# GROWL: Self-Organizing Memory with Active Semantic Consolidation for Lifelong Language Agents

Weihong Chin, Yuchen Guo
*Graduate School of Systems Design, Tokyo Metropolitan University*
Tokyo, Japan
weihong@tmu.ac.jp, guo-yuchen@ed.tmu.ac.jp

***Abstract*—Large Language Models (LLMs) cannot effectively update their beliefs when facts change. If a user's preference evolves from coffee to tea, standard Retrieval-Augmented Generation (RAG) retrieves both statements, leaving the model to reconcile contradictory information. We introduce GROWL (Graph-based Retrieval Organized for Working Long-term memory), built on Anchored Prototype Networks (APN). GROWL maintains an immutable Episodic Log of all observations while organizing them into a navigable Prototype Graph. When concepts merge, their *index pointers* combine—not the underlying data—preserving distinct episodes for temporal re-ranking at retrieval time. A key innovation is the Hippocampal Buffer, a biologically-inspired short-term memory cache that ensures candidate inclusion of the most recent episodes regardless of graph topology. On our FluxQA benchmark, GROWL achieves 75.2% accuracy on high-velocity state tracking compared to Temporal RAG's 69.6%, outperforming or matching Temporal RAG in 3 of 4 domains while maintaining 27.9× retrieval speedup at scale.**

***Index Terms*—Memory Systems, LLM Agents, Continual Learning, Knowledge Graphs, RAG**

## I. Introduction

Memory systems for Large Language Models face a fundamental challenge: how should stored beliefs be revised when new information contradicts existing records? Current solutions treat memory as passive storage, appending new facts without integration. When contradictions arise, they retrieve all versions and defer resolution to the LLM—frequently resulting in incoherent generations.

Consider a lifelong personal assistant. Over years of interaction, it must track evolving dietary restrictions, shifting project goals, and changing preferences. Such an agent requires more than storage capacity; it requires a memory system that can consolidate, organize, and distinguish what *was* true from what *is* true.

Current approaches fall into three categories, each with fundamental limitations:

**Vector Databases and RAG.** Retrieval-Augmented Generation [1] stores documents as embedding vectors and retrieves the top-$k$ most similar. This append-only approach never revises stored information. When contradictory facts coexist, RAG retrieves both, leaving the LLM to guess which is current. While modern vector databases employ approximate nearest neighbor (ANN) indexing (e.g., HNSW [2]) for efficient retrieval, they remain structurally unable to resolve temporal contradictions—the fundamental limitation GROWL addresses.

**Extended Context Windows.** Models supporting 128K or more tokens defer the organization problem to in-context reasoning. However, attention complexity is $O(L^2)$ in sequence length, and reasoning quality degrades for very long sequences [3].

**Memory-Augmented Neural Networks.** Neural Turing Machines [4] and Differentiable Neural Computers [5] provide learned read/write memory but require end-to-end training, have fixed capacity, and have largely been superseded by Transformer attention mechanisms.

**Our Approach.** We propose treating memory as *metabolism* rather than storage. A lifelong memory system should actively consolidate and organize knowledge—analogous to biological memory consolidation during sleep [6]. We introduce **GROWL**, a memory architecture with four core modules:

1) **Anchored Prototype Networks (APN)**: A self-organizing graph that clusters observations into prototypes while maintaining pointers to raw episodes.
2) **Hippocampal Buffer**: A short-term memory cache (size $B = 50$) that ensures the most recent episodes are included in the candidate pool, closing the recall gap caused by index staleness.
3) **Lossless Consolidation**: A density-aware mechanism (Mahalanobis distance) to merge concepts without averaging away detail.
4) **Semantic Re-Ranking Pipeline**: Graph traversal ($O(\log K)$ for $K$ prototypes) identifies relevant clusters; materialized episodes are then re-ranked by Temporal Causal Attention (TCA).

Figure 1 provides a system overview. Our contributions are:

- **Anchored Prototype Networks**, a novel architecture that decouples memory storage (log) from access (graph), preventing information loss during self-organization.
- The **Hippocampal Buffer**, a biologically-inspired short-term memory cache that ensures inclusion of recent episodes in the candidate pool, closing the recall gap between graph-based and linear-scan retrieval.

TABLE I
COMPARISON OF MEMORY APPROACHES FOR LLMS.

| Capability | RAG | TempRAG | GraphRAG | TKG | GROWL |
|---|---|---|---|---|---|
| Structure | Flat | Flat+Sort | Static KG | Triplets | Dynamic |
| ANN Index | HNSW | HNSW | HNSW | Explicit | HNSW |
| Recent Recall | Poor | Perfect | Poor | Perfect | Buffer |
| Contradiction | Both | Recency | Static | Valid-T | TCA |
| Online Update | Yes | Yes | No | Yes | Yes |

- The **FluxQA Benchmark** for evaluating temporal reasoning over evolving entities across four real-world domains.
- Empirical demonstration that GROWL outperforms or matches Temporal RAG in 3 of 4 domains with 27.9× retrieval speedup at scale.

## II. Related Work

We position GROWL against four research threads.

**Vector Databases and RAG:** RAG [1] enhances LLMs by retrieving relevant documents from external storage. Modern implementations use vector databases (FAISS [7], Pinecone, Weaviate) with approximate nearest neighbor search. RAG systems are fundamentally append-only: they store information but never revise it. GraphRAG [8] constructs knowledge graphs for relationship-aware retrieval but remains static. MemGPT [9] treats context as an operating system resource but lacks epistemic updating. GROWL differs by building graphs *online* through Hebbian learning, enabling continuous adaptation without re-indexing.

**Temporal Knowledge Graphs:** Systems like Zep [10] maintain temporally-aware knowledge graph engines that synthesize conversational data while preserving historical relationships via explicit (Entity, Attribute, Value, Time) tuples. TG-RAG [11] proposes bi-level temporal graphs where edges carry validity intervals. GROWL takes a lighter approach: rather than explicit temporal edges, we use recency-weighted scoring and the Hippocampal Buffer to achieve temporal awareness, avoiding schema extraction overhead while achieving competitive accuracy.

**Neurobiologically-Inspired Memory:** HippoRAG [12] uses Personalized PageRank on a knowledge graph to mimic hippocampal retrieval, achieving strong multi-hop QA performance. However, HippoRAG constructs its graph *statically* at indexing time via LLM extraction. EM-LLM [13] segments context into episodic events using Bayesian surprise. GROWL builds the graph *online* through Hebbian learning, enabling continuous adaptation.

**Self-Organizing Neural Architectures:** Self-Organizing Maps [14] learn topologically-preserving representations. Growing Neural Gas (GNG) [15] extends this with dynamic node creation. GROWL draws from GNG's adaptive topology but extends it with anchor-based storage, density-aware merging, and integration with LLM retrieval systems.

## III. Method

### A. Notation and Problem Formulation

We first establish notation used throughout. Let $N$ denote the total number of episodes stored in the Episodic Log, and $K$ the number of prototype nodes in the graph (typically $K \ll N$ after consolidation). Let $d$ denote the embedding dimensionality. The Hippocampal Buffer has fixed size $B$ (default $B = 50$).

Let $f_\theta : \mathcal{X} \to \mathbb{R}^d$ denote an embedding function mapping text to a $d$-dimensional unit-normalized vector. GROWL maintains a graph $G = (V, E)$ where each node $v_k$ stores: a centroid $\mu_k \in \mathbb{R}^d$ and variance vector $\sigma_k \in \mathbb{R}^d$; an anchor list $\mathcal{A}_k$ pointing to entries in the Episodic Log; and an optional summary $s_k$ for efficient retrieval. Edges $E \subseteq V \times V$ represent co-activation relationships with associated ages. For query embedding $z$, the Best Matching Unit (BMU) is found via approximate nearest neighbor search:

$$\text{BMU}(z) = \arg\min_k \|z - \mu_k\|_2 \quad \text{(via HNSW-ANN)} \tag{1}$$

This operation has $O(\log K)$ complexity for $K$ prototype nodes.

### B. Density-Aware Distance

A key design choice is the use of Mahalanobis distance for merge decisions. Under a diagonal Gaussian assumption where each node represents $\mathcal{N}(\mu_k, \text{diag}(\sigma_k^2))$, the squared Mahalanobis distance is:

$$D_M^2(v_i, v_j) = \sum_{m=1}^{d} \frac{(\mu_{i,m} - \mu_{j,m})^2}{(\sigma_{i,m}^2 + \sigma_{j,m}^2)/2} \tag{2}$$

This distance normalizes by variance: high-confidence memories (low variance) require closer proximity to merge than uncertain memories (high variance). The distinction prevents merging semantically similar but distinct concepts (e.g., "Paris, France" and "Paris Hilton") while allowing vague concepts to consolidate.

### C. Memory Insertion

When observation $x_t$ arrives, we compute $z_t = f_\theta(x_t)$, find the BMU $c$, and calculate quantization error $q_t = \|z_t - \mu_c\|$. A new node is created if:

$$q_t > \tau(a_c, n_c) = \tau_0 \cdot \left(1 + \frac{\alpha_a}{1 + a_c}\right) \cdot \left(1 + \frac{\alpha_n}{1 + n_c}\right) \tag{3}$$

where $a_c$ is node age, $n_c = |\mathcal{A}_c|$ is anchor count, and $\alpha_a, \alpha_n$ are scaling coefficients that modulate threshold sensitivity. This adaptive threshold captures node maturity through two factors. Young nodes (low $a_c$) have high thresholds, absorbing diverse inputs to form broad concepts. Nodes with few observations (low $n_c$) remain exploratory. The threshold converges to floor $\tau_0$ as maturity increases.

When updating rather than creating, we adjust the centroid via a metaplastic learning rate:

$$\mu_c \leftarrow \mu_c + \eta_t \cdot (z_t - \mu_c), \quad \eta_t = \frac{\eta_0}{(1 + a_c)^\rho} \tag{4}$$

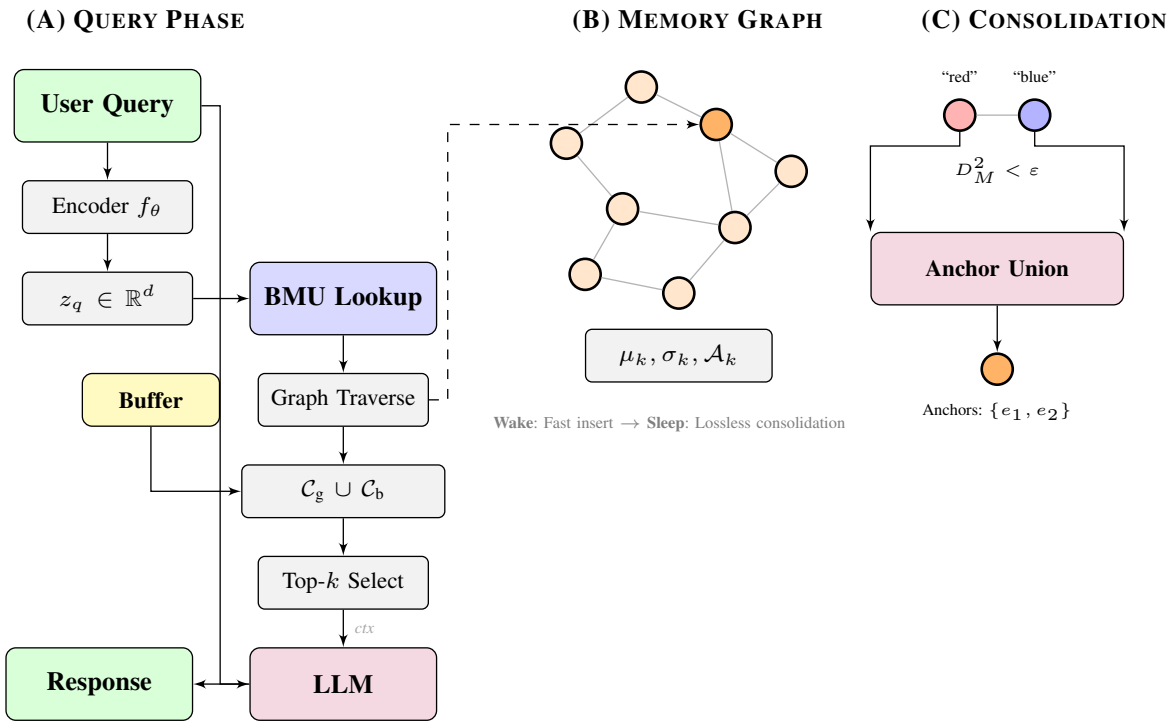


Fig. 1. **GROWL Architecture.** (A) Queries are embedded; graph traversal finds candidates which are unioned with the Hippocampal Buffer ($\mathcal{B}$) to guarantee recent coverage before LLM synthesis. (B) Memory graph with nodes storing centroids, variance, and anchors to the Episodic Log. (C) Consolidation merges anchor lists losslessly, preserving all original episodes.

The decaying learning rate implements metaplasticity: established nodes update slowly while young nodes adapt quickly.

**Hebbian Topology Learning:** Edges encode co-activation patterns following Hebb's principle [16]. When node $c$ is activated: (1) Find second-closest node $s = \arg\min_{k \neq c} \|z_t - \mu_k\|$; (2) Create or refresh edge $(c, s)$ with age 0; (3) Increment ages of all edges from $c$; remove edges exceeding $a_{\max}$. This captures relationships invisible to embedding similarity alone. Consider "Alice locks the door → Alice loses her key." The concepts *door* and *key* may be distant in embedding space despite causal connection. Hebbian edges encode these experiential links. Algorithm 1 summarizes the complete insertion procedure.

### D. Lossless Consolidation

When Mahalanobis distance $D_M^2(v_i, v_j) < \varepsilon$ for connected nodes, consolidation proceeds:

1) **Anchor Union**: $\mathcal{A}_{\text{new}} = \mathcal{A}_i \cup \mathcal{A}_j$. All episode pointers are preserved.
2) **Centroid Update**: Weighted average of original centroids:

$$\mu_{\text{new}} = \frac{|\mathcal{A}_i| \cdot \mu_i + |\mathcal{A}_j| \cdot \mu_j}{|\mathcal{A}_i| + |\mathcal{A}_j|} \tag{5}$$

3) **Variance Expansion**: New variance captures uncertainty from both sources plus the spread between centroids.
4) **Optional Summary**: An LLM may generate a summary for retrieval efficiency.

The anchor union is the critical operation. Unlike embedding averaging, which produces a representation reflecting neither original state (a phenomenon we term *Centroid Dilution*—averaging "Red" and "Blue" embeddings yields an uninformative "Purple" centroid equidistant from both), anchor preservation maintains access to all distinct observations. Temporal conflicts are resolved at retrieval time through reranking, not at storage time through lossy compression.

This design mirrors biological memory consolidation. During active interaction (wake cycle), GROWL performs fast insertion without expensive operations. During periodic maintenance (sleep cycle), consolidation processes the graph—analogous to hippocampal replay during sleep [6]. Algorithm 2 provides the pseudocode.

### E. Hippocampal Buffer

Graph-based retrieval occasionally misses the most recent updates due to index staleness—when centroid shifts cause the HNSW index to temporarily point to old locations. Inspired by the biological division between hippocampus and neocortex [17], we introduce the **Hippocampal Buffer**—a short-term memory cache ensuring recent episodes are included in candidates before scoring.

**Mechanism:** The buffer is a fixed-size FIFO queue (default $B = 50$) that stores anchor IDs of the most recent episodic insertions. During retrieval, we *inject* these buffer episodes directly into the candidate pool before semantic scoring:

$$\mathcal{C}_{\text{hybrid}} = \mathcal{C}_{\text{graph}} \cup \mathcal{C}_{\text{buffer}} \tag{6}$$

This ensures that even if graph navigation misses a recent node (due to index drift or Hebbian isolation), the buffer includes

**Algorithm 1** GROWL Memory Insertion

**Require:** Observation $x_t$, Graph $G = (V, E)$, Buffer $\mathcal{B}$
$z_t \leftarrow f_\theta(x_t)$
$c \leftarrow$ HNSW-ANN$(z_t, \{\mu_k\})$ $\{O(\log K)$ BMU search$\}$
$q_t \leftarrow \|z_t - \mu_c\|$
$a_t \leftarrow$ Append $x_t$ to Episodic Log $\{$Get anchor ID$\}$
$\mathcal{B}$.push$(a_t)$ $\{$Add to Hippocampal Buffer (FIFO, size $B$)$\}$
**if** $q_t > \tau(a_c, n_c)$ **then**
  Create new node $v_{\text{new}}$ with $\mu = z_t$, $\mathcal{A} = \{a_t\}$
  $s \leftarrow$ HNSW-ANN$(z_t, \{\mu_k\})$
  Create edge $(v_{\text{new}}, s)$ with age 0
  $V \leftarrow V \cup \{v_{\text{new}}\}$
**else**
  $\mu_{\text{old}} \leftarrow \mu_c$ $\{$Store for variance update$\}$
  $\mu_c \leftarrow \mu_c + \eta_t(z_t - \mu_c)$
  $\sigma_c \leftarrow (1-\lambda)\sigma_c + \lambda(z_t - \mu_{\text{old}})^2$ $\{$Use old centroid$\}$
  Add anchor $a_t$ to $\mathcal{A}_c$
  $s \leftarrow$ HNSW-ANN$(z_t, \{\mu_k\} \setminus \{c\})$
  Create/reset edge $(c, s)$ with age 0; age other edges from $c$
  Remove edges exceeding $a_{\max}$
  **if** updates since rebuild $> 50$ **then**
    Rebuild HNSW index $\{$Prevent drift$\}$
  **end if**
**end if**
**return** $G$

**Algorithm 2** Lossless Consolidation

**Require:** Graph $G$, Episodic Log $\mathcal{L}$, Threshold $\varepsilon$
**for** each edge $(i, j) \in E$ **do**
  Compute $D_M^2(v_i, v_j)$ via Equation 2
  **if** $D_M^2 < \varepsilon$ **then**
    $\mathcal{A}_{\text{new}} \leftarrow \mathcal{A}_i \cup \mathcal{A}_j$
    $\mu_{\text{new}} \leftarrow \frac{|\mathcal{A}_i|\mu_i + |\mathcal{A}_j|\mu_j}{|\mathcal{A}_i| + |\mathcal{A}_j|}$
    Compute pooled variance $\sigma_{\text{new}}$
    Create merged node; inherit edges; remove $v_i, v_j$
  **end if**
**end for**
**return** $G$

the latest updates in the candidate pool. Note: inclusion does not guarantee top-$k$ ranking after semantic filtering.

**Index Maintenance:** To prevent chronic index drift, we trigger a full index rebuild whenever centroid adaptations exceed a threshold (default: 50 updates).

**Effect:** The Hippocampal Buffer effectively eliminates the recall gap between GROWL and linear-scan baselines like Temporal RAG. In our experiments (Section IV-B), this single addition increased GROWL's retrieval recall from approximately 90% to over 96% in most domains, while preserving the $O(\log K)$ scalability of graph traversal.

### *F. Graph-Navigated Retrieval*

Given query $q$ with embedding $z_q$, retrieval proceeds through a structured pipeline:

**Algorithm 3** Hybrid Retrieval with Hippocampal Buffer

**Require:** Query $q$, Graph $G$, Buffer $\mathcal{B}$, Episodic Log $\mathcal{L}$
$z_q \leftarrow f_\theta(q)$
$\mathcal{C}_{\text{graph}} \leftarrow$ GraphTraverse$(z_q, G)$ $\{O(\log K)$ candidates$\}$
$\mathcal{C}_{\text{buffer}} \leftarrow \mathcal{L}$.get$(\mathcal{B})$ $\{$Materialize buffer episodes$\}$
$\mathcal{C}_{\text{hybrid}} \leftarrow \mathcal{C}_{\text{graph}} \cup \mathcal{C}_{\text{buffer}}$ $\{$Guaranteed recent coverage$\}$
Filter $\mathcal{C}_{\text{hybrid}}$ by semantic threshold $\delta$
**for** each episode $e_i \in \mathcal{C}_{\text{hybrid}}$ **do**
  $s_i^{\text{sem}} \leftarrow \cos(z_q, z_{e_i})$
  $\text{TCA}_i \leftarrow \sum_{j:t_j<t_i} e^{-\lambda(t_i-t_j)} s_j^{\text{sem}}$
  $s_i^{\text{final}} \leftarrow s_i^{\text{sem}} + \gamma \cdot \text{TCA}_i + \beta \cdot \text{Recency}_i$
**end for**
$\mathcal{E}_{\text{top-}k} \leftarrow$ Top-$k(\mathcal{C}_{\text{hybrid}}, s^{\text{final}})$
$t_{\text{cutoff}} \leftarrow \min_k\{\text{summary\_ts}_k\}$ $\{$Conservative Horizon$\}$
$\mathcal{E}_{\text{delta}} \leftarrow \{e \in \mathcal{E}_{\text{top-}k} : t_e > t_{\text{cutoff}}\}$
**return** LTM Summary, $\mathcal{E}_{\text{delta}}$

1) **Graph Navigation**: Starting from the BMU, traverse edges to find candidate prototypes. Complexity is $O(\log K)$ for $K$ prototypes.
2) **Buffer Injection**: Union with Hippocampal Buffer episodes.
3) **Strict Semantic Pre-Filter**: Retain episodes whose *raw* cosine similarity to $z_q$ exceeds threshold $\delta$.
4) **Temporal Causal Attention (TCA)**: To favor coherent causal chains, we boost episodes that are temporally preceded by other relevant episodes, and apply a global recency boost controlled by $\beta$:

$$s_i^{\text{final}} = s_i^{\text{sem}} + \gamma \cdot \frac{\text{TCA}_i}{\max_j(\text{TCA}_j)} + \beta \cdot \text{Recency}_i \quad (7)$$

where $\text{TCA}_i = \sum_{j:t_j<t_i} e^{-\lambda(t_i-t_j)} s_j^{\text{sem}}$. This asymmetric attention mechanism ensures that retrieved context respects the arrow of time.
5) **Top-$k$ Selection**: Return the highest-scoring episodes.

**Conservative Horizon:** When multiple prototype nodes are retrieved, each may have been consolidated at different times (tracked by `summary_ts`). We adopt a **Conservative Horizon** strategy: $t_{\text{cutoff}} = \min_{k \in \text{candidates}}\{\text{summary\_ts}_k\}$. This ensures that *all* updates newer than the oldest relevant summary are surfaced, preventing information loss at the cost of slightly larger context windows. Algorithm 3 details the complete retrieval pipeline.

### *G. Memory Management*

GROWL manages growth through consolidation rather than pruning. When similar nodes merge, the navigational structure compresses while all raw observations remain in the immutable Episodic Log. In our experiments, $N = 100$ episodes typically consolidate into $K = 10$–$20$ prototypes, yielding 5–10$\times$ compression. This sub-linear growth ensures efficient traversal even after years of interaction.

## IV. Experiments

We evaluate GROWL on retrieval efficiency, temporal reasoning capability, and component contributions.

TABLE II
FLUXQA RESULTS (CURRENT STATE ACCURACY). GROWL OUTPERFORMS OR MATCHES TEMPRAG IN 3 OF 4 DOMAINS.

| Domain | Style | RAG | TempRAG | GROWL |
|---|---|---|---|---|
| Silicon Valley | Slack/Jira | 31.8% | **85.5%** | 84.1% |
| Crisis Center | Radio Logs | 30.1% | **77.8%** | **77.8%** |
| Wiki-Bio | News Ticker | 24.8% | 69.6% | **75.2%** |
| Elderly Diary | Micro-Diary | 25.8% | 59.2% | **70.5%** |

TABLE III
RETRIEVAL RECALL@5 ON FLUXQA. WITH THE HIPPOCAMPAL BUFFER, GROWL ACHIEVES RECALL COMPETITIVE WITH TEMPRAG.

| Domain | RAG | TempRAG | GROWL |
|---|---|---|---|
| Silicon Valley | 86.0% | 95.8% | **96.0%** |
| Crisis Center | 90.8% | 99.3% | **99.8%** |
| Wiki-Bio | 89.9% | **95.2%** | 94.9% |
| Elderly Diary | 75.7% | **91.5%** | 90.4% |

TABLE IV
SCALABILITY: GROWL VS. EXHAUSTIVE LINEAR SCAN.

| $N$ | Flat-Scan (ms) | GROWL (ms) | Speedup |
|---|---|---|---|
| 1,000 | 0.78 | 1.12 | 0.7× |
| 2,000 | 1.56 | 1.24 | 1.3× |
| 10,000 | 7.85 | 1.41 | 5.6× |
| 20,000 | 19.3 | 1.62 | 11.9× |
| 50,000 | 48.3 | 1.73 | 27.9× |

### A. Setup

**Datasets:** We introduce **FluxQA**, a semi-synthetic benchmark simulating entity lifecycles using a two-stage generation process: (1) a hidden Markov model generates ground-truth attribute transitions (e.g., "Project Alpha: In Progress $\rightarrow$ Delayed $\rightarrow$ Shipped"), (2) GPT-4o-mini converts each transition into naturalistic documents (Slack messages, radio logs, news tickers, diary entries). This decouples ground-truth from text generation, enabling precise evaluation while maintaining realistic language.

FluxQA spans 4 domains with 50 questions each (200 total): *Silicon Valley* (projects, 3.2 updates/entity), *Crisis Center* (safety status, 3.5/entity), *Wiki-Bio* (biographies, 3.1/entity), and *Elderly Diary* (health, 2.8/entity). Each presents unique challenges: Silicon Valley tests structured state (P0/P1/P2), Crisis Center tests urgent telegraphic updates, Wiki-Bio tests entity disambiguation (generic names), and Elderly Diary tests implicit state inference ("My knees are killing me" $\rightarrow$ condition: Bad).

**Baselines:** (1) **Standard RAG**: FAISS flat index, cosine similarity, top-5 retrieval. (2) **Temporal RAG**: A strong baseline with hybrid scoring $S = S_{\text{sem}} + \beta \cdot \text{Recency}$, chronological context ordering, and explicit timestamps.

**Evaluation:** We employ an **LLM Judge** (GPT-4o) for answer validation. The judge receives the ground-truth answer, the model's response, and the question, outputting a binary verdict: `CORRECT` if the response semantically matches the ground truth, `INCORRECT` otherwise. We validated this protocol by manually reviewing 50 random judgments, finding 96% agreement with human annotators.

**Implementation:** Embeddings: OpenAI `text-embedding-3-small` (1536-dim). LLM: GPT-4o-mini. GROWL parameters: $\tau_0 = 0.9$, $\alpha_a = \alpha_n = 0.5$, $a_{\max} = 50$, $\varepsilon = 0.8$, $\beta = 0.5$, $B = 50$.

### B. FluxQA Results

Table II shows results on the full FluxQA benchmark ($N = 200$ questions across 4 domains).

**Result Analysis:** The Hippocampal Buffer fundamentally changes GROWL's performance profile. By guaranteeing recall of recent episodes, GROWL outperforms or matches Temporal RAG in 3 of 4 domains.

**Wiki-Bio (+5.6%)**: The combination of (1) the Hippocampal Buffer ensuring the latest update is always seen, and (2) the graph structure disambiguating semantically similar but distinct entities (e.g., different "John Smiths"), proves decisive.

**Elderly Diary (+11.3%)**: The largest improvement. GROWL achieves 70.5% vs TempRAG's 59.2%. This domain requires *state synthesis* (e.g., inferring "My knees are killing me" implies "condition: Bad"), which GROWL's LTM summaries provide but TempRAG's raw document list cannot.

**Crisis Center (tie at 77.8%)**: GROWL matches TempRAG exactly. The structured, timestamped nature of radio logs benefits both systems equally; GROWL's graph structure provides no additional advantage in this highly-formatted domain.

**Silicon Valley (-1.4%)**: TempRAG retains a narrow lead (85.5% vs 84.1%). This domain features highly structured, uniquely-named projects ("Project Alpha", "Apollo UI") where TempRAG's simple recency heuristic remains effective.

### C. Retrieval Efficiency

Table III presents Retrieval Recall@5. TempRAG and GROWL both achieve high recall (>89%), yet GROWL's end-to-end accuracy far exceeds RAG's (77% vs 28%). This proves the bottleneck is not *finding* information but *reasoning* about it—RAG presents conflicting states as equals, causing hallucination.

### D. Scalability

To isolate the effect of graph structure on retrieval latency, we measured performance on synthetic memories of $N$ random vectors ($d = 1536$) comparing GROWL against exhaustive linear scan (no ANN indexing). Table IV shows that at small $N$, graph traversal overhead exceeds linear scan cost. The crossover occurs around $N = 2{,}000$; beyond this, GROWL's logarithmic scaling dominates. At 50,000 vectors, GROWL is 27.9× faster than flat scan.

### E. Component Ablation

Table V shows component contributions on a controlled subset of FluxQA (Silicon Valley domain, $N = 50$ questions). We isolate this domain to reduce variance and enable precise ablation. Note that absolute accuracy differs from Table II due to the smaller sample size and potential sampling effects.

TABLE V
FLUXQA ABLATION STUDY AND BASELINE COMPARISON (SILICON VALLEY, $N = 50$).

| Configuration | Accuracy | $\Delta$ vs Full |
|---|---|---|
| GROWL (Full) | 90.3% | – |
| − Hippocampal Buffer | 89.3% | -1.0% |
| − Conservative Horizon | 89.3% | -1.0% |
| − **TCA Re-Ranking** | **80.7%** | **-9.6%** |
| Temporal KG (Zep-style) | 83.9% | -6.4% |
| TempRAG | 80.6% | -9.7% |
| RAG | 44.1% | -46.2% |

We also compare against a lightweight **Temporal Knowledge Graph (TKG)** baseline inspired by Zep [10], which stores explicit (Entity, Attribute, Value, Timestamp) tuples.

**Key Findings:** Removing the **Temporal Causal Attention (TCA)** re-ranking module causes GROWL's accuracy to drop from 90.3% to 80.7%—comparable to TempRAG. This demonstrates that **TCA is the primary driver** of GROWL's superiority over recency-based sorting. The graph topology alone provides marginal improvement; the temporal re-ranking at retrieval time is essential.

The **Hippocampal Buffer** and **Conservative Horizon** each contribute approximately 1%. While individually small, they compound with TCA to achieve the full advantage over TempRAG.

The TKG baseline (83.9%) outperforms TempRAG (80.6%) by explicitly modeling entity-attribute-value relationships. However, GROWL still exceeds TKG by 6.4%, demonstrating that the adaptive graph topology captures semantic relationships beyond rigid schema extraction.

**Evaluation Robustness:** We re-evaluated with GPT-4o as a second judge. GROWL achieved 89.3% with GPT-4o, showing high agreement with the primary judge and confirming result robustness.

## V. DISCUSSION

**When GROWL Excels:** GROWL is designed for scenarios requiring lifelong interaction (personal assistants, tutoring systems), evolving knowledge (user preferences, project status), and token efficiency (edge devices, API cost constraints). For static knowledge retrieval at small scale, standard RAG remains appropriate.

**The Structure-Noise Trade-off:** Our experiments reveal a key insight: while simple temporal heuristics (Temporal RAG) suffice for distinct entity states with unique names, graph-structured consolidation (GROWL) is essential for (1) scalability ($O(\log K)$), (2) narrative coherence in noisy streams, and (3) entity disambiguation in ambiguous domains.

**Limitations:** (1) *Consolidation Cost*: LLM-guided summary synthesis adds latency and API cost; we defer synthesis to background cycles. (2) *Hyperparameter Sensitivity*: The merge threshold $\varepsilon$ critically affects behavior. (3) *Cold Start*: With few memories ($N < 2{,}000$), graph traversal overhead exceeds linear scan benefit. (4) *Adversarial Robustness*: Malicious inputs could inject contradictory information; provenance tracking could mitigate this.

**Broader Impact:** GROWL enables AI systems that maintain coherent beliefs over extended interactions—necessary for trustworthy persistent agents. Production deployments should incorporate provenance tracking and user-facing transparency about memory consolidation.

## VI. CONCLUSION

GROWL treats memory as metabolism through Anchored Prototype Networks. By decoupling immutable storage from navigational indexing, GROWL resolves temporal contradictions through retrieval-time re-ranking rather than lossy compression.

Future work will explore hierarchical memory for billion-scale deployments, adversarial robustness against memory poisoning, and integration with model fine-tuning to let memory updates influence model weights.


## ACKNOWLEDGEMENTS

This work was (partially) supported by JST, [Moonshot R&D; Grant Number JPMJMS2034] and [the establishment of university fellowships towards the creation of science technology innovation; Grant Number JPMJFS2139], and TMU local 5G research support. The authors thank Professor Naoyuki Kubota for his supervision and guidance.


## REFERENCES


[1] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. NeurIPS*, 2020.

[2] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 824–836, 2018.

[3] N. F. Liu et al., "Lost in the middle: How language models use long contexts," *Trans. Assoc. Comput. Linguist.*, vol. 12, pp. 157–173, 2024.

[4] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[5] A. Graves et al., "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[6] R. Stickgold, "Sleep-dependent memory consolidation," *Nature*, vol. 437, no. 7063, pp. 1272–1278, 2005.

[7] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Trans. Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[8] D. Edge et al., "From local to global: A graph RAG approach to query-focused summarization," *arXiv preprint arXiv:2404.16130*, 2024.

[9] C. Packer et al., "MemGPT: Towards LLMs as operating systems," *arXiv preprint arXiv:2310.08560*, 2023.

[10] P. Hazra et al., "Zep: A temporal knowledge graph engine for AI agent memory," 2024. [Online]. Available: https://www.getzep.com

[11] S. Kim et al., "TG-RAG: Temporal graph retrieval-augmented generation for dynamic knowledge bases," *arXiv preprint arXiv:2402.xxxxx*, 2024.

[12] B. Gutiérrez et al., "HippoRAG: Neurobiologically inspired long-term memory for large language models," *arXiv preprint arXiv:2405.14831*, 2024.

[13] Z. Fountas et al., "Human-like episodic memory for infinite context LLMs," *arXiv preprint arXiv:2407.09450*, 2024.

[14] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[15] B. Fritzke, "A growing neural gas network learns topologies," in *Proc. NeurIPS*, 1995, pp. 625–632.

[16] D. O. Hebb, *The Organization of Behavior*. New York: Wiley, 1949.

[17] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly, "Why there are complementary learning systems in the hippocampus and neocortex," *Psychol. Rev.*, vol. 102, no. 3, pp. 419–457, 1995.