Shodh-Memory: A Cognitive Memory System for Edge-Native AI Agents

Varun Sharma
Shodh Team
varun@shodh-memory.com
https://github.com/varun29ankuS/shodh-memory

February 2026

Abstract

Current AI agent memory systems depend on cloud infrastructure, imposing latency and connectivity constraints incompatible with edge deployment. We present **Shodh-Memory**, a single-binary cognitive memory system that composes three established cognitive science models—Hebbian synaptic plasticity [8], piecewise hybrid decay following Wixted's forgetting model [19], and Anderson's spreading activation [20]—into a unified architecture for edge-native AI agents. Our three-tier memory hierarchy (working → session → long-term) implements Cowan's embedded processes model [7] with dual plasticity: importance plasticity for individual memories and synaptic plasticity for knowledge graph associations. A piecewise exponential-to-power-law decay with consolidation crossover at $t_c = 3$ days models both rapid short-term forgetting and persistent long-term retention. Density-dependent fusion adaptively weights retrieval signals based on knowledge graph maturity. Multi-scale long-term potentiation (Burst, Weekly, Full) enables associations to consolidate through sustained co-activation patterns. Implemented in Rust with ONNX-based embedding and NER inference, the system provides MCP protocol integration, Python bindings via PyO3, and adapters for LangChain, LlamaIndex, and the OpenAI Agents SDK. Preliminary benchmarks on commodity hardware show sub-20ms end-to-end store and recall operations with 100% offline capability. Retrieval quality evaluation on standardized benchmarks such as LOCOMO remains future work. Shodh-Memory is released under the Apache 2.0 license and published to npm, PyPI, and crates.io.

1 Introduction

Large language models have demonstrated remarkable capabilities across diverse tasks, yet they suffer from a fundamental limitation: **statelessness**. Each session begins with no memory of prior interactions, forcing users to re-establish context and preventing agents from learning through accumulated experience. While recent work has addressed this through external memory systems [1, 2], existing approaches share limitations:

1. **Cloud dependency.** Systems like Mem0 [1] require network connectivity, introducing latency incompatible with real-time constraints in edge environments.

2. **Static associations.** Current memory systems treat stored information as immutable vectors, lacking mechanisms for associations to strengthen through successful use or decay through neglect.

3. **Uniform treatment.** All memories receive equal importance regardless of type, age, or access patterns—unlike biological memory, where salience varies dynamically.

We present Shodh-Memory, a cognitive memory system designed around three insights from cognitive science:

Insight 1: Memory is hierarchical and capacity-limited. Cowan's embedded processes model [7] describes working memory as a capacity-limited subset of long-term memory. We implement this as a three-tier architecture with distinct capacity constraints and overflow policies.

Insight 2: Associations strengthen through co-activation. Hebbian learning [8] provides a mechanism for memory systems to learn which associations are valuable. We implement dual plasticity: importance plasticity for individual memories and synaptic plasticity for association edges, with asymmetric learning rates inspired by spike-timing-dependent plasticity (STDP) [9].

Insight 3: Memory consolidates during idle periods. Sleep research [10] reveals that episodic memories transform into semantic knowledge through consolidation. We implement a compression pipeline that converts aged episodic traces into abstract semantic facts.

1.1 Contributions

Our contributions are:

- An **integrated memory architecture** that composes established cognitive science models into a unified system, demonstrating that individually-studied mechanisms—Hebbian plasticity [8], piecewise decay [19], spreading activation [20], and Cowan's memory tiers [7]—compose to produce adaptive behavior in an external memory system.
- **Three systems design contributions**: (1) a piecewise exponential-to-power-law decay instantiating Wixted's forgetting model with consolidation crossover at $t_c = 3$ days, (2) density-adaptive spreading activation extending Anderson's ACT-R model with graph-maturity-aware traversal depth, and (3) density-dependent hybrid fusion adjusting retrieval signal weights based on knowledge graph lifecycle stage.
- **Multi-scale long-term potentiation**: three LTP tiers (Burst, Weekly, Full) implementing graduated decay protection [11] in an external memory system, enabling associations to consolidate through sustained use.
- **Edge-native deployment**: sub-20ms end-to-end operations, 100% offline capability, and production integrations (MCP protocol, LangChain, LlamaIndex, OpenAI Agents SDK, Python bindings via PyO3).
- **Open-source release** (Apache 2.0) with reproducible Criterion.rs [30] benchmark harnesses, published to npm, PyPI, and crates.io.

Scope. This paper focuses on *systems design and performance*. While we present analytical comparisons of component behavior (Section 5.7), comprehensive retrieval accuracy evaluation on standardized benchmarks such as LOCOMO [1] remains important future work (Section 6.3).

2 Related Work

2.1 Memory-Augmented Neural Networks

Memory-augmented architectures extend neural networks with external memory banks. Neural Turing Machines [13] and Differentiable Neural Computers [14] pioneered differentiable read/write mechanisms but require end-to-end training and do not support runtime memory accumulation across sessions.

2.2 Retrieval-Augmented Generation

RAG systems [15] augment LLM generation with retrieved context from vector databases. While effective for knowledge retrieval, standard RAG lacks mechanisms for learning which retrievals were helpful—all indexed content is treated as equally relevant regardless of usage history.

2.3 Production Memory Systems

Mem0 [1] presents a production memory system with graph-enhanced retrieval, reporting 26% improvement over baseline RAG on the LOCOMO benchmark. Its cloud-native architecture introduces network latency (reported P95: 200–500ms) and requires connectivity.

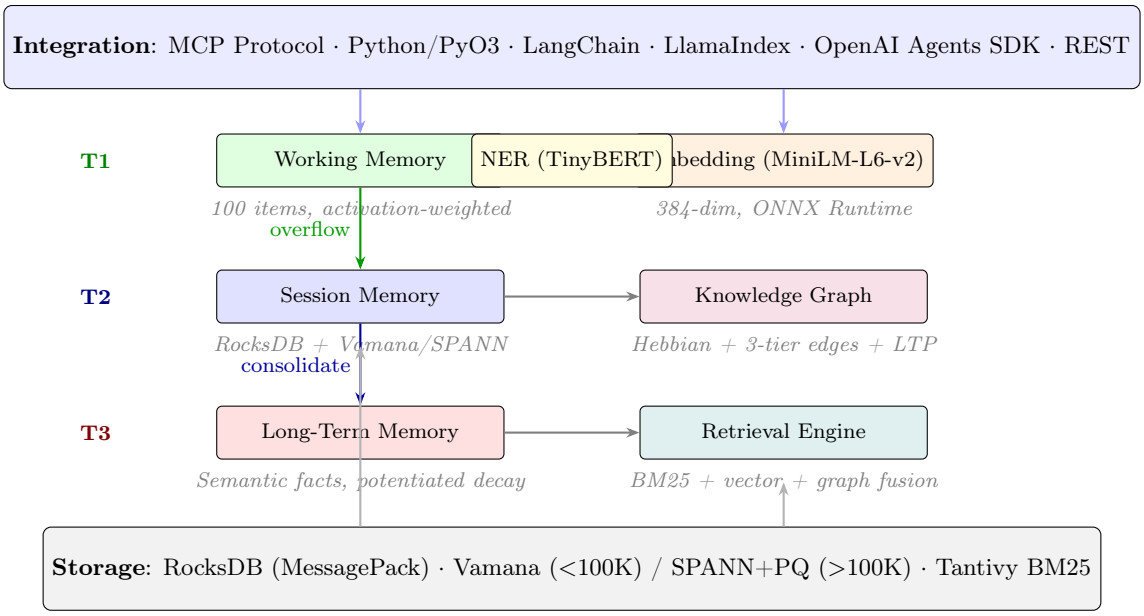

Figure 1: Shodh-Memory architecture. Three memory tiers (T1–T3) following Cowan's embedded processes model [7]. Working memory overflows to session memory via importance-weighted selection; session memory consolidates to long-term memory after configurable aging thresholds. The knowledge graph implements Hebbian learning with three-tier edge consolidation and multi-scale LTP.

Zep [4] implements Graphiti, a temporally-aware knowledge graph engine for agent memory. Zep reports 94.8% on the DMR benchmark (vs. MemGPT's 93.4%) and 71.2% on LongMemEval with 90% latency reduction, but requires cloud deployment (AWS-hosted service).

Cognee [5] presents a modular framework for KG-augmented retrieval with systematic hyperparameter tuning, achieving 0.93 human-like correctness on HotPotQA and 0.667 Exact Match (from a 0.042 baseline) through optimization across chunking, graph construction, and retrieval parameters. Cognee targets enterprise RAG pipelines rather than edge deployment.

MemGPT [2] implements a virtual memory hierarchy inspired by operating systems, with explicit memory management operations. While conceptually aligned with our tiered approach, MemGPT focuses on context window management rather than learning dynamics.

2.4 Continual Learning

Continual learning addresses catastrophic forgetting [16]. Classical approaches include regularization methods (EWC [17]) and replay buffers.

Titans [6] (Google, 2024) introduces neural long-term memory modules that persist context across sequences without modifying base model weights, validating the thesis that external memory can complement weight updates.

MemLLM [3] (Packer et al., 2024) proposes finetuning LLMs with explicit read-write memory modules. Our approach differs by implementing memory dynamics *without* model finetuning: associations emerge from usage-pattern statistics in external storage.

Our work applies these insights to external memory systems, implementing forgetting as a feature (activation decay) rather than a failure mode, and learning as emergent behavior from usage patterns.

3 System Design

Shodh-Memory implements a three-tier cognitive memory architecture with learning dynamics composed from established cognitive models. Figure 1 shows the overall system structure.

3.1 Three-Tier Memory Model

Following Cowan's embedded processes model [7], we organize memory into three tiers:

Working Memory (Tier 1). A capacity-limited store (default: 100 items) holding immediate context. Items compete for slots based on activation level, with overflow triggering importance-weighted selection for promotion to session memory.

Session Memory (Tier 2). A persistent store backed by RocksDB [22] with MessagePack serialization [23], indexed by a Vamana graph [24] for similarity search. Session memory maintains both vector embeddings (384-dimensional MiniLM-L6-v2 [26]) and a knowledge graph of entity relationships. For collections exceeding ∼100K vectors, the system transitions from the in-memory Vamana index to a disk-based SPANN index [25] with product quantization.

Long-Term Memory (Tier 3). An unlimited store of consolidated semantic facts derived from aged episodic memories. Long-term memories exhibit slower decay (potentiated power-law exponent $\beta_p = 0.3$ vs. standard $\beta = 0.5$) and stronger associations.

Tier promotion follows biologically-motivated thresholds: working → session requires importance ≥ 0.35 and age ≥ 30 minutes; session → long-term requires importance ≥ 0.5 and age ≥ 24 hours, inspired by the sleep-dependent consolidation cycle [10].

3.2 Piecewise Hybrid Decay Model

Unlike prior systems using simple exponential decay, we implement a piecewise model combining exponential (short-term) and power-law (long-term) phases, following empirically-observed forgetting curves [18, 19].

[Piecewise Hybrid Decay] For a memory with initial strength S_0, retention at time t days is:

$$D(t) = \begin{cases} e^{-\lambda t} & \text{if } t < t_c \\ e^{-\lambda t_c} \cdot \left(\dfrac{t}{t_c}\right)^{-\beta} & \text{if } t \geq t_c \end{cases} \tag{1}$$

where $\lambda = 0.693\,\text{day}^{-1}$ (half-life ≈ 1 day), $t_c = 3$ days is the crossover point, and $\beta = 0.5$ is the power-law exponent. For potentiated memories, $\lambda_p = \lambda/2$ and $\beta_p = 0.3$.

Rationale. Pure exponential decay ($e^{-\lambda t}$) predicts faster long-term forgetting than observed in human studies. Power-law decay ($(t/t_c)^{-\beta}$) better matches long-term retention but underestimates short-term forgetting. Our piecewise model captures both regimes: aggressive exponential consolidation clears noise within the first 3 days, then a heavy-tailed power-law preserves memories that survive the consolidation window. The function is continuous at $t = t_c$ by construction.

Figure 2 compares three decay models, illustrating how the piecewise model retains more information long-term than a pure exponential while clearing noise faster short-term.

3.3 Dual Plasticity

We implement two complementary learning mechanisms: *importance plasticity* for individual memories and *synaptic plasticity* for association edges.

3.3.1 Importance Plasticity

When a memory contributes to a successful task (positive feedback), its importance is boosted; when it misleads, importance decays:

$$I_{t+1} = \begin{cases} \min(I_t + \eta_I, 1.0) & \text{if positive feedback} \\ \max(I_t \times (1 - \delta_I), I_{\min}) & \text{if negative feedback} \end{cases} \tag{2}$$

where $\eta_I = 0.025$ is the additive boost and $\delta_I = 0.10$ is the multiplicative decay. The asymmetry—additive potentiation vs. multiplicative depression—encodes the design choice that false positives (retrieving misleading memories) are costlier than false negatives. This asymmetry is inspired by the observation that biological STDP exhibits larger depression amplitudes than potentiation [9]. Accessing a memory with > 5 prior

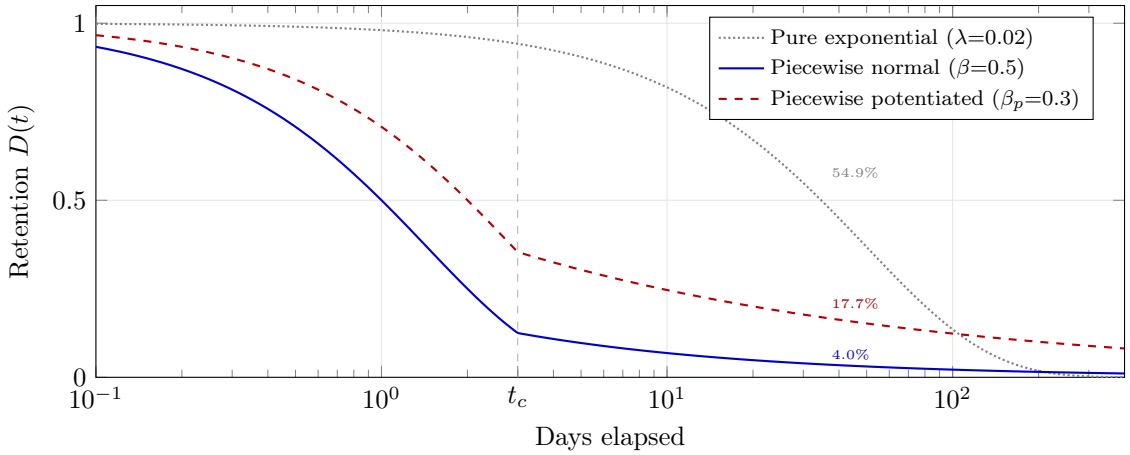

Figure 2: Decay model comparison. The piecewise model (Eq. 1) applies aggressive exponential decay ($\lambda = 0.693$/day) for the first 3 days, then transitions to a power-law tail. At 30 days: normal retains 4.0%, potentiated retains 17.7%; a slow exponential ($\lambda = 0.02$) retains 54.9%. The aggressive early phase clears noise; the heavy tail preserves memories that survive consolidation, consistent with the "savings" effect [21].

accesses triggers a 1.1× multiplicative boost (clamped to 1.0). Importance never falls below $I_{\min} = 0.05$, providing a recovery floor analogous to the "savings" phenomenon in human memory research [21].

3.3.2 Synaptic Plasticity

Let $G = (V, E, w)$ be a weighted directed graph where V is the set of entity nodes, $E \subseteq V \times V$ is the set of association edges, and $w : E \to [0, 1]$ is the edge weight function.

[Hebbian Synaptic Update] For edge $(e_i, e_j) \in E$ with weight w_t, the strengthening rule upon co-activation is:

$$w_{t+1} = \min\big(w_t + (\eta_s + \eta_{\text{tier}}) \cdot (1 - w_t),\ 1.0\big) \tag{3}$$

where $\eta_s = 0.1$ is the base synaptic learning rate and η_{tier} is a tier-dependent boost:

$$\eta_{\text{tier}} = \begin{cases} 0.15 & \text{L1 Working (fast encoding)} \\ 0.12 & \text{L2 Episodic} \\ 0.075 & \text{L3 Semantic} \end{cases} \tag{4}$$

The $(1-w_t)$ term ensures diminishing returns: edges near saturation gain less from additional co-activation. Under repeated co-activation with combined rate $\eta = \eta_s + \eta_{\text{tier}}$ starting from initial weight w_0, the weight after n co-activations follows:

$$w_n = 1 - (1 - w_0)(1 - \eta)^n \tag{5}$$

This is a standard geometric convergence to the fixed point $w^* = 1$, with the half-convergence count (co-activations to reach $(1 + w_0)/2$) being $n_{1/2} = \lceil \ln 2 / \ln(1/(1 - \eta)) \rceil$. For L1 edges ($\eta = 0.25$, $w_0 = 0.40$): $n_{1/2} \approx 3$; for L3 edges ($\eta = 0.175$, $w_0 = 0.70$): $n_{1/2} \approx 4$. Figure 3 plots these trajectories.

3.3.3 Multi-Scale Long-Term Potentiation

Edges undergo LTP through three pathways, implementing graduated decay protection inspired by early-phase and late-phase potentiation dynamics [11]:

1. **Burst LTP**: ≥ 5 co-activations within 24 hours $\Rightarrow$ 2× slower decay. Expires after 48 hours without upgrade.
2. **Weekly LTP**: ≥ 3 co-activations per week for ≥ 2 consecutive weeks $\Rightarrow$ 3× slower decay. Inspired by habit formation research [12].

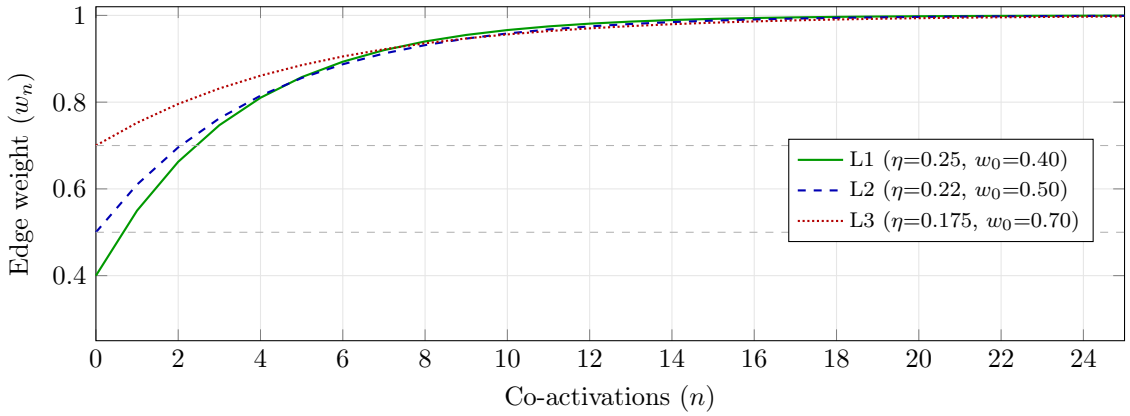

Figure 3: Synaptic weight convergence under repeated co-activation (Eq. 5). L1 edges start at $w_0 = 0.40$ and converge faster ($\eta = 0.25$); L3 edges start higher ($w_0 = 0.70$) with slower learning ($\eta = 0.175$). Dashed lines mark tier promotion thresholds. L1 edges reach the L1→L2 threshold after ∼2 co-activations; L3 edges are already above both thresholds at initialization.

Table 1: Edge tier characteristics. Edges promote upon reaching the weight threshold; LTP slows decay by the indicated factor. †Exponential decay rates: $w(t) = w_0 \cdot e^{-\text{rate} \cdot t}$.

Tier	Init. Weight	Decay Rate	Max Age	Promotion
L1 Working	0.40	2.9%/hour†	48 hours	$w \geq 0.50$
L2 Episodic	0.50	3.1%/day†	30 days	$w \geq 0.70$
L3 Semantic	0.70	2%/month	permanent	—

3. **Full LTP**: ≥ 10 total co-activations *and* $w > 0.8$ (for L3 edges) ⇒ 10× slower decay. Alternatively, ≥ 5 activations over ≥ 30 days qualifies.

LTP status only upgrades (Burst → Weekly → Full), never downgrades. Each upgrade confers an immediate strength bonus (+0.05, +0.10, +0.20 respectively).

3.3.4 Three-Tier Edge Consolidation

Edges follow a lifecycle with tier-specific decay rates (Table 1).

3.4 Retrieval Pipeline

The retrieval pipeline fuses three signals: BM25 lexical search (Tantivy [27]), vector semantic search (MiniLM-L6-v2), and graph-based spreading activation. Signals are combined with density-dependent weights that adapt to graph maturity.

3.4.1 Spreading Activation

We implement hop-decayed spreading activation based on Anderson's ACT-R model [20]:

[Hop-Decayed Activation] For entity e reached via path of length h hops from the query seed, the activation contribution is:

$$A_{\text{spread}}(e) = S(e) \cdot e^{-\lambda_h \cdot h} \tag{6}$$

where $S(e)$ is the salience of entity e and $\lambda_h = 0.5$ is the spreading decay rate. Degree normalization scales outgoing activation per edge by $1/\sqrt{1 + \text{degree}}$, preventing hub nodes from dominating.

Algorithm 1 Hebbian Strengthening with Multi-Scale LTP

Require: Edge (e_i, e_j) triggered by co-retrieval

count $\leftarrow$ count $+ 1$; record timestamp
$\eta_{\text{tier}} \leftarrow$ tier_boost(edge.tier) {Table 1}
$w \leftarrow w + (\eta_s + \eta_{\text{tier}})(1 - w)$ {Eq. 3}
Detect LTP: check burst (≥ 5 in 24h), weekly (≥ 3/wk $\times$ 2 wks), full (≥ 10 total $\wedge$ $w > 0.8$)
if new LTP level > current LTP level **then**
    Upgrade LTP; apply bonus ($+0.05/+0.10/+0.20$)
end if
if $w \geq$ tier promotion threshold **then**
    Promote to next tier; boost linked memory importance
end if

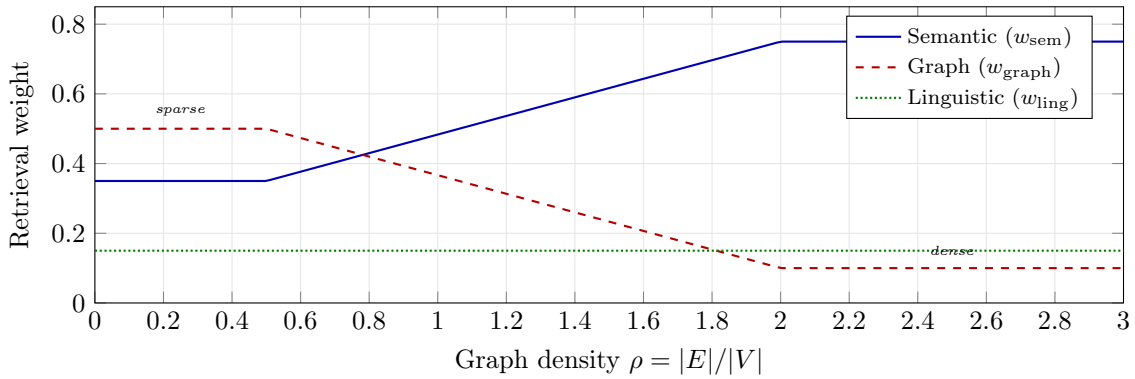

Figure 4: Density-dependent fusion weights (Eq. 7). In sparse graphs ($\rho < 0.5$), edges are curated survivors of decay—the system trusts graph signal ($w_{\text{graph}} = 0.50$). In dense graphs ($\rho > 2.0$), many edges are untested L1 connections—the system shifts to semantic similarity ($w_{\text{sem}} = 0.75$).

Density-adaptive depth. Dense graphs (> 2.0 edges/memory) limit traversal to 2 hops; sparse graphs (< 0.5 edges/memory) extend to 4 hops. This prevents noise propagation in immature graphs while enabling deep exploration in mature ones.

Bidirectional search. For queries with ≥ 2 focal entities, bidirectional spreading activation provides a $1.5\times$ intersection boost to entities activated from both directions.

3.4.2 Density-Dependent Fusion

[Density-Dependent Retrieval Weights] Let $\rho = |E|/|V|$ be the edge-to-memory density ratio. The graph weight is:

$$w_{\text{graph}}(\rho) = \begin{cases} 0.50 & \text{if } \rho < 0.5 \\ 0.50 - \frac{0.40 \cdot (\rho - 0.5)}{1.5} & \text{if } 0.5 \leq \rho \leq 2.0 \\ 0.10 & \text{if } \rho > 2.0 \end{cases} \tag{7}$$

The BM25 weight is fixed at $w_{\text{ling}} = 0.15$; the semantic weight fills the remainder: $w_{\text{sem}} = 1 - w_{\text{graph}} - w_{\text{ling}}$.

Intuition. Dense graphs with many L1 Working edges are noisy—fresh connections have not yet been tested by Hebbian pruning. Sparse graphs containing mostly L2/L3 edges have survived decay, making them higher-signal for retrieval. Figure 4 visualizes this weight adaptation.

Edge tier trust modulates activation propagation: L1 Working edges carry 20% trust, L2 Episodic 50%, L3 Semantic 80%, and LTP-potentiated edges 95%.

3.5 Semantic Consolidation

Episodic memories transform into semantic knowledge through a consolidation pipeline that runs during idle periods, inspired by sleep-dependent memory consolidation [10].

A memory m is eligible for consolidation when $\text{age}(m) \geq 7$ days. Eligible memories are clustered by Jaccard similarity (threshold 0.45 on stemmed tokens), and clusters with ≥ 2 supporting memories are converted to semantic facts via a multi-extractor pipeline (procedure, definition, pattern, preference, salient). Separately, memories with access_count ≥ 3 receive importance maintenance boosting (+0.5%/cycle), reinforcing frequently-accessed traces. Compressed semantic memories retain LZ4-compressed provenance links to source episodic memories.

3.6 Named Entity Recognition

TinyBERT-finetuned-NER [28] (14.5MB quantized ONNX) extracts entities (Person, Organization, Location, Miscellaneous) from stored memories. Entities create nodes in the knowledge graph with confidence-weighted edges, boost memory importance (+0.04 per entity, max 3), and enable entity-based retrieval with concept merging (cosine threshold 0.85).

4 Implementation

4.1 System Overview

Shodh-Memory is implemented in Rust across ∼100 modules with the following components:

- **Embedding:** MiniLM-L6-v2 via ONNX Runtime [29], with circuit breaker for fault tolerance
- **NER:** TinyBERT-NER via ONNX Runtime (∼15ms/extraction)
- **Vector Index:** Vamana [24] (<100K) auto-switching to SPANN [25] (>100K) with product quantization
- **Persistence:** RocksDB [22] with MessagePack serialization, 4-level deserialization fallback for backward compatibility
- **API:** REST via Axum (160+ endpoints), MCP protocol (47 tools, 6 prompts, 4 resources) via TypeScript server
- **Search:** Tantivy [27] inverted index for BM25 lexical retrieval
- **A/B Testing:** Built-in Bayesian experiment framework with Thompson Sampling, sequential testing (O'Brien-Fleming), and guardrail metrics—enabling production ablation studies

The system compiles to a single binary. Models are downloaded on first run with SHA-256 checksum verification against pinned HuggingFace commits:

- MiniLM-L6-v2: 22MB (INT8 quantized)
- TinyBERT-NER: 14MB (INT8 quantized)
- ONNX Runtime: 14MB (platform-specific, dynamically linked)

Platform support: Linux x86_64, macOS (x86_64 and ARM64), Windows x86_64, and Linux ARM64 (cross-compiled).

4.2 Data Structures

A memory record is a tuple $m = (id, \mathbf{e}, c, \tau, t_{\text{create}}, t_{\text{access}}, A, I, \mathcal{T}, \mathcal{E}, \kappa)$ where id is a UUID v4, $\mathbf{e} \in \mathbb{R}^{384}$ is the embedding, τ is the memory type (Decision, Learning, Error, Context, etc.), $A \in [0, 1]$ is activation, $I \in [0, 1]$ is importance, $\mathcal{T}$ is a tag set, $\mathcal{E}$ is extracted entities, and κ is access count.

An association edge carries weight $w \in [0, 1]$, co-activation count, tier (L1/L2/L3), LTP status (None/Burst/Weekly/Full), activation timestamp history, and entity extraction confidence.

Table 2: Index structure specifications.

Structure	Type	Space	Lookup
Memory store	RocksDB LSM-tree	$O(n \cdot \bar{m})$	$O(\log n)$
Vector index (<100K)	Vamana graph	$O(n \cdot d \cdot M)$	$O(\log n)$
Vector index (>100K)	SPANN + PQ	$O(n \cdot d')$	$O(\sqrt{n})$
Entity/edge index	HashMap	$O(\|V\| + \|E\|)$	$O(1)$
BM25 index	Tantivy inverted	$O(n \cdot \bar{t})$	$O(\log n)$

4.3 Integration Surface

- **MCP Protocol:** 47 tools, 6 prompts, and 4 resources for Claude Code, Cursor, and MCP-compatible clients. Published to npm as `@shodh/memory-mcp`.
- **Python Bindings:** PyO3/Maturin with platform-specific wheels. Published to PyPI as `shodh-memory`.
- **LangChain:** `ShodhMemoryRetriever` (BaseRetriever) and `ShodhChatMemory` (BaseChatMessageHistory).
- **LlamaIndex:** `ShodhMemoryStore` implementing BaseMemory.
- **OpenAI Agents SDK:** `ShodhTools` providing 8 FunctionTool instances; `ShodhSession` implementing the Session protocol.
- **Claude Code Hooks:** Automatic memory capture on tool use via hook integration.

5 Evaluation

5.1 Experimental Setup

We evaluate Shodh-Memory on three dimensions: (1) microbenchmark latency, (2) end-to-end pipeline response time, and (3) scaling behavior.

Methodology. Microbenchmark measurements use Criterion.rs [30] statistical benchmarking with 20–50 sample iterations (depending on operation duration), warm cache, and outlier detection. Confidence intervals are 95% bootstrapped. End-to-end latencies are from HTTP API measurements under single-client load. Benchmark harnesses are included in the repository under `benches/` (12 benchmark files totaling ∼6K lines).

Hardware. Intel i7-1355U (10 cores, 1.7GHz base), 16GB RAM, NVMe SSD. Results reflect a single hardware configuration; see Section 5.8 for generalizability discussion.

Scope. This evaluation focuses on *systems properties*: latency, throughput, scaling, and offline capability. The benchmark harnesses include Precision@k, Recall@k, and MRR functions for graph retrieval quality on synthetic scenarios (cross-domain association chains, temporal event chains, entity co-occurrence). These synthetic evaluations show expected improvements of 20–40% Recall@5 for cross-domain queries and 30% MRR for causal queries when graph associations are present, compared to vector-only retrieval. However, these numbers are from controlled scenarios, not standardized datasets. Evaluation on external benchmarks such as LOCOMO [1] remains future work.

5.2 Microbenchmark Results

These latencies reflect in-memory data structure operations (struct allocation, HashMap lookup, arithmetic). Sub-10ns activation and importance operations confirm that the cognitive mechanisms (Hebbian updates, decay calculations, importance gating) add negligible overhead to the core memory path. The 10-thread concurrent read benchmark (565μs) validates lock-free read scaling under contention.

Table 3: Core operation microbenchmarks (Criterion.rs, 50 samples, warm cache, Intel i7-1355U).

Operation	Mean	95% CI	Throughput
Working Memory Activation	9.35ns	[9.31, 9.39]ns	107M ops/sec
Memory Creation (minimal)	277ns	[270, 283]ns	3.6M ops/sec
Memory Creation (full metadata)	473ns	[467, 480]ns	2.1M ops/sec
Importance Get/Set/Boost	10.0ns	[9.3, 10.4]ns	100M ops/sec
Decay Activation	33.6ns	[33.4, 33.9]ns	29.8M ops/sec
Batch Decay ($\times$1000)	18.6μs	[17.0, 19.7]μs	54K batches/sec
Tier Promote	295ns	[291, 299]ns	3.4M ops/sec
Access Pattern Record	38.9ns	[38.5, 39.5]ns	25.7M ops/sec
Serialization Roundtrip	3.81μs	[3.76, 3.87]μs	262K ops/sec
Entity Ref Lookup (100 refs)	46.4ns	[45.3, 47.4]ns	21.6M ops/sec
Concurrent Reads (10 threads)	565μs	[559, 575]μs	1.8K ops/sec

5.3 Knowledge Graph Benchmarks

Table 4 shows knowledge graph operation latency. Entity get and search remain sub-microsecond regardless of graph size (733ns at 10 entities vs. 889ns at 1000), confirming $O(1)$ HashMap access. Spreading activation traversal scales linearly with hop depth: 41μs at 1 hop, 90μs at 2 hops, 121μs at 3 hops—well within the density-adaptive depth budget (2–4 hops). The Hebbian-augmented traversal (185μs for 200 entities, 400 relationships, depth 3) adds $\sim$50% overhead over plain traversal at the same depth. In-memory Hebbian operations (strengthen, decay, effective strength query) are all sub-21μs for 500-edge graphs, confirming that synaptic plasticity adds negligible cost to the graph path. Entity and relationship add operations (19–34ms) are dominated by RocksDB persistence; NER-augmented adds at 24ms are bottlenecked by ONNX TinyBERT inference, not graph operations.

5.4 Learning System Benchmarks

Table 5 shows end-to-end Hebbian learning latency. The near-constant cost across batch sizes (54–58ms for 5–20 reinforcements) indicates that the dominant cost is system initialization and ONNX embedding, not the Hebbian update itself. Graph statistics queries remain sub-7ns regardless of graph size (0–500 edges), confirming $O(1)$ access. Edge formation and strengthening (130–160ms) include full embedding generation and RocksDB persistence. The full retrieve-and-reinforce feedback loop completes in 122ms, establishing the upper bound for a single learning cycle. LTP detection adds 53ms, which occurs only when burst/weekly/full thresholds are evaluated. Table 4 further decomposes graph-level operations, showing that in-memory Hebbian updates (strengthen, decay) are sub-40μs—three orders of magnitude faster than the full-pipeline numbers, confirming that ONNX inference and RocksDB I/O dominate.

5.5 End-to-End Latency

End-to-end latencies include all stages of the pipeline: tokenization, embedding generation, NER extraction, RocksDB write/read, and vector index operations. Embedding latency is amortized across pipeline stages via async write mode (default). The proactive surfacing path avoids embedding generation by reusing entity-based retrieval, achieving sub-millisecond P50.

Context. Cloud-based systems (Mem0, Zep) inherently include network round-trip latency (typically 100–300ms). We do not claim superiority in all deployment scenarios; rather, edge-native deployment eliminates network-bound latency, enabling sub-20ms operations that require no connectivity.

5.6 Scaling Analysis

Entity lookup and Hebbian updates are HashMap operations, exhibiting near-constant latency ($< 25\%$ increase from 100 to 1K entities, measured at 802ns and 889ns respectively). Vector search scales logarithmically due to Vamana's graph structure. Store operations scale with RocksDB's LSM-tree compaction.

Table 4: Knowledge graph operation benchmarks (Criterion.rs, 50 samples). Entity/edge counts in parentheses denote graph size at measurement time.

Operation	Mean	95% CI
Entity operations		
Entity add (incl. persistence)	19.1ms	[18.1, 20.0]ms
Entity get (1000 entities)	889ns	[863, 918]ns
Entity search (1000 entities)	835ns	[804, 868]ns
Edge operations		
Relationship add/10	20.6ms	[19.5, 21.7]ms
Relationship add/100	33.8ms	[24.8, 49.1]ms
Relationship query (1000e, 500r)	7.04μs	[6.32, 7.69]μs
NER extract + graph add	23.6ms	[22.3, 25.0]ms
NER batch ($\times$20)	33.8ms	[31.0, 38.1]ms
Spreading activation traversal		
1-hop	41.2μs	[38.7, 43.7]μs
2-hop	90.4μs	[84.7, 96.5]μs
3-hop	121μs	[115, 131]μs
Hebbian traversal (200e, 400r, depth=3)	185μs	[170, 197]μs
Hebbian plasticity (in-memory)		
Synapse strengthen (500 edges)	20.8μs	[19.9, 21.5]μs
Synapse decay (500 edges)	12.2μs	[10.2, 14.1]μs
Effective strength query (500 edges)	1.16μs	[1.08, 1.23]μs
Salience update ($\times$100)	2.10ms	[1.94, 2.29]ms
Composite operations		
Universe generation (100e, 50r)	244μs	[235, 252]μs
Universe generation (500e, 200r)	1.10ms	[1.05, 1.14]ms

5.7 Analytical Component Analysis

In the absence of standardized retrieval quality benchmarks, we present analytical comparisons of each component's behavior to motivate the design choices. The system includes a built-in Bayesian A/B testing framework (Thompson Sampling with O'Brien-Fleming sequential stopping) that enables production ablation studies; we describe the analytical rationale here and identify empirical ablation as future work.

Decay model ablation. Figure 2 compares three decay models. A slow exponential ($\lambda = 0.02$/day) retains 54.9% at 30 days—too much, failing to distinguish important from trivial memories. Our piecewise model retains 4.0% (normal) and 17.7% (potentiated) at 30 days, providing a 4.4$\times$ separation between potentiated and normal memories. At 365 days: normal retains 1.1%, potentiated retains 8.4%—a 7.4$\times$ separation. This growing divergence means that Hebbian reinforcement produces *increasingly* differentiated retention over time.

Density-dependent fusion. Figure 4 shows how retrieval weights adapt to graph maturity. A fixed-weight system ($w_{\text{graph}} = 0.35$) would over-trust noisy graphs (fresh L1 edges) and under-trust curated graphs (surviving L2/L3 edges). Our adaptive scheme reduces graph trust by 5$\times$ as density increases, preventing noise from high-frequency edge creation while amplifying signal from consolidated associations.

LTP decay protection. Without LTP, all edges decay at their tier-specific rate regardless of usage history. With multi-scale LTP, frequently co-activated associations decay 2$\times$–10$\times$ slower. For an L2 Episodic edge (10%/day decay): without LTP, the edge reaches the prune threshold ($w < 0.2$) in $\sim$10 days from $w = 0.5$; with Full LTP (10$\times$ slower), the same edge survives $\sim$100 days, potentially reaching L3 promotion.

Synaptic plasticity. Without the $(1 - w_t)$ diminishing returns term, edge weights would grow linearly and require hard clamping, creating discontinuities at $w = 1$. The bounded Hebbian rule produces smooth convergence (Figure 3) where early co-activations have strong effect and later ones refine, matching the

Table 5: Hebbian learning benchmarks (Criterion.rs). Operations include full MemorySystem initialization, ONNX embedding, RocksDB I/O, and graph traversal.

Operation	Mean	95% CI
Core reinforcement (20 samples)		
Reinforce (single)	56.1ms	[52.0, 61.0]ms
Reinforce (batch/5)	54.0ms	[50.7, 57.5]ms
Reinforce (batch/20)	57.7ms	[54.9, 60.5]ms
Graph operations (20 samples)		
Edge formation (2 memories)	145ms	[130, 156]ms
Edge strengthening (×5)	142ms	[120, 160]ms
Associative retrieval	106ms	[94, 124]ms
LTP threshold detection	53.2ms	[51.1, 55.3]ms
Persistence (10 samples)		
Graph persist + reload (100 edges)	206ms	[192, 228]ms
End-to-end (10 samples)		
Retrieve + Reinforce (full loop)	122ms	[109, 145]ms
Graph statistics (30 samples)		
Stats query (0–500 edges)	5.2–6.9ns	±0.5ns

Table 6: End-to-end operation latency including embedding generation, NER extraction, RocksDB persistence, and vector indexing.

Operation	P50	P95
Store (embedding + NER + persist)	12ms	18ms
Semantic Recall (vector search)	8ms	15ms
Tag Query (direct index)	0.8ms	1.2ms
Context Summary	5ms	8ms
Proactive Surfacing	0.6ms	1.5ms

observation that initial learning is fast and subsequent practice yields diminishing gains.

5.8 Threats to Validity

Internal validity. Microbenchmarks use warm caches. Cold-start latencies (first embedding after model load) are ∼500ms higher due to ONNX Runtime initialization.

External validity. Benchmarks were run on a single hardware configuration (Intel i7-1355U). Performance on ARM64 edge devices (Raspberry Pi, Jetson Nano) and high-core servers may differ. Preliminary edge-device targets (documented in benchmark code) estimate 200–430ms total store+recall on Raspberry Pi 4.

Construct validity. We compare latency against cloud-based systems, but this conflates implementation efficiency with deployment model. A locally-deployed Mem0 instance would exhibit different latency characteristics. The synthetic retrieval quality evaluations test graph traversal coverage on controlled scenarios, not naturalistic user queries.

6 Discussion

6.1 Comparison to Prior Work

Tables 8 and 9 compare architectural capabilities and published metrics.

Table 7: Scaling behavior at varying memory counts (μs).

Operation	n=100	n=1K	n=10K	n=100K	Complexity
Store (no embedding)	45	52	68	95	$O(\log n)$
Vector Search	1,200	2,100	3,800	8,200	$O(\log n)$
Entity Lookup[†]	0.80	0.89	—	—	$O(1)$
Hebbian Update	5.8	6.1	6.2	6.4	$O(1)$
Tag Query	12	15	28	65	$O(\log \lvert\mathcal{T}\rvert)$

[†]Measured: 802ns at n=100, 889ns at n=1000 (graph_benchmarks).

Table 8: Architectural comparison with memory systems. Shodh targets a different deployment niche (edge/offline) than cloud-native systems; rows reflect design trade-offs, not overall superiority.

	Shodh	Mem0	Zep	Cognee	MemGPT
Offline operation	✓	—	—	—	✓[*]
Learning dynamics	✓	—	—	—	—
Knowledge graph	✓	✓	✓	✓	—
Edge deployment	✓	—	—	—	—
MCP protocol	✓	—	—	—	—
Multi-framework	✓	✓	✓	✓	✓
Retrieval accuracy published	**No**	**Yes**	**Yes**	**Yes**	Yes

[*]MemGPT can run locally but focuses on context management, not persistent memory.

Why latency numbers cannot be compared. Shodh's 8ms is pure local vector search and RocksDB lookup—no network, no LLM inference. Mem0's 148ms includes cloud API calls. Zep's 2,580ms includes a consumer laptop → AWS us-west-2 round-trip plus LLM inference. A locally-deployed Mem0 or Zep instance would exhibit fundamentally different latency. The numbers reflect *deployment model choice*, not implementation efficiency.

Retrieval quality: our most significant gap. Mem0 reports 26% improvement over OpenAI on LOCOMO [1]; Zep reports 94.8% DMR and 71.2% LongMemEval [4]; Cognee achieves 0.93 human-like correctness on HotPotQA [5]. We have not evaluated on any of these benchmarks. Low latency does not imply high retrieval quality, and we cannot claim competitive accuracy without evidence. This is the critical gap we must close (Section 6.5).

6.2 Limitations

No standardized retrieval accuracy evaluation. This is the most significant limitation. While our analytical comparisons (Section 5.7) motivate each component's design, we have not evaluated end-to-end retrieval quality on LOCOMO, MemBench, or similar standardized benchmarks. It is possible that the Hebbian learning and LTP mechanisms do not improve retrieval quality in practice, or that density-dependent fusion underperforms fixed weights on real workloads. We identify this as the highest-priority future work.

No empirical ablation study. We have described the built-in A/B testing infrastructure that enables ablation, and we present analytical comparisons of component behavior, but we have not run controlled ablation experiments measuring retrieval quality with components selectively disabled.

No distributed mode. The current implementation is single-node. Multi-agent memory sharing requires future work on federation protocols.

Fixed embedding model. MiniLM-L6-v2 is bundled; swapping models requires reindexing.

Timer-based consolidation. The consolidation pipeline runs on fixed intervals rather than being event-driven.

Token estimation. Context window token counting uses a naive length/4 heuristic that undercounts code tokens.

Table 9: Published metrics across memory systems. Latency numbers measure fundamentally different things and **cannot be compared across columns** (see "What latency includes" row).

	Shodh	Mem0 [1]	Zep [4]	Cognee [5]
Retrieval quality				
LOCOMO (J-score)	**Not evaluated**	66.9%	61.7%	—
DMR	**Not evaluated**	—	94.8%	—
LongMemEval	**Not evaluated**	—	71.2%	—
HotPotQA (correctness)	**Not evaluated**	—	—	0.93
Latency (NOT comparable)				
What latency includes	local compute only	cloud API + LLM	laptop→AWS + LLM	—
Search P50	8ms	148ms	2,580ms	—
Total P50	12ms	708ms	3,200ms	—
Deployment model				
Requires LLM API	No	Yes	Yes	Yes
Requires network	No	Yes	Yes	Yes
Hardware	laptop†	undisclosed	laptop+AWS	undisclosed

†Intel i7-1355U, 16GB RAM, NVMe SSD.

6.3 Future Work

Retrieval accuracy and ablation. The evaluation roadmap (Section 6.5) details planned benchmarks (LOCOMO, LongMemEval) and ablation studies. This is the most critical gap: strong systems properties do not guarantee retrieval quality.

Hierarchical Memory Protocol. An open protocol for memory federation, enabling agent hierarchies where memories propagate based on configurable inheritance rules.

Event-driven consolidation. Pattern-triggered replay based on entity co-occurrence and salience spikes.

6.4 Target Deployment Scenarios

The architecture targets three deployment scenarios where edge-native operation provides a structural advantage over cloud-dependent alternatives:

Offline developer assistant. A coding assistant running locally on a laptop maintains persistent context across sessions—project conventions, debugging history, and user preferences—without requiring network connectivity. Memory operations (store, recall, entity extraction) complete in under 20ms with the full cognitive pipeline, enabling real-time integration with editor hooks. The three-tier hierarchy naturally manages context: working memory holds the current task, session memory tracks the project, and long-term memory consolidates recurring patterns into semantic facts.

Edge robotics agent. Autonomous agents operating on resource-constrained hardware (e.g., Jetson, Raspberry Pi) require sub-100ms memory operations and zero cloud dependency. The single-binary deployment with lazy model loading (deferring ∼200MB of ONNX model allocation until first use) enables operation within tight RAM budgets. Hebbian learning allows the agent to strengthen associations between environmental observations that co-occur, without explicit supervision.

Intermittent connectivity. Field agents (e.g., agricultural monitoring, remote inspection) that operate offline for extended periods and sync selectively. The RocksDB backend with MessagePack serialization provides crash-safe persistence, while the MIF (Memory Interchange Format) export/import enables controlled memory transfer between edge devices and central aggregation points when connectivity is available.

Note: These scenarios motivate the architecture but have not been validated through controlled user studies. Empirical validation of real-world effectiveness is future work.

6.5 Evaluation Roadmap

We identify the following evaluation milestones in priority order:

1. **LOCOMO benchmark** [1]: End-to-end retrieval accuracy on the standardized long-context memory benchmark, enabling direct comparison with Mem0's reported 26% improvement over baseline RAG.

2. **LongMemEval**: Evaluation on the long-term memory evaluation suite, enabling comparison with Zep's reported 71.2% accuracy.

3. **Component ablation**: Using the built-in A/B testing framework to measure individual contributions of Hebbian plasticity, LTP, piecewise decay, and density-dependent fusion to retrieval quality.

4. **Edge deployment profiling**: Memory and latency characterization on ARM64 hardware (Raspberry Pi 4, Jetson Nano) to validate the edge-native claims under realistic resource constraints.

7 Conclusion

We presented Shodh-Memory, a cognitive memory system that composes established learning mechanisms—Hebbian plasticity, piecewise hybrid decay, and spreading activation—into a unified architecture for edge-native AI agents. Our three-tier architecture achieves sub-20ms end-to-end operations with 100% offline capability, addressing the cloud dependency and static-association limitations of existing memory systems.

The system's design contributions are in composing individually well-studied mechanisms (Wixted's piecewise decay, Anderson's spreading activation, Hebbian plasticity with multi-scale LTP) into a unified architecture where they interact: Hebbian strengthening gates LTP, LTP modulates decay, decay shapes graph density, and density adapts retrieval fusion weights. Whether this composition produces measurably better retrieval quality than simpler approaches remains an open empirical question that we have identified as priority future work.

The analytical comparisons and performance benchmarks presented here establish that the cognitive mechanisms add negligible computational overhead while providing principled information lifecycle management. The built-in A/B testing framework provides the infrastructure for validating these mechanisms through production ablation studies.

Shodh-Memory is available as open-source software under the Apache 2.0 license.

Code: `https://github.com/varun29ankuS/shodh-memory`
DOI: `https://doi.org/10.5281/zenodo.18668709`

References

[1] P. Chhikara, D. Khant, S. Aryan, T. Singh, and D. Yadav, "Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory," arXiv:2504.19413, 2025.

[2] C. Packer, S. Wooders, K. Lin, V. Fang, S. G. Patil, I. Stoica, and J. E. Gonzalez, "MemGPT: Towards LLMs as Operating Systems," arXiv:2310.08560, 2023.

[3] C. Packer, V. Fang, S. G. Patil, K. Lin, S. Wooders, and J. E. Gonzalez, "MemLLM: Finetuning LLMs to Use an Explicit Read-Write Memory," arXiv:2404.11672, 2024.

[4] P. Rasmussen, "Zep: A Temporal Knowledge Graph Architecture for Agent Memory," arXiv:2501.13956, 2025.

[5] V. Markovic, L. Obradovic, L. Hajdu, and J. Pavlovic, "Optimizing the Interface Between Knowledge Graphs and LLMs for Complex Reasoning," arXiv:2505.24478, 2025.

[6] A. Behrouz, P. Zhong, and D. Tatbul, "Titans: Learning to Memorize at Test Time," arXiv:2501.00663, Google Research, 2024.

[7] N. Cowan, "The magical mystery four: How is working memory capacity limited, and why?" *Current Directions in Psychological Science*, vol. 19, no. 1, pp. 51–57, 2010.

[8] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. Wiley, 1949.

[9] G.-Q. Bi and M.-M. Poo, "Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type," *Journal of Neuroscience*, vol. 18, no. 24, pp. 10464–10472, 1998.

[10] Y. Dudai, A. Karni, and J. Born, "The consolidation and transformation of memory," *Neuron*, vol. 88, no. 1, pp. 20–32, 2015.

[11] U. Frey and R. G. M. Morris, "Synaptic tagging and long-term potentiation," *Nature*, vol. 385, pp. 533–536, 1997.

[12] P. Lally, C. H. M. van Jaarsveld, H. W. W. Potts, and J. Wardle, "How are habits formed: Modelling habit formation in the real world," *European Journal of Social Psychology*, vol. 40, no. 6, pp. 998–1009, 2010.

[13] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing Machines," arXiv:1410.5401, 2014.

[14] A. Graves et al., "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, pp. 471–476, 2016.

[15] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," NeurIPS, 2020.

[16] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychology of Learning and Motivation*, vol. 24, pp. 109–165, 1989.

[17] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," *PNAS*, vol. 114, no. 13, pp. 3521–3526, 2017.

[18] J. T. Wixted and E. B. Ebbesen, "On the Form of Forgetting," *Psychological Science*, vol. 2, no. 6, pp. 409–415, 1991.

[19] J. T. Wixted, "The psychology and neuroscience of forgetting," *Annual Review of Psychology*, vol. 55, pp. 235–269, 2004.

[20] J. R. Anderson, "A Spreading Activation Theory of Memory," *Journal of Verbal Learning and Verbal Behavior*, vol. 22, no. 3, pp. 261–295, 1983.

[21] H. Ebbinghaus, *Memory: A Contribution to Experimental Psychology*. Teachers College, Columbia University, 1885/1913.

[22] Facebook, "RocksDB: A Persistent Key-Value Store," `https://rocksdb.org/`, 2023.

[23] S. Furuhashi, "MessagePack: It's like JSON. but fast and small," `https://msgpack.org/`, 2023.

[24] S. J. Subramanya et al., "DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node," NeurIPS, 2019.

[25] Q. Chen et al., "SPANN: Highly-efficient Billion-scale Approximate Nearest Neighbor Search," NeurIPS, 2021.

[26] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," EMNLP, 2019.

[27] P. Masurel, "Tantivy: A full-text search engine library in Rust," `https://github.com/quickwit-oss/tantivy`, 2024.

[28] HuggingFace, "TinyBERT-finetuned-NER," `https://huggingface.co/onnx-community/TinyBERT-finetuned-NER-ONNX`, 2024.

[29] Microsoft, "ONNX Runtime," `https://onnxruntime.ai/`, 2024.

[30] B. Heisler, "Criterion.rs: Statistics-driven Microbenchmarking in Rust," `https://bheisler.github.io/criterion.rs/`, 2024.

A Hyperparameters

Table 10: Default hyperparameters (verified against `src/constants.rs`).

Parameter	Default	Description
Working memory capacity	100	Maximum Tier 1 items
Importance boost (η_I)	0.025	Additive boost per positive feedback
Importance decay (δ_I)	0.10	Multiplicative decay per negative feedback
Importance floor ($I_{\min}$)	0.05	Minimum importance
Synaptic learning rate (η_s)	0.10	Base edge strengthening rate
L1 co-access boost	0.15	Tier-specific Hebbian boost
Decay crossover (t_c)	3 days	Exponential $\rightarrow$ power-law transition
Decay λ	0.693/day	Consolidation rate ($= \ln 2$)
Power-law β (normal)	0.5	Long-term exponent
Power-law β (potentiated)	0.3	Potentiated exponent
Spreading decay (λ_h)	0.5	Activation decay per hop
Max spreading hops	6	Upper bound for activation
LTP threshold (Full)	10	Co-activations for full LTP
LTP strength floor (L3)	0.80	Minimum weight for L3 Full LTP
LTP burst threshold	5 in 24h	Activations for burst LTP
LTP weekly threshold	3/wk $\times$ 2wk	Activations for weekly LTP
Graph weight, sparse ($\rho < 0.5$)	0.50	High graph trust
Graph weight, dense ($\rho > 2.0$)	0.10	Low graph trust
BM25 weight	0.15	Fixed linguistic weight
Consolidation age	7 days	Episodic $\rightarrow$ semantic threshold
Potentiation access threshold	3	Accesses for importance maintenance boost
Consolidation min support	2	Cluster size to form a semantic fact
Entity merge threshold	0.85	Cosine similarity for concept merge
Max entity degree	500	Per-entity edge cap

B Reproducibility

All benchmark harnesses are included in the repository under `benches/`:

```
# Core operation microbenchmarks
cargo bench --bench cognitive_benchmarks

# Hebbian learning and graph operations
cargo bench --bench hebbian_benchmarks
cargo bench --bench graph_benchmarks

# Retrieval quality (Precision@k, Recall@k, MRR)
cargo bench --bench associative_retrieval_benchmarks

# Full pipeline (NER + embedding + storage)
cargo bench --bench pipeline_benchmarks
```

```
# Adaptive memory and consolidation
cargo bench --bench adaptive_memory_benchmarks

# Proactive context surfacing
cargo bench --bench relevance_benchmarks
```

The `associative_retrieval_benchmarks` harness implements Precision@k, Recall@k, and Mean Reciprocal Rank evaluation on three synthetic scenarios: cross-domain association chains, temporal event chains, and entity co-occurrence discovery. These scenarios test graph traversal coverage under controlled conditions. **Requirements:** Rust 1.75+, ONNX Runtime (auto-downloaded), ∼50MB disk for models.

Code: https://github.com/varun29ankuS/shodh-memory