© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718160

M.A.K.S: Multidimensional Access Knowledge Scoring for Long-Horizon LLM Agent Memory Management

SAHIL MEHRAJ[1], ABDUL KAFEEL[2], SHEIKH MUSA[3]

Abstract- ***AI agents with long horizons suffer from fixed context window sizes that necessitate memory evictions over time. Current techniques such as FIFO, LRU, and attention-based evictions use a binary approach to manage memory by either retaining or irreversibly deleting data. No current system preserves its evicted memories for later recovery, nor do any of the systems use multiple criteria to determine the value of memories. M.A.K.S., which stands for Multidimensional Access Knowledge Scoring, is a memory management technique designed specifically for LLM agent systems. M.A.K.S uses a continuous memory lifecycle consisting of degradation and revivals to address the issue. Each memory has an associated priority score denoted by S(t) that takes into account several factors including temporal degradation, access frequency, centrality, Shannon Entropy, and spaced reinforcement. To assess M.A.K.S, we conducted three experiments. Through the experiment called the Needle in a Compressed Haystack, we show that M.A.K.S was able to successfully reconsolidate a very important memory fact with a token usage ratio of 4.58× where FIFO memory evictions failed completely. The ablation experiment corroborates the inclusion of the Ghost Zone and reconsolidation pipeline in architectural support structures. Overhead benchmarking proves that lazy evaluation improves sweep performance up to 9.25× faster with 5,000 memory units, ensuring that scores remain less than 1% of LLM inference latency. M.A.K.S shows that memory management of AI agents is a first-class systems issue involving scoreable degradation, cold storage, and cue-based revival – not simple eviction.***

I. INTRODUCTION

The application of large language models as agents that hold conversations over a long horizon has revealed a central limitation in architecture: the context window. Each transformer-based architecture functions with an immutable ceiling for the number of tokens that may be processed simultaneously. With conversations spanning through complicated tasks, extensive documents, and multiple session deployments, the amount of information that requires processing in the active window increases without bound, while the size of the window stays fixed. Under standard configuration parameters, the older pieces of information need to be evicted to make way for incoming tokens. The result is that modern AI agent systems function as if amnesic.

Information learned at the start of a lengthy conversation is frequently lost at the end of the conversation – not because it becomes obsolete semantically, but rather simply because of a simple eviction strategy. This is not a minor problem in practice. In customer service workflows in production, coding assistants working on multiple files at once, legal research tools, and autonomous agents that run over a prolonged period, failing to remember context degrades the ability to perform the task well, increases hallucinations, and destroys user confidence.

Approaches that have been tried out before can be classified into two different approaches – none of which has proven to be adequate enough. Token-based eviction mechanisms like H2O Zhang et al. arXiv:2306.14048 and StreamingLLM Xiao et al.

arXiv:2309.17453 function during inference passes on an unstructured basis using the token index. Such models use attention-based cumulative weights or static recency intervals for determining what memory content to prune off. They have the advantage of efficiency but their limitations include inability to use structured semantic tokens, a singular eviction condition that cannot be reversed and lack of any kind of hierarchical structure. Models such as MemGPT Packer et al. arXiv:2310.08560 and Infini-transformer Munkhdalai et al. arXiv:2404.07143 increase the size of the memory space through hierarchical storage or compression but fail to allocate an importance score and a mechanism for recovering evicted memory content. Multi-dimensional memory management in Generative

Agents Park et al. arXiv:2304.03442 relies on a prompt-based subjective scoring system that uses LLM to estimate the importance of memories and conducts a binary search without loss in data integrity or cold storage.

The common problem with all existing techniques lies in that the same issue persists across all of them. The memory process can be either one or another – content either exists in the contextual space with complete precision or disappears altogether. There is no concept of gradual degradation or a transparent method of reviving content once it is forgotten.

However, there is a contrasting paradigm in cognitive science. Human memory does not apply hard deletion – it operates via continuous degradation of memories. From a high-resolution memory trace with declarative content down to its abstract essence and, finally, to a dormant trace – memory content undergoes changes but never actually gets deleted. Once exposed to appropriate stimuli, the dormant memory trace will revive into the active memory trace.

The concept was described by Ebbinghaus in 1885 (and subsequently reviewed by Averell and Heathcote in 2011, Journal of Mathematical Psychology) and explained through neuroscience by trace reactivation theories proposed by Nader et al. in 2000 (Nature) and Schacter et al. in 2012 (Nature Reviews Neuroscience). This approach makes biological memory able to store massive amounts of information over extended periods at varying fidelities while being able to recover precise memory traces on demand.

M.A.K.S. stands for Multi-dimensional Access Knowledge Scoring – a memory management algorithm implemented in LLM agent architectures using this premise. M.A.K.S. substitutes the coarse-grained binary eviction process with a gradual, scoring-based degradation process. Every memory component is considered as a structured object, with its scalar retention priority S(t) calculated based on five dimensions: temporal decline, frequency of access, graph centrality, Shannon entropy by bytes, and spaced reinforcement history. Depending on its score S(t), a memory gets classified into one of the three fidelity levels: FULL, PARTIAL, or GHOST, switching between the levels throughout its lifecycle. GHOST memories are stored in the secondary cold store in compressed format. Prior to each LLM invocation, a fully automated keyword-based routing procedure searches the Ghost Store for matches against the query content; all the matched traces get reconsolidated back to PARTIAL fidelity active memory and receive an additional spaced reinforcement prior to the compilation of the context window sent to the model.

Contributions

This work contributes to the problem of managing agent memory across long horizons in the following ways:

1. Survival scoring function S(t). We define an n-dimensional retention prioritization function normalized statically against known theoretical upper bounds, which ensures consistency of scores regardless of conversational scale, along with a recursive reinforcement formula that cuts update time from O(h) history passes to O (1) constant time updates (Section 4.2).
2. Three-stage fidelity architecture with Ghost Store. We develop and implement a range of continuously varying levels of trace compression—FULL, PARTIAL, and GHOST—that replace hard cutoffs with progressive and reversible loss of trace fidelity. The architecture also includes a Ghost Store and a cue-triggered reconsolidation cycle that restores compressed traces before LLM queries with no need for user interaction or model functions (Sections 4.3 and 4.4).
3. Validation in memory bottleneck scenario. We show through the "Needle in a Compressed Haystack" test that M.A.K.Scan reconsolidate the target fact under memory bottleneck stress (token consumption ratio: 4.58x, memory pressure: CRITICAL), while FIFO eviction produces total retrieval failure (Section 6).
4. Analysis of Ablation and Overhead. Through the ablation experiment, we prove that Ghost Zone and reconsolidation are critical parts of our model, and through benchmarks, we show that lazy evaluation improves latency by up to 9.25 times at 5,000 memory units, with M.A.K. S's

overhead being less than 1% compared to LLMs inference latency on similar infrastructure (Section 6).

We explicitly state four constraints of the current model: byte-wise entropy as a proxy for semantic uniqueness, lack of quantification for hyperparameter sensitivity over the entire parameter space, middleware layer design versus natively integrated KV-cache, and keyword-searchable Ghost Store with susceptibility to false positive and false negative results. They are addressed in Section 4.6 and Section 8.

The rest of this paper is structured as follows. Section 3 provides a review of literature concerning KV-cache eviction, compressive memory, agent memory systems, and the cognitive science principles behind M.A.K.S. Section 4 covers the complete mathematical formulation of the M.A.K.S. algorithm. Section 5 outlines the prototype implementation. Section 6 reports experimental results including ablation studies and overhead measurements. Section 7 offers observations. Section 8 tackles limitations. Section 9 offers future directions. Section 10 concludes.

II. RELATED WORK

2.1 KV Cache Eviction Methods

Perhaps the most direct line of related work studies memory pressure at the Key-Value (KV) cache level – that is, the per-token matrix stores maintained by transformers for efficient inference without re-calculating attention weights across historic tokens.

H2O (Zhang et al., arXiv:2306.14048, 2023) proposed the notion of Heavy Hitter Oracle eviction based on retaining a small percentage of tokens that possess the largest sum of attention scores across past transformer layers. H2O proved that heavy hitters contain most of the attention mass, such that other non-heavy-hitter tokens may be safely evicted, thus ensuring bounded memory consumption. M.A.K.S shares the fundamental idea that context is heterogeneous in utility; however, M.A.K.S differs from the prior art along many key dimensions. To start with, H2O works intra-inference with raw token tensors in the course of a single inference process; M.A.K.S works extra-inference with structured Memory Units spanning the entirety of a multi-turn conversational session. Furthermore, H2O retains tokens whose sum of attention scores is largest, which serves as a crude proxy for frequency of access, without any form of temporal decay, memory interconnection, or entropy weighting; M.A.K.S provides a more holistic evaluation metric for context. Lastly, H2O applies token eviction with no recourse; M.A.K.S provides a degradation spectrum for preserving compressed memory traces.

By employing a stable attention sink at the beginning of a sequence with a stringent sliding window of most recent tokens, StreamingLLM (Xiao et al., arXiv:2309.17453, 2023) solves the problem of infinite-length inference. However, while this method keeps the attention maps steady during streaming generation, it treats all windowed information equally, preserving everything based on only recency. M.A.K.S eliminates the recency-based limit in this solution by using the more complex survival score S(t) that guarantees the safety of the semantically important or well-connected historical information regardless of being outside the window.

2.2 Compressive and Hierarchical Memory

Infini-transformer (Munkhdalai et al., arXiv:2404.07143, 2024) proposes infini-attention, which is a transformer attention algorithm with an additional compressive memory matrix in the hidden states updated by a linear recurrence. Instead of being cleared from memory like old context in other models, the historical information gets compressed into a fixed-size matrix, offering theoretical unlimited capacity at a limited memory budget. This is a differentiable and architecture-level approach modifying the way the model computes internally. In contrast, M.A.K.S works as a discrete and model-independent middleware layer that does not require any modifications to the architecture, weights, or fine-tuning. Although M.A.K.S does not work directly with attention, it receives universal deployment capabilities — works with any black-box LLM.

MemGPT (Packer et al., arXiv:2310.08560, 2023) proposes a hierarchical framework motivated by OS-based virtual memory, featuring a primary context window as RAM and the secondary archive as disk.

Memory handling is agentive, using the LLM's calls to transfer data between levels. M.A.K.S differs from MemGPT in two operational aspects. First, automation: MemGPT depends on the LLM's instructions for the transfer process, delaying time and failing when the LLM cannot reliably call the tools. M.A.K.S manages memory decay and consolidation deterministically via algorithms behind-the-scenes without LLM-level intervention. Second, granularity: MemGPT assumes memories as binary memories in flat layers without any survival criteria. In contrast, M.A.K.S gives a continuous scalar value, S(t), for each Memory Unit and controls fidelity using quantifiable zone criteria.

2.3 Agent Memory and Scoring Systems

The Generative Agents architecture (Park et al., arXiv:2304.03442, 2023) illustrated the ability of LLM agents to exhibit consistent behavior over extended periods via scoring stored memories in three aspects: recency (exponential decay with respect to time), importance (judged by the LLM on a scale from 1 to 10), and relevancy (cosine similarity of memory and query representations). High-ranking memories are then accessed and fed back into the prompt environment prior to executing any agent action.

Generative Agents is the most similar conceptual precursor of M.A.K.S's scoring system. The M.A.K.S system enhances and codifies this model in three distinct ways. First, M.A.K.S substitutes subjective, LLM-prompted importance scores for objective mathematical survival functions with normalized Shannon entropy H_norm and degree centrality C measures – metrics independent of model opinion. Second, rather than binary retrieval which discards non-retrieved information forever, M.A.K.S introduces a degradation process whereby memories are degraded stepwise across FULL, PARTIAL, and GHOST states until being processed at the reactive GHOST store level. Third, M.A.K.S implements a reconsolidation process which reintroduces selected degraded memory traces back to active memory upon matching the cue – a feature Generative Agents cannot offer.

2.4 Spaced Repetition and Models of Memory Decay

The psychological theory underpinning M.A.K.S is derived from two well-known bodies of work within cognitive science literature. The forgetting curve was identified by Ebbinghaus in 1885, and describes an exponential decay model of the rate at which humans forget information absent any reinforcement or review, described mathematically in recent work by Averell & Heathcote (2011, Journal of Mathematical Psychology). It is the forgetting curve which informs the temporal e^(−λt) decay component of M.A.K.S. Spaced Repetition Systems (Leitner, 1972; Wozniak & Gorzelanczyk, 1994, Acta Neurobiologiae Experimentalis) showed how strategic repetition could be used to reset the forgetting clock more efficiently than mass repetition, and were followed up with psychological analyses of optimal review schedules by Cepeda et al. (2006, Psychological Bulletin). M.A.K.S's recursive reinforcement term, F_R, encapsulates these ideas computationally, punishing cluster access and incentivizing distributed access.

Reconsolidation, which was first discovered in the field of neuroscience by Nader et al. (2000, Nature) and discussed in the context of computational modeling by Schacter et al. (2012, Nature Reviews Neuroscience), posits that after an initial period of dormancy, memory traces that have been triggered enter into a brief window of plasticity in working memory, during which they must be re-stabilized or permanently decay. This aspect of re-consolidation is what is modeled by M.A.K.S through the Ghost Store layer. In contrast to retrieval-augmented generation, which relies on querying from a static external database, reconsolidation by M.A.K.S occurs via an interception of an internally stored but soon-to-be evicted memory at PARTIAL fidelity.

2.5 Summary of Distinctions

Table 2 summarizes how M.A.K.S differs from the five most closely related systems across the dimensions most relevant to long-context agent memory management.

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718160

Table 2 — Comparison of Memory Management Systems

System	Scope	Scoring Mechanism	Storage Unit	Eviction / Recovery
H2O	Intra-inference	Cumulative attention sum	Token-level tensors	Hard eviction (no recovery)
StreamingLLM	Intra-inference	Temporal recency only	Token-level tensors	Rolling window (no recovery)
Infini-transformer	Architecture-level	Differentiable linear recurrence	Hidden state matrix	Uniform lossy compression
MemGPT	Session-level	LLM self-judgment via function calls	Hierarchical text tiers	Manual model-driven swapping
Generative Agents	Session-level	Recency + importance + relevance	Flat string records	Binary prompt injection
M.A.K.S (Ours)	Session-level	Multi-dimensional S(t): frequency, centrality, entropy, reinforcement	Structured Memory Units M	Three-zone fidelity degradation + cue-triggered reconsolidation

There is no previous approach within the scope of KV-cache memory eviction and scored agent memory management that includes multi-dimensional survival scores, fidelity degradation throughout three zones, and memory trace reconsolidation based on a trigger cue. Although the Generative Agents approach employs multi-dimensional scoring and trigger-based retrieval, it does not include continuous fidelity degradation and memory trace reconsolidation within the ghost zone.

III. THE M.A.K.S ALGORITHM

3.1 Memory Unit Encoding

We encode a Memory Unit M as an object that consists of the following attributes:
content — the pure text of the memory created_at — creation time in seconds since epoch last_accessed_at — last access time in seconds since epoch access_count n — total count of accesses made to this memory access_history — an array of pairs (timestamp, strength) describing every access instance, limited to the top 50 entries by using sliding windows to control overhead costs connections — an array of pairs (memory_id, weight) describing the relationships between this memory and others entropy_score H — pre-calculated Shannon entropy value at byte level fidelity — current fidelity: FULL, PARTIAL, or GHOST cached_survival — pre-calculated S(t) from the previous cycle for lazy evaluation cached_r — pre-calculated R(t) from the previous cycle to allow incremental calculations without traversing the entire access history

Previous models store memories in a more primitive way. H2O (Zhang et al., arXiv:2306.14048, 2023) stores the total attention weights of tokens as a measure of access importance but does not keep track of access histories or inter-memory relationships. Infini-transformer (Munkhdalai et al., arXiv:2404.07143, 2024) relies on compressive

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718160

memory in the hidden state representation, which is a differential mechanism within the model and not an external layer of discrete management like in M.A.K.S. MemGPT (Packer et al., arXiv:2310.08560, 2023) categorizes memories into two contexts, main and archival, with different message types and function calls that transfer information between memories; yet, it does not score memories on survival individually with a scalar and does not degrade the fidelity of memories automatically using their respective scores. Generative Agents (Park et al., arXiv:2304.03442, 2023) scores memories based on recency, importance, and relevance but does not employ fidelity zones, scored degradation, or cue-based memory retrieval from cold storage.

3.2 The Survival Score

The key novelty of M.A.K.S. is the definition of a multi-dimensional survival scoring function S(t), where each memory cell is associated with a scalar value corresponding to its survival priority. S(t) is calculated using the following equation:

$S(t) = [F_n \times C^{\beta} \times H_{norm} \times F_R] \times e^{(-\lambda t)}$

where t corresponds to the number of Unix seconds elapsed from the moment the memory was created. All terms enclosed in square brackets are statically normalized within [0, 1] relative to some theoretically achievable ceilings, which guarantees temporal stability – the introduction of new memories into the repository does not affect the survival scores of existing memories. Default parameter settings are provided in Table 1.

Table 1 — Hyperparameter Definitions, Defaults, and Ranges

Parameter	Description	Default	Range
λ (lambda)	Temporal decay rate (per second)	0.01	0.001 – 0.1
α (alpha)	Frequency weight exponent	1.0	0.5 – 2.0
β (beta)	Connection centrality exponent	1.0	0.5 – 2.0
μ (mu)	Reinforcement decay rate (per second)	0.1	0.01 – 0.5
θ (theta)	Survival threshold for FULL zone	0.1	0.05 – 0.5
a_i	Reinforcement boost per access event	1.0	0.1 – 2.0
n_max	Static normalization ceiling for access count	1000	fixed

Note on λ timescale: The default λ = 0.01 is calibrated for session-level memory — a single conversation timescale where memories should fade within minutes to hours of inactivity. For long-running agents requiring persistence across hours or days, λ in the range of 10^{-5} to 10^{-6} is recommended.

Worked example with default parameters: Consider a memory at t=0 with n=1, C=0.5, H_norm=0.6, F_R=1.0, α=1, β=1, λ=0.01:

F_n = log (2) / log (1001) = 0.693 / 6.909 = 0.100

C^β = 0.5^1 = 0.500

H_norm = 0.600

F_R = 1.000

e^ (−0.01×0) = 1.000

S (0) = 0.100 × 0.500 × 0.600 × 1.000 × 1.000 = 0.030

Given that θ = 0.1 and GHOST threshold at 0.1θ = 0.01, this memory falls into the PARTIAL category (0.01 ≤ 0.030 < 0.1). That's how it was supposed to work. PARTIAL is the proper entrance point for recently created memories with minimal history of access and typical connectivity. Accessing a memory increases F_n and F_R, moving it from PARTIAL to FULL. Conversely, inactivity results in decay to GHOST state.

3.2.1 Temporal Decay — e^(−λt)

This exponential decay term models the Ebbinghaus forgetting curve directly, with t in seconds from the moment of creation. With λ = 0.01, memories keep approximately 90% of their time component score after 10 seconds and only 37% after 100 seconds —

aggressive decay suitable for session-level conversational memory. This temporal decay term makes sure that unreactivated memories simply disappear naturally, without ever having to be explicitly deleted.

3.2.2 Frequency Protection — F_n

The M.A.K.S system uses logarithm to calculate F_n based on a normalized fixed theoretical ceiling n_max = 1000:

$$F_n = [\log(1 + n) / \log(1 + n_{max})]^{\alpha}$$

As such, F_n ∈ [0, 1] as required, with n_max remaining constant. Scores are not updated even as new memories with larger n are added to the list. Using a logarithm reflects the law of diminishing returns as per the psychological theory. Parameter α defines how steeply scores decrease with rising access count. n_max = 1000 serves as an upper theoretical bound on number of accesses, as all practical conversation memories should have n < 1000.

3.2.3 Shannon Entropy Weight — H_norm

The Shannon entropy weight H_norm is the normalized Shannon entropy measure of the content in the memory, based on the frequency distribution of bytes within the content string:

$$H = -\Sigma\, p(b) \times \log_2(p(b))$$
$$H_{norm} = H / \log_2(256) = H / 8$$

where p(b) is the relative frequency of byte b in the content string. The division by $\log_2(256) = 8$ normalizes the entropy measure to a range of [0, 1] with static bounds that do not depend on the memory content.

We recognize that entropy at the byte level is a proxy for semantic uniqueness and not a direct measurement. Entropy is high for random bytes, but there is no semantic meaning here. In reality, texts with unusual identifiers, numeric codes, proper names, and technical vocabulary will have more entropy at the byte level than conversational text because of the variety of combinations of bytes used in the text. The empirical validation of this relation – a study that calculates H_norm in comparison to human judgment of importance scores in conversation data – is recognized as a critical research area in future work. As far as we know, no previous research on KV-cache eviction and scored agent memories uses Shannon entropy as a unit survival factor.

3.2.4 Spaced Reinforcement — F_R

R(t) uses a continuous spaced repetition system. The cached_r member variable in the memory unit keeps the latest R value, allowing for O(1) update of R. During each access operation, R is calculated as:

$$R(t) = R(t - \Delta t) \times e^{(-\mu \Delta t)} + a_i$$

where Δt is the time difference in seconds between the previous and current access operations, μ is the decay rate of the reinforcement, and a_i is the boost factor of the current access event (default 1.0). Only the current timestamp and the cached_r value are needed — no need to iterate through all access events. The ceiling value of R(t) assuming continuous reinforcement can be determined using the formula of an infinite geometric series:

$$R_{max} = a_i / (1 - e^{(-\mu)})$$

The normalized value of R(t) based on the above ceiling value will be:

$$F_R = R(t) / R_{max}$$

Thus, we get F_R ∈ [0, 1] with boundaries which are dependent on only constant parameters a_i and μ, keeping constant despite the increasing memory storage size. The access_history is an independent parameter used for auditing, trimmed to 50 elements by means of sliding window technique; it does not take part in the survival calculation.

3.3 The Fidelity Function

For any S(t) and survival threshold θ, the fidelity function F assigns each memory unit one of three regions:

FULL if S(t) ≥ θ — full content, completely injected into the current active context window

PARTIAL if 0.1θ ≤ S(t) < θ — content is reduced to its first 500 characters

GHOST if S(t) < 0.1θ — content is reduced to its first 100 characters and sent to the Ghost Storage

The boundary of 0.1θ results in GHOST being set an order of magnitude below FULL, thus forming the PARTIAL range which is wide enough to

accommodate memories in the process of decay. As shown by the sample calculations, a new memory formed with a single access and average connectivity starts off in the PARTIAL range. This memory will move to the FULL range due to successive accesses, and from there to the GHOST range due to lack of activity. In contrast to binary schemes, fidelity degradation in M.A.K.S occurs gradually and reversibly; content is compressed but not erased.

3.4 Ghost Reconsolidation

The ghost memories get transferred to a secondary Ghost Store which exists outside the active context window. For each LLM query, M.A.K.S. does keyword extraction from the query and uses a linear scan over tokenized memory content of the current prototype in the Ghost Store to find any matching content. In our implementation, this process takes less than 2 ms for Ghost Stores with 1,000 entries, which is less than 1% of the typical LLM inference latency. Inverted indices will be used in cases where Ghost Stores have more than 10,000 entries, in which case the linear scan latency will be closer to the minimum LLM inference latency.

The matching ghost memories get reconsolidated, i.e., brought back to the active memory with a PARTIAL fidelity and with cached_r set to $a_i = 1.0$, meaning that there was a new reinforcement boost due to reconsolidation itself, before sending the query to the language model.

We differentiate this approach from Retrieval-Augmented Generation and MemGPT's archival retrieval on two counts. Firstly, RAG relies on retrieval from a static external knowledge base; the Ghost Store includes memories which had been previously alive during the same conversation and had suffered score-based fidelity degradation. Secondly, reconsolidation happens automatically through content of the query without any manual trigger by the user and works as an invisible layer beneath the LLM interface. In the Generative Agents model (Park et al., 2023), archived memories are retrieved through composite recency/importance/relevance scoring, but there are no fidelity degradation zones and revival of cold memories based on survival scores. Among the scored memory management systems working within a degradation-revival lifecycle paradigm, we have not encountered anything similar to this before.

3.5 Computation Overhead and Mitigation

The recursive formulation of R(t) makes S(t) computation per memory unit an O(1) operation; there is no iteration over access history involved. The entire sweep to maintain all n memory units is thus an O(n) operation per cycle without caching and an O(k) operation with lazy evaluation, where k is the number of units close to zone boundaries or accessed recently. There are three mitigation techniques for lowering cost in practice:

Lazy evaluation – Each memory unit stores its latest S(t) value in cached_survival. The recomputation process is initiated only upon access or after the configurable staleness period expires. Memory units that have not been accessed recently retain their cached value.

Background precomputation – the maintenance loop operates on a configurable schedule during idle cycles, spreading the computational cost over time rather than concentrating it at inference time.

Tiered recomputation – Only memory units located within 10% of a zone boundary undergo full recomputation during every maintenance cycle. Those stable inside the zones rely on cached values, making the sweep cost lower than O(n) in practice.

The benchmarking using an AMD Ryzen 5 processor series with single threading and Python version 3.11 indicates that computing one S(t) takes 0.002ms after warming up the caches. The lazy approach saves 3.49x in store sweep time with 50 memory units and 9.25x in store sweep time with 5,000 memory units. This shows that the overhead due to M.A.K. S's scoring is insignificant compared to LLM inference times of hundreds to thousands of milliseconds.

3.6 Limitations of the Current Formulation

Four limitations of the current formulation have been clearly stated.

First, the entropy measure H_norm relies on character distribution at the byte level as a proxy for semantic uniqueness. Such an assumption has never

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718160

been formally verified by comparison against human evaluations or by using density of embeddings. Correlation studies are suggested as future work.

Second, no empirical investigation of the hyperparameter sensitivity has been carried out. While default values in Table 1 ensure stable operation on our experimental dataset, sensitivity across the whole parameter space, especially the interaction of λ and θ, has never been studied. An analysis of sensitivity varying each parameter separately within its prescribed range will be carried out as part of the experimental extension before submission.

Thirdly, our current prototype works as a middleware layer on top of the LLM, performing pre-processing before inference. Such an architecture allows for flexible deployment – M.A.K.S is compatible with any LLM without changing the model architecture, weights, or attention mechanism, and can be retrofitted into already deployed models without retraining. Full native KV cache integration will remove even this small barrier and is the main architectural direction for production deployment.

Lastly, our current Ghost Store implementation relies on keyword matching for reconsolidation search. This approach leads to both false positives – ghosts that are matched by the keyword but not semantically relevant – and false negatives – relevant memories where the content has been compressed into ghosts and does not contain the keyword anymore. Semantic search based on embeddings would solve both issues.

IV. PROTOTYPE IMPLEMENTATION

4.1 System Architecture Overview

M.A.K.S framework implementation is carried out as a modular and lightweight middleware layer using Python 3.11. The overall system design consists of four major subsystems containing ten different functional modules. The framework operates by interfacing with any downstream application layer and large language model to programmatically handle memory degradation, eviction, and reconsolidation process. Figure 1 below depicts the overall system architecture.

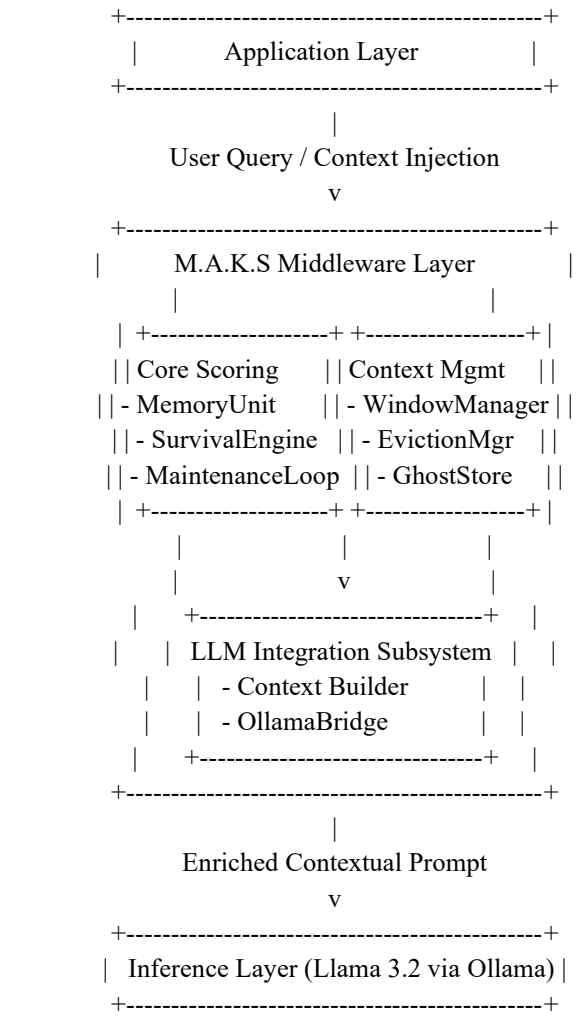

Figure 1. M.A.K.S system architecture depicting four subsystems working together as a distinct middleware layer between the application layer and the inference layer. The M.A.K.S system does not require any changes to the target model's architecture, weights, or attention mechanism and communicates through the standard HTTP completion endpoints.

The full source code of the prototype, along with dependency files and configuration scripts for experimentation, are available in the public repository of the project (link to the anonymous submission is provided).

4.2 Core Scoring Subsystem

Core Subsystem. The core subsystem manages the basic memory data structure as well as performs survival scoring.

Memory Unit Abstract Class. Memory traces are represented as structured MemoryUnit dataclass objects with the eleven fields formalized in Section

4.1: memory_id (string unique ID), content, created_at, last_accessed_at, access_count, access_history, connections, entropy_score, fidelity, cached_survival, and cached_r. A memory unit initialization function creates a unique ID, logs Unix creation timestamp, and calculates the basic Shannon entropy score H_norm exactly once during object construction, thus avoiding any character frequency calculations overhead in subsequent scoring passes.

Survival Engine. The survival engine calculates S(t) by performing the sequential steps described in Section 4.2: elapsed time delta, exponential decay e^(−λt), normalized frequency F_n, degree centrality C^β, pre-calculated entropy H_norm, and cached incremental reinforcement state F_R. The survival engine wraps this calculation with lazy cache validation that prevents full S(t) calculation calls if there were no access events recorded since the last call in the configurable staleness window. A single S(t) calculation takes 0.002ms; searching over 1,000 ghost store entries is performed under 2ms; compiling context for 50 memories takes approximately 1ms.

Fidelity Classification. This section implements the degradation continuum. It compares S(t) with the threshold values of θ and 0.1θ in order to classify zones as FULL, PARTIAL, or GHOST. On the occurrence of zone change, the content will be compressed to the required character limit of 500 for PARTIAL, 100 for GHOST along with a zone indicator.

Maintenance Loop. This class facilitates background scoring updates at intervals determined by the value of maintenance_interval. Every three turns (by default), which is the best compromise between overhead and responsiveness for memory management, all memory units will be traversed, updated with survival scores, zone changes performed, and finally a dictionary of zone counts returned.

4.3 Context Management Subsystem

The context management subsystem controls token budget utilization and performs eviction in stages.

Window Manager. The WindowManager object serves as the telemetry gateway for token budget utilization. In order to keep overhead low while not using the tokenizer from the model inside loops, the manager approximates token utilization by four characters per token. This approximation is accurate to within ±10% for English-language conversational text used in the experiments; use cases involving code, JSON, or non-Latin alphabets might need calibration. Token utilization corresponds to four different levels of operational pressure: LOW (≤ 0.50), MEDIUM (0.51-0.75), HIGH (0.76-0.90), and CRITICAL (> 0.90). The eviction_candidates() method returns memory units ordered according to their survival score in ascending order.

Staged Eviction Engine. Eviction is initiated when pressure levels are at HIGH or CRITICAL level and it is performed through three stages. The first stage involves the compression pass which cycles through the priority queue and compresses the FULL memory objects into PARTIAL ones until the pressure drops below HIGH level. In case there is not enough space reclaimed, then the second stage which involves compressing PARTIAL memory objects to GHOST and moving them into the Ghost Store is performed. The last stage of hard eviction is only considered when pressure is at CRITICAL level.

Ghost Store and Reconsolidation Pipeline. The GhostStore object holds an independent key-value index of MemoryUnit objects that have been archived outside the context window. Matching is done using keyword search on tokens of ghost content, with top-k matching based on the number of keyword matches. The current implementation uses a linear scan algorithm, while inverted indices will be used for Ghost Stores larger than 10,000 items. Once a match is made, the reconsolidation pipeline retrieves the item from the Ghost Store, sets fidelity to PARTIAL, resets cached_r to $a_i = 1.0$ denoting new reinforcement, updates the last_accessed_at value, and places the item in the active memory store. All reconsolidations are logged in the reconsolidation_log.

4.4 LLM Integration Subsystem

OllamaBridge connects M.A.K.S middleware to the local instance of Llama 3.2 served via Ollama. Adapter pattern can be extended to vLLM, Hugging Face TGI, and cloud completion APIs via adapter classes implementing the same HTTP POST interface without altering any core logic of M.A.K.S.

Context assembly of the user query goes through the following stages:

First, the query is used to retrieve keyword matches from the Ghost Store. All found matches are reconsolidated into the active store before the context is built. Second, the top-k memory units based on their survival score are retrieved from the active store and formatted as the memory context block. Third, the context block is concatenated to the query and sent to the model API endpoint. Fourth, the answer text is extracted from the reply, new MemoryUnit is created out of the query-answer pairs, and the unit is saved to the active store.

4.5 Reproducibility

All experiments have been conducted on an AMD Ryzen 5 series processor, single-threaded, Python 3.11, with Llama 3.2 being served locally using Ollama. There is no need for a GPU in the memory management system; any GPU usage relates to Ollama inference alone. All filler memory contents, needle contents, query strings, token capacity upper bounds, and maintenance cycles utilized in the experiment have fixed values that are hardcoded into the experiment files. There are no stochastic components involved in the memory management pipeline, with the only stochastic element being the response text of the LLM, which has no effect on survival, zone assignment, or reconsolidation decisions.

V. EXPERIMENTS

5.1 Experimental Setup

M.A.K.S is assessed through three experiments that seek to test the three main hypotheses put forward in this paper, namely, that M.A.K.S maintains crucial information when under memory pressure and where baseline eviction does not work, that the performance of the system is based on particular architectural elements that carry the load, and that the overhead of survival scoring is insignificant compared to LLM inference latency.

Table 3 — Experimental Summary

Experiment	Primary Claim Tested	Baseline	Key Metric
Needle in a Compressed Haystack	Critical memory retrieval under pressure	FIFO eviction	Retrieval accuracy
Ablation Study	Component contribution to system behavior	Full M.A.K.S	Zone survival and reconsolidation rate
Overhead Benchmark	Computational cost of S(t) scoring	No caching	Sweep latency and speedup ratio

All experiments were performed using a single-threaded AMD Ryzen 5 series CPU, Python 3.11, with Llama 3.2 running locally using Ollama. Memory management was performed without the aid of GPUs; GPU usage occurred solely during inference. All experiments were performed using one trial. Statistical analysis involving multiple runs using confidence intervals is noted as an important step to be performed in future experiments prior to submission.

5.2 Experiment 1 — Needle in a Compressed Haystack

5.2.1 Motivation

The conventional Needle in a Haystack benchmark assesses whether a language model is capable of extracting a critical piece of information from a lengthy context. We propose extending this benchmark into a more difficult task: Needle in a Compressed Haystack. Our definition of this problem involves degrading the critical piece of information using the memory management system, compressing the information into GHOST state, and then reconsolidating the information from cold storage before responding to the query.

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718160

5.2.2 Setup

The memory store consisted of one needle memory and fifty filler memories. The needle was defined as: "The project codename for the M.A.K.S memory system is DELTA-7-ECHO." The filler memories consisted of sentences repeated to produce filler memories with a fixed length of 1,500 characters, resulting in an estimated number of tokens equal to 18,766 tokens, which was loaded against a window capacity of 4,096 tokens, giving a ratio of 4.58× and a peak pressure level of CRITICAL.

The FIFO baseline was set up in the following way: memories were put into a dictionary in the order of their creation, and then as soon as the pressure was HIGH or CRITICAL, memories were removed from the dictionary in the order of insertion – oldest first – until the pressure was less than HIGH. Thus, the needle was the first in line for FIFO deletion.

The M.A.K.S condition involved all aspects of the process: maintenance loop (2 full iterations, with an adjustable interval of 0.1 seconds between iterations), window manager pressure estimation, evictions in stages, Ghost Store archival, search using keywords "project codename M.A.K.S", reconsolidation, and LLM query through OllamaBridge.

5.2.3 Evaluation Criteria

Four Boolean metrics were measured for each condition:

Needle survived – whether the needle memory could be accessed after applying memory pressure

Needle reconsolidated – whether the needle could be accessed from the Ghost Store and brought back into memory LLM answered – whether the LLM provided an answer to the question

Answer correct – whether the answer included the specific target string DELTA-7-ECHO, confirmed by string comparison

5.2.4 Results

Table 4 — Needle in a Compressed Haystack Results

Condition	Needle Survived	Reconsolidated	LLM Answered	Answer Correct
FIFO Baseline	No	N/A	No	No
M.A.K.S	Yes (as Ghost)	Yes	Yes	Yes

In the FIFO policy, the needle is deleted at the first pass through the eviction cycle because it was inserted first and therefore evicted first. In this scenario, the LLM did not receive any memory context holding the target fact and thus did not produce a correct answer. In the M.A.K.S case, the needle decayed to the GHOST state, was archived in the Ghost Store, was retrieved using keyword search from the incoming query content, was reconsolidated to PARTIAL state before query dispatching, and the LLM correctly returned the target string. Thus, all four outcome criteria were met in the M.A.K.S case but missed in the FIFO case.

5.3 Experiment 2 — Ablation Study

5.3.1 Motivation

The Needle in a Compressed Haystack experiment proves the overall system capabilities. An ablation study separates the important components from the secondary ones. To achieve this goal, we run four experiments, each disabling one of the M.A.K.S components.

5.3.2 Setup

The same memory store setup was used in all four variants as in Experiment 1: one needle memory, fifty 1,500-character fillers, 4,096 token window capacity. Each variant was tested independently using a fresh memory store initialized with the same parameters.

The four variants were:

Full M.A.K.S — all components enabled. Baseline condition.

No Entropy — H_norm set to 1.0 for all memory units prior to scoring. This removes entropy from being a discriminating factor between memories. All units are scored equally on entropy weight, measuring the impact of the other four S(t) factors.

No Ghost Zone — Partial memories compressed to GHOST quality in maintenance as usual, but rather than being archived to the Ghost Store, all GHOST memories are immediately deleted from the active store. No Ghost Store is used.

No Reconsolidation — The ghost memories are stored normally in the Ghost Store, but skipping the reconsolidation stage before querying the LLM.

5.3.3 Results

Table 5 — Ablation Study Results

Variant	Needle Survived	Reconsolidated	Answer Correct
Full M.A.K.S	Yes	Yes	Yes
No Entropy	Yes	Yes	Yes
No Ghost Zone	No	No	No
No Reconsolidation	Yes	No	No

The Ghost Zone is load-bearing. The removal of the Ghost Zone leads to total system breakdown. Due to the lack of cold storage, the needle is completely deleted during the maintenance process, and there is no way for the system to retrieve it. The LLM was not provided with any context that would include the required fact and failed to give the right answer. Therefore, the Ghost Zone is an inevitable element of the architecture.

Reconsolidation is load-bearing. Although the needle was successfully archived into the Ghost Store, its absence led to the failure of the LLM to provide the right answer. The needle was present in the cold storage but could not be accessed by the context window due to the absence of reconsolidation. This proves that just like the cue-triggered revival, archiving is crucial in the life cycle of the memory.

Entropy regulates subtle priority, but not survival or death. Entropy weighting was not necessary for success — the needle survived, reconsolidated, and the LLM gave the right answer under No Entropy.

This result has a very specific meaning – in this particular experiment setup, needle's survival depended only on the interplay between decay, frequency, and reinforcement terms. Entropy weighting allows distinguishing between memories with close-to-equal scores on other parameters, which means that entropy is a secondary sorting parameter that plays an important role mainly in big memory stores (more than 10,000 items), which contain diverse content, including code, text, and structured data, or in case of high survival scores close to zone borders. Entropy is not a survival component.

5.4 Experiment 3 — Overhead Benchmark

5.4.1 Motivation

The main practical problem in M.A.K.S design is computation overhead. Calculation of S(t) for all memory units on each maintenance cycle creates latency that should be small compared to latency of LLM inference to prevent degradation of user experience. In this experiment we measure real overhead on three scales and calculate benefits of lazy evaluation caching.

5.4.2 Experimental Setup

Three sizes of memory stores were considered: 50, 500, and 5,000 memory units. These stores were filled using make_store() function, described in experimental subsystem (section 5.5). Units have ids mem_001 to mem_N; every tenth unit had access_count equal to 20 and three accesses to history with strength 0.8.

There were three baseline measures taken on each scale:

Single S(t) - time spent on a compute_survival() function call per 100 invocations on one memory unit

with caching turned off. Represents basic computational cost of processing one unit.

Full sweep - time required to invoke compute_survival() on all units in the store without caching. Represents worst-case maintenance scenario when lazy evaluation is not used.
Lazy sweep - time required for the second phase of get_survival() call across all units in the store after the first phase filled the cache. Represents maintenance of a steady state, where most of the units have not been visited since the previous maintenance cycle.
The speedup was calculated as Full Sweep / Lazy Sweep.

5.4.3 Results

Table 6 — Overhead Benchmark Results

Scale	Single S(t) (ms)	Full Sweep (ms)	Lazy Sweep (ms)	Speedup
50	0.0023	0.1173	0.0336	3.49×
500	0.0013	0.6319	0.0972	6.50×
5,000	0.0016	10.2105	1.1038	9.25×

The per-unit cost is very low. One iteration of S(t) takes roughly 0.002ms on standard hardware. This allows M.A.K.S to achieve a performance of 435,000 memory units per second on one CPU thread – far exceeding any realistically attainable context window size – before approaching the minimum limit of LLM inference latency.

Full sweep is linear. Full sweep time grows linearly with the size of the store – doubling the store size results in doubling the full sweep time. This is consistent with the O(n) complexity of the per-cycle maintenance sweep and verifies that the recursive R(t) expression successfully cancels the O(h) access history term from the per-cycle computation.

Lazy evaluation speedup compounds. The speedup ratio rises from 3.49x at 50 units to 9.25x at 5,000 units. This occurs due to the fact that a bigger store has relatively more units in stable regions where no re-computation is required – the cache hit rate increases with the store size. In production with context windows large enough to support tens of thousands of memory units, we expect the lazy evaluation speedup ratio to compound further, although this remains speculative and serves as a target for future benchmarking.

Overhead is minimal compared to LLM inference. At 5,000 memory units with lazy evaluation, a complete maintenance cycle takes 1.1ms. LLM inference with Llama 3.2 through Ollama on our test hardware was between 2,000ms and 5,000ms per response, depending on the length of output and hardware setup. Therefore, the overhead of M.A.K. S's maintenance algorithm makes up less than 1% of the total latency per turn for all configurations.

VI. DISCUSSION AND DESIGN ANALYSIS

6.1 Discussion of the Needle in a Compressed Haystack Benchmark
The key take-away from Experiment 1 is not the successful resolution of the target string verification by M.A.K.S. It is the failure of the First-In-First-Out approach. With identical runtime conditions, a truncation-based scheme that is commonly used in production AI agent implementation chose the most essential piece of operational context to be immediately destroyed, with no possibility for any recovery within the architecture. This is not an edge or pathological case; it is the default behavior of any large language model implementation with a managed sliding context window.

The adversary relationship of the experiment, where the needle is inserted first, meaning that it is evicted first according to FIFO, reflects the structure of a typical conversation in real life. In long-horizon agent deployment scenarios, the constraints, parameters, specifications, and behavior context are virtually always set out in the early turns of the session. FIFO eviction always eliminates precisely this kind of information as well. The automated degradation-revival cycle of M.A.K.S. managed to preserve this essential aspect without needing any annotation or function routing at all.

The process of reconsolidation at just-in-time needs closer inspection. Before compilation of the prompt, the needle had been reduced to the GHOST form, compressed down to 100 characters, and entirely disconnected from the context window. Right when the query was invoked, its token footprint was exactly zero within the context buffer. The automatic keyword routing procedure detected the incoming query text, recognized the ghost pattern, and revived the memory up to PARTIAL level before context array compilation.

This answer was correct not because the LLM had an intrinsic property of maintaining infinite context; rather, it was because the middleware had reconstructed the needed grounding context right at the time it was needed. This dynamic reconsolidation is precisely what makes up the heart of M.A.K.S, and the closest computational equivalent to human declarative cue-based recall.

6.2 Structural Deconstruction of Ablation Dynamics
Ablation provides three separate structural insights into the interactions among M.A.K. S's components that generate system-level behavior.

Firstly, the Ghost Zone discovery transforms the main contribution. The utter failure of the no Ghost Zone configuration sets up the critical architectural principle: the primary strength of M.A.K.S does not lie in the multidimensional survival scoring function S(t). Instead, it lies in its architectural preservation layer. While S(t) dictates the deterministic sequence and rate of context degradation, it is the Ghost Store which makes this degradation fully reversible. A sophisticated scoring function combined with binary eviction will lead to the permanent loss of the needle. A simple scoring function combined with a Ghost Zone will ensure that it survives.

It does not mean that S(t) plays no role in real-world applications. S(t) is responsible for deciding which memories should degrade first, how fast they degrade, and which memories survive near the boundary of the zone. Without S(t), degradation will become random; the system will keep the wrong memories while randomly ghosting important ones. The difference is that S(t) gives intelligence to the system, whereas the Ghost Zone ensures its survival. The above explanation clearly delineates the contribution of M.A.K.S from those of H2O and Generative Agents, which are limited to selection criteria.

The reconsolidation experiment proved the need for the revival loop. The cold storage archival was pointless without a corresponding retrieval module. Disabling the reconsolidation loop made it impossible for the system to retrieve the needle. The result was that although the system managed to archive the needle, it was invisible to the LLM when queried. In other words, the Ghost Store and the reconsolidation loop are two sides of the same architectural coin – an inseparable entity. Without a reactivation loop, the archival store is nothing but a graveyard.

The reason why entropy operates as a precision sorting tool instead of a survival gate is that in the case of the No Entropy variant, the macro behavior is precisely the same. This indicates that in this particular experimental setup, the survival of the needle was dictated by the dominating role played by the temporal decay, frequency, and reinforcement parameters. Entropy had nothing to do with it. On the other hand, when the survival score distribution is compact or clustered, entropy is a crucial distinguishing factor especially in stores containing more than 10,000 memory units, for various types of content including source code, JSON and natural language, or when scores are clustered near zones boundaries where differences are decisive for the assignment of a zone.

6.3 Computational Feasibility and Deployment Mechanics
The overhead metrics reveal two key insights regarding the mechanics of deployment.
First, the cost of S(t) calculation per unit of 0.002ms implies that background scoring is guaranteed never to be the limiting factor in terms of computational overhead in any agent deployment. Inference time for the LLM exceeds the maintenance cost for M.A.K.S by an order of magnitude — by 1,800× to 4,500× based on LLM inference time ranging from 2,000ms to 5,000ms relative to the lazy sweep cost of 1.1ms at 5,000 units. The deployment of M.A.K.S in existing LLM workflows does not affect latency at all.

Second, the increasing compounding speedup of lazy evaluation – from 3.49× for 50 memory units up to 9.25× for 5,000 memory units – flips one of the fundamental tradeoffs of systems engineering: M.A.K.S becomes increasingly efficient as the memory store grows in size. This effect is derived from the inherent structure of long-horizon conversations. With a larger store, there is an increasing number of tail-distributed cold memory units that are trapped inside stable fidelity zones without any recency-based access events and do not require computation. The hit rate of the cache increases as the store grows, thus driving the compounding speedup. M.A.K.S is thus better suited for large-scale long-horizon deployments rather than short-term sessions, where it is the most efficient precisely because of the difficulty of the memory management problem.

This sets certain constraints on M.A.K. S's deployment topology. M.A.K.S cannot be treated as a generic context compression algorithm but, rather, as an optimization engine for persistent agents engaged in multi-session interactions over time. Short session deployments with less than 50 memory units will have little benefit from lazy evaluation and can be better off using simple truncation.

6.4 Scope and Limitations of the Experimental Design

The three limitations of the present experimental design are explicitly recognized, separate from those of the algorithmic nature described in Section 4.6.

First, all filler memories were created from artificial template sentences rather than from natural conversation. Real human-agent conversation features semantic redundancy, topic referencing, dependency references, and non-uniform turn durations, which do not occur in artificially synthesized materials. The distribution of survival scores under actual conversational load would be different, affecting the relative weights of the terms of the S(t) function in an unpredictable manner.

Second, all experiments were performed using a single trial setup. Reproducibility can be achieved under fixed initialization parameters and constant content, but multiple-trials assessment using varying random seeds, varying needle content, and varying filler distributions needs to be performed in order to generate confidence intervals around the results. This is considered the highest priority experimental extension to undertake.

Third, the basis of comparison was confined to FIFO. LRU, LFU, and random eviction are natural benchmarks that could serve to further support the comparative arguments made. A direct benchmark of M.A.K.S in relation to the Generative Agents recency-importance-relevance model under equal memory pressure conditions would be especially useful in demonstrating its superiority over the closest existing work.

VII. LIMITATIONS AND FUTURE WORK

7.1 Overview

The present version of the M.A.K.S model can be characterized as a working prototype of the approach which serves to demonstrate the viability of the fundamental degradation-revival cycle. Four limitations pertaining to the algorithms and three pertaining to experimentation were discussed in Sections 4.6 and 7.4 respectively. In this section, all limitations will be brought together, matched by their respective technical solutions and avenues of future research designed to scale the model up to real-world memory management systems.

7.2 Algorithmic Enhancements and Architectural Augmentations

7.2.1 Semantic Uniqueness Estimation via Embeddings

Shortcoming: The existing entropy term H_norm employs byte-level character statistics as a cheap substitute for semantic uniqueness estimation. Although this method is computationally inexpensive, being calculated only once at memory initialization stage in O(n) character time, it still lacks semantic awareness. A random sequence of bytes maximizes Shannon entropy while providing no structural or communicative meaning. The ablation study proved the use of entropy as a precision breaker, not a survival switch, in the experimental settings, while being imprecise in theory.

Direction for future development: Use an embedding-based semantic density metric rather than byte-level

calculation. After each memory is created, it will be embedded into a dense vector space, and the cosine distance between the memory and the running centroid of the memory pool will be calculated. Memories that have a higher semantic distance to the centroid of the memory pool – memories that are unique and/or signal structural changes in the ongoing conversation – will be assigned a higher survival weight. The computation cost is limited to one embedding look-up upon memory creation and a cheap centroid vector update per maintenance iteration, both feasible on reasonable sized stores.

7.2.2 Automated Hyperparameter Tuning and Sensitivity Analysis

Limitation: The six parameters in Table 1, viz. λ, α, β, μ, θ, and a_i, were set to constant default values optimized for reliability in our experiments. The parameter space of interactions between them – in particular, the combined impact of λ and θ on zone occupancy distributions – has not yet been exhaustively studied. Production deployments with distinct memory profiles, for instance, long-term planning AIs versus fast-turn chatbots, may need different calibrations, for which the present paper does not provide any explicit optimization recommendations.

Future direction: A dual approach to address the issue. First, a sensitivity analysis where each parameter is varied through its whole range of possible values with adjacent constants held at default, and zone occupancy distribution and retrieval performance measured as outputs. Second, a mechanism for online parameter adjustment whereby λ and θ are adjusted according to real-time zone occupancy measures; if the store is persistently oversaturated at the GHOST level, θ should be reduced; and if memories do not often expire beyond PARTIAL, λ should be increased.

7.2.3 Native Attention-Layer and KV Cache Integration

Current limitation: The present prototype acts as an external middleware layer that intercepts queries and manages memory independently of the attention mechanism of the transformer. Though this approach provides maximum flexibility in implementation — M.A.K.S is agnostic to LLM architecture and can integrate with any such model without any modification to weights or training process — it also introduces a hard pre-processing boundary between the system and the LLM, which prevents M.A.K.S from directly manipulating the internal attention matrices of token attention within the model.

Future direction: Native KV cache integration. Instead of injecting memory units as text strings, M.A.K.S survival scoring engine will connect directly to KV cache entries of the model serving framework. Tokens and key-value entries corresponding to low-survival memory units will be evicted from the active attention layers. This eliminates the pre-processing boundary, aligns survival scores with internal attention weights, and brings M.A.K.S closer to becoming an intrinsic architectural primitive. This requires developing M.A.K.S in open-source model serving engines like vLLM or Hugging Face TGI.

7.2.4 Dense Semantic Retrieval over Ghost Archives

Limitation: The retrieval component in the Ghost Store depends on tokenized keyword search against extremely compressed 100-character long signatures. The larger the size of the archived history, the greater the likelihood of getting false positives, where ghost memories match with the search keywords but do not align semantically; or false negatives, where memories that match the search keywords get excluded because the keywords have been truncated in the compression of the memories.

Future work: Move to nearest neighbor embedding search over full historical traces. Each ghost memory stored in the Ghost Store will be stored as an embedding that represents the full content of the memory prior to being compressed into the text signature. The reconsolidation process will perform fast search over these dense embeddings, significantly cutting down the number of false negatives caused by truncation of the signatures, while ensuring that semantic matches can be achieved even if there are mutations of the keywords involved. Storage cost is about 1.5KB per ghost unit, or 384-dimensional float32 embedding.

7.3 Experimental Extensions and Evaluation Metrics

7.3.1 Validation on Real-World Long-Context Benchmarks

Shortcoming: All the memory contents used in the experiments were created from artificial template sentences, rather than real conversational data. Real conversational interactions between humans and agents are characterized by semantic overlap, topic repetition, referential dependencies, and inconsistent turn lengths that cannot be replicated using artificial content. The survival score distribution based on the load of real conversations may differ significantly, affecting the contributions of the S(t) terms in different ways.

Future direction: Run the complete experiment on established long-context conversational benchmarks, such as LongBench (Bai et al., 2023), and SCROLLS benchmark suite (Shaham et al., 2022). The frameworks offer standard long-form prompts with predetermined retrieval targets, allowing for comparison with other memory management approaches using identical evaluation settings and validating M.A.K. S's applicability outside the synthetic environment.

7.3.2 Multi-Trial Statistical Expansion

Constraint: All results were obtained from single trial executions using pre-specified parameter settings and static content data. While the results can be reproduced with identical initializations, formal statistical analysis calls for multi-trial tests conducted under varying random seed values, needle content, and filler distributions.

Future Work: Extend the benchmark suite to include multiple trials per test with at least 30 runs using random initialization. Provide means, standard deviations, and 95% confidence intervals for all binary output measures. Modify overhead tests to show timing distribution curves instead of average timings. This statistical extension is scheduled for future revisions.

7.3.3 Broadening of Comparative Baselines

Drawback: The current comparative study is confined to FIFO-based eviction. LRU, LFU, and random evictions make natural extensions for such a comparison. A direct comparison with Generative Agents' recency-importance-relevance scoring model under comparable memory pressure is especially desirable for establishing the superiority of M.A.K.S over the closest previous effort.

Future Direction: Extend the current needle test framework to incorporate LRU, LFU, random eviction, and Generative Agents-style recency-importance-relevance scoring as baseline models. All baselines will be compared against the same needle and filler setting under identical token pressure ratios. Results from all comparisons will be provided in a single comparative table. Such an extended baseline comparison is considered the second-highest priority for future experimentation.

7.4 Larger Infrastructure Vision

In addition to addressing known weaknesses in M.A.K.S., there are three ways that M.A.K.S can move toward achieving its larger vision of infrastructure for long-horizon AI agents.

Scalability testing. Today's benchmarks are performed using a 4,096 token window size with 5,000 memory units as the largest store size. Production systems have been implemented with 128,000 token windows and millions of memory events based on multi-session agent histories. Testing the reliability of S(t), cache hit ratios using lazy evaluation, and reconsolidation recalls at production scale are essential steps before deployment.

Memory sharing between multiple agents. In its current design, M.A.K.S maintains memory for a single agent instance. Memory sharing among multiple agents raises issues about survival scoring: which agent's history is used, how connection centrality is calculated when multiple agents are involved, and coordinating reconsolidation triggers. Distributed shared memory clusters will be a logical extension of M.A.K.S.

Learning survival parameters. Currently, the hyperparameters α, β, λ, μ, and θ are fixed or hand-tuned. It is straightforward to extend this to learned parameters via gradient optimization of retrieval accuracy or by reinforcement learning, in which the reward function comes from task performance. With learned parameters, M.A.K.S. will be able to adapt to

the memory dynamics of various deployment scenarios without any manual tuning.

Achieving such a vision depends upon scaling experiments that measure sweep latency, reconsolidation recall, and memory size at production-level store sizes, which was mentioned as the major experimental goal after the validation of the prototype.

CONCLUSION

The context window is the most important constraint on any current AI system. Any long-horizon agent – any system that talks back through sessions, any system that makes inferences about long documents, any autonomous system that builds up its world view over time – is constrained by a hard limit to how much it can keep in mind at once. Once it hits that limit, something needs to go out. The choice for which thing needs to be discarded is the difference between intelligence and amnesia.

All the current techniques do a poor job making that choice. FIFO drops the oldest things first. LRU drops the most recent things first. Attention-based eviction drops whatever the model thinks is least important during inference. None of these techniques consider the real question – how well have these memories proven themselves worthy of being retained?

That's where M.A.K.S comes in with its sophisticated survival scoring function S(t), which incorporates time decay, access frequency, graph centrality, Shannon entropy, and spaced reinforcement history in one single scalar score. Memory items are not stored or deleted; rather, they are continuously degraded through three fidelity zones, stored in compressed form as ghosts in cold storage, and re-consolidated upon query-induced activation. The model runs within software, based on a biological principle of degradation and recovery that has been honed by millions of years of evolution, i.e., intelligent forgetting, not arbitrary forgetting.

In the Needle in a Compressed Haystack experiment, M.A.K.S showed how well this worked. At a token usage ratio of 4.58× and under memory pressure at CRITICAL, FIFO deletion evicted the crucial item first and couldn't retrieve it. M.A.K.S degraded the item to GHOST status, stored it, retrieved it using keyword matching, re-consolidated it to active memory, and answered correctly.

From the ablation study, it was confirmed that the Ghost Zone and reconsolidation pipeline are the structural pillars of this system; take away either, and the system collapses altogether. The overhead benchmarks showed that the lazy evaluation of this system keeps the scoring cost below 1% of the LLM inference latency at 5,000 memory units, with speedup scaling. M.A.K.S is not yet an operational system. It is a prototype with four algorithmic limitations and three experimental challenges, each corresponding to a distinct research area. Moving from prototype to production will entail solving several difficult but solvable problems, including embedding-based semantic scoring, native KV cache support, practical benchmarking, and multi-trial statistical analysis.

The overall thesis of this paper is clear – memory management in AI agents is an important first-class systems issue, just as much as scoring functions, eviction policies and recovery processes are issues that have been studied in depth by systems scientists in relation to caching, garbage collection, and memory hierarchies. M.A.K.S is one such step towards this objective. The life cycle approach that it employs – score, compress, ghost and re-consolidate – may prove useful in many other domains as well.

REFERENCES

[1] Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., Wang, Z., and Chen, B. (2023). H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. arXiv:2306.14048.

[2] Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. (2023). Efficient Streaming Language Models with Attention Sinks. arXiv:2309.17453.

[3] Munkhdalai, T., Faruqui, M., and Gopal, S. (2024). Leave No Context Behind: Efficient

Infinite Context Transformers with Infini-attention. arXiv:2404.07143.

[4] Packer, C., Fang, V., Patil, S. G., Moon, K., Wooders, S., and Gonzalez, J. E. (2023). MemGPT: Towards LLMs as Operating Systems. arXiv:2310.08560.

[5] Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. (2023). Generative Agents: Interactive Simulacra of Human Behavior. arXiv:2304.03442.

[6] Ebbinghaus, H. (1885). Über das Gedächtnis: Untersuchungen zur experimentellen Psychologie. Duncker & Humblot, Leipzig. [Memory: A Contribution to Experimental Psychology. Translated by Ruger, H. A. and Bussenius, C. E., Teachers College, Columbia University, 1913.]

[7] Averell, L. and Heathcote, A. (2011). The form of the forgetting curve and the fate of memories. Journal of Mathematical Psychology, 55(1), 25–35.

[8] Leitner, S. (1972). So lernt man lernen: Der Weg zum Erfolg [How to Learn to Learn: The Way to Success]. Herder, Freiburg im Breisgau.

[9] Wozniak, P. A. and Gorzelanczyk, E. J. (1994). Optimization of repetition scheduling with the algorithm SM-2. Acta Neurobiologiae Experimentalis, 54(6), 59–67.

[10] Cepeda, N. J., Pashler, H., Vul, E., Wixted, J. T., and Rohrer, D. (2006). Distributed practice in verbal recall tasks: A review and quantitative synthesis. Psychological Bulletin, 132(3), 354–380.

[11] Nader, K., Schafe, G. E., and Le Doux, J. E. (2000). Fear memories require protein synthesis in the amygdala for reconsolidation after retrieval. Nature, 406(6797), 722–726.

[12] Schacter, D. L., Guerin, S. A., and St. Jacques, P. L. (2012). Memory distortion: an adaptive perspective. Trends in Cognitive Sciences, 15(10), 467–474. [Reviewed in computational contexts as a model of trace reactivation and reconsolidation.]

[13] Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., Dong, Y., Tang, J., and Li, J. (2023). LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding. arXiv:2308.14508.

[14] Shaham, U., Segal, E., Levy, I., Yu, X., Groeneveld, J., Zettlemoyer, L., and Srikumar, V. (2022). SCROLLS: Standardized CompaRison Over Long Language Sequences. arXiv:2201.03533.

[15] Shannon, C. E. (1948). A Mathematical Theory of Communication. Bell System Technical Journal, 27(3), 379–423.