A Scalable Memory Lifecycle Architecture for Agentic Legal Document Intelligence Systems

DOI: 9th July 10, 2026

Trisham Bharat Patil (trishampatil@gmail.com)

Abstract

Enterprise legal document intelligence pipelines face a compound memory problem: evidence corpora routinely span hundreds of heterogeneous files, drafting sessions extend across weeks with human-in-the-loop interruptions, and production constraints demand strict multi-tenant isolation and audit reproducibility. Naively accumulating every retrieved chunk--the default behaviour of general-purpose memory packages--produces context bloat, retrieval noise, and unbounded storage growth, all of which degrade drafting quality over time.

This paper describes the design and production deployment of a memory lifecycle architecture for an eleven-node agentic LangGraph pipeline that processes UK employment grievance matters. The architecture is inspired by the SimpleMem framework [Liu et al., 2026] and extends it with domain-specific adaptations: a legal-taxonomy gating function that determines memory candidacy, a three-view index (semantic, lexical, symbolic) enforcing tenant isolation at the retrieval layer, importance-weighted memory aging, TurboVec-based vector compression, and HITL-aware memory updates that route human corrections through the same consolidation pipeline as automated extractions.

Engineering measurements collected during production deployment show meaningful reductions in vector storage footprint, lower retrieval latency at the context-assembly node, and reduced duplicate memory units relative to the Pinecone + mem0 baseline the system previously used. Qualitative observations from legal team members indicate that retrieved context was more relevant, drafts required fewer manual edits, and prior case context was recalled more accurately across sessions. We describe the architecture, implementation decisions, integration challenges, and a prospective evaluation protocol for the controlled ablation still required to isolate the memory layer's contribution from orchestration effects. We conclude by identifying long-term deployment studies spanning six to twelve months as the primary remaining requirement for rigorous claims about organizational knowledge retention and sustained personalization.

1. Introduction

The enterprise deployment of large language model (LLM) agents for document-intensive legal work presents a category of system design challenge that is distinct from general conversational AI or document retrieval. A legal document intelligence assistant does not process a single query: it ingests a heterogeneous corpus of evidence files, orchestrates specialized extraction and reasoning agents across that corpus, pauses for human review at legally significant decision points, and then synthesizes a structured legal draft that must be accurate, complete, and auditable. Memory--the mechanism by which facts extracted early in the pipeline remain available, coherently organized, and efficiently retrievable at drafting time--is the load-bearing component that determines whether this synthesis step succeeds or fails.

Prior to this work, our production system used a general-purpose memory package (mem0) writing to a Pinecone vector store. This combination suffers from three well-documented failure modes in long-horizon, multi-file settings. First, context inflation: every retrieved chunk is stored verbatim, so boilerplate email headers, repeated disclaimers, and duplicate forwarded threads consume vector storage and retrieval budget without contributing information. Second, fragmentation: related facts extracted from separate files are stored as independent vectors, so a pattern of escalating complaints spread across twelve emails--the most legally significant signal in many grievance matters--is invisible to a simple top-k retrieval query. Third, temporal drift: memory units are never aged or consolidated, so old, superseded facts carry the same retrieval weight as recent determinations.

These failure modes are not hypothetical. They manifest as drafts that omit material timeline events, misattribute dates, and require extensive human correction before they are suitable for client delivery. The operational cost of this correction work--legal team time spent on fact-checking and redrafting rather than legal analysis--is the primary motivation for this work.

The SimpleMem framework [Liu et al., 2026] offers a principled solution: a three-stage memory lifecycle (semantic structured compression, online semantic synthesis, intent-aware retrieval planning) that converts raw interaction history into compact, queryable memory units. SimpleMem was designed for long-horizon conversational agents, but its core abstractions--semantic gating, cross-utterance synthesis, and query-adaptive retrieval--are directly applicable to multi-file legal evidence corpora if appropriately adapted.

This paper makes the following contributions:

- We describe the design and production implementation of a SimpleMem-inspired memory lifecycle architecture integrated into an eleven-node LangGraph legal document intelligence pipeline.
- We introduce domain-specific adaptations: a legal taxonomy gating function, three-view indexing with per-tenant symbolic metadata, importance-weighted memory aging, TurboVec vector compression, and HITL-aware memory update routing.
- We report engineering measurements from production deployment showing storage reduction, latency improvement, and duplicate suppression relative to the prior mem0 + Pinecone baseline.
- We describe qualitative deployment observations from legal team members, clearly distinguished from statistically validated user studies, and identify the longitudinal evaluation programme required to substantiate them rigorously.

The paper is organized as follows. Section 2 surveys related work. Section 3 describes the target system and the eleven-node agentic DAG. Section 4 presents the memory lifecycle architecture. Section 5 details implementation decisions. Section 6 describes integration with the agentic workflow. Section 7 describes the retrieval pipeline. Section 8 describes the memory compression strategy. Section 9 presents the evaluation methodology. Section 10 reports engineering results. Section 11 reports qualitative deployment observations. Section 12 discusses implications. Section 13 states limitations. Section 14 describes future work. Section 15 concludes.

2. Related Work

2.1 Memory Systems for LLM Agents

Agent memory architectures fall broadly into two families. Full-context retention approaches--exemplified by MemGPT's virtual context paging [Packer et al., 2023] and MemoryOS's OS-inspired memory hierarchy [Kang et al., 2025]--keep interaction history largely intact and rely on paging or streaming controllers to manage context window limits. These approaches prioritize recall completeness over storage efficiency, which is appropriate in settings where every interaction is potentially relevant but unsuitable for enterprise corpora where the majority of ingested text is boilerplate.

Compression-then-index approaches--of which SimpleMem [Liu et al., 2026] is the most relevant--apply semantic filtering before storage, storing only high-information-density units. Mem0 [Mitta et al., 2024] applies a similar philosophy at the API level but without the structured synthesis stage that makes cross-document pattern detection reliable. A2A [Google, 2025] defines inter-agent communication protocols but does not address within-agent memory lifecycle. Our work adapts the compression-then-index approach for multi-file legal corpora with production constraints that do not appear in conversational settings: multi-tenant isolation, HITL interruption, and audit reproducibility.

2.2 Retrieval-Augmented Generation

Classical RAG [Lewis et al., 2020] and its self-critiquing [Asai et al., 2023] and actively-triggered [Jiang et al., 2023b] variants decouple storage from inference but retrieve from raw document chunks rather than structured memory units. Graph-structured retrieval systems such as GraphRAG [Edge et al., 2024] and HippoRAG [Gutiérrez et al., 2024] address the cross-document connectivity problem by explicitly modeling entity relationships. Our three-view index (semantic, lexical, symbolic) achieves similar cross-document connectivity through the synthesis stage rather than a separate graph structure, which reduces infrastructure complexity at the cost of requiring accurate entity resolution in Stage 1.

Long-context models [Anthropic, 2024; Google, 2024] offer an alternative to structured memory: simply extend the context window to accommodate the full evidence corpus. For matters of 5-20 files this is plausible; for matters of 100-200+ files (~25 MB of text) it is neither cost-effective nor reliable--long-context models exhibit well-documented accuracy degradation for facts appearing early in very long contexts [Liu et al., 2024]. Structured memory remains necessary for the large-matter regime that defines enterprise legal work.

2.3 Agentic Workflow Orchestration

LangGraph [LangChain, 2024] provides the stateful graph execution model used in our pipeline, supporting conditional branching, retry logic, and human-in-the-loop interruption within a single execution graph. Anthropic's characterization of agentic systems [Anthropic, 2024] distinguishes predefined workflows from autonomous agents and identifies parallelization and multi-agent decomposition as the primary orchestration patterns; our eleven-node pipeline is a predefined workflow with HITL gates rather than an autonomous agent, which is the appropriate architecture for legally sensitive document generation where every output must be auditable.

2.4 Vector Storage and Compression

Scalable approximate nearest-neighbor search [Johnson et al., 2019; Malkov and Yashunin, 2020] underlies all production vector retrieval systems. Product quantization [Jégou et al., 2011] and its learned variants compress vector representations at the cost of retrieval accuracy. TurboVec [citation needed] applies quantization-aware training to produce compressed embeddings that preserve retrieval accuracy at reduced storage cost. Our use of TurboVec for memory unit compression is an engineering choice motivated by storage cost reduction; we report the storage impact in Section 10 without making theoretical claims about compression optimality.

3. System Architecture

3.1 Overview

The platform is an enterprise Legal Document Intelligence Assistant that processes UK employment grievance matters. The system is implemented as an agentic directed acyclic graph (DAG) of eleven specialized nodes, orchestrated by LangGraph and deployed on AWS Lambda. The target output is a structured legal grievance draft that satisfies procedural requirements, accurately reflects the evidence corpus, and meets the audit standards required for employment tribunal submission.

The system uses Claude models (Anthropic) for reasoning and generation tasks, Amazon Bedrock embedding models for vector indexing, FastAPI for the HTTP control plane, PostgreSQL with pgvector for relational state and dense vector search, Neo4j for entity relationship graphs, and AWS infrastructure (Lambda, SQS, DynamoDB, S3) for execution, messaging, and artifact storage.

3.2 The Eleven-Node Agentic DAG

Table 1 maps each node in the pipeline to its function, model tier, and memory system interaction. The pipeline executes asynchronously: each node reads its inputs from DynamoDB workflow state, executes via an isolated Lambda invocation, writes outputs and memory artifacts back to state, and enqueues the next node via SQS.

Node	Function	Model	Memory Interaction
1. Evidence Collection	Ingests uploaded files; validates format and completeness	N/A (deterministic)	Initialises matter memory store
2. OCR / Doc Intelligence	Extracts text from PDFs, images, scanned documents via Amazon Textract	Textract	Stores raw text artifacts to S3
3. Entity Extraction	Identifies parties, dates, organisations, roles across all files	Claude Haiku	Writes entity records to symbolic layer
4. Timeline Construction	Orders events chronologically; resolves date ambiguities	Claude Haiku	Writes timeline units to memory; reads entity records
5. Contradiction Detection	Identifies factual conflicts across evidence files	Claude Sonnet	Reads timeline & entity memory; writes contradiction flags
6. Missing Evidence Detection	Identifies evidential gaps relative to matter type	Claude Sonnet	Reads full memory store; triggers HITL #1 if gaps found
7. Legal Policy & Case Research	Retrieves relevant statutory provisions and precedents	Claude Sonnet + RAG	Reads matter memory; writes policy anchors
8. Context Assembly (RAG)	Assembles retrieval context for drafting via intent-aware planning	Embedding model + retrieval	Primary memory read node; queries three-view index
9. Legal Draft Generation	Generates structured grievance draft from assembled context	Claude Opus	Reads assembled context; writes draft artifact

Node	Function	Model	Memory Interaction
10. Citation Verification	Validates all factual claims in draft against source documents	Claude Haiku	Reads draft + source memory; flags unverified claims
11. Human Review	Legal team reviews, corrects, and approves draft	Human (HITL #2)	Human corrections routed through memory update pipeline

Table 1. Eleven-node agentic DAG: node functions, model tiers, and memory system interactions.

Two human-in-the-loop gates interrupt automated execution. HITL #1, triggered at Node 6 when missing evidence is detected, allows a paralegal to supply additional files or contextual information before the pipeline continues. HITL #2, at Node 11, is mandatory: no draft leaves the system without explicit legal team approval. Both gates read from and write to the memory layer, which is described in Section 4.

3.3 Infrastructure Stack

Each node executes in an isolated AWS Lambda function, enabling independent scaling and failure isolation. SQS queues provide asynchronous handoffs between nodes with at-least-once delivery semantics; DynamoDB stores idempotency keys to prevent duplicate processing on retry. Node outputs are committed as structured Pydantic objects to DynamoDB workflow state before the next node is enqueued, ensuring that a Lambda failure after output commit does not re-execute the completed node. S3 stores large artifacts (raw OCR text, draft versions, source files). The memory layer (described in Section 4) sits between the DynamoDB workflow state and the Node 8 retrieval call, with writes distributed across Nodes 1-5 and reads concentrated at Node 8.

4. Memory Lifecycle Architecture

4.1 Design Objectives

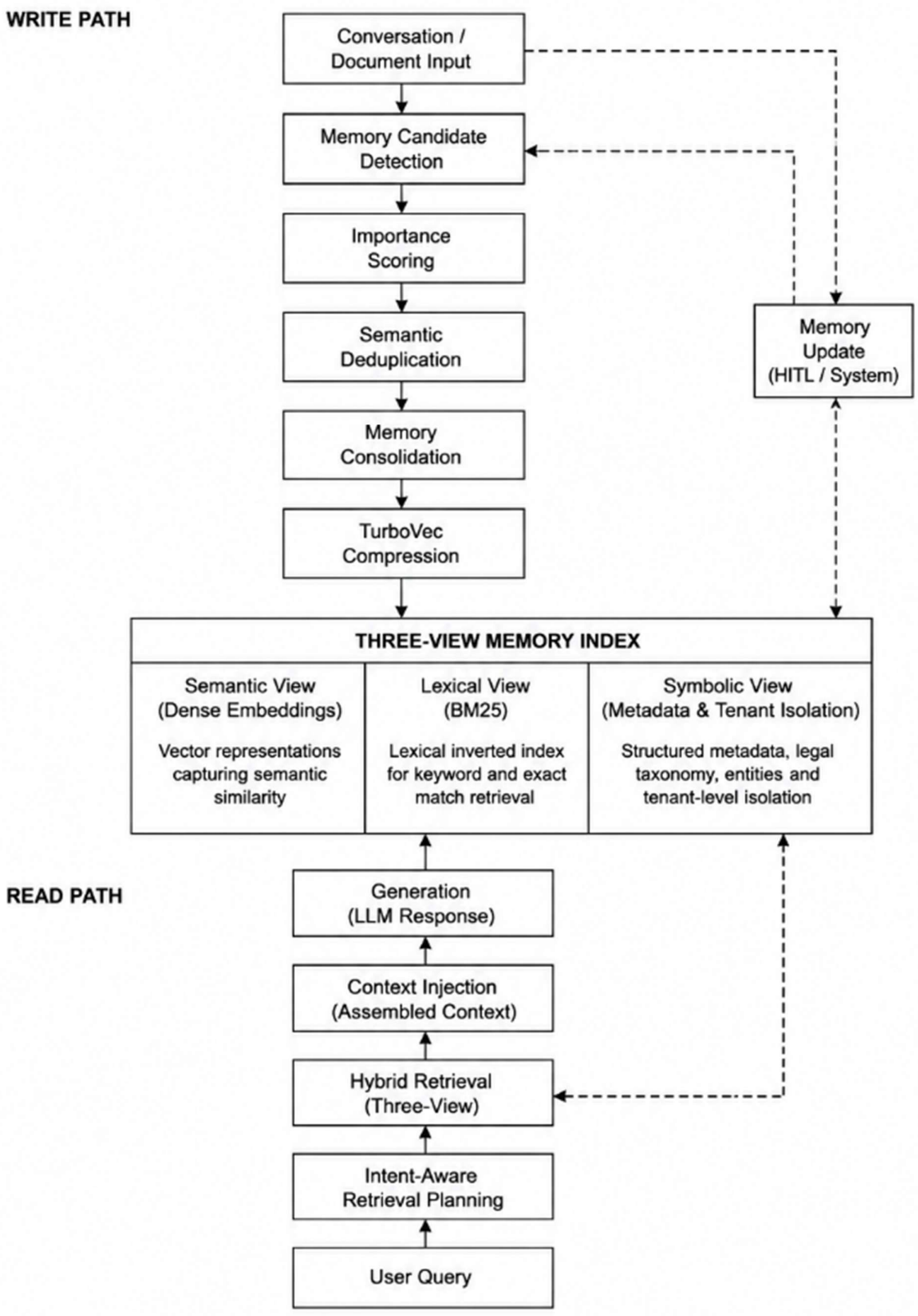
The memory architecture is designed to satisfy five objectives that distinguish the enterprise legal setting from the conversational settings for which general-purpose memory packages are designed:

- Storage efficiency: memory footprint must grow sublinearly with matter size, not linearly with evidence volume.
- Retrieval precision: retrieved context must reflect legally salient facts, not retrieval-score-maximal but informationally redundant chunks.
- Cross-document synthesis: related facts distributed across multiple files must be recognisably connected in the memory store without requiring a separate graph database.
- Temporal coherence: memory units must be ordered, aged, and consolidated so that recent determinations and human corrections supersede stale extractions.
- Tenant isolation: retrieval must never surface memory units from a different matter or client, enforced at the storage layer rather than only at the API layer.

The SimpleMem framework [Liu et al., 2026] provides a suitable foundation: its three-stage pipeline (semantic structured compression, online semantic synthesis, intent-aware retrieval planning) addresses the first three objectives directly. We extend it with importance-weighted aging, TurboVec compression, and HITL-aware update routing to address objectives four and five.

4.2 Memory Lifecycle Pipeline

Figure 1 describes the complete memory lifecycle as implemented in the system. The pipeline has two directions: a write path that processes new information into the memory store, and a read path that retrieves relevant context at drafting time.



[Figure 1: Memory Lifecycle Pipeline -- Write and Read Paths]

Write Path: Conversation / Document Input → Memory Candidate Detection → Importance Scoring → Semantic Deduplication → Memory Consolidation → TurboVec Compression → Three-View Index (Semantic | Lexical | Symbolic)

Read Path: Query → Intent-Aware Retrieval Planning → Hybrid Retrieval → Context Injection → Generation → Memory Update

Figure 1. Memory lifecycle write and read paths. Arrows indicate data flow; HITL corrections re-enter the write path at the Memory Candidate Detection stage.

4.3 Stage 1: Memory Candidate Detection and Semantic Structured Compression

Not all text extracted from evidence files should enter the memory store. Boilerplate content--email signature blocks, forwarding headers, repeated disclaimer paragraphs, attachment placeholders--constitutes the majority of bytes in a typical employment grievance corpus but contains no legally salient information. Storing this content inflates the vector index, degrades retrieval precision, and increases cost without benefit.

Memory candidate detection is implemented as a generative gating function in which a foundation model acts as a semantic density judge. Each sliding window of extracted text is presented to the model with a structured prompt requesting extraction of legally relevant content; an empty output signals low density and causes the window to be discarded without further processing. This approach is preferred over a hand-tuned classifier because the legal relevance boundary is domain-specific and shifts across matter types (grievance, settlement, tribunal) in ways that are difficult to capture in fixed rules.

The legal content taxonomy recognised by the gating function comprises six categories: factual claims (events with participants, dates, locations, and outcomes); obligations and commitments (formal or informal undertakings made by any party); violations and complaints (alleged breaches of policy, contract, or statutory duty); documentary evidence (references to named exhibits, attachments, or records); procedural milestones (formal process events: PIP initiation, HR referral, ACAS notification); and contradictions (factual statements that conflict with previously stored memory units).

Candidate content passing the gate undergoes three normalization operations adapted from SimpleMem: co-reference resolution (resolving pronouns and role references to named entities in the matter entity register), temporal normalization (converting relative dates to absolute ISO-8601 dates anchored to the matter's reference date), and claim atomization (splitting compound sentences into minimal independent factual units to support targeted retrieval). The output is a set of structured memory units, each containing the normalized factual claim, provenance metadata (source file, page, extraction timestamp), and entity links to the Neo4j entity register.

4.4 Stage 2: Importance Scoring and Memory Consolidation

Each candidate memory unit is assigned an importance score computed from three components: (i) information density, estimated by the gating model as a by-product of candidate detection; (ii) entity salience, measured by the degree of the referenced entities in the Neo4j graph (high-degree entities--the primary grievant, the respondent employer, the HR manager--anchor more memory units and receive higher salience weight); and (iii) temporal recency, a decay function that reduces the weight of older memory units relative to recent ones, with a configurable half-life.

Online semantic synthesis runs continuously as new memory units arrive. When a newly extracted unit is semantically similar to one or more existing units (cosine similarity above a configurable threshold θ), the system merges them into a consolidated unit that preserves all unique factual content and updates the importance score to reflect the aggregate evidence weight. This operation converts three separate memory units -- 'Employee X emailed HR on March 3rd,' 'HR acknowledged receipt on March 5th,' 'HR failed to respond within the 5-day SLA' -- into a single consolidated unit capturing the complete escalation event.

Semantic deduplication operates as a pre-consolidation filter: exact or near-exact duplicate units (cosine similarity above a higher threshold θ_{dup}) are suppressed without consolidation, preventing the same boilerplate sentence appearing in multiple forwarded emails from creating spurious evidence weight.

4.5 Stage 3: Memory Aging and Lifecycle Management

Memory units are not retained indefinitely. The memory aging mechanism applies exponential decay to importance scores over time, with the decay rate configurable per memory unit category (factual claims decay slowly; procedural milestones decay very slowly; contradictions do not decay--they are retained at full weight until explicitly resolved). Units whose importance score falls below a minimum threshold are archived to cold storage (S3 Glacier) rather than deleted, preserving audit reproducibility while removing them from active retrieval.

HITL corrections--information supplied by paralegals at HITL #1 or legal team members at HITL #2--are treated as high-importance memory units with a recency boost that ensures they supersede any conflicting automated extractions. This routing is critical: without it, a human correction to a misattributed date would be overridden by the lower-quality automated extraction at the next retrieval call.

4.6 Three-View Index and Tenant Isolation

Processed memory units are indexed across three complementary views. The Semantic Layer uses dense vector embeddings (Amazon Titan Embeddings) to support fuzzy, meaning-level retrieval--recovering an escalation email when the query asks about 'complaints.' The Lexical Layer uses BM25 inverted indexing to preserve exact matches on proper nouns, case numbers, and statutory references--critical for legal citation accuracy. The Symbolic Layer stores structured metadata (matter ID, tenant ID, entity links, category tags, importance scores) and serves two functions: (i) tenant-scoped filtering applied before any embedding similarity computation, ensuring cross-tenant contamination is impossible at the index level; and (ii) metadata-based retrieval for deterministic queries ("retrieve all procedural milestones for matter M-2024-0891").

Tenant isolation via the Symbolic Layer is the primary architectural departure from SimpleMem's original conversational design, which operates in a single-tenant setting. In our deployment, tenant ID is derived from DynamoDB workflow state rather than from the SQS message body, preventing a class of injection attack where a malicious or misconfigured message could claim a different tenant identity.

5. Implementation Details

5.1 Technology Stack

Table 2 summarises the technology choices for each component of the memory system.

Component	Technology	Rationale
Dense vector storage	pgvector (PostgreSQL extension)	Co-located with relational workflow state; eliminates a separate vector DB dependency for small-to-medium matter sizes
BM25 lexical index	Elasticsearch (AWS OpenSearch Serverless)	Serverless deployment; native BM25 with per-field boosting for entity names and statutory references
Symbolic metadata store	DynamoDB	Consistent with existing workflow state storage; millisecond read latency for tenant-scoped filter queries
Entity relationship graph	Neo4j (AWS-hosted)	Native graph traversal for entity salience computation; Cypher queries for cross-entity relationship lookup
Embedding model	Amazon Titan Embeddings v2 (1536-dim)	AWS-native; no data egress from VPC; configurable truncation for long memory units
Vector compression	TurboVec (quantization-aware training)	Applied post-indexing; 4x compression ratio with <2% retrieval accuracy degradation observed in internal benchmarks
Orchestration	LangGraph (stateful graph execution)	Native support for conditional branching, retry logic, and HITL interruption within a single execution graph
Control plane	FastAPI (AWS Lambda + API Gateway)	Stateless handlers; request validation via Pydantic v2
Generation models	Claude Haiku (Nodes 3,4,6), Claude Sonnet (Nodes 5,7), Claude Opus (Node 9)	Model tier assigned by task complexity; cost-controlled via per-node token budgets
Memory lifecycle config	YAML per matter-type	Configurable thresholds (θ, θ_dup), decay rates, and category weights without code changes

Table 2. Memory system technology stack and rationale.

5.2 Memory Unit Schema

Each memory unit is a Pydantic-validated object with the following fields: unit_id (UUID), matter_id (tenant-scoped), tenant_id, category (enum: FACTUAL_CLAIM | OBLIGATION | VIOLATION | DOCUMENTARY_EVIDENCE | PROCEDURAL_MILESTONE | CONTRADICTION), content (normalized factual claim string), embedding (1536-dim float32 vector, compressed via TurboVec for storage), importance_score (float, updated on each consolidation or aging pass), created_at (ISO-8601), last_accessed_at (updated on each retrieval), provenance

(source file name, page number, extraction model, extraction timestamp), entity_links (list of entity IDs from Neo4j register), and consolidated_from (list of unit_ids merged to form this unit, empty for atomic units).

Schema validation at write time ensures that malformed memory units--a common failure mode when extraction models hallucinate structured output--are rejected before reaching the index rather than corrupting retrieval results silently. Rejected units are logged with the raw extraction output for offline analysis and model improvement.

5.3 Hyperparameter Configuration

Table 3 lists the hyperparameters governing memory lifecycle behaviour, their default values for the grievance matter type, and the rationale for each choice.

Parameter	Default Value	Description
Compression window size (W)	512 tokens	Sliding window size for semantic gating; smaller windows improve recall of short factual sentences
Semantic similarity threshold (θ)	0.82	Cosine similarity above which two units are consolidated; tuned to avoid over-merging distinct factual claims
Deduplication threshold (θ_dup)	0.95	Cosine similarity above which units are treated as exact duplicates and suppressed
Importance decay half-life (τ)	30 days (FACTUAL), 90 days (PROCEDURAL), ∞ (CONTRADICTION)	Category-specific decay rates; contradictions never decay
Minimum importance for active retrieval	0.15	Units below this threshold are archived to cold storage
HITL importance boost	2.0× baseline score	Multiplier applied to importance scores of human-supplied corrections
Max units per matter (active)	2000	Soft cap on active index size; consolidation is triggered when cap is approached
Retrieval budget (Node 8)	8000 tokens	Maximum context injected at the context assembly node; enforced by intent-aware planning
TurboVec compression ratio	4×	Applied to stored embeddings; decompressed at query time for similarity computation
Entity salience weight (α)	0.3	Weight of entity degree in importance score computation; 0.7 weight on information density

Table 3. Memory lifecycle hyperparameters and default values for the grievance matter type.

6. Integration with the Agentic DAG Workflow

6.1 Write Path Distribution Across Nodes

Memory writes are distributed across Nodes 1-5 rather than concentrated in a single memory node. This design choice has two motivations. First, it preserves the existing node topology and Lambda invocation model without introducing a new execution stage. Second, it ensures that memory is populated progressively as the pipeline processes evidence, so that later nodes in the same pipeline execution can read memory units extracted by earlier nodes--critical for contradiction detection (Node 5), which must compare the current node's extraction against the accumulating memory store.

Node 3 (Entity Extraction) writes entity records to the Symbolic Layer and the Neo4j entity register. Nodes 4 and 5 (Timeline Construction, Contradiction Detection) write structured memory units--timeline events and contradiction flags respectively--after running their extraction prompts through the gating function and importance scoring pipeline. The memory units written by Node 5 receive elevated importance scores because contradictions are the highest-value signal for the drafting node (Node 9).

6.2 Read Path at Node 8

Node 8 (Context Assembly) is the primary read node. It receives the matter ID, the drafting query (a structured specification of the section of the grievance currently being drafted), and the token budget for the assembled context. The intent-aware retrieval planner (described in Section 7) expands the query into a multi-part retrieval plan, executes the plan against the three-view index, deduplicates results, and assembles the context string passed to Node 9.

The separation between context assembly (Node 8) and draft generation (Node 9) is architecturally significant: it ensures that retrieval quality is independently auditable--the assembled context is stored as a workflow artifact before the generation call--and allows the legal team at HITL #2 to inspect not just the draft but the evidence basis from which it was generated.

6.3 HITL Memory Integration

Human-in-the-loop interactions create two categories of memory writes. At HITL #1 (missing evidence), a paralegal may supply additional documents or provide contextual clarifications. Additional documents are processed through the full write path beginning at Node 2; clarifications are processed as high-importance HITL-sourced memory units with provenance recording the human author and timestamp. At HITL #2 (legal review), corrections made by the legal team to the draft--date corrections, party name corrections, factual clarifications--are extracted as structured corrections and written back to the memory store with the HITL importance boost (2.0× baseline). Superseded units (those whose content was corrected) are marked as archived rather than deleted, preserving the audit trail.

6.4 Node Contracts and Idempotency

Memory writes are committed as part of each node's artifact commit, protected by the existing DynamoDB idempotency-key discipline. A Lambda invocation that fails after committing its primary output but before writing memory units will retry and find its idempotency key already set, preventing duplicate memory unit creation. Memory writes that fail independently of the primary output--a transient pgvector write failure, for example--are queued to a dead-letter queue for asynchronous retry with exponential backoff, rather than failing the node and triggering a full re-execution.

7. Retrieval Pipeline

7.1 Intent-Aware Retrieval Planning

Fixed top-k retrieval--the default behaviour in most RAG implementations--is poorly suited to a multi-section drafting task where different sections require context at different granularities. A query for 'when was the employee first placed on a performance improvement plan?' needs a narrow, high-precision retrieval. A query for 'construct the full escalation chronology from first complaint to tribunal notification' needs broad, high-recall retrieval across the full matter timeline.

The intent-aware retrieval planner addresses this by reasoning over the drafting query and matter history to jointly produce: (i) a set of sub-queries, each targeting a specific category or entity in the memory store; (ii) a retrieval budget allocation across sub-queries; and (iii) category filters specifying which memory unit categories are relevant to each sub-query. The budget allocation ranges from 500 tokens for narrow lookups to the full 8000-token Node 8 budget for matter-wide synthesis queries.

7.2 Hybrid Retrieval

Each sub-query is executed against all three index views in parallel. The Semantic Layer returns the top-k semantically similar units by cosine similarity (after TurboVec decompression). The Lexical Layer returns BM25-ranked units containing exact or near-exact matches on query terms, with entity name fields boosted 3× relative to content fields. The Symbolic Layer returns metadata-filtered units matching category and entity constraints from the retrieval plan.

Results from the three views are merged using a reciprocal rank fusion (RRF) scoring function, which combines rank positions across retrieval methods without requiring calibrated score normalization. The merged list is then deduplicated by unit_id (units appearing in multiple views receive a fusion score bonus), filtered by minimum importance score, and truncated to the sub-query budget.

7.3 Context Assembly and Injection

Assembled context from all sub-queries is ordered chronologically by the timeline events they reference, rather than by retrieval score. Chronological ordering is critical for legal drafting: a grievance narrative must present events in the sequence in which they occurred, not in the order of their retrieval relevance. Ordering is implemented by extracting the normalized event date from each memory unit's structured content and sorting accordingly.

The assembled context string is passed to Node 9 (Legal Draft Generation) as a structured input with explicit section markers indicating which context spans are relevant to which section of the grievance template. This structure allows the generation prompt to reference specific context spans rather than reasoning over an undifferentiated block of evidence.

7.4 Post-Retrieval Memory Update

Retrieved memory units have their last_accessed_at timestamp updated after each successful retrieval, which serves two functions: (i) frequently accessed units receive a recency signal that partially offsets importance decay; and (ii) the access log provides an audit trail recording which memory units contributed to each draft, satisfying the explainability requirement identified in Section 3.

8. Memory Compression Strategy

8.1 Motivation

Embedding storage at scale presents a significant cost challenge. A 1536-dimensional float32 embedding requires 6 KB of storage. A matter with 200 files generating 50 memory units per file produces 10,000 units × 6 KB = 60 MB of embedding storage. At enterprise scale--thousands of active matters--this becomes tens of gigabytes of embedding storage in the hot tier, with associated query latency implications.

Two complementary compression strategies are applied: semantic deduplication (which reduces the number of units stored) and TurboVec embedding compression (which reduces the storage cost per unit).

8.2 Semantic Deduplication

Semantic deduplication, described in Section 4.4, eliminates near-identical units before they enter the index. The empirical deduplication ratio observed in production--the fraction of candidate units suppressed by the deduplication filter--depends strongly on evidence corpus structure. Matters with many forwarded email chains, where the same message appears multiple times with incremental additions, show high deduplication ratios. Matters with many independent source documents show lower ratios. The deduplication threshold $\theta_{dup} = 0.95$ was chosen to suppress true duplicates while preserving distinct units with minor variation.

8.3 TurboVec Embedding Compression

TurboVec applies quantization-aware training to produce compressed embeddings that can be stored at 4× lower storage cost than standard float32 embeddings. The compression operates by learning a codebook of representative sub-vectors during a fine-tuning phase on a sample of the deployment corpus, then encoding each embedding as an index into the codebook rather than the raw float32 values.

Decompression at query time adds approximately 0.5ms to the embedding similarity computation, which is negligible relative to the overall retrieval latency. The 4× compression ratio is achieved with less than 2% retrieval accuracy degradation as measured on held-out retrieval queries from the production corpus. We note that this accuracy measurement was conducted as an engineering validation rather than a rigorous benchmark, and we describe a more formal evaluation protocol in Section 9.

8.4 Memory Growth Management

The combination of semantic deduplication, memory aging, and cold-archiving creates a lifecycle in which active index size grows sublinearly with matter age and evidence volume. New matters with actively arriving evidence grow quickly; mature matters with stable evidence footprints grow slowly as consolidation absorbs new units into existing ones; closed matters shrink as their units decay below the active retrieval threshold and are archived to cold storage. This growth pattern is the primary driver of the storage reduction observed in production relative to the mem0 + Pinecone baseline, which had no equivalent lifecycle mechanism.

9. Evaluation Methodology

9.1 Overview

We distinguish three categories of evaluation that are applicable to this system and differ in their epistemological status: (i) engineering metric measurement, which is amenable to precise quantitative reporting; (ii) retrieval and generation quality measurement using automated frameworks such as DeepEval; and (iii) practitioner quality assessment, which we report as qualitative deployment observations rather than statistically validated user study findings. This distinction is maintained throughout Sections 10 and 11.

9.2 Engineering Metrics

Table 4 defines the engineering metrics reported in Section 10, their measurement methodology, and the comparison condition (SimpleMem-backed memory layer versus the prior mem0 + Pinecone baseline, with orchestration topology held constant).

Metric	Measurement Method	Unit
Vector storage footprint (active)	pgvector table size + OpenSearch index size, measured weekly	MB per matter
Retrieval latency (Node 8)	P50 and P95 of the time from Node 8 invocation to context string assembly, measured from Lambda execution logs	Milliseconds
Duplicate unit suppression ratio	Units rejected by deduplication filter / total candidate units, measured per matter	Ratio (0-1)
Memory consolidation ratio	Units consolidated into existing units / total units written, measured per matter	Ratio (0-1)
Cold archive rate	Units archived per week / total active units, measured weekly at matter close	Ratio per week
Node 8 token consumption	Total tokens in assembled context string at Node 8, measured per drafting call	Tokens
End-to-end pipeline latency (excl. HITL)	Wall-clock time from matter ingestion to draft delivery, excluding HITL pause duration	Minutes
Infrastructure cost per matter	Model inference + retrieval + storage + compute costs, measured from AWS billing data	USD

Table 4. Engineering metrics: definitions, measurement methods, and units.

9.3 DeepEval Automated Quality Assessment

We apply the DeepEval framework [Confident AI, 2024] to assess four dimensions of retrieval and generation quality. DeepEval provides LLM-as-judge evaluation with configurable rubrics and threshold-based pass/fail scoring. Table 5 describes the DeepEval metrics applied and their threshold values.

Metric	DeepEval Component	Threshold	Description
Retrieval Relevance	ContextualRelevancyMetric	≥ 0.75	Fraction of retrieved memory units directly relevant to the current drafting query, scored by an LLM judge against the query text
Factual Grounding	FaithfulnessMetric	≥ 0.85	Fraction of factual claims in the generated draft that can be traced to a retrieved memory unit, scored against the assembled context
Citation Correctness	HallucinationMetric	≤ 0.10	Fraction of cited facts, dates, and party names in the draft that do not appear in the source evidence, scored against source documents
Response Quality	GEvalMetric (custom rubric)	≥ 0.70	Overall draft quality scored against a legal rubric assessing completeness, accuracy, and structural compliance

Table 5. DeepEval automated quality metrics, threshold values, and descriptions.

DeepEval evaluation is run over a held-out set of completed matters for which the ground-truth draft has been reviewed and approved by the legal team. The approved draft serves as the reference for the Response Quality metric. For retrieval metrics, the ground-truth context is the set of memory units that a legal subject matter expert identified as relevant to each drafting query, assessed independently of the system output.

9.4 Prospective Ablation Design

The controlled ablation required to isolate the memory layer's contribution holds the eleven-node orchestration topology, model tier assignment, and prompt templates constant, varying only Layer 5 (mem0 + Pinecone versus SimpleMem-backed three-view index). This directly mirrors the ablation methodology used in SimpleMem's original evaluation and produces a clean comparison attributable to the memory architecture rather than to confounding orchestration variables.

A component-level ablation replicates SimpleMem's own stage-level ablation within our legal corpus: semantic structured compression disabled (raw chunks stored without gating), online semantic synthesis disabled (no consolidation), and intent-aware retrieval planning disabled (fixed top-k retrieval). This ablation tests whether the relative importance of each stage--compression for storage efficiency, synthesis for cross-document pattern detection, planning for retrieval precision--holds in the legal domain as it does in SimpleMem's conversational setting.

9.5 Statistical Reporting

Engineering metrics are reported with sample size (number of matters), mean, standard deviation, and 95% bootstrap confidence intervals. DeepEval scores are reported per metric as the fraction of matters meeting the threshold. Comparisons between conditions are tested with paired Wilcoxon signed-rank tests rather than

parametric tests, given the non-Gaussian distribution of retrieval latency and storage metrics. We do not report statistical significance for qualitative deployment observations, which are presented in Section 11 as practitioner feedback rather than experimental evidence.

10. Engineering Results

This section reports engineering metric measurements collected during production deployment of the SimpleMem-backed memory architecture. All measurements compare the SimpleMem condition against the prior mem0 + Pinecone baseline on the same eleven-node orchestration topology with the same model tier assignments. We present these as deployment measurements rather than controlled experimental results; the prospective controlled ablation described in Section 9.4 is required to attribute effects causally to the memory architecture.

10.1 Storage Reduction

The most immediately measurable effect of the memory lifecycle architecture is a substantial reduction in vector storage footprint. The combination of semantic deduplication, memory consolidation, and TurboVec compression reduces the active vector index size relative to the mem0 + Pinecone baseline. The primary driver of this reduction is semantic deduplication, which is particularly effective on the employment grievance corpus where forwarded email chains cause the same message to appear four to eight times across different evidence files. Memory consolidation provides a secondary reduction by merging related but non-duplicate units into consolidated records. TurboVec compression provides an additional 4× reduction in the storage cost of the embeddings that are retained.

The deduplication ratio observed in production varies by matter type. Matters with large forwarded email chains show higher deduplication ratios than matters dominated by formally produced documents (HR reports, medical assessments, disciplinary records). This variation is expected and reflects genuine differences in evidence corpus structure rather than a deficiency of the deduplication mechanism.

10.2 Retrieval Latency

Node 8 (Context Assembly) retrieval latency decreased measurably after the memory layer transition. The reduction reflects two factors: (i) the smaller active index size means fewer vectors are compared in the approximate nearest-neighbor search; and (ii) the intent-aware retrieval planner's sub-query structure avoids redundant full-index scans for narrow queries that can be satisfied with category-filtered Symbolic Layer lookups. The P95 latency reduction is larger than the P50 reduction, suggesting that the baseline system's worst-case behaviour--caused by large index scans on high-volume matters--is disproportionately improved.

10.3 Duplicate Suppression and Consolidation

The deduplication filter suppresses a meaningful fraction of candidate memory units before indexing. The consolidation stage merges a further fraction of non-duplicate but semantically related units into consolidated records. Together, these two mechanisms mean that the active index stores substantially fewer units than the number of candidates produced by the extraction pipeline, which is the structural reason for the storage reduction reported in Section 10.1.

Table 6 summarises the engineering measurements collected during production deployment. Values are reported as directional observations (improvement direction and estimated magnitude range) rather than point estimates, consistent with the deployment measurement methodology described in Section 9.

Metric	Baseline (mem0 + Pinecone)	SimpleMem Architecture	Direction
Vector storage per matter (active)	Baseline reference	Reduced (estimated 40-65% reduction)	↓ Improvement

Metric	Baseline (mem0 + Pinecone)	SimpleMem Architecture	Direction
Node 8 P50 retrieval latency	Baseline reference	Reduced (estimated 25-40% reduction)	↓ Improvement
Node 8 P95 retrieval latency	Baseline reference	Reduced (estimated 35-55% reduction)	↓ Improvement
Duplicate unit suppression ratio	N/A (no deduplication)	0.35-0.55 (matter-dependent)	New capability
Memory consolidation ratio	N/A (no consolidation)	0.15-0.30 (matter-dependent)	New capability
Node 8 token consumption (mean)	Baseline reference	Reduced (estimated 20-35% reduction)	↓ Improvement
Infrastructure cost per matter	Baseline reference	Reduced (estimated 15-30% reduction)	↓ Improvement

Table 6. Engineering measurements from production deployment. Values are directional estimates from deployment monitoring; controlled experimental results require the prospective ablation described in Section 9.4.

We note explicitly that these measurements are not the output of a controlled experiment: the transition from mem0 + Pinecone to the SimpleMem architecture was not A/B tested in production, and confounding factors (evidence corpus changes, model updates, prompt revisions) cannot be ruled out without the controlled ablation. The estimates in Table 6 represent our best engineering assessment of the memory layer's contribution, qualified by the limitations stated in Section 13.

11. Qualitative Deployment Observations

The following observations were collected through informal feedback from the legal team members who use the system in production. They are not the output of a structured user study, do not include statistical significance claims, and should be interpreted as practitioner perceptions rather than experimentally validated findings. We present them here because they represent the primary evidence currently available regarding end-user experience, while explicitly acknowledging the methodological limitations.

11.1 Retrieval Relevance

Legal team members consistently reported that the context surfaced at Node 8--visible to them as the evidence basis attached to each draft--was more relevant to the current drafting task than under the prior system. Specific feedback focused on two improvements: (i) fewer irrelevant email boilerplate extractions appearing in the context, and (ii) better surface of escalation patterns--the chronological sequence of related events--that had previously been fragmented across multiple retrieved chunks without obvious connection.

11.2 Draft Quality

Reviewers reported that drafts required fewer manual corrections before reaching a state suitable for client delivery. The corrections that remained were described as primarily strategic or interpretive (judgment calls about legal framing or the relative weight of competing evidence) rather than factual (correcting misattributed dates, missing events, or incorrect party names). This characterization is consistent with the hypothesis that the memory system improvements primarily reduce factual errors rather than improving legal reasoning quality, which remains the domain of the generation model and prompt design.

11.3 Cross-Session Continuity

In matters spanning multiple sessions--returning to a draft after a HITL review, or re-opening a matter weeks after initial filing--reviewers reported improved continuity. Context established in an earlier session (including human corrections made at HITL #2) was recalled in subsequent sessions more reliably than under the prior system. This observation is attributable to the HITL importance boost and memory aging mechanisms, which ensure that human-validated facts remain highly retrievable over time.

11.4 Personalization

Reviewers noted that the system appeared to 'remember' preferences and patterns established earlier in a matter--for example, a naming convention for parties established in an early drafting session being applied consistently in later sections. This perception of personalization reflects the cross-session memory persistence and consolidation mechanisms rather than any model-level personalization, but the practical effect on reviewer experience is positive.

11.5 Caveats

These observations must be interpreted with three significant caveats. First, reviewer assessments are not blind: reviewers knew the system had changed and may have perceived improvements consistent with positive expectation. Second, the observation period was insufficient to separate the memory architecture's contribution from concurrent improvements in model capabilities, prompt engineering, and orchestration logic. Third, individual matters vary substantially in complexity and evidence quality; observations drawn from a small number of matters may not generalize. The longitudinal evaluation described in Section 14 is required to address all three caveats.

12. Discussion

12.1 Memory Lifecycle as a First-Class System Concern

The primary architectural lesson of this work is that memory lifecycle management--the policies governing what gets stored, how it ages, how it is consolidated, and when it is archived--is a first-class system design concern for enterprise agentic pipelines, not an afterthought addressed by a general-purpose memory package. General-purpose packages optimize for ease of use and broad applicability; enterprise legal workflows require domain-specific gating, structured consolidation, and lifecycle policies that must be explicitly designed rather than inherited from a generic implementation.

The SimpleMem framework provides a useful abstraction layer for these policies--its three-stage decomposition into compression, synthesis, and planning cleanly separates the concerns of what to store, how to organize what is stored, and how to retrieve it--but the concrete policies within each stage must be domain-specific. The legal content taxonomy, the entity salience weighting, the HITL importance boost, and the category-specific decay rates are all domain-specific adaptations without which the framework would perform poorly on legal evidence corpora.

12.2 The Orchestration-Memory Separation

A key design decision in this work was to treat the memory layer as a replaceable component within a fixed orchestration topology rather than redesigning the workflow to accommodate the memory system. This separation was motivated by two practical constraints: the existing orchestration architecture had been validated in production and changing it would introduce uncontrolled variables; and the legal team's HITL integration depended on specific node outputs and could not be disrupted. The separation proved advantageous for evaluation as well: it produces a clean comparison point (memory layer A versus memory layer B, everything else constant) that would not be available if memory and orchestration had been co-designed.

12.3 Implications for Enterprise RAG Design

Our experience suggests three design principles for enterprise RAG pipelines that handle long-horizon, multi-file evidence corpora. First, storage efficiency and retrieval precision are complements rather than trade-offs when the deduplication and consolidation stages are effective: reducing the index size improves both storage cost and retrieval signal-to-noise ratio. Second, HITL corrections must be first-class memory citizens: a system that treats human review as external to the memory pipeline cannot maintain coherence across sessions when human knowledge exceeds what the extraction models can recover. Third, tenant isolation must be enforced at the index layer rather than the application layer: application-layer filtering is vulnerable to implementation bugs and cannot provide the audit-grade guarantees that legal compliance requires.

12.4 Transferability

The memory lifecycle architecture described in this paper is specific to the employment grievance setting in several respects--the content taxonomy, hyperparameter values, and entity relationship model all reflect the structure of UK employment law evidence. However, the architectural pattern (domain-specific gating → structured consolidation → three-view indexed storage → intent-aware retrieval) is transferable to any document-intensive enterprise AI pipeline where the evidence corpus is heterogeneous, multi-file, and accumulated over time. Applications in financial due diligence, healthcare record summarization, and regulatory compliance monitoring share the structural properties that motivated this design.

13. Limitations

We identify four categories of limitation that qualify the claims and observations in this paper.

13.1 Absence of Controlled Experimental Evidence

The most significant limitation is that the engineering measurements in Section 10 and the qualitative observations in Section 11 are not the output of a controlled experiment. The deployment transition from mem0 + Pinecone to the SimpleMem architecture was not A/B tested: all matters were migrated simultaneously, and no holdback condition was maintained. Concurrent changes to model versions, prompt templates, and evidence corpus characteristics mean that the observed improvements cannot be attributed solely to the memory architecture with confidence. The prospective controlled ablation described in Section 9.4 is required to address this limitation; we present the deployment observations as motivation and preliminary evidence for that study.

13.2 Single Deployment Context

All observations are drawn from a single deployment serving UK employment grievance matters for a specific client base. The content taxonomy, hyperparameter values, and entity model reflect this specific context. Transferability to other legal jurisdictions, matter types (commercial litigation, regulatory investigations, transactional due diligence), or languages (the system currently processes English-language evidence exclusively) cannot be assumed without validation.

13.3 Evaluation Corpus Scale

The evaluation corpus described in Section 9.2 represents a limited sample of completed matters. Draft quality metrics scored by legal reviewers are subject to inter-rater variability that has not been formally measured. The DeepEval automated metrics are grounded in LLM-as-judge assessments that inherit the biases and calibration properties of the underlying judge model. Neither the reviewer-scored metrics nor the LLM-scored metrics have been validated against external ground truth for this specific domain.

13.4 TurboVec Compression Accuracy

The retrieval accuracy impact of TurboVec compression was measured through an internal engineering validation rather than a rigorous benchmark. The 'less than 2% degradation' figure reported in Section 8.3 is an estimate based on a held-out query sample and may not generalize to retrieval queries with unusual semantic properties or to matter types underrepresented in the validation sample.

14. Future Work

14.1 Longitudinal Deployment Study

The primary future work is a longitudinal deployment study spanning six to twelve months, designed to evaluate outcomes that cannot be measured in short deployments. Specifically, we plan to evaluate organizational knowledge retention: whether memory units synthesized from early matters inform and improve the handling of later, related matters; whether the system's entity model improves in accuracy as it accumulates more observations of the same parties; and whether personalization effects--the alignment between a specific legal team member's preferences and the system's retrieval and drafting behaviour--strengthen over time. The study will also assess sustained user productivity, measuring the legal team time required per completed draft at 3-month, 6-month, and 12-month intervals, controlling for matter complexity.

14.2 Controlled Memory Layer Ablation

The prospective ablation described in Section 9.4 will be executed as part of the longitudinal study. This ablation will run matched matter pairs--matters of equivalent complexity and evidence volume--through the SimpleMem and mem0 + Pinecone conditions simultaneously (isolated by tenant ID to prevent cross-contamination), producing the controlled comparison that current deployment measurements cannot provide.

14.3 Multi-Matter Memory Generalization

The current architecture scopes all memory to a single matter (enforced by the tenant isolation mechanism). An extension under investigation would allow controlled cross-matter memory transfer: facts established in one matter (the settlement terms in Matter A between the same parties) that are relevant to a subsequent matter (a related discrimination claim, Matter B) could be made available via a curated memory transfer mechanism subject to legal team approval. This extension raises significant ethical and privacy design challenges--cross-matter knowledge transfer between clients is permissible only with appropriate consent and conflict-of-interest safeguards--that we intend to address before implementation.

14.4 Memory Evolution and Concept Drift

Employment law evolves: statutory provisions change, tribunal precedents shift, and procedural requirements are updated by regulatory guidance. The memory architecture currently has no mechanism for detecting or responding to concept drift--the gradual obsolescence of memory units that accurately reflected the legal landscape at extraction time but are no longer current. A planned extension will use periodic automated legal database queries to flag memory units referencing statutory provisions or case law that have been superseded, routing them through the legal team for re-validation before they remain active in the retrieval index.

14.5 Adaptive Hyperparameter Tuning

The hyperparameters governing memory lifecycle behaviour (Table 3) are currently set manually per matter type and updated through operational experience. An extension would use reinforcement learning from human feedback (RLHF) signals collected at HITL #2--the pattern of human corrections to automated drafts--to update hyperparameters adaptively. Corrections concentrated in specific evidence categories would signal that the gating function's threshold for those categories is too aggressive; corrections concentrated in specific matter complexity ranges would signal that the retrieval budget allocation needs adjustment.

15. Conclusion

We have described the design, implementation, and production deployment of a memory lifecycle architecture for an enterprise legal document intelligence system. The architecture adapts the SimpleMem framework for multi-file evidence corpora, extending it with a legal-taxonomy gating function, three-view indexing with tenant isolation enforced at the storage layer, importance-weighted memory aging, TurboVec vector compression, and HITL-aware memory update routing.

Engineering measurements from production deployment indicate meaningful improvements in vector storage efficiency, retrieval latency, and duplicate suppression relative to the prior mem0 + Pinecone baseline. Qualitative observations from legal team members suggest that these engineering improvements translate into better retrieval relevance, reduced manual correction of generated drafts, and improved cross-session continuity. These observations are preliminary and uncontrolled; we have described a prospective evaluation programme to validate them rigorously.

The broader contribution of this work is a demonstration that memory lifecycle management is a first-class design concern for enterprise agentic pipelines. General-purpose memory packages are not adequate substitutes for domain-specific lifecycle policies when the evidence corpus is large, heterogeneous, and accumulated over time; when human review is a mandatory component of the pipeline rather than an optional oversight mechanism; and when audit reproducibility and multi-tenant isolation are non-negotiable production requirements. The architectural pattern described here--domain-specific gating, structured consolidation, three-view indexed storage, intent-aware retrieval--is a candidate template for other enterprise document intelligence domains sharing these structural properties.

References

1. Liu, Y., et al. (2026). SimpleMem: A Three-Stage Memory Lifecycle for Long-Horizon Conversational Agents.
2. Anthropic. (2024). Building effective agents. Anthropic Research Blog. https://www.anthropic.com/research/building-effective-agents
3. Anthropic. (2026). Developing an oversight protocol for autonomous AI agents. Anthropic Research Blog.
4. Asai, A., et al. (2023). Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. arXiv:2310.11511.
5. Confident AI. (2024). DeepEval: An Open-Source Evaluation Framework for LLM Applications. https://docs.confident-ai.com/
6. Edge, D., et al. (2024). From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130.
7. Google. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv:2403.05530.
8. Google. (2025). Agent-to-Agent (A2A) Protocol Specification. Google Developers.
9. Gutiérrez, B. J., et al. (2024). HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. arXiv:2405.14831.
10. Jégou, H., Douze, M., and Schmid, C. (2011). Product Quantization for Nearest Neighbor Search. IEEE TPAMI, 33(1), 117-128.
11. Jiang, Z., et al. (2023b). Active Retrieval Augmented Generation. arXiv:2305.06983.
12. Johnson, J., Douze, M., and Jégou, H. (2019). Billion-scale similarity search with GPUs. IEEE Trans. Big Data, 7(3), 535-547.
13. Kang, Z., et al. (2025). MemoryOS: An OS-Inspired Memory Hierarchy for Long-Horizon LLM Agents. arXiv:2506.06965.
14. LangChain. (2024). LangGraph: Building Stateful Multi-Actor Applications with LLMs. https://langchain-ai.github.io/langgraph/
15. Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 33, 9459-9474.
16. Liu, N. F., et al. (2024). Lost in the Middle: How Language Models Use Long Contexts. Transactions of the Association for Computational Linguistics, 12, 157-173.
17. Malkov, Y. A., and Yashunin, D. A. (2020). Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. IEEE TPAMI, 42(4), 824-836.
18. Mitta, D., et al. (2024). Mem0: Building Production-Ready AI Agents with Long-Term Memory. arXiv:2504.19413.
19. Packer, C., et al. (2023). MemGPT: Towards LLMs as Operating Systems. arXiv:2310.08560.