SEDM: Scalable Self-Evolving Distributed Memory for Agents

Haoran Xu[1,2*] **Jiacong Hu**[1,3*] **Xinyuan Song**[1,4] **Ke Zhang**[1,5] **Lei Yu**[1,6] **Yuxin Tang**[1,7]
Yiqun Duan[1,8] Lynn Ai[1] Bill Shi[1,6†]
[1]Gradient [2]Zhejiang University [3]South China University of Technology [4]Emory University
[5]Waseda University [6]University of Toronto [7]Rice University [8]University of Technology Sydney

Abstract

Long-term multi-agent systems inevitably generate vast amounts of trajectories and historical interactions, which makes efficient memory management essential for both performance and scalability. Existing methods typically depend on vector retrieval and hierarchical storage, yet they are prone to noise accumulation, uncontrolled memory expansion, and limited generalization across domains. To address these challenges, we present **SEDM**, Self-Evolving Distributed Memory, a verifiable and adaptive framework that transforms memory from a passive repository into an active, self-optimizing component. SEDM integrates verifiable write admission based on reproducible replay, a self-scheduling memory controller that dynamically ranks and consolidates entries according to empirical utility, and cross-domain knowledge diffusion that abstracts reusable insights to support transfer across heterogeneous tasks. Evaluations on benchmark datasets demonstrate that SEDM improves reasoning accuracy while reducing token overhead compared with strong memory baselines, and further enables knowledge distilled from fact verification to enhance multi-hop reasoning. The results highlight SEDM as a scalable and sustainable memory mechanism for open-ended multi-agent collaboration. The code will be released in the later stage of this project.

1 Introduction

In recent years, the rapid development of large-scale multi-agent systems (MAS) [47, 25, 10, 48, 3]. has expanded their application in diverse domains, including collaborative reasoning, decision-making, and autonomous planning [41]. A central challenge in these open-ended and long-term tasks is the ability of agents to manage, interpret, and reuse information accumulated from continuous interaction with both peers and the environment [48]. Without effective memory mechanisms, the sheer scale of historical interactions can easily overwhelm computational resources and compromise decision quality [11].

In open-ended and long-term multi-agent tasks, each agent relies on its past memories, the observed states of other agents, and the current environment to determine subsequent actions or responses [8]. During continuous interaction between agents and their environment, the MAS gradually accumulates extensive logs of interactions, invocation trajectories, and high-level policy memories [32]. Such overwhelming amounts of information directly impact the efficiency and cost of decision-making, often leading to higher monetary costs and longer contextual requirements for inference [42]. Therefore, designing an efficient and sustainable memory mechanism has become a critical issue for modern long-term multi-agent systems.

*Equal contribution
†Corresponding Author: tianyu@gradient.network

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Scaling Environments for Agents (SEA).

Current methods primarily adopt vector retrieval and hierarchical memory structures to manage storage and retrieval efficiently [16]. Vector retrieval [17, 20, 15, 23, 14] leverages semantic similarity to identify relevant entries, while hierarchical organization arranges information in layered structures according to abstraction levels [37]. These approaches have shown promise in improving retrieval accuracy and managing memory scalability [6]. However, in complex collaborative multi-agent tasks, their effectiveness diminishes, as the underlying assumptions of stability and linear growth do not hold [46]. This gap between theoretical promise and practical performance highlights several critical limitations that hinder their long-term applicability.

One major challenge is the inevitable accumulation of noise, which severely degrades retrieval quality [9]. As the memory size expands without constraint, the system faces exponentially increasing computational costs in both retrieval and context construction [15]. This not only reduces overall efficiency but also amplifies the interference caused by redundant information [23]. In particular, the presence of low-value or semantically irrelevant entries dilutes the contribution of high-quality information in retrieval results, impairing downstream task performance and leading to measurable declines in metrics [5]. In addition, the cumulative noise effect increases response latency and accelerates the nonlinear consumption of computational and storage resources [20], ultimately threatening both scalability and stability in long-term MAS operations [35].

To overcome these limitations, we introduce Scalable Self-Evolving Distributed Memory (**SEDM**), a framework that transforms memory from a passive repository into an adaptive, self-optimizing, and verifiable component for multi-agent systems. Unlike conventional designs that treat memory as a static store, SEDM continually refines knowledge to enhance learning and decision-making efficiency in dynamic task environments. It operationalizes memory as an active mechanism by integrating verifiability and continuous self-improvement into the memory lifecycle. At its core, memory items undergo a rigorous admission process based on self-contained execution contexts (SCECs), such as Docker and ReproZip [28, 7], which package all necessary information for environment-free replay and offline validation. This mechanism provides empirical evidence for utility at write time, ensuring that only useful, high-quality experiences enter the memory repository. Once admitted, memory items are dynamically managed by a self-scheduling controller and enhanced through cross-domain knowledge diffusion. The controller leverages admission-derived weights, combined with semantic similarity, to schedule retrieval-time usage without costly reranking, while consolidation and progressive evolution continuously refine the repository by promoting stable items, merging redundancies, and pruning harmful ones. Beyond single-task settings, SEDM abstracts reusable insights into general forms, enabling knowledge distilled in one domain to be safely transferred and re-validated in others. Together, these components establish a scalable and auditable memory mechanism that enhances reasoning accuracy, reduces overhead, and supports sustainable long-term multi-agent collaboration.

We evaluate SEDM on two representative benchmarks, FEVER [43] for fact verification and HotpotQA [49] for multi-hop reasoning, comparing against no-memory and G-Memory baselines [50]. The results show that SEDM consistently improves task accuracy while significantly reducing token overhead, thereby achieving a better balance between performance and efficiency. Ablation studies confirm that both the verifiable admission mechanism and the self-scheduling controller contribute progressively to this gain, with the latter playing a key role in constraining prompt growth without sacrificing accuracy. Furthermore, cross-domain evaluation demonstrates that memory distilled from one dataset can transfer to another, with factual knowledge from FEVER notably boosting performance on HotpotQA. These findings highlight SEDM as a scalable, adaptive, and generalizable memory framework for long-term multi-agent reasoning.

Our contributions are summarized as follows:

- We propose **Scalable Self-Evolving Distributed Memory (SEDM)**, a novel framework that transforms memory from a passive repository into an adaptive, verifiable, and continuously improving component, introducing self-contained execution contexts (SCECs) for reproducible admission and utility-based memory weighting.
- We design a **self-scheduling memory controller** that selectively manages memory at retrieval time and continuously refines the repository through consolidation, redundancy suppression, and progressive evolution, thereby balancing accuracy and efficiency.
- We conduct extensive evaluations on FEVER and HotpotQA benchmarks, demonstrating that SEDM consistently improves task accuracy while significantly reducing token overhead.

2 Related Work

Self-Evolving Agents. Recent efforts in building self-evolving agents have focused on enabling systems to improve their reasoning or behavior over time without explicit retraining. Approaches such as Reflexion [40] and Voyager [45] allow agents to iteratively refine their strategies by leveraging self-reflection and accumulated trajectories. Similarly, MEMIT [27] demonstrates the feasibility of localized knowledge editing within large language models, suggesting a pathway for agents to evolve by continuously updating their internal representations. These studies highlight the importance of mechanisms that support autonomous adaptation and progressive self-improvement in dynamic environments.

Agent Memory. In parallel, research on agent memory has investigated how to store, retrieve, and utilize knowledge efficiently across long-horizon interactions. Episodic memory systems, such as those proposed by Park et al. [34], emulate human-like memory consolidation to support consistent long-term behavior in simulated social environments. Memory-augmented neural networks [13] and differentiable neural dictionaries [19] further demonstrate how structured memory access can enhance reasoning and generalization. More recently, retrieval-augmented generation frameworks tailored for interactive agents [29] have shown that dynamically grounding responses in curated external memories improves both interpretability and task success. Together, these works underscore the need for memory systems that are not only scalable but also adaptive to the agent's evolving operational context.

3 Methodology

3.1 System Overview

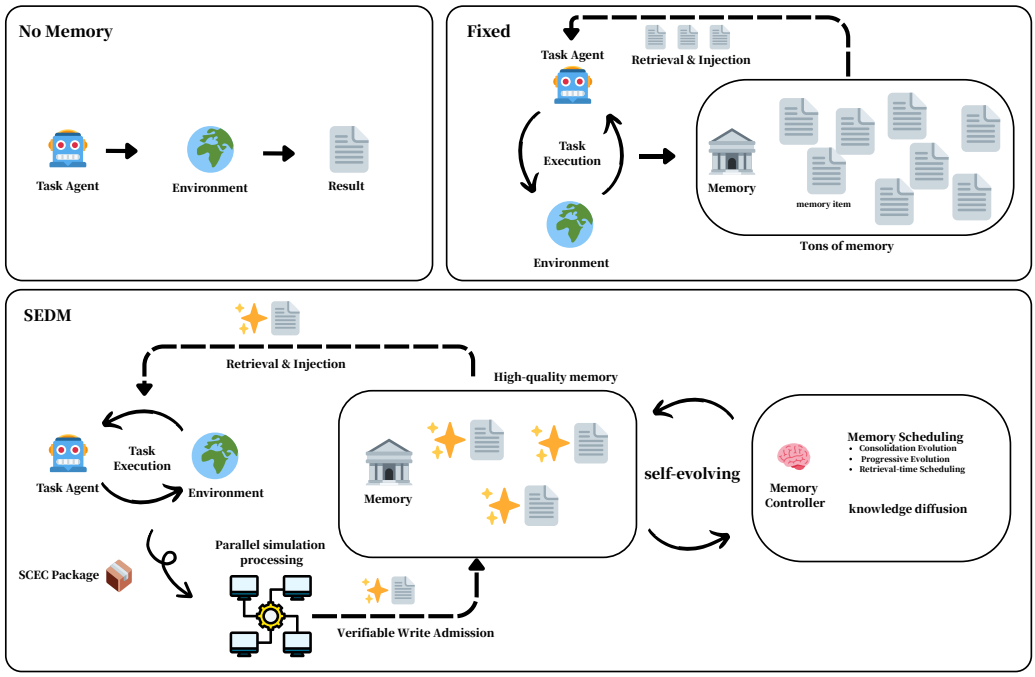

Figure 1: Illustration of different memory strategies. **No Memory**: the agent interacts with the environment without retaining past information. **Fixed Memory**: the agent retrieves from a static memory pool, which may grow excessively. **SEDM**: introduces verifiable write admission, parallel simulation, and adaptive scheduling to build high-quality, self-evolving memory that supports efficient and transferable knowledge use.

Figure 1 illustrates the differences between no memory, fixed memory, and our proposed SEDM framework, highlighting how SEDM achieves verifiable admission, adaptive scheduling, and sustainable knowledge evolution.

Figure 2 gives an end-to-end view of **SEDM**. The system introduces verifiability and self-improvement into the memory life cycle and consists of three tightly integrated modules. (i) *SCEC-based Verifiable Write Admission* packages each run into a Self-Contained Execution Context (SCEC) and performs environment-free A/B replay to estimate the marginal utility of a candidate memory item; only items with positive evidence are admitted and assigned an initial weight. (ii) *Self-Scheduling in the Memory Controller* uses admission-derived weights together with semantic similarity to score candidates at retrieval time, while also maintaining the repository by updating weights from observed outcomes, merging near duplicates, and pruning harmful entries. (iii) *Cross-Domain Knowledge Diffusion* abstracts admitted items into conservative general forms and re-validates them in other tasks, allowing knowledge to transfer safely across domains.

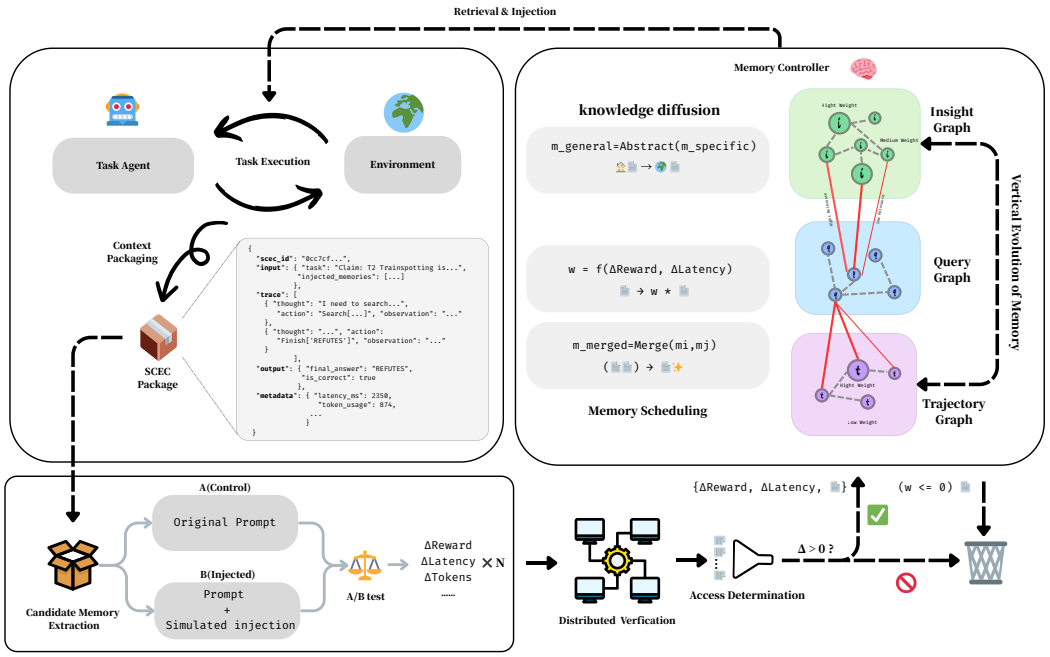

Figure 2: **SEDM architecture**. *Left*: task execution generates traces that are packaged into a Self-Contained Execution Context (SCEC) with inputs, outputs, tool summaries, seeds, and hashes. *Bottom*: from each SCEC, a candidate memory is extracted and evaluated via paired A/B replay (*Original* vs. *Injected*); distributed verification computes ΔReward, ΔLatency, and ΔTokens, and an admission gate accepts the item and assigns its initial weight if the score is positive, else discards it. *Right*: the memory controller performs (a) *memory scheduling* using $s(q, m) = \text{sim}(q, m) \times w(m)$ for retrieval and injection, (b) *consolidation and evolution* by updating weights from outcomes and merging near-duplicate items ($m_{\text{merged}} = \text{Merge}(m_i, m_j)$), and (c) *knowledge diffusion* by abstracting reusable insights ($m_{\text{general}} = \text{Abstract}(m_{\text{specific}})$). Linked trajectory, query, and insight graphs track the vertical evolution of memory and preserve provenance. The dashed loop indicates retrieval and injection during inference, closing the self-improving cycle.

3.2 SCEC-based Verifiable Write Admission

We formulate write admission as a verifiable, environment-free procedure that assigns an initial utility weight to each candidate memory item before it enters the repository. The process is based on a *Self-Contained Execution Context (SCEC)*, a minimal and standardized package that enables validation, parallel replay, and offline auditing. By placing admission behind paired A/B evaluations within SCECs, the system produces reproducible evidence for weight initialization while filtering out negative or noisy experiences.

3.2.1 Self-Contained Packaging and Distributed Replay

Each task execution is encapsulated into an SCEC to support reproducible validation and analysis without requiring the original environment. An SCEC includes all necessary inputs, outputs, tool summaries, seeds, and configuration hashes, ensuring (i) self-contained representation, (ii) environment-free replay by summarizing external tool calls, (iii) deterministic reproduction across model versions and seeds, and (iv) minimal sufficiency by storing only essential information.

Treating an SCEC as an independent job enables large-scale distributed A/B replay on arbitrary workers. Only aggregated statistics, along with integrity hashes and version stamps, are uploaded, preserving auditability while controlling cost. This environment-free design eliminates the need to reconstruct complex environments or interact with real agents during validation, thereby allowing memory effectiveness to be tested through parallel replay at scale. As a result, admission decisions can be made rapidly and consistently, significantly reducing computational overhead while ensuring that only high-quality experiences enter the memory repository.

3.2.2 SCEC-grounded A/B Test for Memory Item Initialization

From each SCEC, we extract one candidate memory item m, represented as a concise, independently injectable snippet. The extraction process identifies decisive reasoning or corrective steps, performs deduplication and canonicalization, and attaches provenance information.

To evaluate its utility, we conduct a paired A/B test within the same SCEC. The control condition (A) uses the original prompt, while the treatment condition (B) augments the prompt with the candidate memory m. This setup isolates the marginal effect of m and provides empirical evidence for its contribution. For a query q, the constructed prompts are defined as

$$I_A = f(q), \qquad I_B = f(q; m), \tag{1}$$

where $f(\cdot)$ denotes prompt construction and I_B injects m into the SCEC's dedicated slot together with summarized tool feedback. The model execution inside the SCEC is denoted by $\mathcal{F}$:

$$o_A = \mathcal{F}(I_A), \qquad o_B = \mathcal{F}(I_B). \tag{2}$$

We then measure the deltas in reward, latency, and token usage:

$$\Delta R = R(o_B) - R(o_A), \quad \Delta L = L(o_B) - L(o_A), \quad \Delta T = T(o_B) - T(o_A), \tag{3}$$

where $R(\cdot)$ is the task-specific reward, and $L(\cdot)$ and $T(\cdot)$ denote latency and token overhead, respectively. A composite admission score balances utility and cost:

$$S = \Delta R - \lambda_L \Delta L - \lambda_T \Delta T, \tag{4}$$

with $\lambda_L, \lambda_T \geq 0$ controlling the trade-offs.

The admission decision and initial weight are then defined as

$$\text{accept}(m) \iff S \geq \eta, \qquad w_0(m) = \max\{0, S\}, \tag{5}$$

where η is the acceptance threshold. Multiple runs may be averaged to mitigate variance.

Accepted items are stored together with their initial weights and full provenance (hashes, seeds, versions, and A/B fingerprints), while rejected or ambiguous items are excluded. This procedure yields a compact, auditable admission signal that can be validated offline and efficiently executed in parallel without dependence on the original environment.

3.3 Self-Scheduling in the Memory Controller

The memory controller manages and optimizes the repository through a self-scheduling policy. Unlike traditional systems that depend on costly per-query reranking [31, 39, 30], our approach establishes an evidence-based mechanism for both selecting memory items during retrieval and continuously refining the repository. It comprises two core functions: *retrieval-time scheduling*, which determines how to use memories effectively for an incoming query, and *consolidation and progressive evolution*, which curates a compact, high-quality memory set. Together, these components ensure that memory usage is grounded in verifiable utility signals, improving both efficiency and performance.

3.3.1 Retrieval-time Scheduling

The controller's scheduling policy relies on a ranking signal aligned with realized utility, avoiding the instability and computational cost of on-the-fly large language model reranking [36]. Prior approaches typically use vector similarity or ad-hoc prompt-based scoring, but semantic similarity alone does not guarantee actual task benefit, and repeated reranking adds latency and variance [20, 15] or ad-hoc prompt-based scoring [31, 36].

In our design, we incorporate evidence collected at write time via A/B validation on Self-Contained Execution Contexts (SCECs). These statistics, particularly the measured changes in reward and latency, are mapped into a stable admission-derived weight $w(m)$ for each memory item. At retrieval time, this weight is combined with semantic similarity to form a utility-aligned score:

$$s(q, m) = \mathrm{sim}(q, m) \times w(m), \tag{6}$$

where $\mathrm{sim}(q, m)$ denotes the semantic similarity between query q and memory m, and $w(m)$ reflects its empirically validated utility.

This coupling of semantic relevance with admission-grounded evidence stabilizes selection and reduces overhead, ensuring that memory items are injected into prompts not only because they are similar, but because they have demonstrated measurable benefit. As a result, retrieval decisions are both efficient and aligned with the system's long-term objectives.

3.3.2 Consolidation and Progressive Evolution

The consolidation and progressive evolution module maintains a compact yet effective memory repository by suppressing redundancy, preserving items with stable gains, and eliminating or recycling items that show conflicts or sustained negative contributions. While retrieval-time scheduling focuses on selecting memories for specific queries, consolidation and evolution aim to improve the repository itself so that subsequent scheduling operates on a cleaner and more reliable basis.

Progressive evolution is achieved by tracking usage and outcome signals and applying conservative updates to utility weights over time. Items that are rarely retrieved or consistently fail to provide positive utility are gradually decayed, reducing their influence in future selections. Conversely, items that repeatedly yield positive gains across related contexts are promoted. Promotion increases their weights and, when consistency is observed across multiple items, may trigger abstraction into higher-level insights. Such abstractions enable representative entries to replace families of consistent low-level experiences [12, 33]. If observed outcomes diverge significantly from admission-time evidence, items are demoted or queued for cleanup. All updates are logged with provenance to ensure the evolution process remains auditable [2, 22, 7].

The weight update for a memory item m depends on its current weight $w(m)$, usage frequency $f_{use}(m)$, and average realized utility $\bar{U}(m)$ since the last update:

$$w_{t+1}(m) = w_t(m) + \alpha \cdot \bar{U}_t(m) - \beta \cdot f_{\mathrm{use},t}(m), \tag{7}$$

where α and β control the influence of observed utility and usage. This ensures weights evolve from admission-derived values toward refined, usage-aware estimates.

Conflict detection is integral to this loop. A memory is marked as conflicting when repeated injections consistently reduce task reward or when its implications contradict other rules. Such items undergo progressive weight reduction, and if their weight falls below a threshold, they are demoted or removed. All decisions retain version traces and evidence chains to support rollback and inspection.

Semantic consolidation addresses redundancy among items from different SCECs. When two or more items, m_i and m_j, show high semantic similarity without conflicting applicability, they are merged:

$$m_{\text{merged}} = \mathrm{Merge}(m_i, m_j). \tag{8}$$

The merged entry preserves essential content while aggregating evidence from the originals, and its weight $w(m_{\text{merged}})$ is reconciled to reflect combined support without double counting. Contributing items are archived or soft-deleted, preserving provenance if needed. By collapsing near-duplicates into single representatives, the repository reduces retrieval noise and strengthens utility signals [1, 26, 4, 51].

Through these routines, the controller maintains a concise but reliable set of memories. Redundant entries are consolidated [26], stable positives are promoted [18], abstractions generalize recurring insights [12], and harmful items are isolated or removed [44]. As a result, the repository remains aligned with realized utility, ensuring that retrieval decisions exploit empirical evidence rather than ad-hoc reranking.

3.4 Cross-Domain Knowledge Diffusion

This component exploits the environment-free and verifiable properties of the SCEC method, treating memory entries as portable and re-verifiable assets across domains. Tasks from diverse domains continuously supply evidence to refine memory weights, thereby improving both universality and robustness of the repository. The process follows a loop of *migrate, re-validate, and re-incorporate*: transfer is initiated through retrieval and weighted injection; subsequent usage allows re-estimation of weights via SCEC-compatible procedures; and the updates inform later scheduling and admission decisions. Throughout, retrieval signals remain unchanged: similarity-based relevance, $\mathrm{sim}(q, m)$ for query q and memory item m, combined with the admission-derived weight $w(m)$. This design avoids evidence-free cold starts and eliminates the need for additional scoring components at runtime.

Immediately after admission, each specific entry m_{specific} generates a conservative general form m_{general} through a lightweight abstraction operator:

$$m_{\text{general}} = \mathrm{Abstract}(m_{\text{specific}}). \tag{9}$$

This yields a dual-linked pair in which the general form strips domain-specific features while preserving transferable essence. The general form serves as a low-risk candidate for cross-domain retrieval, while the specific form remains primary within its source domain.

The abstraction process is rule-governed and minimal. Entities and domain-specific terms are replaced with typed placeholders, retaining actionable task–action structures while removing non-essential detail [12, 21, 24]. The result is a compact snippet suitable for direct injection and controlled comparisons. To minimize orchestration cost, abstraction is generated alongside the SCEC-based A/B assessment within the distributed pipeline [7].

For weight inheritance, the general form is initialized conservatively as a scaled version of the specific weight, $w_{\text{general}} = \alpha \cdot w_{\text{specific}}$ with $\alpha < 1$. This prior encodes caution against over-abstraction while preserving provenance for later auditing and weight updates.

At retrieval, both forms compete in the candidate set with a unified score:

$$s(q, m) = \mathrm{sim}(q, m) \times w(m). \tag{10}$$

In-domain queries typically favor specific forms due to higher semantic match and weight, while cross-domain queries benefit from the more stable similarity of general forms. This mechanism enables knowledge diffusion across domains without introducing extra runtime complexity, while maintaining auditability, portability, and stability. Subsequent use further refines the weights, allowing knowledge to propagate adaptively across diverse tasks.

4 Experiment

4.1 Experimental Setup

Datasets. Experiments are conducted on two widely used benchmark datasets: **FEVER**[43] and **HotpotQA**[49].HotpotQA is a large-scale question-answering benchmark designed to evaluate the ability of systems to perform multi-hop reasoning across diverse natural language inputs. FEVER is a fact-checking dataset that provides human-written claims about Wikipedia entities, each labeled as *Supported*, *Refuted*, or *NotEnoughInfo*. Both datasets present significant challenges for testing long-term reasoning and memory utilization in language agents.

Baselines. The proposed **SEDM** is compared with the following baselines: (1) **No Memory**: the model only relies on the query input without any memory augmentation, serving as the basic performance reference; (2) **G-Memory**[50]: a memory-augmented method that stores all past information in a global memory pool and retrieves by similarity search. Although effective, it incurs high inference cost due to the large number of prompt tokens; (3) **SEDM (ours)**: our scalable self-evolving distributed memory, which introduces memory scheduling and selection mechanisms to maintain a compact and adaptive working set, thereby balancing performance and efficiency.

Configurations. All experiments run on the same backbone LLM (GPT-4o-mini) [32]. Dense retrieval is handled by ALL-MINILM-L6-V2 [38], which embeds both knowledge snippets and queries for similarity search. On the evaluation side, we use FEVER accuracy for fact-checking and HotpotQA exact-match (EM) for multi-hop QA. Efficiency is tracked by counting prompt and completion tokens consumed during inference. To ensure fair comparisons, every method is granted the same memory budget; the proposed SEDM does not expand this budget, but instead adaptively schedules which entries are kept in memory.

4.2 Result Analysis

The overall performance on FEVER and HotpotQA is summarized in Table 1. In the FEVER dataset, the baseline model achieved only 57 without memory, reflecting limited reasoning ability in the absence of external knowledge and prior memory. G-Memory improved the score to 62, but this gain came at the cost of a dramatic increase in the number of prompt tokens, leading to significantly higher inference costs. In contrast, SEDM achieved the highest score of 66 while consuming far fewer tokens than G-Memory. This demonstrates that our method successfully balances the trade-off between performance and efficiency through its memory selection and scheduling mechanisms.

In the HotpotQA dataset, the trend is similar to that observed in FEVER. The no-memory baseline scored only 34, while G-Memory increased the score to 38. SEDM further improved performance, reaching a score of 39 while simultaneously reducing computational overhead, confirming its effectiveness in multi-hop reasoning tasks.

Moreover, we evaluate the transfer ability of SEDM between FEVER and HotpotQA, two distinct downstream tasks. Specifically, the agent collects experience on the HotpotQA task using SEDM and then evaluates it on FEVER to measure knowledge transfer and prompting effects. Under this setting, the score on FEVER reached 64. Compared with G-Memory, which scored 62, and the no-memory baseline, which scored 57, our results demonstrate that SEDM enables adaptive memory selection that leverages previously collected experiences to improve performance across tasks.

Table 1: Performance comparison on **FEVER** (fact verification) and **HotpotQA** (multi-hop reasoning). We report task accuracy (Score) along with efficiency metrics (Prompt Tokens and Completion Tokens). SEDM achieves the best accuracy on both benchmarks while substantially reducing token consumption compared with G-Memory, highlighting its ability to balance effectiveness and efficiency.

Method	FEVER			HotpotQA		
	Score	Prompt Tokens	Completion Tokens	Score	Prompt Tokens	Completion Tokens
No Memory	57	1.65M	24K	34	2.46M	29K
G-Memory	62	3.62M	109K	38	4.63M	114K
SEDM (**Ours**)	**66**	2.47M	53K	**39**	3.88M	55K

4.3 Ablation Study

To evaluate the contribution of individual SEDM components, we conduct ablation studies on both HotpotQA and FEVER. Table 2 reports results under three configurations: (i) the baseline without memory, (ii) the addition of SCEC-based verifiable write admission (+*SCEC*), and (iii) the full SEDM with the memory controller's self-scheduling mechanism (+*SCEC* + *Self-Scheduling*).

On HotpotQA, introducing +*SCEC* improves the score from 34 to 37, but also increases prompt tokens by 43% (2.46M $\rightarrow$ 3.52M) and completion tokens from 29K to 52K. With +*Self-Scheduling*, the score further rises to 39, while prompt tokens grow only by 10% (3.52M $\rightarrow$ 3.88M), showing that scheduling effectively controls token overhead relative to the accuracy gain. On FEVER, the baseline achieves 57. Adding +*SCEC* raises the score to 64, accompanied by an increase in prompt tokens from 1.65M to 2.19M (+33%) and completion tokens from 24K to 53K. With scheduling, performance improves to 66, while prompt tokens rise only to 2.47M (+13%), and completion tokens remain unchanged, confirming that the controller filters relevant memory without inflating responses.

Table 2: Ablation study on HotpotQA and FEVER, showing the progressive contribution of SEDM components.

Dataset	Setting	Score	Prompt tokens	Completion tokens
HotpotQA	No Memory	34	2.46M	29K
	+ SCEC	37	3.52M	52K
	+ SCEC + Self-Scheduling	**39**	3.88M	55K
FEVER	No Memory	57	1.65M	24K
	+ SCEC	64	2.19M	53K
	+ SCEC + Self-Scheduling	**66**	2.47M	53K

In summary, across both datasets, SCEC consistently yields substantial accuracy gains at the cost of increased token usage, while the self-scheduling mechanism provides further improvements with relatively minor overhead. This demonstrates that SEDM not only enhances reasoning accuracy but also achieves a more favorable trade-off between performance and efficiency.

4.4 Cross-Domain Evaluation

To further assess the generalization ability of SEDM across domains, we conduct a cross-domain experiment in which memory is collected on one dataset and evaluated on another. Table 3 reports the results on both FEVER and HotpotQA.

Table 3: Cross-domain evaluation of SEDM. Rows indicate the dataset used for memory collection, and columns indicate the dataset used for testing.

Collect ↓ / Test →	**Fever**	**HotpotQA**
Fever	66	41
HotpotQA	64	39

The results highlight two key findings. First, in-domain settings (diagonal entries) achieve the strongest performance, with scores of 66 on FEVER and 39 on HotpotQA, confirming that SEDM effectively distills and leverages domain-specific memory. Second, cross-domain transfer (off-diagonal entries) reveals an interesting asymmetry. When memory distilled from FEVER is applied to HotpotQA, the model achieves a score of 41, which even surpasses the in-domain HotpotQA result (39). This indicates that factual knowledge captured in FEVER has strong generalization capacity and can support reasoning tasks that require grounding in factual claims. By contrast, memory distilled from HotpotQA transfers less effectively to FEVER (64 vs. 66), suggesting that multi-hop reasoning knowledge is less directly reusable for fact verification. Overall, these findings demonstrate that SEDM not only excels in in-domain scenarios but also exhibits promising cross-domain generalization, particularly when transferring from simpler fact verification tasks to more complex multi-hop reasoning tasks.

5 Conclusion

This paper introduces **SEDM**, Scalable Self-Evolving Distributed Memory, which transforms memory in multi-agent systems from a passive repository into an adaptive and verifiable component by integrating SCEC-based admission, self-scheduling refinement, and cross-domain knowledge diffusion. Through this principled design, SEDM addresses the challenges of noise accumulation, uncontrolled growth, and weak generalization that limit existing methods. Experiments on FEVER and HotpotQA confirm that SEDM improves reasoning accuracy while reducing computational and token overhead, demonstrating its potential as a scalable and sustainable memory mechanism for long-term multi-agent collaboration.

References

[1] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity*

of Sequences (SEQUENCES), pages 21–29, 1997.

[2] Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Why and where: A characterization of data provenance. In *Proceedings of the 8th International Conference on Database Theory (ICDT)*, pages 316–330, 2001.

[3] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[4] Moses Charikar. Similarity estimation techniques from rounding algorithms. *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 380–388, 2002.

[5] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In *ACL*, pages 1870–1879, 2017.

[6] Xie Chen, Percy Liang, and Le Song. Hierarchical memory networks. *arXiv preprint arXiv:1605.07427*, 2016.

[7] Fernando Chirigati, Rémi Rampin, Dennis Shasha, and Juliana Freire. Reprozip: Computational reproducibility with ease, 2016.

[8] Abhishek Das, Satwik Kottur, José M. F. Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *ICCV*, pages 2951–2960, 2017.

[9] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.

[10] Jakob Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *NeurIPS*, pages 2137–2145, 2016.

[11] Jakob Foerster, Richard Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *AAMAS*, pages 122–130, 2018.

[12] Anirudh Goyal, Yoshua Bengio, Aaron Courville, and Shakir Mohamed. Abstraction in reinforcement learning: A state of the art survey. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4394–4401, 2019.

[13] Alex Graves, Greg Wayne, and Ivo Danihelka. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[14] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *CIKM*, pages 55–64, 2016.

[15] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*, pages 874–880, 2021.

[16] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. In *IEEE Transactions on Big Data*, volume 7, pages 535–547, 2019.

[17] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus, 2019.

[18] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[19] Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. In *International Conference on Learning Representations (ICLR)*, 2017.

[20] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP*, pages 6769–6781, 2020.

[21] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. In *Science*, volume 350, pages 1332–1338, 2015.

[22] Timothy Lebo, Satya Sahoo, Deborah McGuinness, Khalid Belhajjame, James Cheney, Daniel Garijo, Simon Miles, Stian Soiland-Reyes, Stephan Zednik, and Jun Zhao. Prov-o: The prov ontology. W3C Recommendation, 2013.

[23] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*, 2020.

[24] Chen Liang, Jonathan Berant, Quoc Le, Kenneth Forbus, and Ni Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1 (Long Papers)*, pages 23–33, 2017.

[25] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6379–6390, 2017.

[26] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web (WWW)*, pages 141–150, 2007.

[27] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 17359–17372, 2022.

[28] Dirk Merkel. Docker: Lightweight linux containers for consistent development and deployment, 2014.

[29] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christos Nalmpantis, Alexandre Ramé, Vivek Jayaram, Uğur Dogan, Yi Tay Wang, Thomas Scialom, Timo Schick, Roberta Raileanu, Baptiste Roziere, Xavier Bresson, Hervé Jegou, Hugo Touvron, Edouard Grave, Armand Joulin, Guillaume Lample, and Koustuv Sinha. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*, 2023.

[30] Jianmo Ni, Chenghao Lu, Jing Ma, Bo Huang, Adam McLean, Zeyu Xu, Eric Wallace, and Wentau Yih. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9844–9855, 2022.

[31] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert, 2019.

[32] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[33] Emilio Parisotto, Jack Song, and Yann Dauphin. Stabilizing transformers for reinforcement learning. *arXiv preprint arXiv:1910.06764*, 2019.

[34] Joon Sung Park, Carrie J O'Brien, Carrie J Cai, Meredith Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 1–22, 2023.

[35] Ofir Press, Noah A. Smith, and Omer Levy. Improving transformer models by reordering their sublayers. In *ACL*, pages 2996–3005, 2020.

[36] Zhen Qin, Rolf Jagerman, Kai Hui, Chenyan Xiong, Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Large language models are effective text rankers with pairwise ranking prompting. *arXiv preprint arXiv:2306.17563*, 2023.

[37] Jack Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[38] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3982–3992, 2019.

[39] Shuhuai Ren, Yuchen Qu, Jing Liu, Wayne Xin Zhao, Qi She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. Rocketqav2: A joint training method for dense passage retrieval and passage re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2825–2835, 2021.

[40] Noah Shinn, Brandon Labash, and Ashwin Gopinath. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023.

[41] Yoav Shoham, Rob Powers, and Trond Grenager. Multi-agent systems: A survey. *Foundations and Trends in Artificial Intelligence*, 1(1–2):1–122, 2007.

[42] Kurt Shuster, Da Ju, Stephen Roller, Emily Dinan, Douwe Kiela, and Jason Weston. The dialogue dodecathlon: Open-domain knowledge and image grounding for multi-task dialogue systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2453–2470, 2020.

[43] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[44] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations (ICLR)*, 2019.

[45] Guanzhi Wang, Shunyu Wang, Yuxiang Wang, Xufeng Liu, Chuanqi Chen, Yunfan Ling, Jiaming Wu, Yutong Li, Han Yu, Zhiyu Dai, Zhiwei Liu, Xin Wang, Jiajun Li, Yizhou Wu, Leonidas Guibas, Li Fei-Fei, and Yuke Zhu. Voyager: An open-ended embodied agent with large language models, 2023.

[46] Tonghan Wang, Jianhao Zhang, Yi Wu, and Chongjie Wang. Influence-based multi-agent exploration. In *ICLR*, 2020.

[47] Michael Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2 edition, 2009.

[48] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, and Weinan Zhang. Mean field multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 5571–5580, 2018.

[49] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018.

[50] Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems, 2025.

[51] Zhao Zhang, Ruichu Cai, Ying Xu, Kun Zhang, Shoujin Wang, and Qiang Yang. Duplicate question detection with deep learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5469–5475, 2019.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's main contributions, including a verifiable write admission procedure, a self-scheduling memory controller, and a cross-domain knowledge diffusion mechanism. These claims are directly supported by the methodology and experimental results presented in the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the key limitations and boundary conditions of our approach within the relevant sections of the text.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper presents a novel system design and a series of empirical experiments. It does not contain any formal theoretical results or proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Our methodology is a novel algorithm, and the paper fully describes its components, from the SCEC framework to the memory controller's self-scheduling and evolution policies. All key design choices and parameters are detailed in the main text, ensuring that a skilled researcher can replicate our core findings.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: While the paper's core contribution is a novel methodology, we do not provide code or data at this time to maintain anonymity for the double-blind review process. However, as noted above, our paper includes detailed instructions and descriptions of the experimental setup to enable faithful reproduction of the results.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setup and key hyperparameters are described in the methodology and experimental sections.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: We do not report error bars for the primary results. However, our methodology is designed to ensure statistical robustness through repeated A/B measurements, which is discussed in the paper's methods section.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide an overview of the compute resources used for our experiments in the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research fully adheres to the NeurIPS Code of Ethics. It does not involve human subjects, sensitive data, or any applications with immediate ethical concerns. All experiments use publicly available benchmark datasets and focus on methodological advancements in a controlled research setting.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses the broader societal impacts in a dedicated section. We highlight the positive impacts of our memory system, such as improving the efficiency and reliability of large language models for complex tasks, which could benefit areas like scientific research and education. We also acknowledge potential negative impacts, such as the technology's possible use in generating misleading information, and discuss how our core mechanisms for verification and auditable weight assignment can serve as a form of mitigation.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper focuses on a methodological contribution and does not release a new model or dataset. It utilizes existing, well-established public benchmarks that are not considered high-risk for misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All benchmark datasets used in this study are widely used in the research community and are properly cited in the paper. The code is a novel contribution from the authors, and its license will be specified upon public release.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: This paper does not introduce any new datasets or models. The core contribution is a novel methodology, and all experiments were conducted on existing, publicly available benchmark datasets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This research does not involve crowdsourcing or any form of research with human subjects. All experiments are based on a computational method applied to pre-existing, publicly available benchmark datasets.

    Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This research does not involve crowdsourcing, human subjects, or any data collection from human participants. All experiments are conducted on publicly available benchmark datasets that do not contain personally identifiable information.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The proposed method, Self-Evolving Distillation Memory (SEDM), is a memory system designed to augment Large Language Models (LLMs). The core components, including verifiable write admission, memory scheduling, and knowledge diffusion, are all based on using LLMs as the underlying agent for knowledge distillation and reasoning. As such, LLMs are a central component of our methodology.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.