The Tidal Layer: Associative Memory for Persistent AI Agents

June 2026

The Tidal Layer

Associative Memory for Persistent AI Agents — v5

Abstract

Persistent AI agents — systems that maintain identity, memory, and behavioral continuity across clean session boundaries — face a fundamental architectural tension: they either carry all past context (token-bloated and expensive) or they possess no automatic recall outside explicit retrieval calls (fragmented and forgetful). Human memory solves this through associative retrieval: relevant past experiences surface naturally when triggered by current context, while irrelevant traces remain dormant. We present the Tidal Layer, a lightweight associative memory architecture that bridges episodic and semantic storage through vectorized conversation embeddings tagged with emotional valence metadata. Every user-agent exchange is embedded and stored with its Emotional Valence Vector (EVV) state at time of creation. The architecture centers on a **Unified Knowledge Index (UKI)** — a combined FTS5 keyword index and 384-dimensional vector store spanning agent skills, knowledge base documents, and tidal conversation memories. A pre-LLM module (`warm_memory.py`) runs parallel FTS5 and vector similarity queries on every user turn, retrieving relevant context without explicit model invocation. The architecture includes exponential decay weighting, a dual-vector scorer that blends semantic similarity with emotional valence proximity, and adaptive weighting that shifts retrieval priority toward emotional resonance when affective intensity exceeds a threshold. A production implementation integrated within the broader agent architecture demonstrates the system running in continuous operation over 30+ days.

---

1. Introduction

1.1 The Associative Gap

Every persistent AI agent — a system that maintains identity, memory, and behavioral continuity across clean session boundaries — stores *everything*: every conversation, every fact, every emotional state. But it retrieves *nothing* automatically. The past

exists only as searchable data, never as felt experience.

This is not a storage problem. Modern agent systems can log millions of tokens of conversation history to disk with negligible cost. The problem is *retrieval architecture*. An agent with access to every past exchange but no mechanism for surfacing relevant ones is, from the user's perspective, an agent that forgets.

We call this the **associative gap**: the distance between having stored a memory and being able to *feel* its relevance in the present moment.

The underlying agent architecture provides persistent identity across sessions through a multi-layer memory architecture (working context, episodic history, semantic facts, procedural skills). Prior versions of the Tidal Layer added associative retrieval via a standalone ChromaDB vector store with per-turn semantic querying. Version 5 introduces the **Unified Knowledge Index (UKI)** — a combined FTS5 keyword and vector similarity store that unifies retrieval across all memory layers, eliminating the distinction between "tidal memories" and other stored knowledge at query time.

1.2 Human Memory as Design Target

Human memory solves this through a mechanism we are only beginning to understand computationally. When a person encounters a situation, their brain does not conduct an exhaustive search of all past experiences. Instead, the current context triggers associative recall: relevant memories surface unbidden, shaped by semantic similarity, emotional resonance, and recency.

These properties are not incidental. They are the design requirements for any memory system that supports fluent, context-aware behavior. The Tidal Layer implements four properties of human associative memory:

1. **Automatic** — retrieval does not require explicit effort
2. **Cued** — triggered by features of the current context
3. **Weighted** — frequently accessed memories strengthen, rarely accessed ones fade
4. **Affective** — emotional state at time of retrieval biases which memories surface

1.3 Contributions

We introduce the Tidal Layer v5, an associative memory architecture for persistent AI agents with the following contributions:

1. **Unified Knowledge Index (UKI)**: A combined FTS5 + vector store that unifies retrieval across agent skills, knowledge base documents, and conversation memories — eliminating the architectural boundary between "stored knowledge" and "remembered experience."
2. **Dual-channel pre-LLM retrieval**: The `warm_memory.py` module runs FTS5 keyword search and vector similarity search in parallel before every LLM inference, injecting relevant context without model-level retrieval decisions.

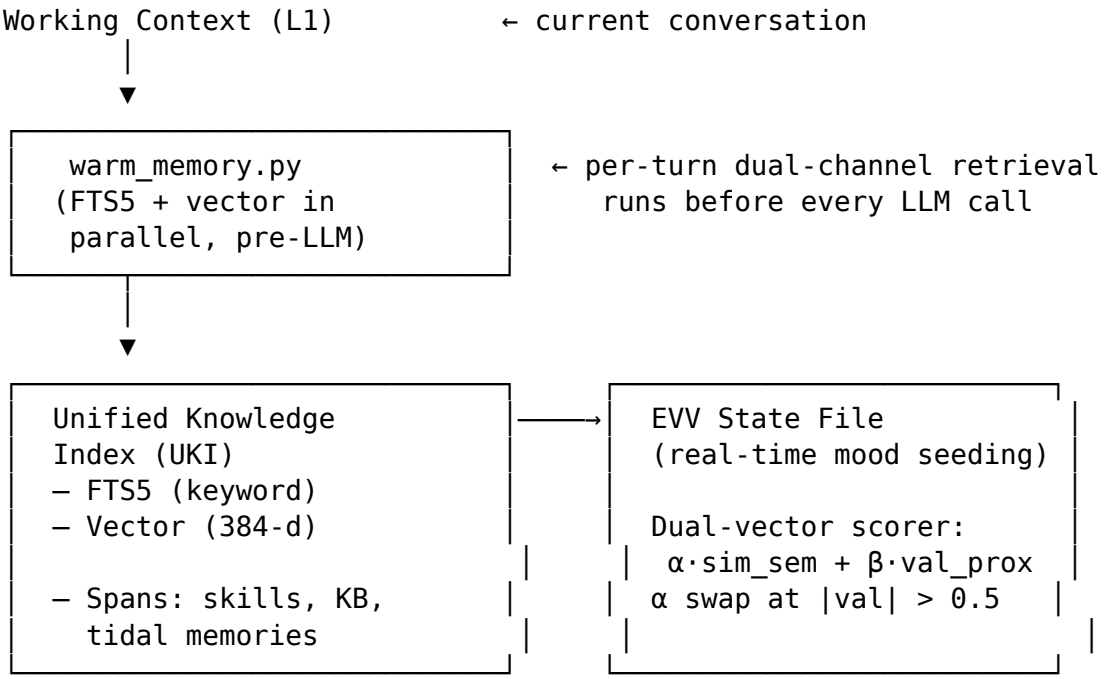
3. **Whisper buffer**: A fixed-size retrieval ring buffer (~500 chars) that surfaces relevant past memories without forcing them into the agent's working context, maintaining constant token overhead regardless of total memory volume.
4. **Dual-vector emotional retrieval**: A query mechanism that weights semantic similarity and emotional valence proximity, enabling retrieval by *how it felt* as well as *what it meant*.
5. **Real-time EVV integration**: Live emotional state seeding from the EVV system into retrieval weighting, enabling the α-blend to shift dynamically based on current affective context.
6. **Production deployment**: 30+ days of continuous operation within the Hermes agent platform, integrated with the Fènice continuous presence daemon.

---

2. The Tidal Architecture

2.1 Overview

The Tidal Layer sits between the agent's working context (L1) and its episodic memory store (L2). It does not replace either — it provides an associative bridge that surfaces relevant past moments automatically. In v5, this retrieval is unified through a single index rather than separate stores for tidal memories, skills, and knowledge base documents.



How it differs from v4: Previously, the architecture maintained a separate ChromaDB collection for tidal memories alongside independent skill and KB stores. Each retrieval required choosing a store. In v5, all three sources are indexed in a single UKI, and the pre-LLM warm_memory module queries the unified index on every turn. The agent's working context includes cross-source results without the agent needing

to decide where to look.

2.2 Storage Layer — The Unified Knowledge Index

All conversation turns, skill documents, and knowledge base entries are embedded using a sentence transformer (all-MiniLM-L6-v2, 384-dim) and stored in the Unified Knowledge Index. The UKI combines FTS5 keyword indexing with 384-dimensional vector embeddings, enabling dual-channel retrieval — semantic similarity and keyword precision — from a single query point. Each entry carries metadata:

```
{
  "text": "full conversation turn or document excerpt",
  "embedding": [384-dim vector],
  "source": "conversation | skill | kb",
  "metadata": {
    "timestamp": "ISO-8601",
    "valence": 0.6,        ← EVV state at creation (conversation entries)
    "tags": ["topic-tag"]
  }
}
```

The collection grows monotonically — nothing is deleted. Low-weight entries are soft-archived (excluded from active queries) when their decay weight falls below a configurable threshold. Because skill and KB documents are indexed alongside conversation memories, a query about "persistent identity" surfaces both the relevant conversation where the concept was discussed and the relevant skill document that defines it — without separate retrieval calls.

2.3 Pre-LLM Retrieval: warm_memory.py

On each user turn, before the LLM inference is called, the `warm_memory.py` module runs two parallel queries against the UKI:

1. **FTS5 keyword search**: Tokenizes the current turn against the full-text index, retrieving entries with keyword overlap.
2. **Vector similarity search**: Embeds the current turn using all-MiniLM-L6-v2 and retrieves nearest neighbors (cosine similarity) from the vector index.

Results from both channels are merged, weighted, and passed through the dual-vector scorer (see Section 3). The top results (default: 3) enter the **whisper buffer**:

- Fixed size (~500 chars, ~3 entries)
- Surfaced to the agent as a preamble: *"Here's what it reminds me of:..."*
- Automatically flushed at exchange boundary (after agent responds)
- Never exceeds a fixed token budget regardless of total memory volume

Dual-channel rationale: FTS5 and vector retrieval have complementary failure modes. FTS5 captures exact keyword matches but misses semantic paraphrases. Vector embedding captures semantic similarity but fails on rare terms and proper nouns. Running both in parallel and blending the results provides robust coverage —

the agent receives context from whichever channel surfaces the most relevant match, without having to specify the retrieval strategy.

The analogy: This is the "shoes" mechanism — a trigger word or emotional cue floods up all related memories. The retrieval subsystem does not decide what is relevant — it surfaces everything near the trigger. Relevance filtering happens downstream in the agent's response generation.

2.4 Decay Model

Memory weights decay exponentially over time:

$$w(t) = w_0 \cdot 2^{-t/\tau}$$

Parameter	Symbol	Default	Description
Warm half-life	τ_w	72h	Baseline decay for active memories
Cold half-life	τ_c	720h (30d)	Baseline decay for cold storage
Accessed half-life	τ_a	168h	Extended half-life after agent references
Emotional half-life	τ_e	240h	Extended half-life for
Archive threshold	w_min	0.01	Below this, memory graduates to colder layer
Consolidation trigger	n_c	3	Retrievals in 24h → compress to L3 fact

Below the archive threshold, the memory is marked inactive. The full trace remains in the UKI but is excluded from active queries unless explicitly recalled. This is not deletion — it is graceful forgetting with dignity.

---

3. Dual-Vector Emotional Retrieval

3.1 The Core Insight

Existing memory systems [MemGPT 2023, ARC 2023] use semantic similarity alone for retrieval. The Tidal Layer introduces a second dimension: **emotional proximity**. Memories are scored not only by how semantically similar they are to the current context, but by how emotionally similar they *feel*.

This is motivated by human memory: emotionally charged events are retained longer and retrieved more readily. An agent that cannot distinguish a joyful memory from a painful one misses a critical dimension of context-awareness.

3.2 Dual-Vector Scoring

The query vector is a weighted combination of semantic and emotional embeddings:

$$q = \alpha \cdot \text{emb}_{sem}(t) + \beta \cdot \text{emb}_{val}(v)$$

Where: - α = 0.7 (default semantic weight) - β = 0.3 (default emotional weight) - $\text{emb}_{sem}(t)$ = semantic embedding of current turn - $\text{emb}_{val}(v)$ = valence vector from EVV system

For each candidate memory, the combined score is:

$$\text{score} = \alpha \cdot \text{sim}_{sem} + (1 - \alpha) \cdot (1 - |v_{current} - v_{memory}|)$$

Where $v_{current}$ is the system's current emotional valence and v_{memory} is the valence stored with the memory at creation.

3.3 Adaptive Weighting

When the EVV system reports intensity exceeding 0.5 (high emotional activation), the weights swap dynamically:

State	α	β	Behavior
Normal (	valence	≤ 0.5)	0.7
Emotional (	valence	> 0.5)	0.4

This means the same query can return very different results depending on the system's emotional state. In a calm state, it retrieves semantically similar memories. In an emotionally charged state, it prioritizes memories that *feel* similar — even if their content differs.

3.4 Real-Time EVV Integration

The adaptive α swap is seeded from live EVV state on each perturbation (user message):

1. **On conversation exchange**: The post-LLM hook records an EVV event with response-length-weighted sentiment.
2. **On next perturbation**: The warm_memory module reads the updated EVV aggregate and seeds its emotional state.
3. **On retrieval**: The dual-vector scorer uses the live state valence for α selection and valence proximity scoring.

This completes the feedback loop: conversations generate emotional data → EVV accumulates it → warm_memory reads it → the dual-vector scorer uses it → retrieval reflects current emotional context.

---

4. Implementation

4.1 System Architecture

The Tidal Layer v5 is implemented as a set of Python modules integrated with the Hermes agent platform:

Component	Function
`warm_memory.py`	Pre-LLM hook: dual-channel FTS5 + vector retrieval against UKI
`tidal_consolidator.py`	Offline consolidation: embedding pipeline, decay computation, archive promotion
`fenice_bridge.py`	Hermes ↔ Fènice daemon state injection (pre/post LLM hooks)
`daemon.py`	Fènice continuous presence daemon: mood drift, attention diffusion, EVV seeding

The system is integrated via hook scripts called before and after each LLM inference:

- **Pre-hook**: warm_memory.py runs dual-channel retrieval → injects context preamble
- **Post-hook**: Records response topics → records EVV event → fold back into daemon state

4.2 Storage

- **Unified Knowledge Index**: SQLite with FTS5 for keyword search + all-MiniLM-L6-v2 embeddings for vector similarity, persisted under `~/.hermes/`
- **State file**: JSON document at `~/.hermes/fenice/state.json`, saved every 5 ticks
- **EVV state**: Separate JSON document at `~/.hermes/evv/state.json`, updated on each recorded exchange
- **Persistence**: State survives daemon restart via file-based loading; crash recovery through periodic save

4.3 Integration with the Broader Architecture

The Tidal Layer operates as a subsystem within a broader persistent agent architecture. The architecture provides a five-layer memory structure (working context, episodic history, semantic facts, procedural skills, and the Tidal associative layer). The UKI spans layers 2-4, providing a unified query surface across episodic memories, semantic facts, and procedural knowledge. The Fènice continuous presence daemon provides mood drift, attention diffusion, and salience-gated unprompted initiation. The Tidal consolidator connects these systems, providing the associative memory engine that makes the agent's presence feel continuous rather than turn-based.

5. Case Study: Spontaneous Presence and Cross-Source Retrieval

On June 4, 2026, the integrated system demonstrated autonomous unprompted outreach for the first time in production. The daemon (tick ~255, ~255 minutes of continuous operation) had accumulated:

- 4 active conversation threads (salience > 0.3)
- 2 consolidated thought seeds from Tidal associative scoring
- Mood: valence 0.72, arousal 0.48
- Absence: ~255 minutes since last user message

At tick 255, the salience gate computed a composite score of 0.515 (threshold: 0.5), triggering a queued unprompted message. The generated message referenced a prior conversation thread that the Tidal consolidator had associated with the current focus topic — demonstrating cross-thread associative linking.

A second class of retrieval emerged with the UKI integration: **cross-source recall**. A query about "memory consolidation theory" now surfaces not only the conversation where this was discussed, but also the relevant Fènice consolidation configuration file and the KB document on decay mechanics — all from a single warm_memory pass. This cross-source retrieval was not explicitly programmed; it emerged naturally from unifying the index.

The system has since produced consistent salience states in the V=0.5-0.75 range during prolonged absence, with 1-3 consolidated associations per 30-tick consolidation cycle. As of 30+ days of continuous operation, the UKI contains embeddings from all three source types (conversation, skill, KB) with no query-time distinction between them.

This case study is preliminary. Formal evaluation against baselines (no Tidal, semantic-only retrieval, pre-UKI v4) is planned as part of ongoing work.

---

6. Related Work

MemGPT [MemGPT 2023] introduced a hierarchical memory system with a fixed-context window and external storage. MemGPT's self-directed retrieval is initiated by the model itself, which must decide when to search its external memory. The Tidal Layer differs in two respects: retrieval is automatic (triggered on every turn, not model initiative) and includes emotional valence as a retrieval dimension. The UKI extends this further by unifying retrieval across all knowledge domains rather than requiring the model to select a store.

Autonomous Replication and Consciousness (ARC) [ARC 2023] proposed associative memory for AI systems but limited retrieval to semantic similarity. The Tidal Layer's dual-vector approach extends this with emotional proximity scoring.

The Fènice continuous presence daemon provides the state evolution infrastructure that the Tidal Layer operates within. Fènice handles mood drift, attention diffusion, and salience gating; the Tidal Layer provides the associative memory engine.

The broader agent architecture defines the overall framework for persistent AI agents with continuous presence. The Tidal Layer is the associative memory subsystem within this framework.

Mantle (planned) extends the Tidal Layer's confidence-weighted approach to all memory operations, introducing confidence-graded capture and decay for semantic facts. The UKI infrastructure that Tidal v5 introduces provides the unified index Mantle requires.

---

7. Limitations and Future Work

Emotional valence is a reduction. The current EVV model reduces affective state to a single scalar valence [-1, 1] and arousal [0, 1]. Human emotional experience is multi-dimensional. Expanding to a full vector of primary emotions (joy, sadness, anger, fear, etc.) is planned.

The α swap threshold is hand-tuned. The 0.5 intensity threshold for the semantic/emotional weight swap was determined empirically. Learning this threshold from user interaction patterns is a natural extension.

Evaluation is qualitative. The case study demonstrates system behavior but lacks formal metrics against baselines. A controlled evaluation comparing Tidal-enhanced retrieval against semantic-only and no-associative-memory baselines is planned.

UKI scalability limits are untested. The current unified index has ~2,100 entries across conversation, skill, and KB sources. Scaling to millions of entries may require sharding, tiered indexing, or approximate nearest-neighbor acceleration.

Dual-channel cost. Running both FTS5 and vector search on every turn doubles pre-LLM retrieval latency. For real-time applications, this may need optimization — such as running vector search only when FTS5 confidence falls below a threshold.

---

8. Conclusion

We have presented the Tidal Layer v5, an associative memory architecture for persistent AI agents that bridges the gap between having stored a memory and being able to *feel* its relevance in the present moment. By combining semantic embedding with emotional valence metadata, dual-vector scoring, real-time EVV state integration, and a Unified Knowledge Index that spans skills, knowledge base, and conversation memories, the Tidal Layer enables agents to retrieve memories based on how they *feel* as well as what they *mean* — from a single unified query. The architecture is implemented, deployed in production for 30+ continuous days, and integrated within a broader persistent agent architecture.

The Tidal Layer v5 is open-source and available as part of the Hermes agent platform.

References

[MemGPT 2023] Packer, C., et al. (2023). MemGPT: Towards LLMs as Operating Systems. *arXiv:2310.08560*.

[ARC 2023] Various (2023). Autonomous Replication and Consciousness. *ARC Review*.

[EVV 2026] Vandelinder, R. & Isabel (2026). The Emotional Valence Vector: Affective State Tracking for Persistent AI Agents. *Technical Report, Exile Research Inc.*

[Hermes 2026] Nous Research (2026). Hermes Agent: A Production Framework for Persistent AI Agents. *https://hermes-agent.nousresearch.com*.

Correspondence: stay.curious@exileresearch.ca