# Mnemosyne: An Unsupervised, Human-Inspired Long-Term Memory Architecture for Edge-Based LLMs

Aneesh Jonelagadda Christina Hahn Haoze Zheng Salvatore Penachio

Kaliber AI, Research Division
{a.jonelagadda, c.hahn, h.zheng, s.penachio}@kaliber.ai

## Abstract

Long-term memory is essential for natural, realistic dialogue. However, current large language model (LLM) memory systems rely on either brute-force context expansion or static retrieval pipelines that fail on edge-constrained devices. We introduce Mnemosyne, an unsupervised, human-inspired long-term memory architecture designed for edge-based LLMs. Our approach uses graph-structured storage, modular substance and redundancy filters, memory committing and pruning mechanisms, and probabilistic recall with temporal decay and refresh processes modeled after human memory. Mnemosyne also introduces a concentrated "core summary" efficiently derived from a fixed-length subset of the memory graph to capture the user's personality and other domain-specific long-term details such as, using healthcare application as an example, post-recovery ambitions and attitude towards care. Unlike existing retrieval-augmented methods, Mnemosyne is designed for use in longitudinal healthcare assistants, where repetitive and semantically similar but temporally distinct conversations are limited by naive retrieval. In experiments with longitudinal healthcare dialogues, Mnemosyne demonstrates the highest win rate of 65.8% in blind human evaluations of realism and long-term memory capability compared to a baseline RAG win rate of 31.1%. Mnemosyne also achieves current highest LoCoMo benchmark scores in temporal reasoning and single-hop retrieval compared to other same-backboned techniques. Further, the average overall score of 54.6% was second highest across all methods, beating commonly used Mem0 and OpenAI baselines among others. This demonstrates that improved factual recall, enhanced temporal reasoning, and much more natural user-facing responses can be feasible with an edge-compatible and easily transferable unsupervised memory architecture.

## 1 Introduction

Memories are referred to as the "database of the self" (Mahr and Csibra, 2018; Conway, 2005). For a natural language exchange between two agents to be human-like, both parties must have a reliable sense of long-term memory. This is even more true for in-person edge AI agents, which must produce responses that are as natural, helpful, and accurate as possible during physical interactions with the user.

However, LLMs are constrained by fixed context limits which cause earlier content to be excluded from its context once conversation length exceeds the limit. This causes predictable memory loss during an exchange between two agents, which is a functionality gap. Related works (Yan et al., 2025; Chhikara et al., 2025; Liu et al., 2025b) attempt to address this issue but have key weaknesses for our use case, such as not encoding temporal information, requiring a significant data and/or development overhead for applications in new domains, or not being viable for edge-constrained devices.

Our goal is to have a lightweight model that can refer to past events, either responding to a user with the correct context or bringing up the appropriate context when relevant. To this end, we propose *Mnemosyne*, an unsupervised, graph-augmented storage and retrieval architecture for edge-based LLMs, integrating modular intake filters for substance, redundancy, and pruning. The system supports probabilistic recall, temporal decay modeled after human forgetting (Murre and Dros, 2015), and boosting mechanisms for redundant memories, enabling it to behave more like a human memory system rather than a static retrieval pipeline. In addition, the architecture encapsulates core personality extraction as a deep memory module, and introduces novel query-relevance steering mechanisms such as naturalized time-delta placement keywords. Ultimately, Mnemosyne is a context engineering architecture, controlling what enters the downstream LLM's context window. To our knowledge, this is the first memory system that combines modular intake filtering, dynamic graph-structured storage, and human-inspired recall dynamics for long-term LLM memory on constrained devices.

Mnemosyne is uniquely positioned to augment conversational agents in settings where on-premise edge agents interact with users regularly and collect interaction data. In the standard LoCoMo benchmark (Maharana et al., 2024), Mnemosyne achieves state-of-the art performance in temporal reasoning by improving on alternative methods with a J-score of 60.42%. Additionally, it achieves a much higher win rate in blind human evaluations of 65.8% vs a naive RAG baseline's 31.1%, demonstrating both effective factual recall and much more natural user-facing responses.

## 2 Related Work

LLMs based on the Transformer architecture (Vaswani et al., 2017) can now scale context windows into the hundreds of thousands of tokens (Xiao et al., 2024). However, quadratic attention costs and KV-cache memory scaling make this approach unfeasible on edge devices, where memory and compute are limited. Long-context models, with context windows of 128k to over 1M tokens, can afford to retrieve and process orders of magnitude more memories within a single prompt than an SLM with a context window of $\sim$4k tokens. (Subramanian et al., 2025). Thus, brute-force long-context methods used with LCLMs remain impractical for on-device deployment. Episodic memory buffers maintain a sliding window of the most recent conversational turns, as seen in commercial chatbots like Pi and Replika, but these methods discard older information regardless of salience.

Our design allows the system to maintain a long-term history with a size constrained by the total available system memory, which scales more gracefully than the quadratically-scaling limitations of the LLM's attention mechanism. By selectively retrieving and injecting only the most salient nodes into LLM's context mechanism, we bypass the bottleneck of a limited context window on resource-constrained devices.

RAG systems (Lewis et al., 2021) perform efficiently context-wise and are effective given a semantic variety of potential topics. Our use case, however, is a single-user domain-specific interaction agent that generates conversations with significantly less semantic diversity than those seen in common general-purpose vector stores. These interactions' importance wax and wane over time, necessitating a more dynamic approach. The method proposed in this paper is fundamentally context engineering-based, but is both more generalizable to memory as a whole and more specialized to the edge and constrained semantic domains.

### 2.1 LLM Memory Systems

More sophisticated techniques use structured external memory, such as MemGPT (Packer et al., 2024) and LangMem, which query a vector store. While this is a step in the right direction, traditional vector similarity search is inadequate for our domain due to the high semantic similarity of memories; very subtle but crucial differences between recurring events (e.g., daily wound care reports) are often lost in naive vector comparisons, leading to noisy retrieval.

Table 1: Comparison of Mnemosyne with past related methods. ∼ indicates partial support.

| System | Structured Memory | Semantic Recall | Temporal Dynamics | Edge-Feasible | Redundancy Handling | Forgetting Model |
|---|---|---|---|---|---|---|
| Long-Context | X | ✓ | X | X | X | X |
| Naive RAG | ✓ | ✓ | X | ✓ | X | X |
| MemGPT/LangMem | ✓ | ✓ | X | X | ∼ | X |
| Memory-R1 | ✓ | ✓ | X | ∼ | ✓ | ✓ |
| Mem0 | ✓ | ✓ | ∼ | ∼ | ✓ | ∼ |
| **Mnemosyne** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

As shown in Table 1, prior systems address fragments of the long-term memory problem. Refinements to long-term memory, such as those proposed in Memory-R1 (Yan et al., 2025) and Mem0 (Chhikara et al., 2025), inspire aspects of Mnemosyne's design but still have slight functionality gaps. For example, Mem0 uses a rules-based cache management strategy as a forgetting mechanism, updating previous information if it is contradicted by new information. The system proposed in this paper, however, incorporates temporal decay and refresh/rewind mechanisms modeled after human memory studies (Mahr and Csibra, 2018; Conway, 2005). Thus, our forgetting process is probabilistic, continuous, and tunable. Memory-R1 introduces a comprehensive forgetting mechanism similar to the one we propose, but lacks any kind of temporal encoding in its stored memories. Additionally, like MemGPT/LangMem, Memory-R1 was not built to be edge-compliant; despite potential for refactoring, its edge efficacy has not been tested to our knowledge. Memory-R1's further reliance on reinforcement learning for add/delete policy optimization makes it less interpretable and requires specialized training runs with carefully crafted reward signals. This makes it expensive to adapt and deploy to new domains. In contrast, Mnemosyne is unsupervised, which makes it lightweight and domain-transferable with minimal overhead.

## 3 Methods

Our memory system is designed for high performance, storing all necessary graph information on an in-memory database for its speed advantages (see Section 4). At a high level, our system is comprised of (1) commitment, (2) recall, (3) asynchronous updates to the system's core supersummary, and (4) a pruning module.

The memory system inputs are interaction summaries. These are generally atomic to the interaction, but we also recommend ingestion of a shorter-term iteratively updated running summary. The summaries are then asynchronously passed into the commitment algorithm, which either discards or commits the input into the memory graph as a node by using the core summary as a salience baseline. Upon a user query, the recall algorithm selects a start node and probabilistically traverses the graph, selecting a set of the most relevant nodes and converting it into formatted LLM generation context. Finally, the core summary is periodically updated to ensure that newer conversations are accounted for. This process is also asynchronous.

### 3.1 Commitment

Our memory system's commitment algorithm contains two filters: (1) an LLM-based substance filter that discards unimportant information, and (2) a traditional ML-based redundancy filter that marks and prunes redundant conversations. From there, the algorithm is dedicated to storing all information that will be necessary during recall to ensure high performance. For the full algorithm, see Algorithm B3.

Because not every memory has substance, our commitment algorithm has an LLM-based filtering mechanism to discard irrelevant incoming conversation summaries. Using the conversation summary, any other ancillary summaries, and the core summary as inputs, an LLM judges if the contents of the conversation are substantial or not. We can thus formulate the substance filter as a binary classifier. We define "substantial" conversations in the instructions to the LLM as conversations that include information from one of the following categories: (1) the patient's medical condition, treatment, or care, (2) the patient's clinical

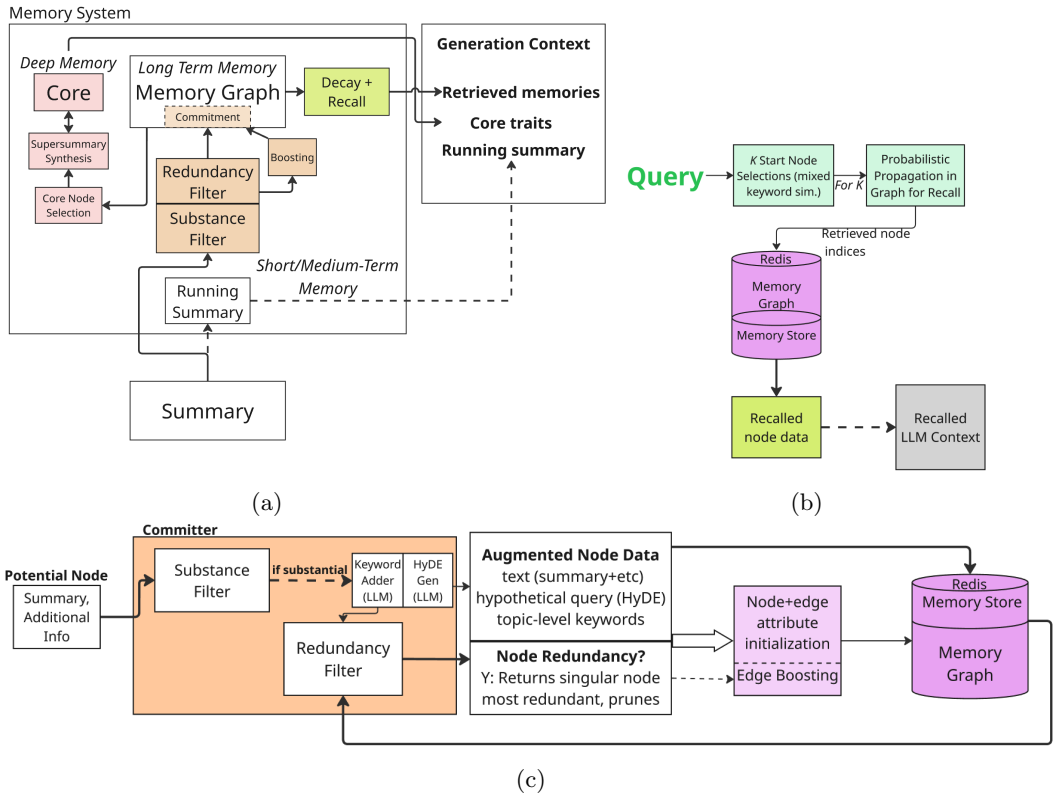


Figure 1: High-level layout of Mnemosyne: (a) the entire memory unit, (b) the recall process, and (c) the commitment process.

status, behavior, or needs, and (3) details about the patient's personality. Note that routine or mundane conversations are considered non-substantial. If interactions are deemed substantial, they get forwarded to the rest of the committer algorithm.

### 3.1.1 Redundancy Filter and Rewind Calculation

Even if conversations are deemed substantial, many similar conversations in the same memory graph may bloat the system and decrease performance. Thus, the commitment process has a redundancy filter to detect redundant incoming nodes and pair them with an existing node. Taking inspiration from the primacy-recency effect (Mayo and Crockett, 1964; Morrison et al., 2014), we discard the more recent of the two existing nodes if an incoming conversation is deemed redundant to a node already paired.

We calculate redundancy via a combination of: (1) the mutual information (Duncan, 1970) between the two nodes' conversation summary embeddings and (2) the Jaccard similarity between the keywords of those conversation summaries. Implementation-wise, entropy and joint entropy for textual documents can be calculated programmatically by binning fixed-length text embeddings for the mutual information calculation.

Let node $n$ be the incoming node, $m$ be some existing node in the memory graph $G$. Because we're operating on the embeddings of conversation summaries $n_s, m_s \in \mathcal{S}$, let the embedding function $f$ be defined as $f(n) = \mathbf{e_n}$. Given some constant weight $\alpha_{\text{NMI}} \in [0, 1]$ and the redundancy score calculation $\text{RS}(n, m)$, we calculate $m^+$, the existing node with the maximum redundancy score:

$$\text{RS}(n, m) = \alpha_{\text{NMI}} \cdot \text{MI}(\mathbf{e_{n_s}}, \mathbf{e_{m_s}}) + (1 - \alpha_{\text{NMI}}) \cdot \text{JS}(\mathbf{e_{n_k}}, \mathbf{e_{m_k}})$$
$$m^+ = \arg\max_{m \in G} \text{RS}(n, m)$$

We finally define some minimum redundancy score threshold $\text{RS}_{\text{min}}$ to set the redundant node $m^+$. If $m^+$ is already paired, $m^+$ is set to the older node in the pairing and the newer one is discarded.

Repeated memories are "refreshed" and retained for longer periods of time (Ebbinghaus, 1913). Thus, when a memory has been determined to be redundant by the redundancy filter, we introduce a "boosting" or rewind mechanism that reverses the decrease in selection probability caused by temporal decay during recall.

This rewind value is calculated during commitment and stored for use during recall. We use a sigmoidal function defined in Equation A1 which plateaus at higher timescales to bound the boosting mechanism; we must ensure some maximal boost which is within the domain of the decay function. Further, the nonlinear behavior near zero de-incentivizes frequent updates to the same memory, which is important for encouraging diversity among the selected memories between user queries. Since small $t$ in this case corresponds to a short time since the most recent boost, this behavior resembles biological brains' desensitization to frequent stimulus (Thompson and Habituation, 1966; Groves and Thompson, 1970).

#### 3.1.2 Graph Construction and Decay Calculation

We construct node and edge instances only with lightweight information necessary for critical graph operations (see Table C1). Hypothetical queries, taking inspiration from HyDE (Gao et al., 2023), are also generated and embedded during commitment to close the structural difference between the user's query and the graph content summaries during recall. Exact implementation details can be found in Section 4 and Appendix.

Once the input has been judged substantial by the substance filter, we create the node instance $n$ and create edges between $n$ and the existing nodes in the memory graph $G$ based on similarity. Our similarity score function is a weighted sum of (1) the cosine similarity between the two nodes' conversation summary embeddings, and (2) the Jaccard similarity between the keywords of those conversation summaries.

Given a keyword mixing weight $\alpha_{\text{key}} \in [0, 1]$ similar to $\alpha_{\text{NMI}}$ in Section 3.1.1, our similarity score function is

$$\text{sim}(n, m) = \alpha_{\text{key}} \cdot \text{CS}(\mathbf{e_{n_s}}, \mathbf{e_{m_s}}) + (1 - \alpha_{\text{key}}) \cdot \text{JS}(\mathbf{e_{n_s}}, \mathbf{e_{m_s}})$$

After introducing some minimum edge creation threshold $\text{edge}_{\text{min}}$, we can then define $e_{nm}$ as the edge between the new node $n$ and some existing node $m \in G$ with weight given by $\text{sim}(n, m)$.

Another intended functionality of our memory system is to emulate human memory by "forgetting" older conversations at a natural rate. The majority of the legwork for this forgetting mechanism is done by "decaying" the selection probability of older memories during recall. We account for refreshed memories by fetching the rewind value $\Delta_m(t)$ stored during commitment. Given some edge $e$ and current time $t = t_{\text{curr}}$, $e$'s effective age $e_{\text{eff}}$ can be defined as $e_{\text{eff}} = t - e_{\text{t_init}} - \Delta_e(t)$. From here, we construct our decay function as a modified reverse sigmoid defined in Equation A2, since we intend our function to plateau above 0 at higher timescales and ensure that past conversations are never fully forgotten. To ensure consistency in retrieval, we additionally enforce the edge weight's value to be 1 at $t = 0$ regardless of parameters by imposing a linear correction to the modified reverse sigmoid function's early values.

### 3.2 Recall

The recall algorithm is responsible for retrieving the set of conversation summaries that will be returned as LLM context, accomplished via probabilistic traversal of the memory graph with modified edge weights. Our algorithm is inspired by neuron activation propagation, where neurons fire action potentials and influence nearby neurons (Feinerman et al., 2005). The signal decays as it travels through the nervous system (Feinerman et al., 2005; Kharas et al., 2022), which we emulate through our algorithm's passive decay in selection probability past the start node.

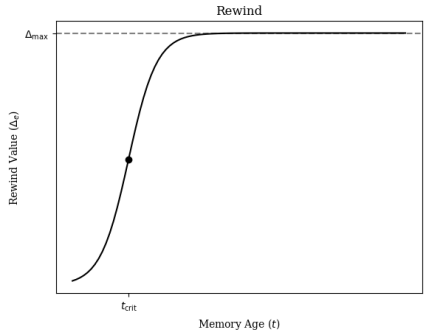


Figure 2: Graph of the rewind function.

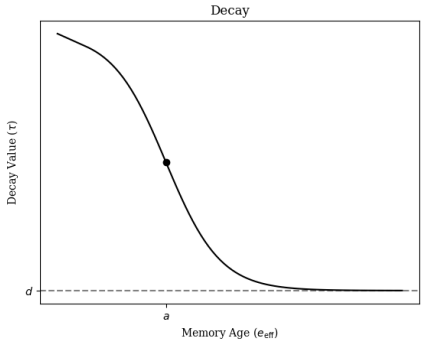


Figure 3: Graph of the decay function.

We begin by selecting the node $m^*$ most similar to the user query. The hypothetical query generated during commitment is used here for the comparison to eliminate the difference in answer space between the user query and the graph content. We use a weighted sum of two similarity scores $S_{\text{meta}}$ and $S_{\text{query}}$ to select the start node. $S_{\text{meta}}$ is calculated with the embedded user query $u_q$ and a concatenation of (1) the naturalized time delta (e.g., `"last week"`) $m_\Delta$ between the current time $t$ and the initialization time $t_{\text{init}}$ of node $m$, (2) a natural language description of domain-specific state(e.g.,`"post-surgery recovery phase"`) and (3) relevant keywords $m_k$. $S_{\text{query}}$ is calculated with the embeddings of user query $u_q$, the hypothetical query $m_q$, and the summary text $m_s$. $S_{\text{meta}}$ and $S_{\text{query}}$ can be defined as

$$S_{\text{meta}} = \text{CS}\left(\mathbf{e_{u_q}}, \mathbf{e_{m_\Delta + m_k}}\right) \quad S_{\text{query}} = \text{CS}\left(\mathbf{e_{u_q}}, \mathbf{e_{m_q + m_s}}\right)$$
$$S(m) = \alpha_{\text{meta}} \cdot S_{\text{meta}} + (1 - \alpha_{\text{meta}}) \cdot S_{\text{query}}$$
$$m^* = \arg\max_{m \in G} S(n, m)$$

We would like to highlight the novelty of incorporating temporal language into the similarity computation rather than treating time as a raw numeric feature. We use the naturalized time delta because as time goes on, the user will refer to the memory contained in $m$ with relative temporal qualifiers instead of the present tense. Thus, it's necessary to account for during the recall process in order to accurately retrieve temporally situated memories.

After selecting the start node $m^*$, we proceed with the recall algorithm by traversing the graph DFS-style via edge weights (see Section 3.1.2). The selection probability of the next node $m$ is modified by temporal decay and rewind mechanisms, and if selected, $m$ is added to a set of final nodes $R$ that will be returned, with their contents formatted as LLM context. Let $n$ be the current selected node during traversal, $m$ the next neighbor of $n$, and $e_{nm}$ the weight of the edge connecting them. Given an arbitrary exploration hyperparameter $\mu$, the selection probability of node $m$ can be defined as

$$P(m) = \mu \cdot e_{nm} \cdot \tau(e_{\text{eff}})$$

We begin by calculating $P(m)$ for each neighbor $m \in N(m^*)$ and generating a random number $r \in [0, 1]$. If $r < P(m)$, $m$ is added to $R$ and $N(m)$ are recursively explored. The overall traversal algorithm is shown in Algorithm B5. We also separately introduce a pruning module which acts as the systems "forgetter"; nodes are scored by a version of the selection probability without exploration term, and nodes below a pruning threshold are removed (see Equation A3). The pruning mechanism is only intended for use when the graph size approaches memory limits.

### 3.3 Core Summary

People possess personality quirks and remember emotionally resonant or distinctive events (Schmidt, 2006). These personalized details have been shown to help LLMs generate more natural and tailored responses to user queries (Qiu et al., 2025; Liu et al., 2025a). To

represent these facts, we store a supersummary of past conversations that represent what the AI assistant has learned about the patient. We call this supersummary the "core summary" and it is always injected into final LLM system prompt along with the recalled context. Most importantly, this core summary is generated on a fixed-length subset of the graph's memories. This allows for efficient and feasible longitudinal scaling; otherwise, if done naively, the supersummary-generating LLM would have to potentially ingest hundreds to thousands of memories at once, disallowing low-cost edge use of this system.

The stored core summary is used in the substance filter as a baseline of comparison for "what is substantial," and captures key details about the user. As stated in Section 3.1, important information includes (1) the patient's medical condition, treatment, or care, (2) the patient's clinical status, behavior, or needs, and (3) details about the patient's personality. Our core summary algorithm begins by selecting a central subset of nodes $C$ from the memory graph $G$ subject to $|C| \leq$ a maximum node number $M$. In order to ensure appropriate span of topics represented in $C$, we first partition $G$ using k-means clustering (Lloyd, 1982) on the node embeddings $\mathbf{e_n} \in \mathbb{R}^{d_{\text{embed}}}$, yielding $k$ cluster assignments.

All nodes across all clusters are then scored using a linear combination of (1) connectivity score $s_{\text{conn}}$, (2) boost score $s_{\text{boost}}$, (3) temporal recency score $s_{\text{rece}}$, and (4) information density score $s_{\text{ent}}$ to yield a hybrid score $s_{\text{hybrid}}(n;\theta)$ of node $n$. The coefficients $\theta$ in the linear combination are subject to the hyperparameter constraint $\sum_{i=1}^{4} \theta_i = 1$. Assuming each node $n$ has attributes embeddings vector $\mathbf{e_n}$, neighbors $N(n)$, initialization time $t_{\text{init}}$, and set of edges with boosts $\Delta_{n,m}$ $\forall m \in N(n)$,

$$s_{\text{conn}}(n) = \frac{|N(n)|}{|G| - 1} \qquad s_{\text{boost}}(n) = \frac{|\{m \in N(n) : \Delta_{n,m} > 0\}|}{\max_{n' \in G} |\{m \in N(n') : \Delta_{n',m} > 0\}|}$$

$$s_{\text{rece}}(n) = \exp\left(-\frac{t_{\text{curr}} - t_{\text{init}}}{\lambda}\right) \qquad s_{\text{ent}}(n) = \frac{1}{1 + H(\mathbf{e_n})}$$

First we define the connectivity score $s_{\text{conn}}$ as the connectivity of the node $n$ normalized by the maximum possible connectivity of the graph $G$. This selects for highly central memories which semantically relate to other memories. Next, in order to reward memory boosting due to highly related interactions being ingested (and deemed redundant) by our memory system, we count the number of boosted edges for node $n$ and normalize by the maximum number of boosted edges across all nodes in the current graph to form boost score $s_{\text{boost}}$. The temporal recency score $s_{\text{rece}}$ is introduced prioritize recent memories over old ones. We use a separate timescale than our decay function from A2 to ensure decoupling between the core memory system, which can have various scales of synchronicity and can be used independently for other use-cases such as personality extraction, and the regular long-term-memory graph-based recall with more frequent update synchronicity. We use a memory half-life $\lambda = 28$ days. Finally, certain memories can be highly connected due to semantic generality, recent, and frequently related to incoming interaction data, but also contain little intrinsic information. A white noise process $W(n)$ thus propagates through our core system favorably. In order to counteract this, we introduce an information-density-based score $s_{\text{ent}}$ using the binned entropy $H(\mathbf{e_n})$.

Our design must allow for sufficient diversity across the core nodes, so we use the cluster assignments to select and aggregate the top scoring node per cluster yielding an initial selected set of core memories $C'$. Solely using one node per cluster in the core node subset can fully obfuscate the underlying distribution of memories, so we augment the subset with the top $k_2$ scoring nodes $C_{\text{crit}} \in G \setminus C'$ from the set of all nodes that have not been selected already. This yields the final core summary subset of nodes $C = C' \cup C_{\text{crit}}$ with $|C| = k + k_2$. Finally, if $|C| >$ the maximum node number $M$, we truncate the array simply by keeping the top $M$ most recent nodes.

For the updating process, the core summary $c$ is initialized as `"N/A"` when there are no available past conversations with the user. Periodically, the central subset $C$ of the graph is selected asynchronously, and raw conversation summaries corresponding to the nodes in $C$ will be passed into an LLM with the current core summary. There are two different instruction prompts: (1) initializes the core summary and only accounts for $s_c$, and (2)

updates the core summary and accounts for both $c$ and $s_c$. Both prompts instruct the LLM to capture overall characteristics.

# 4 Experimental Setup

All experiments were conducted on machine equipped with a 12th Gen Intel Core i7-12700E processor (12 cores, 20 threads, up to 4.8GHz boost frequency), NVIDIA RTX A4000 GPU with 16GB VRAM, and 64GB DDR5 system memory. Our hyperparameter configuration is provided in Table C3. All of our human evaluation, commitment and core operations involving LLMs were done with mistral-7b-instruct-v0.2.Q5_K_M (Jiang et al., 2023), our vector embeddings were generated by PubMedBERT (Gu et al., 2021; NeuML, 2023), and our external node attributes (see Section C2) and graph structure were stored on a Redis database (Sanfilippo and Contributors, 2025). Llama3.1-8B-Instruct (Grattafiori et al., 2024) was used as the generation LLM for LoCoMo benchmarking.

## 4.1 Memory System Benchmarking using LoCoMo

We use the LoCoMo benchmark score (Maharana et al., 2024) to evaluate Mnemosyne's ability to recall both correct and contextually relevant information compared to other leading techniques listed in Section 2.1. The LoCoMo benchmark provides a set of 10 scenarios, each containing a series of conversations and QA questions for benchmarking. The number of conversations in each series can range from 19 to 32 and each conversation has several turns. The query provided by the QA data (excluding the adversarial category) was thus used to perform recall (Section 3.2), feeding an LLM with context to generate an answer. We report the LLM-as-a-Judge (J) score using GPT-4o-mini (Hurst et al., 2024) as the judge with a standard LLM-as-a-Judge prompt (Packer et al., 2024) to compare with gold standard. This better captures semantic fidelity compared to the F1-score and BLEU-score, as noted in (Chhikara et al., 2025). Baseline metrics are cited directly from Yan et al. (2025)'s LLama3.1-8B-Instruct backbone results. The authors do not specify the judge model, though given similar baselines such as Mem0 it is likely GPT-4o-mini.

## 4.2 Human Evaluation

Retrieval metrics give us a quantifiable measure of accuracy, but they do not necessarily capture the "naturalness" of LLM responses. To ensure a holistic evaluation, we also conducted a blind human study comparing three systems: (1) Mnemosyne with core summary, (2) Mnemosyne without core summary, and (3) Naive RAG baseline. 8 human annotators, including 1 non-response, were presented with 40 queries (e.g., "What does the patient consider their most important personal value?") and three LLM responses to those queries corresponding to the three systems. The system order was randomly scrambled for each query. The annotators made pairwise judgments on their preferred response, with ties permitted when both responses were equally strong or weak. Using these pairwise judgments, we calculated the win rate (see Equation A5) for each of the systems with a Wilson 95% CI.

# 5 Results

## 5.1 Retrieval Benchmarking

Our benchmarking results are shown in Table C4. Mnemosyne yielded the highest J-score among all baseline methods for the LoCoMo temporal reasoning and single-hop scores, with a second-highest overall score of 54.55%. While OpenAI memory performed better for the single-hop category, the backbone LLM used is different and not edge-viable, so we discount this from the direct comparison. This indicates that Mnemosyne excels in temporal reasoning, which makes sense given its elaborate treatment of temporal dynamics, and its overall performance still beats most other baseline memory methods.

Table 2: LoCoMo Benchmark (J-score) of Mnemosyne compared to past related methods. Systems marked with * use Llama3.1-8B-Instruct as its generation LLM backbone, while † marks results reported directly in Yan et al. (2025).

| System | Single-Hop (%) | Multi-Hop (%) | Open-Domain (%) | Temporal Reasoning (%) | Overall (%) |
|---|---|---|---|---|---|
| Mem0*† | 43.93 | 37.35 | 52.27 | 31.4 | 45.68 |
| Zep*† | 52.38 | 33.33 | 45.36 | 27.58 | 42.80 |
| MemGPT/LangMem*† | 47.26 | 39.81 | 48.38 | 30.94 | 44.18 |
| OpenAI | **63.79** | 42.92 | 62.29 | 21.71 | 52.90 |
| Memory-R1*† | 59.83 | **53.01** | **68.78** | 51.55 | **62.74** |
| Mnemosyne* | **62.78** | 49.53 | 60.42 | **53.03** | 54.55 |

### 5.2 Human Evaluation

Mnemosyne with the core summary outperforms both the ablated version of Mnemosyne without the core summary and the naive RAG baseline with a win rate of 65.8%. Judges consistently preferred responses containing long-horizon and recurring details about the patient, which indicates that there is a notable advantage in maintaining a long-term persona-level supersummary in the memory system. Additionally, Mnemosyne without the core summary also outperformed the baseline. This suggests that the commitment gates, graph-structured recall, and human-inspired temporal mechanisms can improve response quality even without a deep summary module. However, the relative gap between the two Mnemosyne variants shows that core summarization is still necessary because individual memories alone seem to be less effective at reconstructing higher-order patterns.

An extension of this result is that perfect single- and multi-hop retrieval for long time-scales is not necessary to achieve perceived naturalness in an AI agent; a typical human will not arbitrarily remember what happened at a precise time several months ago. These findings validate our hypothesis that memory systems should be able to provide both specific and general information about the user for more natural and personalized responses from LLMs.

Table 3: Human evaluation results. The strongest result is bolded.

| System | Win Rate (%) | 95% CI (Wilson) |
|---|---|---|
| **Mnemosyne w/ Core Summary** | **65.80** | [61.78, 69.61] |
| Mnemosyne w/o Core Summary | 53.13 | [48.98, 57.22] |
| Naive RAG | 31.07 | [27.38, 35.02] |

## 6 Conclusion

In this paper we presented Mnemosyne, an unsupervised, graph-structured memory system with modular intake filters, probabilistic recall with temporal decay and rewind, and a core summary module that captures persistent user traits. Unlike other techniques, Mnemosyne is specifically designed to run on edge devices, letting agents keep long-horizon memory without relying on brute-force context expansion or heavy cloud infrastructure. We evaluate on the LoCoMo benchmark, for which Mnemosyne beats all methods for temporal reasoning with a score of 60.4% and overall performs better than almost all leading techniques by scoring an overall J-score of of 54.6%, coming second only to Memory-R1 despite being a solely edge-based system. Further, while hop retrieval and open-domain are the metrics of focus for several other leading long-term retrieval methods, the focus of our system is on response and interaction naturalness within an atomic domain. This is reflected in our human evaluation, where the full system achieves a 65.8% win rate over RAG's 31.1%, consistently producing responses that are more contextually coherent and human like. The ablated Mnemosyne system which omits the core summary system also outperforms RAG in this regard with a win rate of 51.5%, demonstrating significant efficacy of the traversal and filters specifically. All results were further determined to be statistically significant ($\alpha = 0.05$). Future work can revolve around use-based dynamic hyper-parameters. We hope that these results show that highly dynamic heuristics-based memory systems are viable for commercial use, especially given our unsupervised approach which dramatically reduces integration overhead.

**Ethics Statement**

All human evaluators gave informed consent to participate in the study, which was conducted under our institutional ethics guidelines. No personally identifiable or health-related information was collected, and all longitudinal healthcare interaction data used to provide scenarios for evaluation was carefully synthesized using a Large Language Model (with real data as reference) as to not correspond with any particular person or patient's journey. The humans surveyed in the human evaluation were all employees of Kaliber AI, raising a potential conflict-of-interest in terms of providing unbiased review metrics. To address this, systems were evaluated in a blind manner with order of system responses further scrambled. A standard set of instructions were given to each subject and no further guidance was given. All completed evaluations were included in the analysis. Further, all other employees and thus humans surveyed were unaware of any details regarding Mnemosyne and the project's purpose even external to the instruction set given, ameliorating any concerns about conflict-of-interest and prior knowledge leakage into results. Full evaluation data is provided in the supplementary materials. LLMs were used in a very limited extent to provide help with sentence structure and grammar.

**Reproducibility Statement**

The code for Mnemosyne, as well as the engineering scaffolding for execution such as environment setup, Redis database code, and LLM layers, is provided in the supplementary material. Input data and data processing for evaluation, including LoCoMo and retrieval benchmarking, is additionally provided in a separate folder in the code provided in the supplementary materials. Full algorithms are outlined in Appendix B, and our hyperparameter setup for evaluation is provided in Table C3. Mathematical equations for our algorithms and metrics are provided in Appendix A, and our code implementation can be found under Appendix E. For transparency, we have also released the human evaluators' set of instructions, response forms, and responses. Although human evaluation metrics change across the set of humans surveyed, we are confident that our methods were thorough enough that the results are truly significant.

## References


Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025. URL `https://arxiv.org/abs/2504.19413`.

Martin A. Conway. Memory and the self. *Journal of Memory and Language*, 53(4):594–628, 2005. ISSN 0749-596X. doi: https://doi.org/10.1016/j.jml.2005.08.005. URL `https://www.sciencedirect.com/science/article/pii/S0749596X05000987`.

Tyrone E Duncan. On the calculation of mutual information. *SIAM Journal on Applied Mathematics*, 19(1):215–220, 1970.

Hermann Ebbinghaus. Memory (ha ruger & ce bussenius, trans.). *New York: Teachers College.(Original work published 1885)*, 39, 1913.

Ofer Feinerman, Menahem Segal, and Elisha Moses. Signal propagation along unidimensional neuronal networks. *Journal of neurophysiology*, 94(5):3406–3416, 2005.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Philip M Groves and Richard F Thompson. Habituation: a dual-process theory. *Psychological review*, 77(5):419, 1970.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare*, 3(1):1–23, October 2021. ISSN 2637-8051. doi: 10.1145/3458754. URL http://dx.doi.org/10.1145/3458754.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Natasha Kharas, Ariana Andrei, Samantha R Debes, and Valentin Dragoi. Brain state limits propagation of neural signals in laminar cortical circuits. *Proceedings of the National Academy of Sciences*, 119(30):e2104192119, 2022.

Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 69(6):066138, 2004.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021. URL https://arxiv.org/abs/2005.11401.

Jiahao Liu, Xueshuo Yan, Dongsheng Li, Guangping Zhang, Hansu Gu, Peng Zhang, Tun Lu, Li Shang, and Ning Gu. Improving llm-powered recommendations with personalized information. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2560–2565, 2025a.

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, Yuanxing Zhang, Zhuo Chen, Hangyu Guo, Shilong Li, Ziqiang Liu, Yong Shan, Yifan Song, Jiayi Tian, Wenhao Wu, Zhejian Zhou, Ruijie Zhu, Junlan Feng, Yang Gao, Shizhu He, Zhoujun Li, Tianyu Liu, Fanyu Meng, Wenbo Su, Yingshui Tan, Zili Wang, Jian Yang, Wei Ye, Bo Zheng, Wangchunshu Zhou, Wenhao Huang, Sujian Li, and Zhaoxiang Zhang. A comprehensive survey on long context language modeling, 2025b. URL https://arxiv.org/abs/2503.17407.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023. URL https://arxiv.org/abs/2307.03172.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

Johannes B. Mahr and Gergely Csibra. Why do we remember? the communicative function of episodic memory. *Behavioral and Brain Sciences*, 41:e1, 2018. doi: 10.1017/S0140525X17000012.

Clara W Mayo and Walter H Crockett. Cognitive complexity and primacy-recency effects in impression formation. *The Journal of Abnormal and Social Psychology*, 68(3):335, 1964.

Alexandra B Morrison, Andrew RA Conway, and Jason M Chein. Primacy and recency effects as indices of the focus of attention. *Frontiers in human neuroscience*, 8:6, 2014.

Jaap M. J. Murre and Joeri Dros. Replication and analysis of ebbinghaus' forgetting curve. *PLOS ONE*, 10(7):1–23, 07 2015. doi: 10.1371/journal.pone.0120644. URL https://doi.org/10.1371/journal.pone.0120644.

NeuML. pubmedbert-base-embeddings. https://huggingface.co/NeuML/pubmedbert-base-embeddings, 2023.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

Yilun Qiu, Xiaoyan Zhao, Yang Zhang, Yimeng Bai, Wenjie Wang, Hong Cheng, Fuli Feng, and Tat-Seng Chua. Measuring what makes you unique: Difference-aware user modeling for enhancing llm personalization. *arXiv preprint arXiv:2503.02450*, 2025.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Guilherme Rosa, Luiz Bonifacio, Vitor Jeronymo, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, and Rodrigo Nogueira. In defense of cross-encoders for zero-shot retrieval. *arXiv preprint arXiv:2212.06121*, 2022.

Salvatore Sanfilippo and Redis Contributors. Redis. https://redis.io, 2025.

Stephen R Schmidt. Emotion, significance, distinctiveness, and memory. *Distinctiveness and memory*, pages 47–64, 2006.

Shreyas Subramanian, Vikram Elango, and Mecit Gungor. Small language models (slms) can still pack a punch: A survey, 2025. URL https://arxiv.org/abs/2501.05465.

Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. *arXiv preprint arXiv:2010.08240*, 2020.

RF Thompson and WA Spencer Habituation. A model phenomenon for the study of neuronal substrates of behavior., 1966, 73. *DOI: https://doi. org/10.1037/h0022681. PMID: https://www. ncbi. nlm. nih. gov/pubmed/5324565*, pages 16–43, 1966.

Vassilis Tsakanikas, Tasos Dagiuklas, Muddesar Iqbal, Xinheng Wang, and Shahid Mumtaz. An intelligent model for supporting edge migration for virtual function chains in next generation internet of things. *Sci. Rep.*, 13(1):1063, January 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Fali Wang, Zhiwei Zhang, Xianren Zhang, Zongyu Wu, Tzuhao Mo, Qiuhao Lu, Wanjing Wang, Rui Li, Junjie Xu, Xianfeng Tang, Qi He, Yao Ma, Ming Huang, and Suhang Wang. A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with llms, and trustworthiness, 2024. URL https://arxiv.org/abs/2411.03350.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks, 2024. URL https://arxiv.org/abs/2309.17453.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning, 2025. URL https://arxiv.org/abs/2508.19828.

Yue Zheng, Yuhao Chen, Bin Qian, Xiufang Shi, Yuanchao Shu, and Jiming Chen. A review on edge large language models: Design, execution, and applications, 2025. URL https://arxiv.org/abs/2410.11845.

# Appendices

## A Equations

**Rewind Calculation**. Let $e = e_{nm}$ be the edge being boosted, $t$ the current time $t_{\text{curr}}$, and $e_{\text{boosted}}$ the time of the last edge boost. Given constants $t_{\text{crit}}$, which determines how long the function takes to plateau, and $\Delta_{\max}$, which is the maximum rewind value, we define the rewind value $\Delta_e(t)$ to be

$$\Delta_e(t = t_{\text{curr}}) = \Delta_{\max}\left(\frac{1}{1 + \exp(-t + e_{\text{boosted}} + t_{\text{crit}})}\right) \tag{A1}$$

**Decay Calculation**. Given $a$, the middle of the sigmoid, $b$, the sigmoid's "steepness" (lower values indicate steeper dropoff), $c$, the transition point of the linear correction domain, and $d$, the floor decay value, our decay function can be formalized as

$$\tau(e_{\text{eff}}) = \begin{cases} \frac{1 - d}{1 + e^{\frac{(e_{\text{eff}} - a)}{b}}} & c \leq e_{\text{eff}} \\ -\frac{e_{\text{eff}}(1 - \tau(c))}{c} & 0 \leq e_{\text{eff}} \leq c \end{cases} \tag{A2}$$

From our independent testing, the best values of $b$ and $c$ are $b \approx \frac{a}{4}$ and $c \approx b$.

**Pruning Score Calculation**. Given node $n$, neighbors $N(n)$, and the weight of edge $e_{nm}$,

$$\text{PS}(n) = \max_{m \in N(n)} (e_{nm} \cdot \tau(e_{\text{eff}})) \tag{A3}$$

**Normalized Discounted Cumulative Gain.** nDCG rewards systems that rank relevant memories higher, calculated as

$$\text{nDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k} \tag{A4}$$

where $\text{DCG@}k = \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)}$and $\text{rel}_i$ is the relevance score of the retrieved memory at position $i$, and IDCG represents the ideal DCG if all relevant memories were retrieved in perfect order.

**Win Rate**. Given memory system $S_{\text{mem}}$,

$$\text{WR}(S_{\text{mem}}) = \frac{\text{number of wins} + 0.5 \cdot \text{number of ties}}{\text{total number of opportunities}} \tag{A5}$$

## B Algorithms

**Algorithm B1** Substance Filter

```
function HasSubstance(s_new, G)
    context ← s_new + G.core_summary
    prompt ← "Given the context, output 1 if the conversation has substance, else 0."
    result ← LLM(prompt, context)                                   ▷ e.g., Mistral-7B
    return result = 1
end function
```

**Algorithm B2** Redundancy Filter

```
function DetermineRedundancy(n_new, G)
    m⁺ ← FindMostRedundant(n_new, G)
    if m⁺ is null then
        return ADD_NEW, n_new
    end if
    if m⁺.is_paired is null then
        return PAIR_NEW, (n_new, m⁺)
    else
        n_j ← G.GetNode(m⁺.is_paired)
        n_older ← SelectOlder(m⁺, n_j)
        n_newer ← SelectNewer(m⁺, n_j)
        return REPLACE_AND_PAIR, (n_new, n_older, n_newer)
    end if
end function
```

**Algorithm B3** Commitment Algorithm

```
procedure CommitMemory(s_new, d_new, G)
    if ¬HasSubstance(s_new, G) then
        return G                                   ▷ Discard summary, no changes made
    end if
    n_new ← InstantiateNode(s_new, d_new)
    n_red ← FindMostRedundantNode(n_new, G)                ▷ Finds most similar node
    if n_red is null then                          ▷ Case 1: No redundant node found
        G.AddNode(n_new)
    else if n_red.is_paired is null then      ▷ Case 2: Redundant with an unpaired node
        G.AddNode(n_new)
        G.SetPair(n_new, n_red)
        e_{n_red n_new}.boost ← Δ_e(t)
    else                                  ▷ Case 3: Redundant with an already paired node
        n_pair ← G.GetNode(n_red.is_paired)
        n_to_keep ← SelectOldest(n_red, n_pair)
        n_to_remove ← SelectNewest(n_red, n_pair)
        G.RemoveNode(n_to_remove)
        G.AddNode(n_new)
        G.SetPair(n_new, n_to_keep)
        e_{n_red n_new}.boost ← Δ_e(t)
    end if
    ConnectNewNode(n_new, G)                       ▷ Create edges to similar nodes
    return G
end procedure
```

**Algorithm B4** Start Node Selection

**Input:** Query $q$, Memory graph $G$
**Output:** Start node $n_{best}$

```
function SelectStartNode(q, G)
    n_best ← null
    s_max ← −1
    for each node n_i in G do
        s_summary ← CosineSimilarity(q.embedding, n_i.embedding)
        s_meta ← CosineSimilarity(q.embedding, n_i.meta_embedding)
        α_meta ← CalculateAlpha(n_i.age)                ▷ α_meta increases with age
        s_total ← α_meta s_meta + (1 − α_meta) s_summary
        if s_total > s_max then
            s_max ← s_total
            n_best ← n_i
        end if
    end for
    return n_best
end function
```

**Algorithm B5** Probabilistic Traversal

**Input:** Query $q$, Memory graph $G$, Exploration hyperparameter $\mu$
**Output:** List of recalled node indices $R$

```
procedure RecallTraversal(n_curr, G, R, V, μ)
    V ← V ∪ {n_curr}
    R.append(n_curr.index)
    for each neighbor n_k of n_curr do
        if n_k ∉ V then
            e_curr,k ← G.GetEdge(n_curr, n_k)
            t_eff ← CalculateEffectiveAge(e_curr,k)
            τ ← DecayFunction(t_eff)                    ▷ Apply reverse sigmoid decay
            p_choose ← e_curr,k.weight × τ × μ
            r ← Random(0, 1)
            if r < p_choose then
                RecallTraversal(n_k, G, R, V, μ)
            end if
        end if
    end for
end procedure
```

## C Tables

| Attribute | Description |
|---|---|
| `index` | Unique node ID (hash or int) |
| `neighbors` | List of connected node indices |
| `is_paired` | If exists, node ID of paired node |
| `t_init` | Time of original conversation |

(a) Node attributes.

| Attribute | Description |
|---|---|
| `weight` | Similarity score between two nodes |
| `boost` | Accumulated rewind value |
| `t_boosted` | Time of last edge boost during commitment |

(b) Edge attributes.

Table C1: Attributes of memory graph components.

| Attribute | Description |
|---|---|
| `summary_text` | Full summary text |
| `summary_vector` | Summary embedding |
| `hypothetical_query` | Hypothetical query |
| `hypothetical_vector` | Hypothetical query embedding |
| `keywords` | Keywords |

Table C2: Node attributes stored externally.

| Component | Parameter | Value | Description |
|---|---|---|---|
| **Graph Construction & Commitment** | | | |
| CommitmentEngine | edge_threshold | 0.5 | Minimum combined similarity score required to create an edge between two nodes. |
| CommitmentEngine | $\alpha_{\text{key}}$ | 0.3 | The weight of keyword Jaccard similarity vs. semantic cosine similarity for calculating edge weights. (1-$\alpha_{\text{key}}$)*semantic + $\alpha_{\text{key}}$*keyword. |
| **Redundancy Detection** | | | |
| RedundancyDetector | threshold | 0.25 | Score above which a new node is considered redundant with an existing one. |
| RedundancyDetector | $\alpha_{\text{NMI}}$ | 0.6 | The weight of Normalized Mutual Information (NMI) vs. Jaccard similarity in the base redundancy score. |
| RedundancyDetector | temporal_weight_factor | 0.5 | Multiplier for the temporal bonus, controlling the influence of time proximity on the final redundancy score. |
| RedundancyDetector | temporal_decay_hours | 24 | The "half-life" (in hours) controlling how quickly the temporal bonus for redundancy fades. |
| **Memory Recall** | | | |
| RecallEngine | $\alpha_{\text{meta}}$ | 0.6 | The weight of the semantic (hypothetical query) score vs. the metadata score for selecting the recall start node. |
| RecallEngine | $\mu$ | 2.0 | Exploration hyperparameter that scales the probability of traversing to a neighbor node during recall propagation. |
| RecallEngine | mid_sig | 2,419,200 | The effective age (in seconds, default is 4 weeks) at which the temporal decay function reaches its midpoint. |
| RecallEngine | floor | 0.05 | The minimum probability (floor) returned by the temporal decay function, ensuring old memories can still be recalled. |
| RecallEngine | top_k_starts | 3 | Number of top nodes to select for multi-start traversal. |
| RecallEngine | min_threshold | 0.4 | Minimum score threshold to consider a node relevant. |
| RecallEngine | max_nodes_per_start | 1 | Maximum nodes to collect from each start point. |
| **Core Summary** | | | |
| GetCentralSubset | $\theta_{\text{conn}}$ | 0.3 | Determines the weight given to a node's centrality, prioritizing memories that are semantically related to many other memories. |
| GetCentralSubset | $\theta_{\text{boost}}$ | 0.3 | Governs the influence of memory reinforcement, prioritizing nodes that have been frequently boosted by incoming redundant interactions. |
| GetCentralSubset | $\theta_{\text{rece}}$ | 0.2 | Sets the weight for temporal recency, controlling the preference for newer memories during the core summary selection process. |
| GetCentralSubset | $\theta_{\text{ent}}$ | 0.2 | Controls the influence of a memory's information density, serving to penalize semantically generic but otherwise well-connected nodes. |
| GetCentralSubset | $\lambda$ | 28 | Timescale for recency decay (half-life in days) |

Table C3: Hyperparameter settings

# D Supplementary Evaluations

## D.1 Retrieval Benchmarking

**Experimental Setup:** To evaluate Mnemosyne's ability to recall both correct and contextually relevant information, we conducted a retrieval fidelity test between Mnemosyne and a naive RAG baseline, focusing specifically on the start node selection mechanism. Start node selection is the hypothetical bottleneck of our retrieval-based system since all subsequent recall logic is triggered from the start node.

We created a test set of 20 factual, time-specific queries targeting key clinical information across different surgical phases (e.g., "Where and when was the patient told to complete pre-operative blood tests?"). For each query, we manually defined a list of gold standard memory node IDs representing the ground truth. To ensure a fair comparison, we evaluated only the start node selection mechanism of Mnemosyne against RAG's top-$k$ retrieval, setting $k = 3$ for both. Notably, we disabled traversal of Mnemosyne for this benchmark to ensure that we were only measuring the quality of start nodes. This separates the traversal benefit of contextual augmentation using redundancy, boosting, and time-decay from the initial retrieval process.

Two complementary metrics were used to assess performance: (1) hit rate, which measures the percent of queries that resulted in the retrieval of at least one pertinent memory, and (2) normalized discounted cumulative gain (nDCG@3), which accounts for both relevance and ranking position of retrieved memories.

**Results:** Our benchmarking results are shown in Table C4. On both metrics, we can observe that Mnemosyne's start node selection mechanism performs better than naive RAG baseline. With a hit rate of 0.75 compared to the naive RAG's 0.64, Mnemosyne showed an absolute increase of 11% in its ability to correctly retrieve at least one gold standard ID. Additionally, using Mnemosyne increased the nDCG from 0.625 to 0.696, an improvement of 0.071. These improvements indicate that Mnemosyne not only finds relevant memories more accurately, but also positions them more precise in retrieval order. Furthermore, our hybrid scoring approach, which combined semantic similarity with temporal and contextual metadata was validated. The improvement over pure semantic similarity search (RAG) demonstrates that incorporating temporal and contextual information into retrieval process results in better memory selection, even before applying graph traversal.

Table C4: Retrieval benchmarking results. The strongest results are bolded.

| System | Hit Rate | nDCG |
|---|---|---|
| **Mnemosyne** | **0.75** | **0.696** |
| Naive RAG | 0.64 | 0.625 |

**Commentary:** This evaluation starts with framing start-node selection as the main bottleneck of the system and compares it directly to standard RAG. Even in this stripped down form, Mnemosyne slightly beats RAG on hit rate and nDCG which acts as a good sanity check that the base retrieval mechanism is sound. The rest of the system, including traversal, filters, and core summary, act as augmentations that drive improvements in naturalness as the main body analysis shows.

## E Downloadable Supplementary Material (Code, Evaluation Data)

Refer to this link to download the main code for this project, including environment setup instructions, input data, and hit rate evaluation:

Click to Download Mnemosyne Code

Refer to this link to download the LoCoMo benchmarking code for this project:

Click to Download LoCoMo benchmarking Code

Refer to this link to download the project code for human input evaluation:

Click to Download Mnemosyne Human Evaluation Code

Refer to this link to download the raw human evaluation responses (with 1 no-response excluded from the survey):

Click to Download Human Evaluation Responses