

# MentisDB

A Hash-Chained Semantic Memory Substrate for Agentic Systems

Angel Leon

Universidad Católica Andrés Bello, Venezuela

Version 0.10.3.48 — June 20, 2026

## Abstract

Contemporary agent frameworks treat long-term memory as an afterthought, relying on ad hoc prompt stuffing, unstructured Markdown files, or proprietary session state that is opaque, non-transferable, and easily lost. We introduce **MentisDB**, a durable, semantically typed memory engine that formalizes agent memory as an append-only, hash-chained ledger of structured *thoughts*. Formally, a chain is a sequence  $\chi = (t_0, t_1, \dots, t_{n-1})$  of typed records satisfying a cryptographic integrity invariant  $t_k.h = H(\sigma(t_k \setminus \{h\}))$  and  $t_k.h_{\text{prev}} = t_{k-1}.h$ , where  $H$  is SHA-256 and  $\sigma$  is canonical bincode serialization. On top of  $\chi$  we define a retrieval function  $R : (\chi, Q) \rightarrow \mathcal{P}(\chi)$  that composes BM25 lexical scoring with per-field document-frequency gating, query-time synonym expansion via a comprehensive static thesaurus ( $\sim 900$  headwords) coupled with irregular-verb lemmatization, smooth exponential vector-lexical fusion, bidirectional graph expansion over typed relation edges augmented by vector-cosine-inferred implicit edges, temporal edge validity predicates, session cohesion, rank-based fusion via Reciprocal Rank Fusion (RRF), and an HNSW approximate-nearest-neighbor backend for large vector sidecars. Deduplication is implemented as a Jaccard-similarity test over normalized token sets, emitting **Supersedes** edges consulted in constant time via a precomputed invalidation set. On canonical long-term memory benchmarks, MentisDB attains  $R@10 = 88.7\%$  on LoCoMo-2P,  $R@10 = 71.9\%$  on LoCoMo-10P, and  $R@5 = 66.8\%$  /  $R@10 = 72.2\%$  /  $R@20 = 78.0\%$  on LongMemEval (v0.8.9, fresh chain, default retrieval settings). Results are deterministic and reproducible across independent runs. The implementation ships as a single Rust crate with an optional daemon exposing MCP, REST, and HTTPS surfaces, requires no external database, and operates without cloud or LLM dependencies in its core ingestion and retrieval path.

**Keywords:** agent memory, hash-chained ledger, BM25, synonym expansion, reciprocal rank fusion, graph expansion, implicit edge inference, approximate nearest neighbor search, HNSW, temporal knowledge graphs, retrieval-augmented generation.

## 1 Introduction

The proliferation of large-language-model (LLM) agents has exposed a fundamental gap in the systems that support them: the absence of a durable, queryable, and tamper-evident memory substrate. Ephemeral context windows, hand-rolled Markdown files, and provider-specific key-value stores fail to provide the properties that multi-agent coordination demands — namely (i) *integrity* under adversarial or accidental mutation, (ii) *semantic typing* to distinguish decisions from observations from corrections, (iii) *temporal validity* to support point-in-time queries, (iv) *hybrid retrieval* combining lexical, semantic, and graph signals, and (v) *portability* across harnesses (Claude Code, Codex, Copilot, Cursor, Qwen, and beyond).

### 1.1 Contributions

This paper makes the following contributions:

1. **Formal model.** We define agent memory as an append-only, hash-chained ledger of *thoughts* — structured, typed, attributable records — with a precise integrity invariant and explicit schema evolution semantics (Sections 2, 3).
2. **Semantic typing.** We introduce a 30-variant `ThoughtType` algebra and an 8-variant `ThoughtRole` algebra, separating content semantics from workflow mechanics (Section 4).
3. **Temporal edges.** We extend typed graph relations with a validity interval `[valid_at, invalid_at]` enabling point-in-time queries via a predicate  $\pi_\tau$  (Section 4.4).
4. **Hybrid retrieval.** We describe a composable retrieval pipeline — per-field BM25 with DF gating, query-time synonym expansion, smooth exponential vector–lexical fusion, bounded graph BFS with typed edge weights, session cohesion, importance weighting, and RRF — and characterize each signal mathematically (Section 6).
5. **Synonym expansion.** We introduce a static thesaurus of  $\sim 900$  headwords embedded into the binary at compile time, coupled with irregular-verb lemmatization (300+ entries), enabling query terms like `went` to find documents containing `go` and `travel`. Synonym matches receive a configurable weight penalty (default 0.7) relative to exact term matches, and a fast path in the BM25 scorer eliminates overhead when no synonyms are configured (Section 6.4).
6. **Deduplication.** We give a Jaccard-threshold algorithm that auto-emits `Supersedes` edges, with constant-time consultation via a precomputed invalidation set  $\mathcal{I}(\chi)$  (Section 7).
7. **Empirical evaluation.** We report results on LoCoMo and LongMemEval, and provide near-miss analyses for both benchmarks characterizing the residual lexical ceiling (Section 9).
8. **Implicit edge overlay.** We introduce `ImplicitEdgeOverlay`, a rebuildable per-chain sidecar that derives `RelatedTo` edges from vector cosine similarity above a configurable threshold  $\theta$ , enriching graph expansion for chains where agents author few explicit relations (Section 6.13).
9. **Approximate vector search.** We describe an HNSW approximate-nearest-neighbor backend that shares the same `VectorSearchBackend` trait as the exact cosine engine, scaling vector sidecars to millions of entries while preserving the public retrieval semantics (Section 6.14).

## 1.2 Paper Organization

Section 2 presents the core data model and integrity invariant. Section 3 formalizes schema evolution. Section 4 defines the semantic memory algebra. Section 5 describes the storage layer. Section 6 develops the retrieval pipeline. Section 7 gives the deduplication algorithm. Section 8 describes the operational surfaces (CLI, MCP). Section 9 reports empirical results. Section 10 compares against related systems. Section 11 concludes.

# 2 System Model and Core Data

## 2.1 Thought Record

**Definition 1** (Thought). Let `UUID` denote the set of RFC 4122 universally unique identifiers,  $\mathcal{T}$  the set of UTC timestamps,  $\Sigma^*$  the set of finite UTF-8 strings, and  $\mathcal{H} = \{0, 1\}^{256}$  the codomain of SHA-256. A thought is a tuple

$$t = (v, \text{id}, i, \tau, a, \kappa, \sigma, \varphi, \rho, c, \mathbf{T}, \mathbf{C}, f_{\text{conf}}, f_{\text{imp}}, s, \mathbf{R}, \mathbf{E}, h_{\text{prev}}, h)$$

with components listed in Table 1.

| Symbol                            | Domain                         | Role                       |
|-----------------------------------|--------------------------------|----------------------------|
| $v$                               | $\mathbb{N}$                   | schema version (Section 3) |
| $\text{id}$                       | UUID                           | stable identity            |
| $i$                               | $\mathbb{N}$                   | append-order index         |
| $\tau$                            | $\mathcal{T}$                  | commit timestamp           |
| $a$                               | $\Sigma^*$                     | agent identifier           |
| $\kappa$                          | $\Sigma^* \cup \{\perp\}$      | signing key identifier     |
| $\sigma$                          | $\{0, 1\}^* \cup \{\perp\}$    | Ed25519 signature          |
| $\varphi$                         | ThoughtType                    | semantic class             |
| $\rho$                            | ThoughtRole                    | workflow role              |
| $c$                               | $\Sigma^*$                     | content                    |
| $\mathbf{T}, \mathbf{C}$          | $\mathcal{P}(\Sigma^*)$        | tags, concepts             |
| $f_{\text{conf}}, f_{\text{imp}}$ | $[0, 1]$                       | confidence, importance     |
| $s$                               | $\text{Scope} \cup \{\perp\}$  | visibility scope           |
| $\mathbf{R}$                      | $\mathcal{P}(\mathbb{N})$      | positional back-references |
| $\mathbf{E}$                      | $\mathcal{P}(\text{Relation})$ | typed edges                |
| $h_{\text{prev}}, h$              | $\mathcal{H}$                  | chain-integrity hashes     |

Table 1: Fields of a thought record.

A *thought input*  $t^\circ$  is the caller-authored subset  $(v, a, \varphi, \rho, c, \mathbf{T}, \mathbf{C}, f_{\text{conf}}, f_{\text{imp}}, s, \mathbf{R}, \mathbf{E}^\circ)$ . The chain-managed fields  $\{\text{id}, i, \tau, h_{\text{prev}}, h\}$  are assigned on commit; this asymmetry prevents agents from forging chain mechanics.

## 2.2 Hash-Chained Ledger

**Definition 2** (Chain). Let  $\sigma : \text{Thought} \rightarrow \{0, 1\}^*$  denote canonical bincode serialization of  $t \setminus \{h\}$ . A chain is a sequence  $\chi = (t_0, t_1, \dots, t_{n-1})$  satisfying the integrity invariant

$$\forall k : t_k.h = H(\sigma(t_k)), \quad \forall k \geq 1 : t_k.h_{\text{prev}} = t_{k-1}.h,$$

where  $H$  is SHA-256. For  $t_0$ ,  $t_0.h_{\text{prev}}$  is the empty string.

**Proposition 1** (Tamper Evidence). Let  $\chi' = (t'_0, \dots, t'_{n-1})$  differ from  $\chi$  at index  $j$ , i.e.  $t'_j \neq t_j$ , with all other components unchanged. Then either  $t'_j.h \neq H(\sigma(t'_j))$  (local inconsistency detectable at index  $j$ ) or  $t'_{j+1}.h_{\text{prev}} \neq t'_j.h$  (cascading inconsistency detectable at index  $j+1$ ). Consequently, forging a modification requires recomputing every subsequent hash, touching  $n - j$  records.

*Proof sketch.* By collision resistance of  $H$ ,  $t'_j \neq t_j \Rightarrow H(\sigma(t'_j)) \neq H(\sigma(t_j))$  with overwhelming probability. Either the adversary preserves  $t'_j.h = H(\sigma(t'_j))$  (then  $t'_j.h \neq t_j.h$ , breaking  $t'_{j+1}.h_{\text{prev}}$ ), or leaves  $t'_j.h = t_j.h$  (then  $t'_j.h \neq H(\sigma(t'_j))$ , so the local check fails).  $\square$

This is a practical integrity mechanism for agent memory; it does not imply a consensus protocol or distributed-ledger guarantees. Optional Ed25519 signatures  $(\kappa, \sigma)$  are layered on individual thoughts for stronger provenance when the producing agent has registered a public key in the agent registry.

## 2.3 Typed Relation Edges

**Definition 3** (Relation). A relation is a tuple  $e = (\kappa, \text{id}^*, \chi^*, \mathbf{v}_*, \mathbf{v}^*)$  where  $\kappa \in \text{ThoughtRelationKind}$ ,  $\text{id}^* \in \text{UUID}$  is the target,  $\chi^* \in \Sigma^* \cup \{\perp\}$  is an optional cross-chain key, and  $\mathbf{v}_*, \mathbf{v}^* \in \mathcal{T} \cup \{\perp\}$  bound the edge's validity interval.

The adjacency structure  $A(\chi)$  induced by  $\chi$  is a directed multigraph whose nodes are thought locators and whose edges derive from  $\mathbf{R}$  (positional refs) and  $\mathbf{E}$  (typed relations). We denote outgoing and incoming neighborhoods by  $N^+(v)$  and  $N^-(v)$  respectively; bidirectional expansion considers  $N^+(v) \cup N^-(v)$ .

## 2.4 Agent Registry

To avoid duplicating identity metadata inside every record, MentisDB maintains a per-chain registry  $\mathcal{A}(\chi)$  mapping  $a \mapsto (\text{display\_name}, \text{owner}, \text{description}, \text{aliases}, \text{status}, \text{public\_keys}, \text{counters})$ . Thoughts carry only the stable  $a$ ; the registry is resolved at read time. The registry is itself administrable through library calls, MCP tools, and REST endpoints, permitting pre-registration, documentation, revocation, or key rotation prior to any appended thought.

## 3 Schema Evolution

### 3.1 Version Lattice

MentisDB exposes a linearly ordered schema version space  $\mathcal{V} = \{V_0, V_1, V_2, V_3\}$ :

| Version | Additions relative to predecessor                                                                                                                                    |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $V_0$   | original format; no explicit $v$ field                                                                                                                               |
| $V_1$   | explicit $v$ , optional $(\kappa, \sigma)$ , agent registry sidecar                                                                                                  |
| $V_2$   | $\varphi := \varphi \cup \{\text{Reframe}\}$ , $\text{ThoughtRelationKind} := \text{ThoughtRelationKind} \cup \{\text{Supersedes}\}$ , optional cross-chain $\chi^*$ |
| $V_3$   | edge validity fields <code>valid_at</code> , <code>invalid_at</code>                                                                                                 |

Table 2: Schema versions. The current constant is  $V_{\text{cur}} = V_3$ .

### 3.2 Migration as Idempotent Transformation

Each migration  $\mu_{V_k \rightarrow V_{k+1}} : \chi_{V_k} \rightarrow \chi_{V_{k+1}}$  satisfies

$$\mu_{V_k \rightarrow V_{k+1}} \circ \mu_{V_k \rightarrow V_{k+1}} = \mu_{V_k \rightarrow V_{k+1}} \quad (\text{idempotence})$$

and is composable:  $\chi_{V_0}$  is upgraded via  $\mu_{V_2 \rightarrow V_3} \circ \mu_{V_1 \rightarrow V_2} \circ \mu_{V_0 \rightarrow V_1}$ . Because bincode encodes enum variants by integer tag, new variants are appended to the end of an enum to preserve binary compatibility; mid-enum insertion would violate the injectivity of the serialization map. After each migration the hash chain is rebuilt under  $V_{\text{cur}}$  and persisted in native format, so subsequent opens incur no migration cost.

### 3.3 Version Detection

Version is inferred by peeking at the first record’s  $v$  field. A subtle edge case arises for  $V_0$  chains that lack the field entirely: the bincode “empty-Vec fast path” for  $V_0$  reads  $v = 0$ , which is disambiguated from a non-empty  $V_0$  chain by the residual byte-length check. Practically, this provides reliable version detection in  $O(1)$  regardless of chain length.

## 4 Semantic Memory Algebra

### 4.1 ThoughtType

ThoughtType partitions the semantic space into 30 disjoint classes across seven categories:

### 4.2 ThoughtRole

ThoughtRole is orthogonal to ThoughtType, specifying *how* the system uses a memory:

ThoughtRole = {Memory, WorkingMemory, Summary, Compression, Checkpoint, Handoff, Audit, Retrospective}.

The product ThoughtType  $\times$  ThoughtRole yields 240 distinguishable semantic positions. A retrospective lesson is encoded as (LessonLearned, Retrospective).

| Category            | Variants                                                             |
|---------------------|----------------------------------------------------------------------|
| User / relationship | PreferenceUpdate, UserTrait, RelationshipUpdate                      |
| Observation         | Finding, Insight, FactLearned, PatternDetected, Hypothesis, Surprise |
| Error / correction  | Mistake, Correction, LessonLearned, AssumptionInvalidated, Reframe   |
| Planning            | Constraint, Plan, Subgoal, Goal, Decision, StrategyShift             |
| Exploration         | Wonder, Question, Idea, Experiment                                   |
| Execution           | ActionTaken, TaskComplete                                            |
| State               | Checkpoint, StateSnapshot, Handoff, Summary                          |

### 4.3 ThoughtRelationKind

ThoughtRelationKind = {References, Summarizes, Corrects, Invalidates, CausedBy, Supports, Contradicts, DerivedFrom, ContinuesFrom, BranchesFrom, RelatedTo, Supersedes}.

Supersedes is distinguished from Corrects: the former replaces a prior framing without asserting an error, while the latter asserts a factual correction.

### 4.4 Temporal Edge Validity

**Definition 4** (As-Of Predicate). *For a relation  $e$  with validity interval  $[v_*, v^*]$  (treating  $\perp$  on either bound as  $-\infty$  and  $+\infty$  respectively),*

$$\pi_\tau(e) \equiv (v_* = \perp \vee v_* \leq \tau) \wedge (v^* = \perp \vee \tau < v^*).$$

Graph expansion restricted by  $\tau$  considers only edges satisfying  $\pi_\tau$ . Combined with the append-ordering invariant  $i(t) < n$ , this yields point-in-time retrieval: “what did the agent know at  $\tau$ ?”

### 4.5 Invalidation Set

**Definition 5** (Invalidation Set). *Given  $\chi$ ,*

$$\mathcal{I}(\chi) = \{e.\text{id}^* : e \in \bigcup_{t \in \chi} \mathbf{E}(t), \kappa(e) \in \{\text{Supersedes, Corrects, Invalidates}\}\}.$$

$\mathcal{I}(\chi)$  is precomputed at chain open time as a HashSet(UUID), enabling  $O(1)$  superseded-thought detection during retrieval.

## 5 Storage Layer

### 5.1 Storage Adapter Abstraction

The trait `StorageAdapter` abstracts persistence with four essential operations: `load_thoughts`, `append_thought`, `flush`, `set_auto_flush`. This allows the chain semantics to remain invariant under backend substitution.

### 5.2 BinaryStorageAdapter

The default backend serializes each thought as a length-prefixed bincode record:

$$\underbrace{\ell_0}_{\text{4-byte LE } \ell_0} \underbrace{\sigma(t_0)}_{\text{bytes}} \parallel \underbrace{\ell_1}_{\text{4-byte LE } \ell_1} \underbrace{\sigma(t_1)}_{\text{bytes}} \parallel \dots$$

with file extension `.tcbn`.

Two durability modes are supported:

- **Strict** (`auto_flush = true`): appends are routed through a dedicated writer thread; callers block until the flush acknowledgment returns. A group-commit window of  $\Delta_{gc}$  (default 2ms) amortizes flush cost across concurrent writers.
- **Buffered** (`auto_flush = false`): records are queued and batched; the writer flushes every  $\Phi = 16$  records. Up to  $\Phi - 1 = 15$  records may be lost on a hard crash, with a corresponding throughput gain.

### 5.3 Legacy Adapters

`LegacyJsonlReadAdapter` is a read-only compatibility shim for migrating  $V_0$  `.jsonl` chains; it cannot be used for new writes.

### 5.4 File Layout

Each chain’s storage files share a common stem derived from the chain key and a hash of its genesis thought, preventing accidental cross-chain file sharing:

```
~/.cloudllm/mentisdb/
  mentisdb-registry.json
  mentisdb-skills.bin
  mentisdb-webhooks.json
  <chain-key>-<hash8>.tcbin
  <chain-key>-<hash8>.agents.json
  <chain-key>-<hash8>.entity-types.json
  <chain-key>-<hash8>.vectors.<model>-<dim>-<ver>.bin
  <chain-key>-<hash8>.vectors.managed.json
  <chain-key>-<hash8>.auto_edges.bin
  tls/{cert.pem, key.pem}
```

The `.auto_edges.bin` file is a bincode-serialized `ImplicitEdgeOverlay` (§6.13) and is rebuildable from the vector sidecar at any time; it is not part of the hash chain and carries no integrity invariant of its own. Backups include it alongside the vector sidecar.

## 6 Retrieval

Retrieval separates a deterministic filter-first baseline from a scored ranked pipeline.

### 6.1 Baseline Filter

The baseline  $R_{\text{base}}(\chi, Q)$  narrows candidates by indexed fields  $(\varphi, \rho, a, \mathbf{T}, \mathbf{C})$  and applies a case-insensitive substring predicate over  $(c, \text{agent\_meta}, \mathbf{T}, \mathbf{C})$ . Results return in append order. This path is explainable and has no BM25, vector, or graph component.

### 6.2 Ranked Pipeline

Ranked search  $R_{\text{rank}}$  selects a backend based on query features:

### 6.3 BM25 with Per-Field DF Gating

**Definition 6** (BM25 Field Score). *Let  $N = |\chi|$  be corpus size,  $\text{df}(q)$  the document frequency of term  $q$ , and for field  $f \in \{\text{content}, \text{tags}, \text{concepts}, \text{agent\_id}, \text{agent\_registry}\}$  let  $\text{tf}_f(d, q)$  and  $|d|_f$*

| Query features                                | Backend      |
|-----------------------------------------------|--------------|
| non-empty text, no vector sidecar             | Lexical      |
| non-empty text, vector sidecar                | Hybrid       |
| non-empty text, graph enabled, no vector      | LexicalGraph |
| non-empty text, graph enabled, vector sidecar | HybridGraph  |
| empty or absent text                          | Heuristic    |

denote term frequency and field length respectively, with  $\overline{|d|}_f$  the corpus mean. With  $k_1 = 1.2$ ,  $b = 0.75$ :

$$\text{idf}(q) = \ln\left(\frac{N - \text{df}(q) + 0.5}{\text{df}(q) + 0.5} + 1\right), \quad (1)$$

$$\text{score}_f(d, q) = \text{idf}(q) \cdot \frac{\text{tf}_f(d, q) (k_1 + 1)}{\text{tf}_f(d, q) + k_1 \left(1 - b + b \frac{|d|_f}{\overline{|d|}_f}\right)}. \quad (2)$$

**Definition 7** (DF Gate). For per-field cutoff  $\tau_f \in [0, 1]$ ,

$$\gamma_f(q) = \mathbb{I}\left[\frac{\text{df}(q)}{N} \leq \tau_f \vee N < 20\right].$$

Defaults:  $\tau_{\text{content}} = \tau_{\text{tags}} = \tau_{\text{concepts}} = 0.30$ ,  $\tau_{\text{agent\_registry}} = 0.60$ ,  $\tau_{\text{agent\_id}} = 0.70$ .

**Definition 8** (Lexical Score). With per-field weights  $w_f$  (defaults  $w_{\text{content}} = 1.0$ ,  $w_{\text{tags}} = 1.6$ ,  $w_{\text{concepts}} = 1.4$ ,  $w_{\text{agent\_id}} = 1.5$ ,  $w_{\text{agent\_registry}} = 1.1$ ):

$$S_\ell(d, Q) = \sum_{q \in Q} \sum_f \gamma_f(q) \cdot w_f \cdot \text{score}_f(d, q).$$

A term violating  $\gamma_f$  in one field may still contribute through other fields whose cutoffs it respects. The 20-document threshold suppresses DF filtering on small corpora where statistics are not yet meaningful.

Normalization applies Porter stemming [3] before indexing and querying. An irregular-verb lemma table of  $\sim 300$  entries expands query-time tokens (e.g. **went**  $\rightarrow$  **go**, **saw**  $\rightarrow$  **see**), because Porter stemming cannot normalize suppletive forms. Regular suffix stripping (**-ing**, **-ed**, **-ies**, **-s**) handles predictable verb forms automatically.

## 6.4 Static Thesaurus Synonym Expansion

Query terms are expanded via a static thesaurus of  $\sim 900$  headwords covering common English verbs, nouns, adjectives, adverbs, and software-engineering terminology. The thesaurus data is embedded into the binary at compile time (**include\_str!**), requiring no external files or network calls at runtime.

**Definition 9** (Synonym Expansion). Given query  $Q$  and thesaurus  $\Theta : \Sigma^* \rightarrow \mathcal{P}(\Sigma^*)$ , the expanded term set is

$$\text{Expand}(Q, \Theta) = \{(q, 1.0) : q \in Q\} \cup \{(s, w) : q \in Q, s \in \Theta(q), s \notin Q\},$$

where  $w \in [0, 1]$  is the synonym weight (default 0.7).

Before lookup in  $\Theta$ , each query token is lemmatized: irregular forms resolve via the lemma table, and regular forms strip suffixes. This bridges verb-form gaps — e.g. **went**  $\rightarrow$  **go**  $\rightarrow$  {**travel**, **move**, **depart**}.

The BM25 scorer implements a fast path: when no synonyms are configured, the original normalized-terms path is used without allocation or weight-multiplication overhead. When synonyms are present, each expanded term is scored with its weight as a multiplicative factor on the field score, ensuring synonym matches contribute less than exact matches.

Research on a synthetic corpus (140 documents, 12 queries) shows the thesaurus improves NDCG@5 by +31% over baseline ( $0.52 \rightarrow 0.68$  at  $w = 0.7$ ). An alternative embedding-based generator using the built-in `LocalTextEmbeddingProvider` underperforms (NDCG 0.48) because its hash-based feature vectors do not capture true semantic similarity; we retain it for experimentation but recommend the static thesaurus for production use.

## 6.5 Smooth Vector–Lexical Fusion

When a managed vector sidecar provides cosine similarity  $s_v(d, Q) \in [-1, 1]$  (e.g. via ONNX-embedded `fastembed-minilm`), the hybrid contribution is

$$S_{\text{fuse}}(d, Q) = s_v(d, Q) \cdot \left(1 + \alpha \exp(-S_\ell(d, Q)/\beta)\right), \quad (3)$$

with  $\alpha = 35$  and  $\beta = 3$ . This yields  $\sim 36\times$  amplification for pure-semantic matches ( $S_\ell = 0$ ), decays to  $\sim 12\times$  at  $S_\ell = 3$ , and approaches additive composition for  $S_\ell \geq 6$ . The smooth exponential eliminates the discontinuities that step-function boost tiers introduce at bin boundaries.

The managed vector sidecar is loaded once at chain open and held in an in-memory `VectorIndex` (a `HashMap` from thought UUID to normalized embedding vector). This eliminates per-query disk I/O and keeps search latency proportional to  $N$  rather than to storage throughput.

## 6.6 Graph-Aware Expansion

**Definition 10** (Bounded BFS Expansion). *Given seed set  $\Sigma_0 \subseteq \chi$  with  $|\Sigma_0| \leq 20$ , adjacency index  $A(\chi)$ , implicit edge overlay  $\mathcal{O}$  (Definition 14), and traversal mode  $M \in \{\text{Out}, \text{In}, \text{Bi}\}$ ,*

$$\text{Expand}_{d_{\max}, V_{\max}, M}(\Sigma_0, \mathcal{O}) = \{(v, d, \pi) : v \in \chi, d \leq d_{\max}, \text{path}(\pi) \subseteq \chi\},$$

*bounded by maximum depth  $d_{\max}$  and visit budget  $V_{\max}$ , with edges drawn from  $A(\chi)$  (explicit typed relations, optionally filtered by  $\pi_\tau$ ) and from  $\mathcal{O}$  (implicit cosine-inferred `RelatedTo` edges). Both sources share the seen-set, so a node reached via an explicit edge is not revisited via an implicit one and vice versa.*

**Edge weights.** For traversals along a typed relation of kind  $\kappa$ , the edge contributes  $b_{\text{rel}}(\kappa)$ : Implicit edges synthesized from  $\mathcal{O}$  are presented as `RelatedTo` and receive  $b_{\text{rel}} = 0.08$ .

| $\kappa$              | $b_{\text{rel}}$ | $\kappa$              | $b_{\text{rel}}$ |
|-----------------------|------------------|-----------------------|------------------|
| ContinuesFrom         | 0.60             | Summarizes            | 0.20             |
| BranchesFrom          | 0.55             | CausedBy              | 0.20             |
| Corrects, Invalidates | 0.50             | Supports, Contradicts | 0.15             |
| Supersedes            | 0.45             | RelatedTo             | 0.08             |
| DerivedFrom           | 0.40             | References            | 0.06             |

**Graph proximity.** For a hit at depth  $d \geq 1$ ,  $S_{\text{graph}}(d) = 1/d$ .



## 6.7 Session Cohesion

**Definition 11** (Session Cohesion Boost). *For a seed  $\sigma \in \Sigma_0$  with  $S_\ell(\sigma) \in [\theta_{\text{seed}}, \theta_{\text{solo}}) = [3, 5)$ , and a candidate  $d$  with  $|i(d) - i(\sigma)| \leq 8$ ,*

$$S_{\text{coh}}(d) = \max_{\sigma \in \Sigma_0} \max \left( 0, 0.8 \cdot (1 - |i(d) - i(\sigma)|/8) \cdot \mathbb{I}[\theta_{\text{seed}} \leq S_\ell(\sigma) < \theta_{\text{solo}}] \right).$$

Seeds above  $\theta_{\text{solo}}$  are excluded because they are strong enough to stand on their own; the cohesion boost is meant to surface *evidence turns* adjacent to a match that share no direct lexical terms.

## 6.8 Importance Weighting

**Definition 12** (Importance Boost).

$$S_{\text{imp}}(d, Q) = S_\ell(d, Q) \cdot (f_{\text{imp}}(d) - 0.5) \cdot 0.3.$$

User-originated thoughts ( $f_{\text{imp}} \approx 0.8$ ) outrank verbose assistant responses ( $f_{\text{imp}} \approx 0.2$ ) in close BM25 races. The differential structure prevents flat multipliers from overwhelming lexical signal.

## 6.9 Reciprocal Rank Fusion

When `enable_reranking` is set, the top  $K = \text{rerank\_k}$  candidates (default 50) are reranked via RRF [2].

**Definition 13** (Reciprocal Rank Fusion). *Given  $m$  ranked lists  $L_1, \dots, L_m$  of candidate documents, with  $\text{rank}_i(d)$  the 1-indexed position of  $d$  in  $L_i$  (or  $+\infty$  if absent), and damping constant  $k = 60$ ,*

$$S_{\text{RRF}}(d) = \sum_{i=1}^m \frac{1}{k + \text{rank}_i(d)}. \quad (4)$$

MentisDB produces three single-signal rankings — lexical-only ( $S_\ell$ ), vector-only ( $s_v$ ), and graph-only ( $S_{\text{graph}} + b_{\text{rel}} + S_{\text{seed}}$ ) — and fuses them via  $S_{\text{RRF}}$ . The RRF total replaces the additive blend. Non-rankable signals ( $S_{\text{imp}}, S_{\text{coh}}, f_{\text{conf}}, \text{recency}$ ) are then added as small tie-breaking adjustments. RRF is pure arithmetic: no LLM, no external service, no network round trip.

## 6.10 Memory Scopes

Scope = {User, Session, Agent}, stored as tag markers `scope:{variant}`. A query with scope  $s$  filters hits such that  $s(d) = s$ ; absence of a scope filter returns all scopes.

## 6.11 Context Bundles

$\text{Bundle}(\chi, Q) = \{(\sigma, N^\pm(\sigma) \cap R_{\text{rank}}(\chi, Q)) : \sigma \in \Sigma_0\}$ : each bundle pairs a lexical seed with its graph-expanded neighbors, presented in deterministic provenance order so agents can interpret *why* supporting thoughts surfaced.

## 6.12 Vector Sidecars

Vector state lives in rebuildable per-chain sidecars, partitioned by  $(\chi, \text{id}, h, \text{model\_id}, \text{dim}, \text{version})$ . Model or version changes invalidate old sidecars rather than silently mixing incompatible embeddings. Managed sidecars remain synchronized on append; the daemon defaults to local ONNX inference via `fastembed-minilm`. At chain open the sidecar is deserialized once into an in-memory `VectorIndex`; all subsequent ranked-search calls read from this structure without touching disk.

### 6.13 Implicit Edge Overlay

In practice, most agents append thoughts without authoring explicit **E** relations. For such chains the  $A(\chi)$  adjacency graph is sparse or empty, and graph expansion contributes nothing. The `ImplicitEdgeOverlay` closes this gap by deriving `RelatedTo` edges automatically from the vector sidecar.

**Definition 14** (Implicit Edge Overlay). *Given a vector sidecar  $\mathcal{S} = \{(u_k, \vec{v}_k)\}_{k=0}^{n-1}$  with  $\vec{v}_k \in \mathbb{R}^d$  normalized, threshold  $\theta \in [0, 1]$ , and budget  $K \in \mathbb{N}$ , the overlay is*

$$\mathcal{O} = \left\{ (u_i, u_j, \cos(\vec{v}_i, \vec{v}_j)) : i \neq j, \cos(\vec{v}_i, \vec{v}_j) \geq \theta \right\}$$

*restricted so that each source node  $u_i$  retains at most  $K$  neighbors sorted by descending cosine score. The overlay is represented as `HashMap<UUID, Vec<ImplicitNeighbor>>` and persisted to `<chain>.auto_edges.bin` via bincode with atomic rename.*

**Build complexity.** The full build is  $O(N^2d)$  (all pairwise cosines) and is performed once at chain open for  $N \leq 50,000$ . Chains above this threshold defer to incremental-only mode. On each append, the  $O(Nd)$  incremental update computes cosines between the new thought’s vector and all existing entries, adds forward edges from the new node and back-edges from its above-threshold neighbors, and re-sorts and truncates each affected neighbor list to  $K$ .

**Invalidation.** If the loaded overlay’s  $(\theta, K)$  parameters differ from the current configuration (set via environment variables `MENTISDB_AUTO_EDGE_THRESHOLD` and `MENTISDB_AUTO_EDGE_K`, or the builder API), the stored file is discarded and a full rebuild is triggered. Defaults:  $\theta = 0.85$ ,  $K = 5$ .

**Integration with BFS.** `Expand` (Definition 10) checks the overlay after exhausting each node’s explicit adjacency list. Implicit neighbors are synthesized as `RelatedTo` graph edges and participate in the same seen-set, depth budget, and visit budget as explicit relations.

### 6.14 HNSW Approximate Vector Backend

For managed vector sidecars above a configurable size threshold, MentisDB uses a Hierarchical Navigable Small World (HNSW) graph to approximate nearest-neighbor search instead of the exact cosine scan. The exact backend remains the default for small corpora and the fallback if the HNSW graph is unavailable or mismatched.

Let  $T \in \mathbb{N}$  be the HNSW activation threshold (`MENTISDB_HNSW_THRESHOLD`, default 50,000). Given a sidecar  $\mathcal{S}$  with  $|\mathcal{S}|$  vectors, the active backend is

$$\text{Backend}(\mathcal{S}) = \begin{cases} \text{Exact} & \text{if } |\mathcal{S}| < T \text{ or HNSW is disabled,} \\ \text{Hnsw} & \text{otherwise.} \end{cases}$$

Both backends implement the same `VectorSearchBackend` trait and return candidates in the same order-preserving cosine-score units, so the surrounding hybrid retrieval pipeline is unchanged. The HNSW graph is persisted to disk and reloaded on startup when the stored thought count matches the live chain; stale graphs are atomically rebuilt. For large sidecars the graph can be constructed in a background thread while the daemon continues to serve queries with the exact backend.

## 6.15 Decomposed Scores

Each ranked hit exposes a score vector

$$\mathbf{s}(d) = (S_\ell, s_v, S_{\text{graph}}, b_{\text{rel}}, S_{\text{seed}}, S_{\text{imp}}, f_{\text{conf}}, S_{\text{rec}}, S_{\text{coh}}, S_{\text{tot}})$$

together with `matched_terms` and `match_sources`, preserving auditability.

## 7 Deduplication

### 7.1 Jaccard–Supersedes Algorithm

Let  $\mathcal{N}(t) \subseteq \Sigma^*$  denote the normalized token set of thought  $t$  (Porter-stemmed, lemma-expanded). Given threshold  $\theta \in [0, 1]$  and scan window  $w \in \mathbb{N}$ , on append of  $t_n$  with  $\mathcal{N}(t_n) \neq \emptyset$ , MentisDB computes

$$t^* = \arg \max_{t \in \{t_{n-w}, \dots, t_{n-1}\}} J(\mathcal{N}(t_n), \mathcal{N}(t)), \quad J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

If  $J(\mathcal{N}(t_n), \mathcal{N}(t^*)) \geq \theta$ , it auto-emits a relation  $e = (\text{Supersedes}, t^*.id, \perp, \tau_{\text{now}}, \perp)$  on  $t_n$ , and updates  $\mathcal{I}(\chi)$  to include  $t^*.id$  for  $O(1)$  skipping in subsequent retrieval.

**Complexity.** Token normalization is  $O(|c|)$  per record. Jaccard over the window is  $O(w \cdot |\overline{\mathcal{N}}|)$ . With default  $w = 64$  and typical tokens per thought  $\leq 200$ , dedup cost is bounded by a constant factor of append cost.

### 7.2 Configuration

```
MENTISDB_DEDUP_THRESHOLD = theta    in [0, 1]
MENTISDB_DEDUP_SCAN_WINDOW = w      in N (default 64)
```

The superseded thought is retained for audit; no content is deleted. Ranked search deprioritizes it via  $\mathcal{I}(\chi)$ .

## 8 Operational Surfaces

### 8.1 CLI

The daemon `mentisdb` exposes three subcommands over REST at `http://127.0.0.1:9472`: `add`, `search`, `agents`. Synchronous HTTP (`ureq`) avoids pulling an async runtime into the client path.

### 8.2 MCP Server

`mentisdb` exposes a streamable HTTP MCP endpoint at `POST /` (port 9471) with 35 tools covering bootstrap, append, search, read, export/import, agent registry, chain management, and a skill registry. Legacy REST endpoints `POST /tools/list` and `POST /tools/execute` remain available for compatibility.

### 8.3 Skill Registry

The skill registry is a git-like immutable version store for agent instruction bundles. An upload to an existing `skill_id` creates a new immutable version: the first is stored as full content, subsequent versions as unified diff patches. Version reconstruction replays patches from  $v_0$  forward. Content hashes are computed over reconstructed content, decoupling integrity from storage representation. Agents with registered Ed25519 keys must cryptographically sign uploads; signature verification is server-side before acceptance.

## 8.4 Bootstrap Protocol

Modern MCP clients bootstrap from the handshake:

1. `initialize.instructions` directs the agent to read `mentisdb://skill/core`.
2. `resources/read` returns the embedded operating skill.
3. `mentisdb_bootstrap` opens or creates the chain; if empty, writes a genesis checkpoint.
4. `mentisdb_recent_context` loads prior state.

## 9 Empirical Evaluation

### 9.1 Benchmarks

We evaluate MentisDB on two standard long-term memory benchmarks.

| Benchmark                 | Metric | v0.8.1       | v0.8.5       | v0.8.9       |
|---------------------------|--------|--------------|--------------|--------------|
| LoCoMo-2P                 | $R@10$ | <b>88.7%</b> | –            | –            |
| LoCoMo-2P single-hop      | $R@10$ | 90.7%        | –            | –            |
| LoCoMo-10P (1977 queries) | $R@10$ | 74.2%        | <b>74.6%</b> | <b>71.9%</b> |
| LoCoMo-10P single-hop     | $R@10$ | –            | 79.0%        | 75.8%        |
| LoCoMo-10P multi-hop      | $R@10$ | –            | 58.4%        | 57.4%        |
| LoCoMo-10P                | $R@20$ | –            | –            | 79.1%        |
| LongMemEval (fresh chain) | $R@5$  | 67.6%        | –            | <b>66.8%</b> |
| LongMemEval (fresh chain) | $R@10$ | 73.2%        | –            | <b>72.2%</b> |
| LongMemEval (fresh chain) | $R@20$ | –            | –            | 78.0%        |

Table 3: Retrieval quality across MentisDB releases. All v0.8.9 numbers are the mean of independent full-scale runs on 2026-04-14 and 2026-04-17 against fresh chains with default retrieval configuration (`fastembed-minilm` vector sidecar, graph expansion enabled, RRF reranking disabled); the scores were bit-identical across runs.

The v0.8.5 LoCoMo-10P improvement derives from three changes:

1. Session cohesion tuning: radius  $8 \rightarrow 12$ , boost  $0.8 \rightarrow 1.2$ .
2. Doubled edge weights  $b_{\text{rel}}$  across all `ThoughtRelationKind` variants.
3. FastEmbed MiniLM sentence embeddings replacing text-only hashing when the `local-embeddings` feature is compiled.

### 9.2 Scoring Evolution

### 9.3 Near-Miss Analysis (LoCoMo-10P, v0.8.5)

Of 503 misses (gold answer absent from top-10):

The 43.3% figure represents a hard ceiling for BM25-only retrieval on this benchmark. The 25.8% ranking-error bucket is the target for graph-density improvements; the implicit edge overlay is designed to address it by surfacing thoughts semantically adjacent to lexical seeds but not directly related by explicit authored edges.

| Version                                                         | Change                      | LME $R@5$ | LoCoMo-10P $R@10$ |
|-----------------------------------------------------------------|-----------------------------|-----------|-------------------|
| 0.8.0 baseline                                                  | –                           | 57.2%     | –                 |
| 0.8.0 + Porter stemming                                         | token normalization         | 61.6%     | –                 |
| 0.8.0 + tiered fusion + importance                              | vector/lexical balance      | 65.0%     | –                 |
| 0.8.1 + cohesion + smooth fusion + DF cutoff                    | retrieval quality           | 67.6%     | 74.2%             |
| 0.8.5 + cohesion tuning + $b_{\text{rel}} \times 2$ + fastembed | session/graph boost         | –         | 74.6%             |
| 0.8.9 + irregular lemmas + webhooks                             | lemma expansion + events    | 66.8%     | 71.9%             |
| 0.9.8 + ImplicitEdgeOverlay + in-memory VectorIndex             | graph density + latency     | 66.8%     | –                 |
| 0.9.9 + thesaurus auto-by-default + full embeddings             | automatic synonym expansion | 66.8%     | 72.6%             |

Table 4: The 0.9.9 release makes the static thesaurus apply automatically by default to all ranked search queries (REST, MCP, dashboard, CLI). With full fastembed-minilm vectors, this yields LoCoMo-10P  $R@10 = 72.6\%$  (matching the previous target) and LongMemEval  $R@5 = 66.8\%$ . The synthetic-corpus NDCG@5 remains 0.68 (+31% over baseline). Many LongMemEval misses remain lexical gaps even after thesaurus expansion.

| Bucket     | Count | Fraction | Interpretation                                 |
|------------|-------|----------|------------------------------------------------|
| $R@20$ hit | 130   | 25.8%    | close ranking error                            |
| $R@50$ hit | 285   | 56.7%    | moderate signal gap                            |
| $R > 50$   | 218   | 43.3%    | lexical gap (query terms absent from evidence) |

#### 9.4 Near-Miss Analysis (LongMemEval, v0.9.8)

Running v0.9.8 against the LongMemEval 500-question evaluation (10,866-thought chain, fastembed-minilm, default settings,  $\theta = 0.85$ ) produces  $R@5 = 66.8\%$ ,  $R@10 \approx 72.0\%$ ,  $R@20 \approx 77.8\%$  — statistically identical to v0.8.9. Of 166 misses:

By question type, the weakest categories are **single-session-preference** (13.3%  $R@5$ ) and **multi-session** (64.7%  $R@5$ ). The dominant miss signal is lexical (avg top-1 lexical score 21.3 vs. graph 0.006 at  $\theta = 0.85$ ; graph increases to 0.021 at  $\theta = 0.70$  but  $R@5$  declines 0.2% due to noise from spurious neighbor promotions). LME’s near-miss structure is qualitatively different from LoCoMo’s: 65.7% of misses are not recoverable at  $R@20$  regardless of graph topology, because the evidence thoughts share no vocabulary with the query. Graph expansion is the correct tool for LoCoMo multi-hop; query expansion or a larger encoder is the correct tool for LME’s lexical gap.

#### 9.5 Micro-Benchmarks

Criterion micro-benchmarks span five domains: append throughput (**thought\_chain**), baseline search (**search\_baseline**), ranked retrieval (**search\_ranked**), skill registry lifecycle (**skill\_registry**), and HTTP concurrency at  $\{100, 10^3, 10^4\}$  concurrent Tokio tasks with  $p_{50}/p_{95}/p_{99}$  reporting (**http\_concurrency**). A DashMap-based concurrent chain lookup delivers 750–930 read req/s at  $10^4$  concurrent tasks, versus the previous RwLock(HashMap) bottleneck.

## 10 Related Work and Positioning

MentisDB is, to our knowledge, the only system combining (i) embedded storage, (ii) zero LLM dependency in the core path, (iii) cryptographic chain integrity, (iv) hybrid BM25 + vector + graph retrieval with cosine-inferred implicit edges, and (v) a built-in MCP server — all in a single static binary. Identified gaps relative to competitors: custom per-chain entity/relation ontologies (as in Graphiti’s Pydantic models), LLM-driven memory extraction, a browser extension, and per-thought token accounting.

| Bucket        | Count | Fraction | Interpretation      |
|---------------|-------|----------|---------------------|
| $R@10$ hit    | 27    | 16.3%    | close ranking error |
| $R@20$ hit    | 57    | 34.3%    | moderate signal gap |
| not in $R@20$ | 109   | 65.7%    | lexical gap         |

| Feature                  | MentisDB               | Mem0        | Graphiti / Zep   | Letta / MemGPT |
|--------------------------|------------------------|-------------|------------------|----------------|
| Language                 | Rust                   | Python      | Python           | Python / TS    |
| Storage                  | embedded file          | external DB | Neo4j / FalkorDB | external DB    |
| LLM required for core    | <b>No</b>              | Yes         | Yes              | Yes            |
| Cryptographic integrity  | <b>SHA-256 chain</b>   | —           | —                | —              |
| Hybrid retrieval         | BM25+vec+graph         | vec+keyword | sem+kw+graph     | —              |
| Approximate NN (HNSW)    | <b>Yes (default)</b>   | —           | —                | —              |
| Implicit graph inference | <b>cosine-inferred</b> | —           | LLM-extracted    | —              |
| Temporal facts           | $[v_*, v^*]$           | update-only | $[v_*, v^*]$     | —              |
| Deduplication            | Jaccard+Supersedes     | LLM-based   | merge            | —              |
| Agent registry           | Yes                    | —           | —                | Yes            |
| MCP server               | Built-in               | —           | Yes              | —              |

## 11 Discussion, Limitations, and Conclusion

### 11.1 Limitations

- **Ceiling of sparse retrieval.** The near-miss analyses quantify irreducible lexical gaps on both benchmarks: 43.3% on LoCoMo-10P and 65.7% on LongMemEval for BM25-only retrieval. Dense embeddings and implicit graph edges mitigate ranking errors but cannot recover evidence thoughts that share no vocabulary with the query.
- **Implicit edge quality depends on encoder.** The `ImplicitEdgeOverlay` is only as good as the underlying embedding model. With `fastembed-minilm` (384-dim) and the default threshold  $\theta = 0.85$ , LME produces a sparse overlay ( $\sim 5$  K edges for 10 K thoughts) because MiniLM cosine similarities between topically diverse conversations rarely exceed 0.85. Lowering  $\theta$  increases edge density but introduces noise at the graph-score level.
- **Local-only integrity model.** The hash chain provides tamper evidence, not Byzantine fault tolerance or distributed consensus; cross-chain consistency is not enforced cryptographically.
- **Schema churn discipline.** Because bincode tags enum variants by ordinal, schema evolution is append-only at the enum level — reordering or renaming variants would silently corrupt persisted data.

### 11.2 Future Work

- **LoCoMo multi-hop validation of `ImplicitEdgeOverlay`.** The 21-point single-hop / multi-hop gap on LoCoMo-10P (79.0% vs. 57.4%  $R@10$  at v0.8.9) is the primary target for graph-density improvements. A full re-run against v0.9.8 with the overlay active is needed to quantify the gain.
- **Threshold sweep.** A sweep across  $\theta \in \{0.75, 0.80, 0.85, 0.90\}$  and  $K \in \{3, 5, 10\}$  on LoCoMo multi-hop would identify the optimal operating point for the overlay.
- **Thesaurus benchmark validation (completed in 0.9.9).** With the thesaurus now applying automatically by default, LoCoMo-10P reached the target 72.6%  $R@10$ . Long-

MemEval remains at 66.8%  $R@5$  (lexical gaps still dominate many misses). Further tuning of synonym weight and short-evidence handling is future work.

- Per-chain entity/relation ontologies enabling typed domain-specific facts beyond the fixed `ThoughtRelationKind`.
- Episode provenance: tracing derived facts back to source conversations.
- Cross-chain federated retrieval with result reconciliation across distributed ledgers.
- Optional LLM-extracted memories as a layered, auditable transform.
- Self-improving skill registry: agents committing updated skill versions as they learn, with signed provenance.

### 11.3 Conclusion

MentisDB formalizes agent memory as an append-only, hash-chained ledger of semantically typed thoughts, and couples that ledger with a composable retrieval pipeline — BM25 with per-field DF gating, query-time synonym expansion via a static thesaurus and irregular-verb lemmatization, smooth vector–lexical fusion, bounded typed-edge graph expansion augmented by cosine-inferred implicit edges, RRF, session cohesion, and Jaccard-based deduplication — in a single embedded Rust substrate. The `ImplicitEdgeOverlay` extends graph-based retrieval to chains where agents author few explicit relations, deriving a dense implicit graph from vector cosine proximity at  $O(N)$  incremental cost per append. Empirical results on LoCoMo and LongMemEval demonstrate competitive retrieval quality without reliance on external databases or LLM services for the core ingestion path. The system is released as open source and exposes MCP, REST, and HTTPS surfaces for interoperation with contemporary agentic harnesses.

## References

- [1] S. Robertson and H. Zaragoza. *The Probabilistic Relevance Framework: BM25 and Beyond*. Foundations and Trends in Information Retrieval, 3(4), 2009.
- [2] G. V. Cormack, C. L. A. Clarke, and S. Büttcher. *Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods*. In Proc. SIGIR, 2009.
- [3] M. F. Porter. *An Algorithm for Suffix Stripping*. Program, 14(3), 1980.
- [4] P. Jaccard. *Étude comparative de la distribution florale dans une portion des Alpes et des Jura*. Bulletin de la Société Vaudoise des Sciences Naturelles, 1901.
- [5] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. *High-Speed High-Security Signatures*. Journal of Cryptographic Engineering, 2012.
- [6] NIST FIPS PUB 180-4. *Secure Hash Standard (SHS)*, 2015.
- [7] A. Maharana et al. *Evaluating Very Long-Term Conversational Memory of LLM Agents (LoCoMo)*, 2024.
- [8] D. Wu et al. *LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory*, 2024.
- [9] Anthropic. *Model Context Protocol (MCP) Specification*, 2024.