International Journal of Engineering and Science Invention (IJESI)
ISSN (Online): 2319-6734, ISSN (Print): 2319-6726
www.ijesi.org ||Volume 15 Issue 8 August 2026 || PP 38-44

HAMM: An Adaptive Memory Management Framework for Long-Term Large Language Model Agents

M. R. Gagan[1] and Manju Sadasivan[2]
[1,2] *School of Science and Computer Studies, CMR University, Karnataka, India*

Abstract: *Long-term large language model (LLM) agents require memory mechanisms that can retain useful information across interactions while avoiding the accumulation of obsolete, redundant, or irrelevant content. Conventional retrieval-augmented generation (RAG) systems generally emphasize semantic similarity during retrieval and may retain memories indefinitely, whereas rigid mechanisms such as sliding windows and fixed time-to-live (TTL) policies can discard information without considering its future utility. This paper proposes HAMM (Human-Inspired Adaptive Memory Management), a hierarchical external-memory framework that dynamically manages memory using an Adaptive Memory Strength Score (AMS). The score integrates intrinsic importance, normalized access frequency, recency, confidence, and user feedback. A time-dependent decay function governs memory persistence, while retrieval reinforces successfully used memories and a threshold-based lifecycle controller moves memories among working, short-term, long-term, archive, and deleted states. A pilot simulation from the original study is retained as an initial proof of concept; it shows that HAMM matched the constructed benchmark's retrieval performance of the permanent-memory baseline while producing a slightly lower mean retrieval latency. However, these pilot results do not establish general superiority. The revised study therefore defines a rigorous evaluation protocol involving multi-session interactions, controlled memory noise, conflicting and changing facts, scalability tests, ablation experiments, repeated trials, and statistical analysis. The proposed framework and methodology provide a stronger basis for studying adaptive memory management in long-term LLM agents.*

Keywords - *Adaptive Memory; Explainable AI; Human-Inspired AI; Large Language Models; Memory Reinforcement; Retrieval-Augmented Generation; Selective Forgetting; Long-Term Agents.*

Date of Submission: 11-08-2026 Date of acceptance: 24-08-2026

I. Introduction

Large language models have evolved from single-turn text-generation systems into agents capable of maintaining state, using external tools [4], planning over multiple steps [5], and interacting with users over extended periods. In such systems, memory becomes a fundamental architectural component because information unavailable in the current context window must be stored and subsequently retrieved. Recent surveys describe agent memory as a rapidly developing research area spanning storage, retrieval, updating, consolidation, and long-term experience.

Retrieval-augmented generation provides a practical mechanism for externalizing information from the model into a retrievable store. Nevertheless, a permanent vector store creates a different problem: the system may accumulate large numbers of historical interactions without an explicit mechanism for determining which memories should persist, which should be strengthened, and which should become inactive. As the memory collection grows, irrelevant or outdated records can compete with useful information during retrieval. Conversely, rigid deletion strategies can remove information that remains important beyond a predefined time window.

Hierarchical memory architectures have demonstrated that external memory can be organized into different levels according to access and persistence requirements. MemGPT, for example, introduced a virtual-context approach that manages information between limited working context and external storage. More recent research has broadened the problem from storage to dynamic memory evolution and has emphasized accurate retrieval, test-time learning, long-range understanding, and selective forgetting as distinct competencies of memory agents. Furthermore, providing agents with adaptive memory banks tuned to long-term interactions has shown to heighten empathy and the model's ability to seamlessly adapt to a user's evolving context [10].

This study proposes HAMM, a Human-Inspired Adaptive Memory Management framework designed to address this problem. HAMM assigns each memory item a dynamic strength value based on importance, access frequency, recency, confidence, and user feedback. The strength influences retrieval priority and memory lifecycle decisions. Frequently accessed and high-value information can be reinforced, while information with low continuing utility can decay and eventually move to archive or deletion states.

The principal contributions are: (1) a formal adaptive memory-strength model for external memory in LLM agents; (2) a hierarchical memory lifecycle covering working, short-term, long-term, archive, and deleted states; (3) a reinforcement and selective-forgetting mechanism; (4) an explicit retrieval algorithm combining semantic similarity with memory strength; and (5) a rigorous evaluation protocol for retrieval quality, forgetting behavior, storage efficiency, latency, scalability, and component-level contributions.

II. Background and Related Work

A. Retrieval-Augmented Generation and External Memory

RAG combines a language model with an external retrieval mechanism so that relevant information can be supplied at inference time. Its principal advantage is that information can be updated without retraining the underlying language model. However, conventional RAG generally treats stored records primarily as retrieval candidates and does not inherently define a complete lifecycle for memory formation, reinforcement, decay, and deletion. Recent investigations into prompt structuring reveal that explicitly defining intermediate reasoning steps—often requiring intermediate memory retrieval—drastically enhances complex problem-solving [5].

B. Memory in LLM-Based Agents

A systematic survey of LLM-agent memory identifies memory as a core mechanism supporting self-evolving interaction and reviews methods for memory design and evaluation [9]. A newer survey argues that contemporary agent memory cannot be described adequately using only short-term and long-term labels; memory should also be considered in terms of forms, functions, and dynamics, including how memories are formed, evolved, and retrieved over time [2]. Extending these capabilities into specialized domains, such as clinical and prognostic predictions, further underscores the necessity of robust, few-shot memory retention [11].

C. Hierarchical Memory

MemGPT established a hierarchical or virtual-memory perspective in which the model manages information between a limited context and external memory [3]. HAMM shares the principle that not all information should occupy the same memory level, but extends it with an explicit quantitative memory-strength model. In HAMM, hierarchy is a consequence of a memory lifecycle controller that evaluates the current survival value of each memory. Organizing working memory hierarchically allows agents to treat subgoals as distinct memory chunks, heavily reducing computational load and context bloat [1].

D. Memory Reinforcement and Selective Forgetting

A long-term agent must retain important information without allowing obsolete content to dominate retrieval. Reinforcement and forgetting are therefore complementary operations. Recent benchmarking work explicitly identifies selective forgetting as one of four core competencies of memory agents, alongside accurate retrieval, test-time learning, and long-range understanding [1]. This supports the central premise of HAMM that forgetting should be treated as an intentional memory-management operation.

E. Research Gap

Table I: Comparison of Memory Management Capabilities

Capability	Permanent RAG	Sliding Window	Fixed TTL	Hierarchical Memory	HAMM
Semantic retrieval	Yes	Yes	Yes	Yes	Yes
Importance-aware retention	No	No	No	Partial	Yes
Adaptive temporal decay	No	No	No	Partial	Yes
Access-based reinforcement	No	No	No	Partial	Yes
Selective forgetting	No	Implicit	Time-based	Partial	Yes
Explicit lifecycle controller	No	No	No	Partial	Yes

The contribution is therefore not the introduction of external memory or hierarchy alone; rather, HAMM combines a quantitative memory-strength function with reinforcement, adaptive decay, and explicit lifecycle transitions.

III. Research Questions and Hypotheses

RQ1: How does adaptive memory-strength scoring affect retrieval accuracy compared with static memory-management strategies?

RQ2: To what extent does selective forgetting reduce retrieval of irrelevant, obsolete, or conflicting memories as memory volume increases?

RQ3: How does memory reinforcement affect retention of frequently accessed and high-importance information?
RQ4: What is the effect of HAMM on retrieval latency and active memory footprint under increasing memory loads?
RQ5: Which HAMM components contribute most to retrieval quality, retention, and storage efficiency?
Hypotheses:

- H1: HAMM achieves higher retrieval F1 than static memory-management baselines under long-session conditions.
- H2: HAMM produces a lower active memory footprint than permanent-memory RAG as session length and memory volume increase.
- H3: HAMM reduces retrieval of irrelevant or obsolete memories compared with permanent-memory RAG.
- H4: Reinforcement improves retention of high-importance memories without proportionally increasing active storage.
- H5: Adaptive decay provides a better accuracy-storage trade-off than fixed TTL deletion.

IV. Proposed HAMM Framework

A. Design Principles

HAMM is based on five principles: memory should have a measurable strength; memory strength should evolve over time; successful retrieval should reinforce useful memories; memory hierarchy should reflect persistence and access requirements; and forgetting should be selective and importance-aware.

B. System Architecture

The architecture contains an input and encoding layer, a Memory Controller, a hierarchical storage subsystem, and a retrieval layer. The storage hierarchy consists of Working Memory, Short-Term Memory (STM), Long-Term Memory (LTM), Archive Memory, and Deleted Memory. Working Memory corresponds to the immediate model context. STM stores recent candidate memories for rapid access. LTM contains reinforced and high-value memories. Archive stores inactive memories outside the primary retrieval path. Deleted Memory represents records that have crossed the forgetting threshold and are permanently removed, subject to the application's data-retention policy.

Interaction → Memory Encoding → Importance Assessment → STM → Reinforcement/Decay → LTM or Archive → Selective Deletion

C. Adaptive Memory Strength

For a memory item m, the Adaptive Memory Strength Score is:

$$S_m(t) = \alpha I_m + \beta F_m + \gamma R_m(t) + \delta C_m + \varepsilon U_m$$

where I denotes intrinsic importance, F denotes normalized access frequency, R(t) denotes recency, C denotes confidence, U denotes user feedback, and α, β, γ, δ, and ε are non-negative weights satisfying $\alpha + \beta + \gamma + \delta + \varepsilon = 1$. All component variables should be normalized to [0,1].

D. Adaptive Recency and Decay

Recency is represented by:

$$R_m(t) = exp[-\lambda_m(t - t\_last,m)]$$

where t_last,m is the most recent successful access time and λ_m is the memory-specific decay constant. The initial implementation uses:

$$\lambda_m = k / [I_m \times max(C_m, 0.1)]$$

The constant k controls the global decay scale. In the final experiments, k and the component weights should be calibrated on a development set rather than selected only from a short pilot simulation.

E. Memory Lifecycle

The controller maps memory strength to lifecycle states using thresholds τL, τA, and τD:

- $S_m(t) \geq \tau L$: retain or promote to Long-Term Memory.
- $\tau A \leq S_m(t) < \tau L$: retain in Short-Term Memory.
- $\tau D \leq S_m(t) < \tau A$: move to Archive Memory and remove from active retrieval.
- $S_m(t) < \tau D$: delete the memory according to the application's retention policy.

F. Memory Reinforcement

When a memory contributes to a retrieval event, its access count and last-accessed timestamp are updated:

$$F\_new = F\_old + 1$$
$$t\_last = current\_time$$

For large-scale deployments, raw frequency should be normalized or transformed logarithmically so repeated accesses do not cause unbounded growth.

G. Retrieval Algorithm

- Encode the query into a dense vector.
- Search STM and LTM for candidate memories.
- Compute cosine similarity for each candidate.
- Compute the current AMS for each candidate.
- Combine semantic similarity and AMS.
- Return candidates exceeding the retrieval threshold.
- Reinforce successfully used memories.
- Re-evaluate lifecycle state after reinforcement.

$$Score(q,m) = CosineSim(q,m) \times S_m(t)$$

H. Computational Considerations

If N candidates are returned by vector search, AMS evaluation is O(N). Ranking is O(N log N) when sorting is required. End-to-end latency should include embedding, vector search, memory scoring, ranking, and controller updates.

V. Experimental Methodology

A. Implementation

The original implementation was developed in Python using all-MiniLM-L6-v2 through sentence-transformers and cosine similarity through scikit-learn. Custom Python components simulate the Memory Controller, temporal decay, and baseline memory policies.

B. Benchmark Design

The original pilot benchmark contained high-importance facts such as user profession and preferences, together with low-importance facts such as weather and meal choices. The revised evaluation should extend this benchmark into multi-session interactions containing stable facts, temporary facts, changing facts, conflicting facts, repeated facts, and irrelevant conversational noise.

Table II: Evaluation Benchmark Scenarios and Purposes

Scenario	Purpose	Example
Stable fact	Test long-term retention	User profession
Temporary fact	Test decay	Current travel plan
Changed fact	Test memory updating	Preferred tool changes
Conflicting fact	Test conflict handling	Old vs. new preference
Repeated fact	Test reinforcement	Important preference repeated
Irrelevant noise	Test memory pollution	Weather or casual conversation
Long-range dependency	Test delayed retrieval	Fact needed many sessions later

C. Baselines

The original study compared HAMM with Vanilla RAG, Sliding Window (size 3), and Fixed TTL (1.5 seconds). These controls are retained. For a full journal evaluation, a hierarchical-memory baseline such as a MemGPT-style architecture should also be included because hierarchical external memory is an established direction [3].

- Vanilla RAG: permanent external memory.
- Sliding Window: retain only the most recent fixed number of interactions.
- Fixed TTL: remove memories after a fixed time interval.
- Hierarchical-memory baseline: separate active context and external memory without HAMM's adaptive strength formulation.
- HAMM: adaptive strength, reinforcement, decay, hierarchy, and selective forgetting.

D. Evaluation Metrics

- Precision and Recall.
- F1 score.
- Irrelevant retrieval rate.
- Conflict retrieval rate.
- Retention accuracy.
- Forgetting accuracy.
- Active storage footprint.
- End-to-end retrieval latency.

- Scalability as memory volume increases.

E. Experimental Protocol

Each configuration should be evaluated over multiple random seeds and repeated trials. Report mean, standard deviation, and 95% confidence intervals. Use a development set for threshold and weight calibration and a held-out test set for final reporting. Statistical comparisons should use an appropriate paired test, such as the Wilcoxon signed-rank test when distributional assumptions are not satisfied, together with effect sizes.

F. Ablation Study

- HAMM − Importance
- HAMM − Frequency
- HAMM − Recency
- HAMM − Confidence
- HAMM − User Feedback
- HAMM − Reinforcement
- HAMM with fixed rather than adaptive decay
- HAMM without selective forgetting

G. Scalability Study

Memory volume should be increased systematically, for example from 100 to 500, 1,000, 5,000, 10,000, and 50,000 memory items where computationally feasible. For each scale, measure F1, irrelevant retrieval rate, active footprint, and end-to-end latency.

VI. Pilot Experimental Results

The following results are retained from the original implementation as a pilot proof-of-concept experiment. They should not be interpreted as definitive evidence of general superiority because the benchmark is small and synthetic and the decay window is short. A full journal submission should replace or substantially extend this table with repeated multi-session experiments and statistical analysis.

Table III: Pilot Simulation Retrieval Performance Metrics

Model	Precision (%)	Recall (%)	F1	Hallucination Rate (%)	Storage Footprint (%)	Avg. Latency (ms)
Vanilla RAG	100.0	100.0	1.00	0.0	100.0	18.2
Sliding Window	0.0	0.0	0.00	100.0	50.0	19.0
Fixed TTL	0.0	0.0	0.00	100.0	0.0	14.0
HAMM	100.0	100.0	1.00	0.0	100.0*	17.6

The original pilot used a 2.0-second temporal window. The low-priority memory did not cross the deep-archive threshold of 0.30, so the active footprint remained 100%. This demonstrates that decay parameters require calibration and that a short synthetic time scale cannot establish long-term storage efficiency.

The pilot shows that HAMM can match the constructed benchmark's retrieval performance while producing a small latency reduction relative to permanent-memory RAG. However, because both systems achieve perfect precision and recall, the pilot does not establish an accuracy advantage. The 17.6 ms versus 18.2 ms latency difference should not be described as statistically significant without repeated measurements and statistical testing.

VII. Discussion

A. Interpretation of the Pilot Results

The pilot demonstrates feasibility of combining semantic retrieval with adaptive memory strength. It also exposes a methodological limitation: the benchmark and time scale are too limited to distinguish HAMM's selective forgetting capability from permanent retention. The 100% active footprint indicates that the lifecycle controller was not sufficiently stressed. The principal scientific value of the revised study should therefore come from long-session, noisy, conflicting, and scalability experiments.

B. Relationship to Existing Memory Architectures

HAMM is related to hierarchical memory approaches because it separates immediate context from external stores, but it places additional emphasis on a quantitative memory-strength variable and explicit lifecycle transitions. MemGPT established the importance of managing information between constrained context and external memory [3]. Recent evaluation work argues that memory-agent evaluation must include selective forgetting and long-range interaction rather than only static retrieval [1].

C. Accuracy-Storage Trade-off

The central objective is not to retain every memory forever, but to make memory persistence proportional to expected future utility. Therefore, evaluation should examine the trade-off between retrieval quality and active storage. A successful HAMM configuration should preserve important memories while reducing exposure to obsolete and irrelevant records.

D. Explainability

HAMM provides an interpretable basis for memory decisions. The controller can expose importance, frequency, recency, confidence, user feedback, and the resulting lifecycle state. This makes it possible to audit why a memory was retained, reinforced, archived, or deleted.

VIII. Threats to Validity and Limitations

- The current pilot benchmark is synthetic and small; perfect retrieval scores should not be generalized to real-world LLM agents.
- Decay constants and lifecycle thresholds are manually specified in the current implementation and require systematic calibration.
- The current implementation uses a single embedding model; different embedding models may alter retrieval behavior.
- The pilot latency comparison lacks repeated-run statistical testing.
- The framework evaluates external-memory forgetting rather than parameter-level continual learning; the final manuscript should therefore avoid using catastrophic forgetting as a direct synonym for external-memory deletion.
- User feedback and confidence require operational definitions and reliable measurement in real deployments.
- Deletion policies have privacy and governance implications and should follow application-specific data-retention requirements.

IX. Future Work

Future work should investigate learnable decay parameters, adaptive weighting of AMS components, conflict-aware memory updating, multimodal memories, and memory consolidation across trajectories. Evaluation should also be extended to contemporary multi-turn memory benchmarks such as MemoryAgentBench, which explicitly evaluates accurate retrieval, test-time learning, long-range understanding, and selective forgetting [1]. A further direction is to compare HAMM with modern experience-oriented memory systems and to learn reinforcement signals automatically rather than specifying them heuristically.

X. Conclusion

This paper presented HAMM, an adaptive memory management framework for long-term LLM agents. HAMM models memory as a dynamic lifecycle rather than a permanent collection of vectors. Its Adaptive Memory Strength Score integrates importance, frequency, recency, confidence, and user feedback, while reinforcement and adaptive decay determine whether memories remain active, move to long-term storage, enter an archive, or are deleted. The pilot implementation demonstrates feasibility but does not, by itself, justify claims of universal superiority. The revised methodology addresses this limitation through multi-session evaluation, controlled noise, conflicting and changing facts, scalability testing, ablation analysis, repeated trials, and statistical validation. These elements provide a stronger foundation for evaluating whether adaptive selective memory management can improve the reliability and efficiency of long-term LLM agents.

References

[1] Hu, Y., Wang, Y., & McAuley, J. (2025). Evaluating Memory in LLM Agents via Incremental Multi-Turn Interactions. *arXiv*.
[2] Luo, J., Tian, Y., Cao, C., Luo, Z., Lin, H., Li, K., Kong, C., Yang, R., Ma, J., et al. (2024). From Storage to Experience: A Survey on the Evolution of LLM Agent Memory Mechanisms. *arXiv*.
[3] Packer, C., Fang, V., Patil, S., Lin, K., Wooders, S., & Khattab, D. E. P. (2023). MemGPT: Towards LLMs as Operating Systems. *arXiv*.
[4] Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., & Scialom, T. (2023). Toolformer: Language Models Can Teach Themselves to Use Tools. *arXiv*.
[5] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv*.
[6] Xu, Z., Jiang, P., Gao, Y., & You, J. (2025). Cloud-Base Retrieval Augmented Language Models in Multi-Hop QA Task: A Meta-Analysis. *Proc. ICAACE*, 1498–1502.
[7] Yang, H. Y., Chen, B. S., Li, Y. J., & Li, C. Y. (2025). A Multi-Strategy Retrieval-Augmented Generation Approach for Enhancing Network Management. *Proc. APNOMS*, 1–4.
[8] Zhang, W., & Zhang, J. (2025). Hallucination Mitigation for Retrieval-Augmented Large Language Models: A Review. *Mathematics*, 13(5), 856.
[9] Zhang, Z., Bo, X., Ma, C., Li, R., Chen, X., Dai, Q., Zhu, J., Dong, Z., & Wen, J.-R. (2024). A Survey on the Memory Mechanism of Large Language Model based Agents. *arXiv*.

[10] Zhong, W., Guo, L., Gao, Q., Ye, H., & Wang, Y. (2024). MemoryBank: Enhancing Large Language Models with Long-Term Memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, *38*(17), 19724-19731.
[11] Chen, Z., Balan, M. M., & Brown, K. (2023). Language Models are Few-shot Learners for Prognostic Prediction. *arXiv*.