

Topological Adaptive Graphs (TAG): A Topic-Scoped Memory Architecture for Persistent Context in Large Language Models

Souren Khetcho
Independent Research

February 2026

Abstract

We present Topological Adaptive Graphs (TAG), a memory architecture for large language models that organizes conversational context into topic-scoped vector spaces with a lightweight metadata routing layer. Unlike flat retrieval-augmented generation (RAG) systems that search a single embedding space, or hierarchical memory systems (e.g., MemGPT) that page context based on token budget pressure, TAG partitions memory by semantic coherence and routes queries through a two-tier retrieval mechanism: coarse-grained topic classification via continuously updated summary documents, followed by fine-grained semantic search within the selected vector space. We introduce a relevancy scoring system inspired by web search ranking algorithms, where user interaction signals, particularly correction events and re-reference frequency, drive automatic promotion and demotion of context between hot and cold storage tiers. The architecture employs a small, specialized router model for topic classification, keeping inference costs low while preserving high routing accuracy. We argue that this approach addresses the fundamental problem of context pollution in long-running conversational AI systems and provides a framework for persistent, personalized memory that scales with user history without degrading retrieval precision.

1 Introduction

Large language models have demonstrated remarkable capabilities in natural language understanding and generation, yet they remain fundamentally stateless. Each inference call operates within a fixed context window, and without external augmentation, no information persists between sessions. This limitation creates a persistent gap between user expectations, that an AI assistant should remember prior conversations, preferences, and ongoing topics, and the reality of token-bounded, session-scoped interactions.

The dominant approach to addressing this limitation has been Retrieval-Augmented Generation (RAG), where external documents are embedded into vector representations, stored in a vector database, and retrieved via cosine similarity at query time. While effective for knowledge-base augmentation, standard RAG architectures suffer from a critical flaw when applied to conversational memory: they treat all stored context as a flat, undifferentiated embedding space. As a user’s conversational history grows, semantically adjacent but topically unrelated content begins to contaminate retrieval results, a phenomenon we term *context pollution*. A query about a user’s career plans may surface fragments from a conversation about their partner’s career, not because the content is relevant, but because the embedding vectors share geometric proximity in a high-dimensional space.

Recent work has proposed more sophisticated memory architectures. MemGPT [Packer et al., 2023] draws on operating system design to introduce hierarchical memory with paging between main context and archival storage. Mem0 [Chhikara et al., 2025] provides a memory

orchestration layer with user, session, and agent scoping, combining vector search with graph-based relationships. A-Mem [Zhang et al., 2025] applies the Zettelkasten method to create self-organizing memory notes with dynamic linking. These systems represent meaningful progress, but each carries architectural limitations. MemGPT’s paging is driven by token budget pressure rather than semantic coherence. Mem0’s scoping is coarse-grained (user-level, session-level) rather than topic-level. A-Mem’s note-linking creates rich associations but lacks a routing mechanism that scales efficiently as the knowledge base grows.

In this paper, we propose Topological Adaptive Graphs (TAG), an architecture that addresses these limitations through three core innovations: (1) topic-scoped vector spaces that partition memory by semantic coherence rather than token budget or session boundaries; (2) a continuously evolving summary-based routing layer that enables efficient topic classification without evaluating the full memory store; and (3) a relevancy scoring system inspired by PageRank [Page et al., 1999] that uses user interaction signals to maintain a hot/cold storage hierarchy, ensuring that retrieval latency and accuracy scale gracefully with conversational history size.

2 Problem Statement

We identify three fundamental limitations in existing LLM memory architectures that TAG is designed to address.

2.1 Context Pollution in Flat Vector Spaces

Standard RAG implementations embed all conversational history into a single vector space. As the volume of stored interactions grows, retrieval quality degrades because cosine similarity does not distinguish between topical relevance and mere semantic proximity. Consider a user who has discussed both their fitness routine and their rehabilitation from a sports injury. Both topics share vocabulary related to exercise, pain, recovery, and physical activity. A flat embedding space will surface fragments from both topics indiscriminately, forcing the LLM to either hallucinate connections or ignore retrieved context that appears relevant but is topically misaligned.

2.2 Budget-Driven vs. Meaning-Driven Memory Management

MemGPT and similar hierarchical systems page context in and out of the active window based on token consumption. When the context window fills, content is summarized or evicted to archival storage. This approach is mechanistically sound but semantically blind: the decision of what to keep and what to archive is driven by recency and space constraints, not by the topical structure of the conversation. A critical piece of context from an earlier topic may be evicted simply because more recent (but less important) content has consumed the available token budget.

2.3 Scalability of Retrieval Routing

As a user’s conversational history grows over weeks, months, or years, the memory store may contain hundreds of distinct topical threads. Evaluating all stored embeddings on every query introduces latency proportional to the total store size. Existing systems either accept this linear scaling or apply crude heuristics (e.g., recency filters) that sacrifice retrieval recall. What is needed is a routing mechanism that narrows the search space to relevant topical partitions before performing fine-grained retrieval, analogous to how a search engine first identifies relevant index shards before ranking individual documents.

3 Architecture

TAG consists of four primary components: (1) Topic Vector Spaces (TVS), unbounded per-topic embedding stores; (2) the TAG Routing Layer, a metadata index of continuously updated topic summaries; (3) a Router Model, a lightweight LLM or classifier responsible for topic selection; and (4) the Relevancy Engine, a scoring system that manages hot/cold storage tiers based on user interaction signals.

Figure 1: TAG Architecture Overview

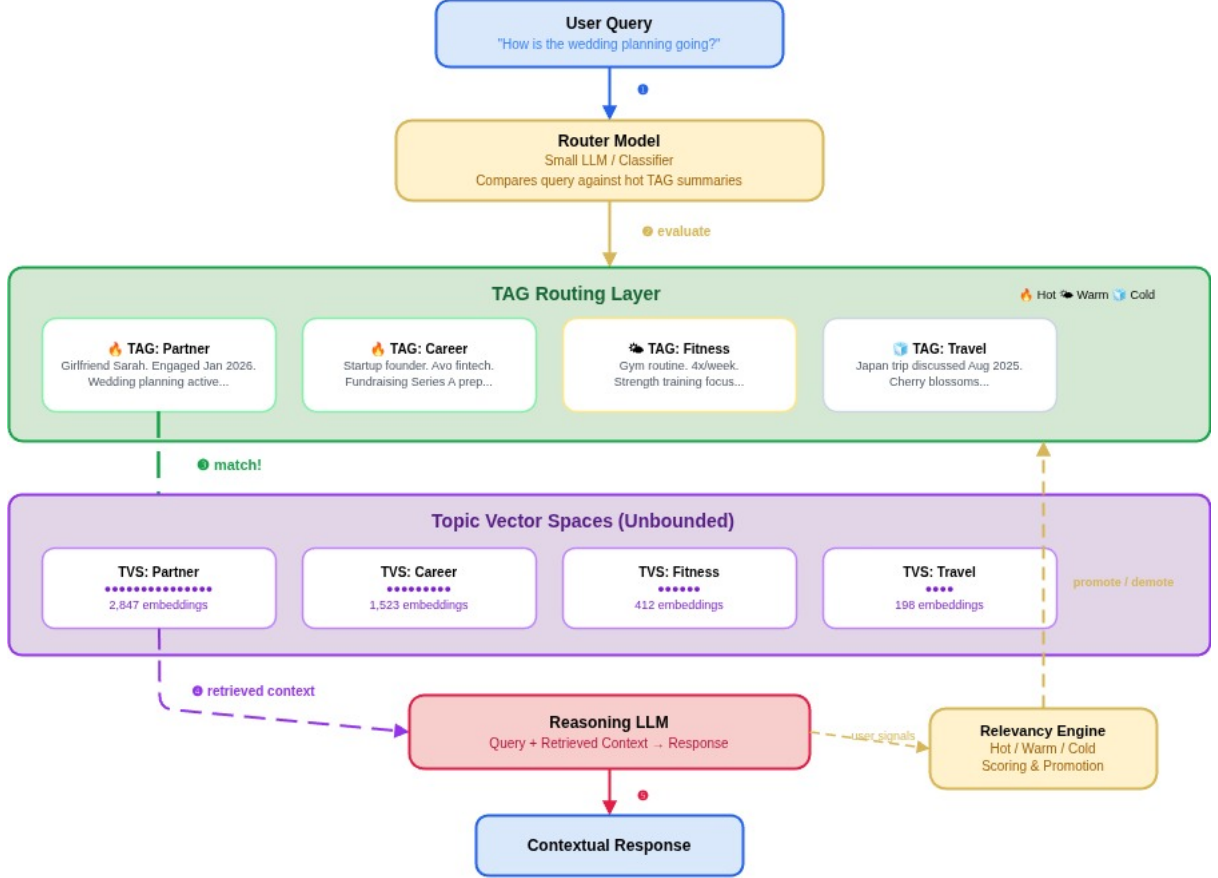


Figure 1: TAG Architecture Overview. User queries are routed through the TAG Routing Layer to the relevant Topic Vector Space. Retrieved context is injected into the reasoning LLM’s prompt. The Relevancy Engine manages hot/warm/cold tiering of TAG summaries.

3.1 Topic Vector Spaces (TVS)

Unlike conventional RAG systems that maintain a single, monolithic vector store, TAG partitions the embedding space into discrete Topic Vector Spaces. Each TVS is an independent vector database (or namespace within a shared database) that stores embeddings exclusively related to a single conversational topic. Critically, TVS are unbounded: they grow as the topic grows, with no predefined token or embedding count limit. A topic vector space for “career planning” may contain embeddings spanning months of conversation, while a vector space for “weekend cooking experiments” may contain a handful of exchanges. The storage scales with the semantic density of the topic, not with an arbitrary budget.

When a new conversational fragment is determined to belong to an existing topic, it is embedded and appended to the corresponding TVS. When no existing topic matches with

sufficient confidence, a new TVS is created, and a corresponding TAG summary document is generated. The decision of whether to create a new TVS or append to an existing one is made by the Router Model (Section 3.3), using the TAG summaries as its classification targets.

3.2 The TAG Routing Layer

The TAG Routing Layer is a lightweight index consisting of one TAG summary document per Topic Vector Space. Each TAG document is not a static label (e.g., “girlfriend”) but a continuously evolving, recency-weighted compact summary of the entire content of its associated TVS. A TAG document for a topic about the user’s partner might read:

“User’s girlfriend Sarah. Together 2.5 years. Recently engaged (January 2026). Previously argued about relocating to Chicago in mid-2025, resolved in favor of staying in Vancouver. She works in healthcare (nursing). Wedding planning is actively underway: venue search, guest list discussions, budget stress. User expresses high emotional investment in this topic.”

This summary serves dual purposes. First, it is the routing target: the Router Model compares the user’s current input against all hot TAG summaries to determine which TVS to query. Second, it is a compressed representation of the TVS content, enabling the Router to make informed decisions without reading the full vector space. The TAG summary is updated incrementally as new content enters the TVS, with a recency bias: more recent conversational turns receive greater emphasis in the summary, while older context is progressively compressed. This mirrors how human memory naturally foregrounds recent events while retaining the gist of older ones.

3.3 The Router Model

A key design decision in TAG is the separation of the reasoning model (the primary LLM generating responses) from the routing model (the classifier selecting which TVS to query). The Router Model is a small, fast, cost-efficient model, either a fine-tuned lightweight LLM (e.g., a 7B parameter model) or a dedicated classifier, whose sole responsibility is topic selection.

At query time, the Router receives the user’s current input and the set of hot TAG summaries (see Section 3.4). It returns a ranked list of TVS identifiers with confidence scores. If the top-ranked TVS exceeds a confidence threshold, the system retrieves context from that TVS via standard semantic search and injects it into the reasoning model’s context window. If no hot TVS exceeds the threshold, the Router falls back to evaluating cold TAG summaries (Section 3.4.3). If multiple TVS are relevant (e.g., a query that spans career and relationship topics), the Router may return multiple TVS identifiers, and context is retrieved from each.

This separation of concerns keeps routing fast and cheap. The Router Model operates on TAG summaries (short documents, typically under 500 tokens each) rather than on the full content of the vector spaces. Even for a power user with hundreds of topics, the Router is evaluating a manageable set of compact summaries, not searching across millions of embeddings.

3.4 The Relevancy Engine

The Relevancy Engine is the mechanism by which TAG manages the scalability of its routing layer. It maintains a tiered storage hierarchy (hot, warm, and cold) for TAG summary documents, and automatically promotes or demotes topics between tiers based on user interaction signals.

Figure 2: Query-Time Routing Flow

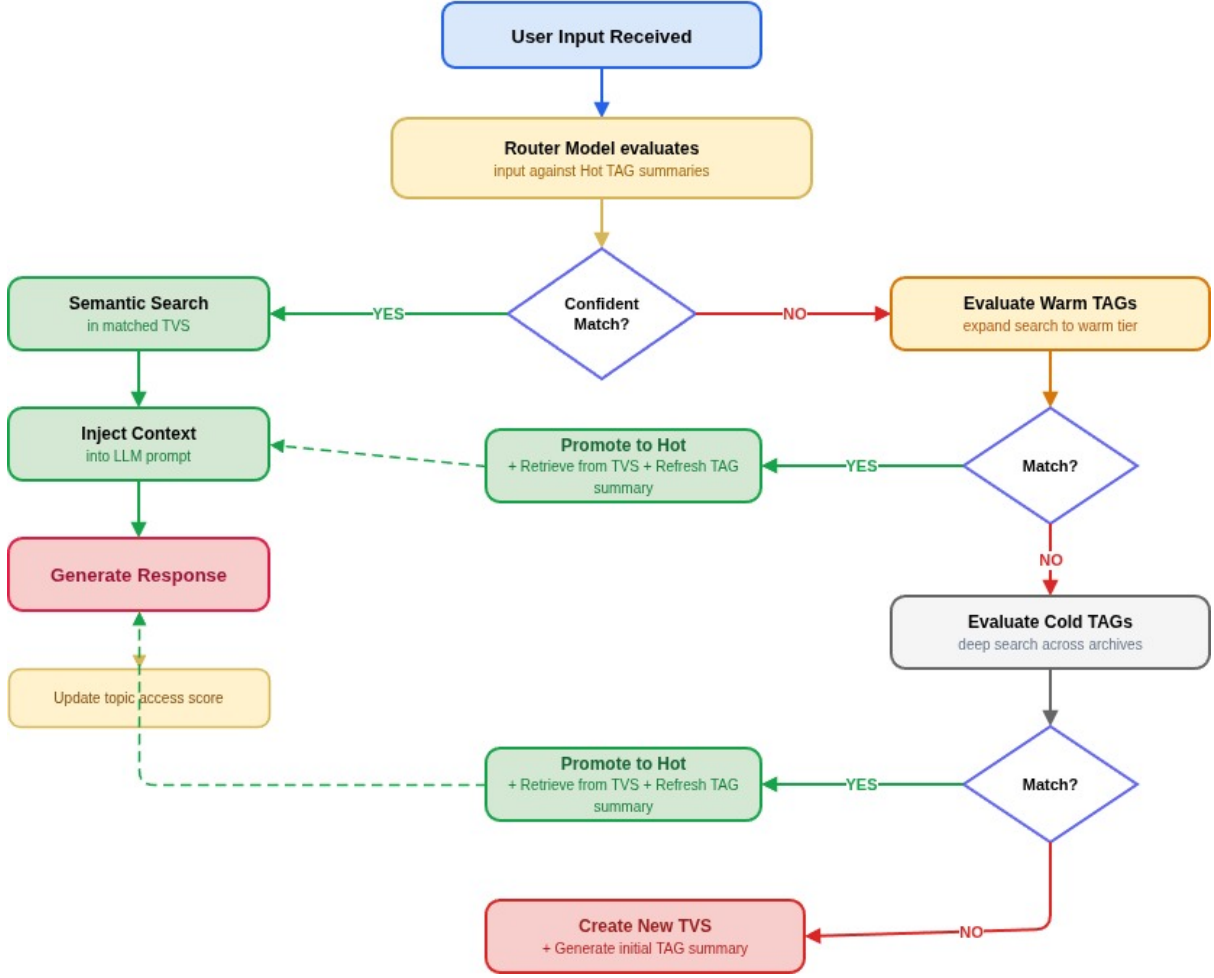


Figure 2: Query-Time Routing Flow. The Router Model evaluates hot TAG summaries first. On a confident match, context is retrieved directly. On failure, the system cascades to warm, then cold tiers. Matches from lower tiers trigger automatic promotion.

3.4.1 Hot/Warm/Cold Tiering

The tiering system operates as follows. Hot topics are those actively referenced in recent conversations. Their TAG summaries are evaluated by the Router on every query. The hot tier is bounded in size (e.g., the 15–25 most active topics) to maintain low routing latency. Warm topics are those not recently referenced but with moderate historical activity. Their TAG summaries are not evaluated by default but are included in the fallback search when no hot topic matches with sufficient confidence. Cold topics are those with low historical activity or extended periods of inactivity. Their TAG summaries are archived and only evaluated on explicit fallback. The underlying TVS data is never deleted; only the routing priority changes.

3.4.2 Promotion and Demotion Signals

The Relevancy Engine draws inspiration from web search ranking algorithms, particularly the insight that user behavior signals are a more reliable indicator of relevance than static content features. We identify two categories of promotion signals.

Implicit signals include frequency of topic reference across sessions, recency of last access, duration of engagement within a topic thread, and cross-referencing (when the user discusses a

Figure 3: Relevancy Engine — Hot/Warm/Cold Tiering

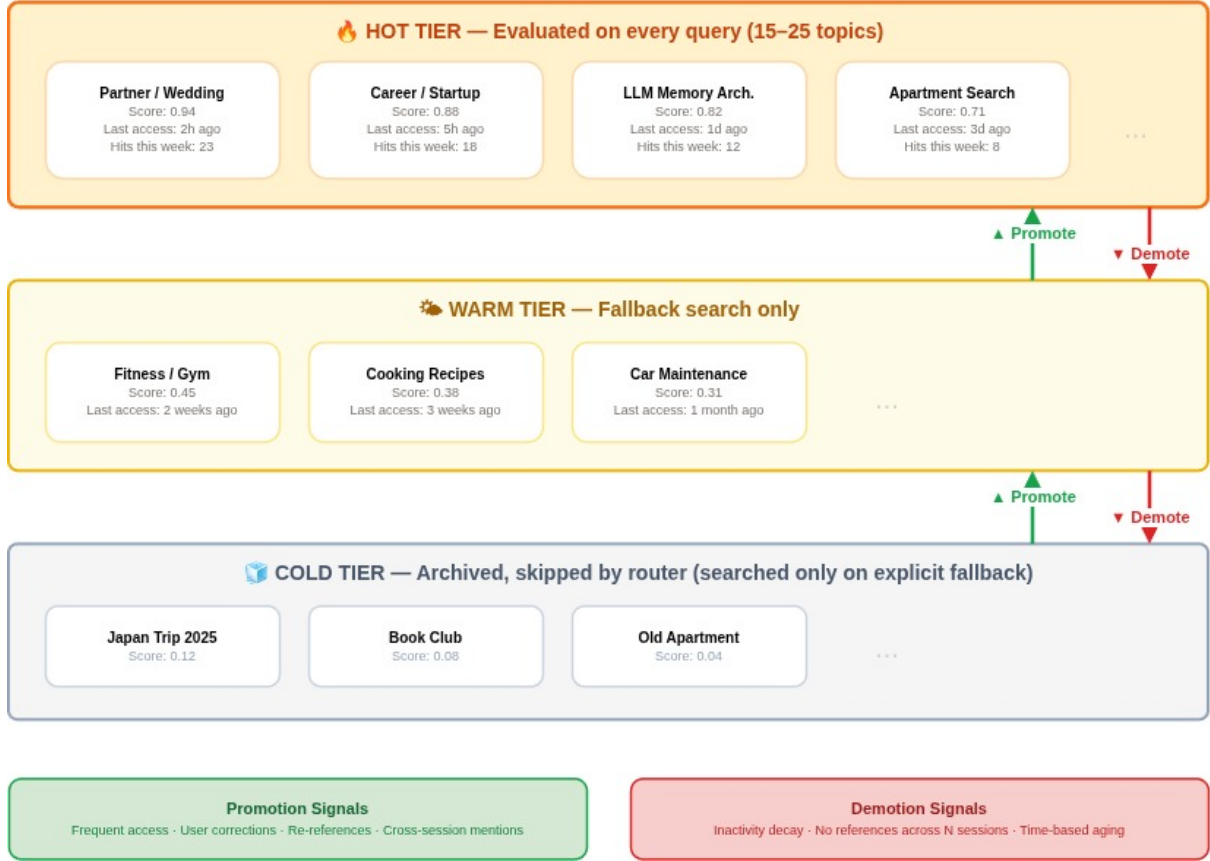


Figure 3: Relevancy Engine Tiering. Topics are scored by access frequency weighted by recency. User correction events trigger immediate promotion. Sustained inactivity causes gradual demotion. The router evaluates only hot-tier summaries by default.

topic without re-explaining background context, implying they expect the system to remember). Implicit signals drive gradual promotion and demotion. A topic discussed daily stays hot; a topic untouched for weeks slowly drifts to warm, then cold.

Explicit correction signals are the most powerful promotion triggers. When a user says “I already told you about this,” “why did you forget?”, “no, I meant X, we discussed this last week,” or otherwise indicates that the system has failed to retrieve expected context, this constitutes a strong signal that the referenced topic should be immediately promoted to the hot tier and its TAG summary should be refreshed. These correction events are analogous to a user clicking on a search result that was ranked too low: they indicate a ranking failure that the system should learn from. Conversely, a sustained conversation where the user never corrects or re-explains is a positive signal that the current routing is accurate and the hot tier composition is appropriate.

3.4.3 Cold Fallback with Live Reassessment

When a user query produces no high-confidence match in the hot tier, the system initiates a cold fallback procedure. The Router evaluates warm and then cold TAG summaries to find the most relevant topic. Once a match is found, the system performs a full semantic search within the corresponding TVS, retrieves the relevant context, and injects it into the reasoning model’s context window. Simultaneously, the Relevancy Engine promotes the matched topic to the hot tier and triggers an update of its TAG summary to reflect the newly accessed context. This

fallback mechanism ensures that no historical context is ever truly inaccessible; it is merely deprioritized in the routing order. The promotion-on-access behavior means that a single reference to a cold topic is sufficient to restore it to active routing, creating a self-healing memory system.

4 TAG Summary Lifecycle

The TAG summary document is the central abstraction in the architecture, and its lifecycle management is critical to system performance. We describe the four phases of a TAG summary’s lifecycle.

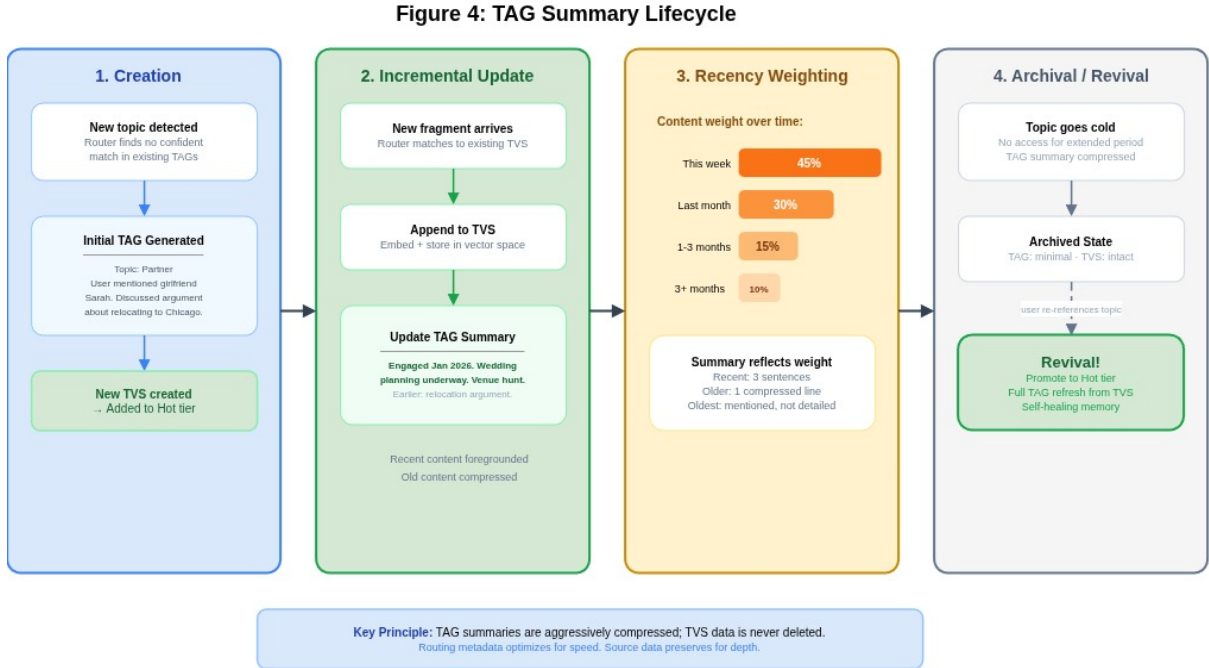


Figure 4: TAG Summary Lifecycle. Summaries are created when new topics are detected, incrementally updated with recency weighting as topics grow, and archived when topics go cold. Archived topics retain full TVS data and can be revived on access.

4.1 Creation

When the Router Model determines that an incoming conversational fragment does not belong to any existing TVS with sufficient confidence, a new TVS is created. The reasoning LLM (or a dedicated summarization model) generates an initial TAG summary from the conversational fragment. This summary captures the topic label, key entities, the user’s apparent intent or emotional valence, and any factual anchors (names, dates, locations, decisions).

4.2 Incremental Update

As new conversational fragments are appended to a TVS, the TAG summary is updated incrementally. The update process is recency-weighted: new information is added at full fidelity, while older content is progressively compressed. The update is performed by a summarization pass that takes the existing TAG summary and the new fragment as input and produces a revised summary. This approach avoids the need to re-read the entire TVS on every update, keeping the update cost proportional to the size of the new fragment rather than the total topic history.

4.3 Recency Weighting

The recency weighting in TAG summaries reflects an empirical observation about conversational patterns: users overwhelmingly reference recent context rather than historical context when continuing a topic. The TAG summary is structured such that the most recent state of a topic occupies the majority of the summary’s content, while older context is retained in compressed form. For example, in the “girlfriend” topic, the summary foregrounds the current wedding planning discussions while retaining the historical context of the relocation argument as a one-line compressed reference. If the user later revisits the relocation topic, the cold fallback mechanism retrieves the full context from the TVS, and the TAG summary is refreshed to re-expand that subtopic.

4.4 Archival and Decay

When a topic enters the cold tier due to sustained inactivity, its TAG summary is archived but not deleted. The summary itself may be further compressed during archival to reduce the evaluation cost if it is ever included in a fallback search. The underlying TVS remains intact and fully searchable. This asymmetry, aggressive compression of routing metadata with full preservation of source data, ensures that the system optimizes for routing speed without sacrificing retrieval depth.

5 Comparison with Existing Architectures

5.1 TAG vs. Standard RAG

Standard RAG treats all stored content as a flat embedding space, performing cosine similarity search across the entire store on every query. TAG partitions this space into topic-scoped regions and routes queries to the relevant partition before performing semantic search. This two-tier approach eliminates context pollution and reduces the effective search space per query, improving both precision and latency.

5.2 TAG vs. MemGPT

MemGPT’s hierarchical memory is inspired by operating system virtual memory, with paging driven by context window pressure. When the active context fills, content is summarized and moved to archival storage. TAG’s hierarchy is driven by semantic coherence and user behavior signals, not by token budget. MemGPT’s archival retrieval requires the LLM to self-direct search operations via function calls; TAG’s routing is handled by a dedicated, lightweight model that operates on precomputed summaries, keeping the reasoning model focused on generation.

5.3 TAG vs. Mem0

Mem0 provides memory scoping at the user, session, and agent level, with an LLM-based fact extraction and deduplication pipeline. TAG operates at a finer granularity, topic-level scoping within a single user’s history, and introduces the TAG summary as a continuously evolving routing index that Mem0 lacks. Mem0’s retrieval is flat semantic search within a user’s memory store; TAG’s retrieval is routed through the summary layer, reducing the effective search space. Additionally, TAG’s Relevancy Engine provides explicit hot/cold tiering with user-signal-driven promotion, whereas Mem0 does not currently implement behavioral relevancy scoring.

5.4 TAG vs. A-Mem (Zettelkasten-Inspired)

A-Mem creates self-organizing atomic notes with dynamic linking, inspired by the Zettelkasten knowledge management method. While A-Mem’s note-linking captures rich associative relationships, it lacks a routing mechanism that scales efficiently as the note count grows. TAG’s summary-based routing provides a coarse-grained index that A-Mem’s flat link traversal does not, enabling faster query resolution at scale. The approaches are potentially complementary: A-Mem’s linking could enhance within-TVS organization in a TAG system.

6 Design Considerations and Open Questions

6.1 Topic Boundary Detection

A critical implementation challenge is determining when a conversational fragment belongs to an existing topic versus when it warrants a new TVS. Multi-topic messages (e.g., “I’m stressed about the wedding budget and it’s also affecting my work performance”) require either multi-tagging or topic splitting. We propose that the Router Model should support multi-routing, where a single message can be appended to multiple TVS and retrieved from any of them. The TAG summaries of related topics can also cross-reference each other, preserving the associative structure without merging the vector spaces.

6.2 TAG Summary Drift

Over hundreds of incremental updates, TAG summaries risk drifting from the actual content of their associated TVS. Periodic full-refresh passes, where the summarization model reads a sample of the TVS content and regenerates the summary from scratch, can mitigate this. The frequency of full-refresh can be tied to the topic’s activity level: hot topics refresh more frequently, cold topics less so.

6.3 Context Window Integration

As context windows grow (current frontier models support 128K–200K tokens, with 1M+ on the horizon), the question arises: does topic-scoped retrieval remain necessary? We argue yes, for two reasons. First, larger windows exacerbate context pollution: more irrelevant content in the window actively degrades model attention and output quality. Second, cost scales linearly with context size; injecting only the relevant topical context keeps inference costs proportional to query complexity, not to total history size.

6.4 Privacy and Data Governance

Topic-scoped storage creates natural boundaries for data governance. Users can inspect, export, or delete individual topic vector spaces without affecting the rest of their conversational history. TAG summaries provide human-readable abstractions of what the system “knows” about each topic, enabling transparent memory auditing. This granularity is difficult to achieve in flat RAG systems where content is interleaved across a single embedding space.

7 Proposed Evaluation Framework

To validate TAG against existing architectures, we propose evaluation across three dimensions.

Retrieval Precision. Using the LOCOMO benchmark [Maharana et al., 2024] and extensions thereof, measure the accuracy of context retrieval across multi-session, multi-topic conversational histories. The key metric is whether the retrieved context is topically aligned

with the query, not merely semantically similar. We expect TAG’s topic-scoped retrieval to significantly outperform flat RAG on cross-topic disambiguation tasks.

Routing Latency. Measure end-to-end retrieval time as a function of total memory store size and number of active topics. TAG’s hot-tier routing should exhibit near-constant latency regardless of total store size, with cold fallback adding predictable latency only when necessary.

User Experience Coherence. In longitudinal user studies, measure the frequency of user correction events (“I already told you this”) as a proxy for memory failures. A well-functioning TAG system should exhibit a decreasing correction rate over time as the Relevancy Engine learns the user’s topical priorities.

8 Conclusion

We have presented Topological Adaptive Graphs (TAG), a memory architecture that rethinks how LLMs maintain persistent conversational context. By partitioning memory into topic-scoped vector spaces, routing queries through continuously updated summary documents, and managing retrieval priority via user-signal-driven relevancy scoring, TAG addresses the fundamental limitations of flat RAG, budget-driven paging, and coarse-grained memory scoping. The architecture is designed to scale gracefully with user history size, maintain high retrieval precision across diverse topical threads, and create the subjective experience of an AI system that genuinely remembers.

The core insight of TAG is that memory management in conversational AI is not primarily a storage problem or a retrieval problem; it is a routing problem. The challenge is not embedding more content or searching more efficiently, but knowing where to look before looking. By solving routing first, TAG transforms the retrieval problem from a global search into a local one, with all the attendant benefits in precision, latency, and cost.

We believe TAG represents a step toward AI systems that maintain the kind of contextual continuity that humans take for granted in their relationships, where every conversation builds on the last, where important context is never lost, and where the system’s understanding of the user deepens over time rather than resetting with each session.

References

- Chhikara, P., Khant, D., Aryan, S., Singh, T., & Yadav, D. (2025). Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory. *arXiv:2504.19413*.
- Maharana, A., Lee, D., Tulyakov, S., Bansal, M., Barbieri, F., & Fang, Y. (2024). Evaluating Very Long-Term Conversational Memory of LLM Agents. *ACL 2024*.
- Packer, C., Fang, V., Patil, S., Lin, K., Wooders, S., & Gonzalez, J. (2023). MemGPT: Towards LLMs as Operating Systems. *arXiv:2310.08560*.
- Qian, H., Liu, Z., Zhang, P., Mao, K., Lian, D., Dou, Z., & Huang, T. (2025). MemoRAG: Boosting Long Context Processing with Global Memory-Enhanced Retrieval Augmentation. *TheWebConf 2025*.
- Zhang, Y., et al. (2025). A-Mem: Agentic Memory for LLM Agents. *NeurIPS 2025*.
- Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank Citation Ranking: Bringing Order to the Web. *Stanford InfoLab*.