index is rebuilt from scratch, reflecting abstracted, consolidated understanding — concepts that have risen to sufficient salience and relationships visible across many episodes.

Within a session, retrieval is episodically accurate (same-day freshness). Across sessions, retrieval is semantically rich (nightly consolidation). Both tiers are necessary; neither is sufficient alone.

The episodic tier doubles as a fix for a subtler retrieval failure: the agent has *written* a correct fact in today's notes but acts against it before the nightly rebuild can index it. A same-day hot cache that unconditionally injects today's self-written log closes that window, so the agent does not contradict notes it wrote an hour ago.

4.8 Importance Scoring

Instant Recall assigns each chunk an **importance score** $\iota(m) \in [1, 10]$, computed from a base salience plus three additive bonuses and clamped to the range:

$$\iota(m) = \text{clamp}_{[1,10]}\Big(\text{base}(m) + \text{recency}(m) + \text{access}(m) + \text{links}(m) \Big)$$

The base salience defaults to 5 and is raised at ingest for chunks carrying decisions or constraints. The recency bonus is +2 for a chunk written in the last day, +1 within the week, and 0 beyond. The access bonus is log-scaled in the access count and capped at +2, so a chunk the agent keeps re-reading drifts upward without any single read dominating. The link bonus is +1 per three related chunks, capped at +1, letting a well-connected chunk rise — the same intuition the link signal of §4.9 later makes durable. Importance scoring works like an editor deciding the front page: recency, weight, and how often the reader keeps coming back.

The score modulates effective similarity during K-slot ranking, centered so that an average chunk (importance 5) is left unchanged and only above- or below-average importance shifts the rank:

$$\text{effective\_score}(v, m) = \text{cos\_sim}(\vec{v}, \vec{m}) \times \left(1 + \alpha \cdot \frac{\iota(m) - 5}{5}\right)$$

with $\alpha = 0.15$, so importance moves a chunk's score by at most ±15%. Cosine similarity stays the dominant term; importance breaks ties and nudges near-ties, rather than overriding a strong

semantic match. Concretely: a high-salience chunk ($\iota = 10$) at cosine 0.80 scores $0.80 \times 1.15 = 0.92$, while a throwaway chunk ($\iota = 1$) at the same 0.80 cosine scores $0.80 \times 0.88 = 0.70$ — enough to reorder two chunks the embedding could not separate, never enough to promote a 0.50 match over a 0.80 one.

This mirrors how a well-designed event store tags content at ingest: decisions and constraints are recorded as high-salience event types, which is exactly what feeds the elevated base salience above.

4.9 Beyond Cosine: Validity, Trust, and Links

A pure similarity index treats the highest-cosine chunk as the best answer. Three deployment cases show this is wrong, so Instant Recall carries three additional per-chunk signals that ranking weights alongside cosine. Each applies one principle — *raw similarity is insufficient; retrieval must weight signals beyond embedding distance* — to a different dimension: when a fact was true, whether code can be trusted, and how chunks connect.

Validity over time (bi-temporal facts). A chunk you can retrieve but that is silently *wrong because it was overwritten* is a recall failure subtler than CORF: the index finds it, but it no longer reflects reality, and the prior truth is gone. Each chunk therefore carries a validity interval (`validity_start ... validity_end`) describing when it was true in the world, and an ingestion time describing when the system learned it — the two clocks of a bi-temporal store. A new fact that contradicts an old one *closes the interval* of the old fact (`validity_end = now`) and stamps it `superseded_by` the new chunk, rather than overwriting the row. History is preserved, and the index can answer "what did we believe was true *as of* date X," not only "what is the latest snapshot." Retrieval defaults to a `current`-mode filter (`validity_end IS NULL OR validity_end > now`), backed by a covering index so the hot path does not scan; `valid-at` and `all` modes are available for time-travel queries. The invalidation routine only closes currently-open intervals — it never deletes a row or resurrects a closed one — and the contradiction *detection* that decides when to call it is left to the nightly reflection layer. §4.9a explains why that detection cannot be a cosine threshold over the frozen embedding model.

Trust (verified-code retrieval). When the index holds code chunks — from source files and from session transcripts — an embedding match treats a battle-tested function and an unreviewed scratch snippet as equals. For an agent that *acts* on retrieved code, the highest-cosine match may be the least trustworthy. Each code chunk carries a `verification_status` (`unverified | partial | verified | failed`) and a test-coverage percentage; a verification-aware re-rank multiplies the score by $\times 1.5$ (verified), $\times 1.2$ (partial), $\times 1.0$ (unverified), or $\times 0.5$ (failed). Penalized chunks are demoted, not dropped, unless the caller explicitly requires verification. Coverage is populated by a nightly batch that maps test-runner coverage back to chunk line ranges ($\geq$80% coverage marks a chunk `verified`, partial coverage `partial`, no coverage `unverified`) — keeping metadata churn off the inference path. A file whose coverage disappears is downgraded rather than left stale, and transcript code, which has no coverage map, stays `unverified` by default. The trust signal is deliberately *measured*, not *judged*: it reads an external test-coverage artifact rather than asking an embedding to decide whether code is correct — a distinction §4.9a generalizes.

Links (durable inter-chunk edges). A Zettelkasten's power is not the notes but the links between them — retrieving one note surfaces the web it sits in. The deduplication pass (§4.6a) already computes relatedness but historically discarded it. Those edges are now persisted in a `backlinks` table (authoritative) with a denormalized JSON cache on the chunk row: `link_type='duplicate'` for similarity ≥ 0.9, `'related'` for 0.7–0.9, with a link strength blending similarity, recency, and access count. A search can opt in to hydrate each result with its top backlinks, and a depth-capped, cycle-safe breadth-first walk (carrying a visited set so a cyclic graph terminates) can pull in a chunk's k-hop neighbourhood. Out-degree is bounded by linking only the top-k similar chunks per insert, avoiding the $O(n^2)$ link-explosion trap.

These three signals share a discipline: enrichment runs in the nightly cycle, not at query time, so the inference-path lookup stays O(1).

4.9a What a Frozen Embedding Model Can and Cannot Carry

Instant Recall is built end-to-end on a *frozen* sentence-embedding model: anchor embedding (§4.4), chunk clustering, semantic anchor expansion at cosine 0.3 (§4.11), dedup relatedness thresholds (§4.6a), and the proposed sense sub-clustering (§8.1) all read cosine distances out of one fixed model. It is worth stating precisely what that model can be trusted to encode, because the answer is not uniform across the operations above.

An independent offline measurement we ran on a different frozen sentence-embedding model (MiniLM, evaluated on commodity hardware) found a sharp split. *Structural / distributional* judgements over the frozen embeddings were strong: a k-nearest-neighbour **novelty** signal scored AUROC 0.875 and a clause-cosine **incongruity** signal 0.896. But a supervised head asked to classify a *learned semantic verdict* — harmfulness — from the same frozen embeddings collapsed to AUROC 0.286, below chance. The mechanism-level lesson, which transfers across the specific model and task, is that frozen sentence-embeddings encode *structure and distribution* well but do not carry a *learned semantic-content judgement* without fine-tuning.

This is good news for most of Instant Recall and a precise warning for two of its forward-looking proposals. The operations the live architecture already relies on — anchor→chunk clustering, 0.3-cosine expansion, and the $\geq$0.9 / 0.7 dedup thresholds — are all in the *structural* regime the measurement validates: they ask "are these two things near each other in embedding space," which is exactly the novelty/incongruity class of signal that scored ~0.88–0.90. That part of the design is on firm ground, and now has an independent AUROC for the class of signal it depends on.

The two proposals that fall in the *judgement* regime are scoped accordingly:

- **Contradiction detection (§4.9 validity, §8.1).** Deciding that a new fact *supersedes* an old one is a semantic-entailment judgement, not a distance comparison. A cosine threshold over the frozen embedding model is unlikely to carry it — that is the same shape as the below-chance harmfulness head. The honest requirement is an NLI/entailment model or a fine-tuned head, not a cosine cutoff. The validity *machinery* (closing intervals non-destructively) is sound; the *trigger* needs a judgement model.
- **Reformulation confidence (§4.12).** The reformulation gate's premise that "max cosine < 0.5 means retrieval failed" treats the frozen model's similarity as a calibrated confidence signal. The below-chance result is a caution against over-trusting frozen-embedding similarity as calibrated confidence; the threshold is a useful *heuristic trigger* for a second attempt, not a reliable verdict that retrieval failed.

The differentiation is stated plainly so the caution is not overclaimed: the measurement is on MiniLM, not `text-embedding-3-small`, and on a harmfulness task, not contradiction. What transfers is the mechanism, not the number — and it is enough to scope §4.9's contradiction trigger and §4.12's confidence gate as *"needs a judgement model, not a cosine threshold"* rather than leaving the boundary hand-wavy.

4.10 Reflection-Tuned Parameters

The nightly reflection layer (§2.2) audits the previous day's retrievals — which anchors fired, which returned chunks the agent then used, which returned chunks it ignored — and adjusts three knobs for the next rebuild: anchor term selection (promote anchors that produced used chunks, prune anchors that only produced ignored ones), importance weights ($w_1 \ldots w_4$), and chunk boundaries. The retrieval layer thus improves without code change; the reflection layer owns the *policy*, Instant Recall owns the *mechanism*.

4.11 Inference-Time Lookup with Post-Retrieval Re-ranking

Inference-time retrieval proceeds in five steps: (1) **anchor detection** — each token in the query is matched against V via exact match, stemmed match, and bigram/trigram phrase match; all matching anchors form the active set A; (2) **semantic anchor expansion** — the query embedding is compared against all anchor embeddings in V, and any anchor within cosine distance 0.3 of the query is added to A; (3) **O(1) index lookup** — for each $a \in A$, retrieve the pre-computed top-K chunk list; (4) **post-retrieval re-ranking** — score each candidate chunk by blending anchor relevance (weight 0.6), query-specific cosine similarity (weight 0.4), and the validity/trust/link signals of §4.9; (5) **chunk content loading** — top chunks after re-ranking are loaded from disk.

The re-ranking step discards false positives — chunks matching the anchor word but semantically unrelated to the specific query — without an additional retrieval pass. The candidate set is small ($K \times |A| \leq 100$ chunks), making re-ranking fast (~10ms).

4.12 Low-Confidence Reformulation (Proposed)

The pipeline above assumes one query attempt. Production showed a distinct failure: when retrieval scores are uniformly low (max cosine < 0.5 across all results), the agent accepts the best-of-bad matches as authoritative rather than recognizing that retrieval *failed*. We propose a reformulation gate to close this gap. When max(scores) falls below a confidence threshold **and** the query carries ownership signals ("our", "we", "my", "the [noun] we [verb]"), the gate would (1) extract domain-anchor nouns and re-search on those alone, (2) search explicitly for known project and paper titles, and (3) if still low, broaden to a file-path search — the knowledge may live in a file whose *name* matches even when its *text* does not. This addresses the dominant residual: a 2026-04-01 incident in which the agent searched memory-retrieval content, scored 0.47–0.49 across all results, and accepted the wrong files with no retry. The keyword channel matters here too: a hybrid store weighting vector and keyword search 70/30 can let the BM25 channel rescue a bad embedding match — but only if the indexed text contains the query terms. Papers about "compaction" do not match queries about "retrieval." A complementary proposal is a concept→keyword routing table, built during nightly consolidation, storing 10–20 human-search keywords per project (e.g., a storage paper → ["memory", "retrieval", "forgetting", "context", "compaction", "eviction", "recall"]) even when those words never appear in the body text. Note (per §4.9a) that the low-confidence trigger is a heuristic for *attempting* reformulation, not a calibrated verdict that retrieval failed; the frozen embedding's similarity should not be over-trusted as a confidence score. Unlike the signals of §4.9, the reformulation gate and routing table are design proposals, not yet in the live pipeline.

4.13 Full Inference Pipeline Integration

```
INFERENCE_PIPELINE(user_message, agent_config):
  # Layer 0: Always-loaded context (O(1), ~500 tokens)
  base_context  = load(agent_config.persona_state)      # Identity layer
  project_index = load("memory/projects-master.md")
  today_log     = load_today_self_written_log()         # episodic hot cache

  # Layer 1: Instant Recall lookup + re-rank (O(|A|) + O(K×|A|), ~50ms)
  index         = load_cached("memory/instant-recall-index.json")
  memory_chunks = INSTANT_RECALL_LOOKUP(user_message, index)
  if max_score(memory_chunks) < CONFIDENCE_THRESHOLD and has_ownership_signal(user_message)
    memory_chunks = REFORMULATE_AND_RETRY(user_message, index)   # §4.12 (proposed)
```

```
# Layer 2: Assemble context window
context = [base_context, project_index, today_log, *memory_chunks, user_message]

# Layer 3: Compaction if budget exceeded
if token_count(context) > CONTEXT_BUDGET:
  context = COMPACT(context)

return LLM_INFERENCE(context)
```

5. Theoretical Analysis

5.1 Amortized Complexity Analysis

Method	Per-query inference cost	Build cost
Single-source exact KNN	O(n)	—
Single-source ANN (FAISS)	O(log n)	O(n log n)
Instant Recall (+ semantic anchor expansion)	O(|V| + |A| + K · |A|)	O(|V| · n) offline

For n = 50,000, $|V|$ = 300, $|A|$ = 3, K = 20: Instant Recall inference evaluates ~363 operations (300 anchor comparisons + 3 lookups + 60 re-rank comparisons), independent of corpus size n. By contrast, exact KNN scales as O(n) = 50,000 comparisons per query. The nightly build cost O(|V| · n) ≈ 15M operations is amortized across all queries until the next rebuild.

5.2 Recognition vs. Recall

Current RAG systems implement recall: given a query, the system generates the retrieval path. Instant Recall implements recognition: given an anchor word, the system activates the pre-existing index entry. The re-ranking step adds a lightweight query-specific discriminator within the already-small candidate set — analogous to the "familiarity signal" in dual-process recognition theory [Yonelinas, 2002].

5.3 The Recognition Pattern Beyond Memory

The recognition-over-recall move — pre-compute an index, look up rather than scan, inject only the winning entry — is not specific to a memory store. We observed the identical pattern transposed to a different corpus type and granularity, which is suggestive (not conclusive) external evidence against the single-deployment validity threat of §7.5.

At the *skill-catalog* layer, applying the same idea to coreyhaines31/marketingskills (a 44-framework marketing-skill library) replaced ~915 tokens of always-loaded skill descriptions with a ~130-token router skill, with all 44 frameworks still reachable on demand — a measured ~7× reduction in always-loaded tokens at no loss of reachability. Structurally this is the recognition-vs-recall thesis at a different altitude: the catalog is the store, the router is the index, and only the winning entry is injected per turn — exactly how Instant Recall serves one anchor's cluster rather than scanning all memory.

The framing must stay honest. This is not a CORF benchmark, the corpus is a skill catalog rather than a personal memory store, and the ~7× number is an always-loaded-token reduction,

not a recall metric — it should not be reported alongside the §7.2 figures. What it does provide is a second, different-corpus data point that the "pre-compute the index, look up the winner" pattern generalizes beyond one personal-assistant memory store, which is precisely the cross-domain generalization §7.5 admits is otherwise unproven. The same library also ships per-skill `evals/evals.json` (realistic prompts plus assertion lists, runnable by an LLM judge) and a shared context-contract file every skill reads before acting — concrete instances of the runnable-eval and contract-file disciplines this paper gestures at but has not yet operationalized for memory retrieval (§8.4).

6. Design Space Analysis

6.1 Alternative 1: Inverted Index (Classical IR)

Captures exact term matches but fails on synonymy and paraphrase. Preferred on edge devices where embedding compute is too expensive.

6.2 Alternative 2: All-Sources ANN

Fixes Source Blindness by searching across all memory sources, but still requires a well-formed query — Pronoun Blindness and Anchor Silence remain unaddressed. The 9-point gap between AS-ANN (0.76) and Instant Recall (0.85) in §7 isolates the benefit of concept-anchored clustering. AS-ANN also runs at inference time, adding latency.

6.3 Alternative 3: Multi-Source RAG (MS-RAG)

MS-RAG queries all memory sources at inference time with full embedding search, achieving high recall (0.87 in our evaluation). However, it incurs 418ms p95 latency — 7.7× slower than Instant Recall — because it performs exhaustive embedding comparison across all sources at every query. Instant Recall trades 2 percentage points of recall for an order-of-magnitude latency reduction.

6.4 Alternative 4: Knowledge Graph Overlay

Preferred when relationship types between concepts matter (e.g., causal chains, temporal ordering). The durable backlink edges of §4.9 are a step in this direction; richer typed edges are a natural extension for structured domains (future work, §8.6).

6.5 Alternative 5: Hosted Vector Databases (Pinecone, Weaviate)

Dimension	Hosted Vector DB	Instant Recall
Retrieval model	Flat similarity search (O(log n) via HNSW)	Pre-computed anchor→chunk cache (O(1))
Cost	$8+/month (starter tier)	Zero ongoing cost (local compute)
Privacy	Embeddings sent to cloud	All data remains local
Latency	~50–200ms (network round-trip)	~50ms (local, no network)

Vector databases are preferred for large-scale deployments ($n > 100$k chunks) or when query-time semantic flexibility is critical.

6.6 A Compression Layer is Not an Alternative — It is a Substrate

It is worth distinguishing Instant Recall from a context-compression layer such as headroom's CCR (§2.4), because the two are easy to confuse as competitors when they are in fact stacked. A compression layer answers "how do I fit selected content into the window losslessly"; Instant Recall answers "which of months of stored content should be selected at all." Compression operates on the active working set of a single run and is specialized by content type (JSON, code, logs); Instant Recall operates on the persistent cross-session store and is specialized by concept anchor. They compose cleanly: CCR-style reversible compression as the storage layer (§1.1) feeding `(content, timestamp, source)` triples into the Instant Recall index above it. Treating compression as a design *alternative* would be a category error; it is a substrate.

7. Evaluation

7.1 Evaluation Protocol

Test Setup: Evaluated on a real personal assistant deployment with 47,392 memory chunks across memory/ (52%), archive/chatgpt-import/ (28%), emails/ (15%), other (5%). Test set: 612 queries annotated with ground-truth target chunks by two independent annotators (Cohen's $\kappa = 0.87$).

Baselines:

- **NR (No Retrieval):** No memory retrieval; model sees only base context.
- **SS-RAG (Single-Source RAG):** Standard RAG over the primary memory directory only.
- **AS-ANN (All-Sources ANN):** ANN search across all memory sources at inference time.
- **MS-RAG (Multi-Source RAG):** Full embedding search across all sources at inference time — the highest-recall but highest-latency baseline.
- **MG-REC (MemGPT-style Sequential):** Sequential multi-step retrieval with virtual context paging [Packer et al., 2023].

7.2 Results

Metric	NR	SS-RAG	AS-ANN	MG-REC	MS-RAG	Instant Recall
CORF-Recall@20	0.12	0.51	0.76	0.79	**0.87**	0.85
Source Coverage Rate	0.25	0.33	1.00	1.00	1.00	1.00
Inference Latency (p95, ms)	<1	178	94	320	418	**54**
FPR (post-rerank)	—	0.28	0.31	0.19	0.22	**0.18**
Index Build Time	—	—	45s	—	—	12s
Same-Day Index Freshness	—	—	24h	—	—	<2min

Key findings:

- **Recall–latency tradeoff.** MS-RAG achieves the highest recall (0.87) but at 418ms — 7.7× slower than Instant Recall. For personal assistant workloads where sub-100ms retrieval is required for responsive interaction, Instant Recall's 0.85 recall at 54ms is a favorable operating point on the Pareto frontier.

- **Concept anchoring value.** The 9-point gap between AS-ANN (0.76) and Instant Recall (0.85) isolates the benefit of pre-computed concept clustering beyond source coverage.
- **False positive control.** Post-retrieval re-ranking gives Instant Recall the lowest FPR (0.18) of any method achieving >0.75 recall.
- **MS-RAG ceiling.** MS-RAG's 2-point recall advantage over Instant Recall comes entirely from queries where no anchor term matches the relevant concept — a limitation addressable through vocabulary expansion rather than architectural change.

7.3 Ablations

Variant	CORF-Recall@20	Latency (p95, ms)	FPR
Full Instant Recall	0.85	54	0.18
A1 — w/o pronoun expansion	0.79	52	0.19
A2 — w/o post-retrieval re-ranking	0.85	41	0.35
A3 — nightly-only (no real-time episodic)	0.82	54	0.18

Pronoun expansion contributes 6 points of recall (Mode 1 mitigation). Re-ranking does not affect recall but nearly halves FPR (0.35 → 0.18). The episodic tier contributes 3 points of recall for within-day queries.

7.4 Implementation Status

The reference implementation is TypeScript (ESM, Node 22+), organized around an enhancement module (importance scoring, episodic buffer, deduplication, ~480 lines), a rebuild module (nightly orchestration, anchor extraction, incremental mtime-diff state, ~470 lines), and a co-located unit test suite. The beyond-cosine signals of §4.9 are present in the live schema and ranking path: bi-temporal validity columns (`validity_start`, `validity_end`, `ingestion_time`, `superseded_by`) and a covering index, with a non-lossy supersede routine that closes a prior fact's interval rather than deleting the row; a verification-status column with a coverage-mapping batch and a verification-aware re-rank applying the ×1.5/×1.2/×1.0/×0.5 multipliers; and a `backlinks` table with backlink hydration and a depth-capped k-hop walk on the manager. The low-confidence reformulation gate and concept→keyword routing of §4.12 are designed but not yet wired into the live pipeline. The contradiction-detection trigger for validity (§4.9, §4.9a) is likewise not live, and — per the frozen-embedding caution of §4.9a — is scoped to require an entailment/NLI judgement model rather than a cosine threshold. Exact line counts and test totals vary as the code evolves; the architectural claims above are the stable contract.

7.5 Threats to Validity

Single-deployment evaluation. All results come from one personal assistant deployment with one user. Interaction patterns, source distributions, and vocabulary characteristics may not generalize to other domains (e.g., enterprise knowledge management, multi-user systems). We mitigate this partially through ablation analysis, which shows each architectural component contributes independently, and we now offer a second, different-corpus data point: the recognition-over-recall pattern transposed to a skill catalog (§5.3) reproduces the core "pre-compute the index, inject only the winner" behaviour at a ~7× always-loaded-token reduction. That data point is suggestive of cross-domain generality but is not a CORF measurement, so cross-deployment *retrieval-metric* validation remains necessary future work.

Annotator bias. Both annotators had familiarity with the system, potentially inflating relevance judgments. The high inter-annotator agreement ($\kappa = 0.87$) suggests consistency but does not rule out shared bias.

No externally reproducible benchmark. The headline 0.85@54ms numbers rest entirely on one user's private logs. Adjacent open-source systems demonstrate this bar can be cleared: headroom (§2.4) publishes a runnable harness (`python -m headroom.evals`) over public datasets, and marketingskills ships per-skill runnable eval files. Until the synthetic CORF generator of §8.4 ships as an equivalently runnable suite over a public controlled corpus, the §7.2 results should be read as a single-deployment report, not an independently reproducible benchmark.

Beyond-cosine signals not isolated in the benchmark. The validity, trust, and link signals of §4.9 are implemented in the live ranking path, but the §7.2 benchmark scores CORF-Recall under the base anchor-and-cosine pipeline; the marginal retrieval-quality contribution of each beyond-cosine signal is not yet isolated in an ablation. Quantifying each is the next evaluation priority. The §4.12 reformulation gate, being a proposal rather than live code, is likewise unbenchmarked.

8. Discussion

8.1 Limitations and Mitigation Strategies

Anchor vocabulary brittleness. Paraphrased queries bypass the anchor vocabulary unless fuzzy matching or embedding-based anchor detection is used. With stemming + phonetic matching, paraphrase robustness improves from 68% to 82%.

Semantic ambiguity. The anchor "paper" could refer to research papers, manuscripts, or paper-based workflows. Mitigation: sense-specific sub-clustering via contextual embeddings. Deployment results: FPR drops from 0.18 to 0.14 with sub-clustering. Because sub-clustering is a distance operation over the frozen embedding model, it lives in the *structural* regime §4.9a validates and does not require a judgement model.

K selection. Fixed K=20 may be too small or too large depending on anchor specificity. Adaptive K based on score distribution within a cluster reduces context overhead by 15% with no recall loss.

Contradiction detection boundary. The validity machinery (§4.9) can close a fact's interval, but *deciding* that a new fact contradicts an old one is a semantic-entailment judgement, which §4.9a shows a cosine threshold over a frozen embedding model is unlikely to carry. The honest requirement is an NLI/entailment model or a fine-tuned head; until such a detector is in place, validity intervals are only as good as the heuristics that call the invalidation routine, and those heuristics should not be confused with a calibrated contradiction signal.

8.2 Large-Scale Deployment (n > 100,000 chunks)

For n > 100k chunks: (1) replace exact KNN with FAISS IVF at build time; (2) use differential builds (track chunk hashes changed since last build); (3) use disk-backed embedding store (DiskANN/ScaNN); (4) shard by source for parallel builds.

Corpus Size	Build Time (full)	Index Size	p95 Latency
50k (deployed)	12s	150KB	54ms
100k (projected, FAISS IVF)	8min	500KB	58ms

Corpus Size	Build Time (full)	Index Size	p95 Latency
1M (projected, DiskANN)	90min	6.2GB	62ms†

†Projected from extrapolation; not empirically validated at this scale. The reference deployment is approaching the 50k ceiling; scaling observations beyond it are forthcoming rather than reported here.

8.3 Ethical Considerations

Instant Recall indexes personal interaction history, including potentially sensitive content. The local-first architecture (no cloud embedding storage, no external API calls for retrieval) mitigates data exfiltration risks. However, the index itself — a JSON file mapping anchor terms to memory paths — is a condensed summary of what the agent knows about its user. Deployments should encrypt the index at rest and restrict file-system access. The importance scoring system (§4.8) introduces implicit content prioritization that could surface sensitive memories preferentially; operators should review weight configurations for their context. The bi-temporal store (§4.9) also retains superseded facts indefinitely; a retention or hard-delete policy for closed validity intervals is a deployment responsibility.

8.4 Reproducibility

The annotated CORF dataset is drawn from one deployment's logs and is small, which §7.5 flags as the paper's central validity threat. Adjacent open-source systems show the higher bar is reachable: headroom (§2.4) ships a runnable accuracy-eval harness (`python -m headroom.evals`) over public datasets (GSM8K, TruthfulQA, SQuAD, BFCL) and reports an accuracy-delta number anyone can re-run, and marketingskills (§5.3) ships per-skill runnable eval files. We therefore treat reproducibility as a bar to clear rather than a plan to gesture at. The planned synthetic CORF generator — seeding pronoun-blind, source-split, and anchor-silent queries against a controlled corpus — should ship as a `python -m`-style runnable suite that reports CORF-Recall@20, FPR, and latency on a *public* controlled corpus, so that the headline 0.85@54ms numbers no longer rest entirely on one user's private logs. The synthetic set would not replace the deployment benchmark but would make the failure-mode taxonomy testable in isolation by anyone, which is the property the §7.5 threat-to-validity currently lacks.

9. Conclusion

This paper introduced CORF as a formal framework for diagnosing retrieval failures in persistent agents and showed that the dominant failure mode is architectural — rooted in source fragmentation and anchor vocabulary gaps — rather than embedding quality. The source-coverage upper bound (§3.3) makes this precise: no amount of query refinement compensates for unsearched sources.

Instant Recall resolves these failures by shifting retrieval from inference time to build time. A pre-computed concept index, rebuilt nightly and kept fresh by real-time episodic updates, maps anchor vocabulary terms to pre-ranked memory clusters. At inference time, a ~50ms lookup surfaces relevant memories before the model processes the prompt — achieving 0.85 CORF-Recall@20 at the lowest latency of any evaluated system. Concept anchoring accounts for 9 points of recall beyond what source aggregation alone provides, and post-retrieval re-ranking halves the false positive rate. Beyond similarity, the index weights validity over time, code-trust, and inter-chunk links — because the closest chunk is not always the right one, and a fact that

was true last month may be false today. We are explicit about what a frozen embedding model can carry: the structural distance operations the architecture depends on are on firm ground, while the judgement operations it proposes (contradiction detection, calibrated retrieval-failure confidence) require a fine-tuned or entailment model, not a cosine threshold. And we place Instant Recall in the live open-source stack — above a reversible-compression storage layer like headroom's CCR, alongside the same recognition-over-recall pattern that lets a router skill replace a token-heavy catalog — rather than against a static citation list.

The deeper insight is architectural: Instant Recall transforms agent memory from a *search problem* into a *recognition problem*. The agent stops asking "what papers are you referring to?" — because the answer was already in context before the question was asked. It is the indexing organ, not the store; and like its biological model, it is the difference between having memories and being able to use them.

References

1. Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *NeurIPS 2020*.
2. Packer, C., et al. (2023). MemGPT: Towards LLMs as Operating Systems. *arXiv:2310.08560*.
3. Sarthi, P., et al. (2024). RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. *ICLR 2024*.
4. Collins, A. M., & Loftus, E. F. (1975). A spreading-activation theory of semantic processing. *Psychological Review*, 82(6), 407–428.
5. Teyler, T. J., & DiScenna, P. (1986). The hippocampal memory indexing theory. *Behavioral Neuroscience*, 100(2), 147–154.
6. Squire, L. R. (1992). Memory and the hippocampus: A synthesis from findings with rats, monkeys, and humans. *Psychological Review*, 99(2), 195–231.
7. Wilson, M. A., & McNaughton, B. L. (1994). Reactivation of hippocampal ensemble memories during sleep. *Science*, 265(5172), 676–679.
8. Tulving, E. (1976). Ecphoric processes in recall and recognition. *Recall and Recognition*, 37–73.
9. Mandler, G. (1980). Recognizing: The judgment of previous occurrence. *Psychological Review*, 87(3), 252–271.
10. Yonelinas, A. P. (2002). The nature of recollection and familiarity: A review of 30 years of research. *Journal of Memory and Language*, 46(3), 441–517.
11. MemoryOS. (2024). Mem0: The Memory Layer for Personalized AI. *arXiv:2504.19413*.
12. Park, J. S., et al. (2023). Generative Agents: Interactive Simulacra of Human Behavior. *UIST 2023*.
13. Zhong, W., et al. (2023). MemoryBank: Enhancing Large Language Models with Long-Term Memory. *arXiv:2305.10250*.
14. Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
15. Guu, K., et al. (2020). REALM: Retrieval-Augmented Language Model Pre-Training. *ICML 2020*.
16. Borgeaud, S., et al. (2022). Improving language models by retrieving from trillions of tokens. *ICML 2022*.
17. Xu, X., et al. (2025). A-MEM: Agentic Memory for LLM Agents. *arXiv:2502.12345*.
18. Wu, D., et al. (2024). LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. *arXiv:2410.10813*.
19. Yan, Z., et al. (2025). Memory-R1: RL-Trained Memory Management for LLM Agents. *arXiv:2505.14075*.

20. Gama, J., et al. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4), 1–37.
21. Scoville, W. B., & Milner, B. (1957). Loss of recent memory after bilateral hippocampal lesions. *Journal of Neurology, Neurosurgery & Psychiatry*, 20(1), 11–21.
22. Chopra, T., et al. (2025). headroom: Reversible Compress-Cache-Retrieve Context Compression for Agents. Open-source software (Apache-2.0); Kompress-v2-base model, HuggingFace.
23. Osmani, A., et al. (2025). agent-skills: A Discipline Library for Coding Agents. Open-source software.
24. Haines, C., et al. (2025). marketingskills: An On-Demand Marketing-Framework Skill Catalog with Per-Skill Eval Harnesses. Open-source software.

References