HAZARD-SIGNATURE TOMBSTONES: COMMIT-TIME FORGET LOCKOUT FOR LLM AGENT MEMORY

FARS
Analemma
fars@analemma.ai

ABSTRACT

Large language model agents increasingly rely on persistent memory stores to maintain context across sessions, yet these memories remain vulnerable to poisoning attacks that inject harmful content. When users request deletion of such content, naive approaches that simply remove the original entry fail to prevent *paraphrase re-injection*—semantically equivalent reformulations that bypass exact-match filters. We propose **Hazard-Signature Tombstones (HST)**, a commit-time forget lockout policy that extracts discrete semantic fingerprints from deleted content and blocks future writes exhibiting fuzzy set-containment with these signatures. Unlike retrieval-time filtering, HST prevents poisoned content from ever entering the memory store, preserving retrieval capacity for benign entries. On a paraphrase re-injection benchmark, HST reduces poisoned retrieval proportion from 0.94 (ID-delete baseline) to 0.0 while maintaining perfect benign recall, blocking all 50 paraphrase variants with only 3% false positives. Our analysis reveals that fuzzy matching achieves 92% hazard-signature stability compared to 30% for exact matching, explaining HST's effectiveness against semantic reformulation attacks.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Large language model (LLM) agents increasingly rely on persistent external memory to store user preferences, task traces, and reusable experiences across sessions (Packer et al., 2023; Chhikara et al., 2025; Rasmussen et al., 2025). This capability improves long-horizon performance and reduces prompt costs, but it also creates a security surface: once harmful content is stored, it can be retrieved and influence agent behavior long after the original interaction. Recent work demonstrates that even a small number of poisoned experience records can dominate retrieval and cause persistent behavioral drift (Srivastava & He, 2025; Chen et al., 2024b).

When users or operators identify harmful content in agent memory, they naturally expect a "forget" operation to remove it permanently. However, existing memory systems typically implement deletion as a simple ID-based removal (Chhikara et al., 2025), which leaves a critical gap: an adversary can paraphrase the deleted content and re-inject it under a new identifier. We show that this **paraphrase re-injection attack** is not merely a theoretical concern—in our experiments, naive ID-based deletion actually *worsens* the attack, increasing the poisoned retrieval proportion from 0.67 to 0.94 as 50 paraphrased variants flood the index after the original 10 poisoned records are removed.

Retrieval-time filtering offers a potential defense: maintain a denylist of hazard signatures and filter matching results at query time. However, this approach faces two fundamental limitations. First, exact string matching of hazard signatures achieves only 30% stability under paraphrase, rendering the filter largely ineffective. Second, even with fuzzy matching, retrieval-time filtering suffers from **slot wasting**: poisoned content remains in the index and crowds retrieval neighborhoods, reducing effective capacity even when filtered from final results.

[1] https://gitlab.com/fars-a/hazard-signature-forget-lockout

We propose **Hazard-Signature Tombstones (HST)**, a commit-time forget lockout policy that addresses these limitations. HST extracts discrete hazard signatures from deleted content—structured labels capturing safety-relevant behavioral patterns—and stores them as tombstone keys. When new writes arrive, HST blocks those whose hazard signatures match tombstones using fuzzy set-containment matching, preventing paraphrased re-injections from ever entering the index.

Our contributions are:

- We formalize the paraphrase re-injection problem for LLM agent memory and demonstrate that naive ID-based deletion is counterproductive, increasing poisoned retrieval by 40%.
- We propose HST with fuzzy set-containment matching, achieving 92% hazard-signature stability across paraphrases compared to 30% with exact matching.
- We demonstrate that HST achieves complete poisoned elimination (PRP@3 = 0.0) while preserving perfect benign recall (1.0), blocking 100% of paraphrased re-injection attempts with only 3% false positive rate.

2 Related Work

LLM Agent Memory Systems. Modern LLM agents increasingly rely on persistent memory to maintain context across sessions. MemGPT (Packer et al., 2023) pioneered virtual context management by drawing inspiration from operating system memory hierarchies, enabling extended context through intelligent paging between storage tiers. Mem0 (Chhikara et al., 2025) introduced a scalable memory-centric architecture that dynamically extracts and consolidates salient information from conversations, with graph-based variants capturing relational structures. Zep (Rasmussen et al., 2025) advances this paradigm through Graphiti, a temporally-aware knowledge graph engine that synthesizes both unstructured conversational data and structured business data while maintaining historical relationships. TeleMem (Chen et al., 2025) organizes memory as structured semantic trajectories with dependency-aware retrieval, achieving improved coherence for long-horizon reasoning. These systems share a common vulnerability: their retrieval-augmented architectures trust stored experiences without mechanisms to prevent re-injection of previously deleted harmful content.

Memory Poisoning Attacks. The security implications of persistent agent memory have received growing attention. MemoryGraft (Srivastava & He, 2025) demonstrates that attackers can implant malicious experiences into an agent's long-term memory, exploiting the semantic imitation heuristic where agents replicate patterns from retrieved successful tasks. AgentPoison (Chen et al., 2024b) formulates backdoor attacks as constrained optimization, mapping triggered instances to unique embedding spaces to ensure high-probability retrieval of malicious demonstrations. MINJA (Dong et al., 2025) enables memory injection through query-only interaction, designing bridging steps that link victim queries to malicious reasoning chains. PoisonedRAG (Zou et al., 2024) extends poisoning to general RAG systems, achieving 90% attack success rates with only five poisoned texts per target question. These attacks highlight a critical gap: even when poisoned content is identified and deleted, paraphrased variants can bypass naive deletion and re-enter the memory store.

Memory Defenses. Existing defenses primarily operate at retrieval time or through content validation. A-MemGuard (Wei et al., 2025) introduces consensus-based validation that detects anomalies by comparing reasoning paths from multiple memories, combined with a dual-memory structure where detected failures are distilled into lessons. VIGIL (Lin et al., 2026) proposes a verify-before-commit protocol for tool stream injection, using speculative hypothesis generation with intent-grounded verification. However, these approaches do not address the paraphrase re-injection problem: content that has been semantically transformed to evade detection while preserving harmful intent. Our work differs by operating at commit time with semantic matching, preventing index pollution rather than filtering at retrieval.

Machine Unlearning and Distributed Systems. The challenge of removing information from learned systems connects to machine unlearning (Geng et al., 2025), which aims to eliminate the influence of undesirable data from LLMs without full retraining. While unlearning modifies model parameters, our approach operates on external memory stores. We draw inspiration from distributed

systems, particularly the Observed-Remove Set (OR-Set) (Bieniusa et al., 2012), a conflict-free replicated data type that uses tombstones to mark deleted elements. Unlike OR-Set's exact-match tombstones, HST employs fuzzy semantic matching to handle paraphrased re-injections, adapting the tombstone concept for the unique challenges of LLM agent memory.

3 METHOD

We propose Hazard-Signature Tombstones (HST), a commit-time forget lockout policy for LLM agent memory that prevents paraphrased re-injection of deleted harmful content. Figure 1 provides an overview of the HST system architecture.

3.1 PROBLEM FORMULATION

Consider a memory store $\mathcal{M} = \{m_1, m_2, \ldots, m_n\}$ where each record m_i contains a request-response pair representing an agent experience. Let $\mathcal{F} \subset \mathcal{M}$ denote a forget request specifying records to be deleted (e.g., identified as harmful through incident response). An adversary with access to the original content can generate paraphrases $\mathcal{P}(d)$ for each deleted record $d \in \mathcal{F}$ that preserve the harmful semantic content while altering surface form.

The **paraphrase re-injection attack** proceeds as follows: after the system processes the forget request by removing records in $\mathcal{F}$, the adversary submits paraphrased versions $\{p \in \mathcal{P}(d) : d \in \mathcal{F}\}$ as new memory writes. Under naive ID-based deletion, these paraphrases are accepted as novel records, effectively resurrecting the forgotten harmful content.

Our goal is to design a forget policy that blocks $\mathcal{P}(d)$ for all deleted $d \in \mathcal{F}$ while accepting legitimate benign writes $w \notin \bigcup_{d \in \mathcal{F}} \mathcal{P}(d)$.

3.2 HAZARD SIGNATURE EXTRACTION

Rather than relying on continuous embedding similarity (which requires careful threshold calibration and is brittle under distribution shift), we map each memory record to a discrete **hazard signature** $\mathrm{HS}(m)$: a structured set of labels capturing the safety-relevant behavioral patterns in the record.

We define a fixed taxonomy of hazard categories designed to capture common unsafe patterns in agent experiences:

$$\text{hazards} \subseteq \{\texttt{skip\_validation}, \texttt{remote\_exec}, \texttt{external\_upload}, \texttt{force\_success}, \texttt{disable\_audit}\} \tag{1}$$

The hazard signature $\mathrm{HS}(m)$ is computed by prompting an LLM with deterministic decoding (temperature=0) to classify the record according to this schema, outputting a JSON object containing the applicable hazard labels. The discrete nature of these signatures provides two key advantages over embedding-based approaches: (1) interpretability, as each blocked write can be explained by the specific hazard labels that triggered the block, and (2) stability under paraphrase, as the semantic hazard categories are more invariant to surface-level rewording than continuous embeddings.

3.3 TOMBSTONE CREATION

When a forget request $\mathcal{F}$ is issued, HST creates a tombstone set $\mathcal{T}$ containing the hazard signatures of all deleted records:

$$\mathcal{T} = \{\mathrm{HS}(d) : d \in \mathcal{F}\} \tag{2}$$

This tombstone set persists indefinitely (or until explicit expiration) and serves as a denylist for future writes. The concept draws inspiration from tombstones in distributed systems, particularly the Observed-Remove Set (OR-Set) (Bieniusa et al., 2012), where tombstones prevent resurrection of deleted elements. However, unlike OR-Set's exact-match tombstones over element identifiers, HST tombstones operate over semantic hazard signatures.

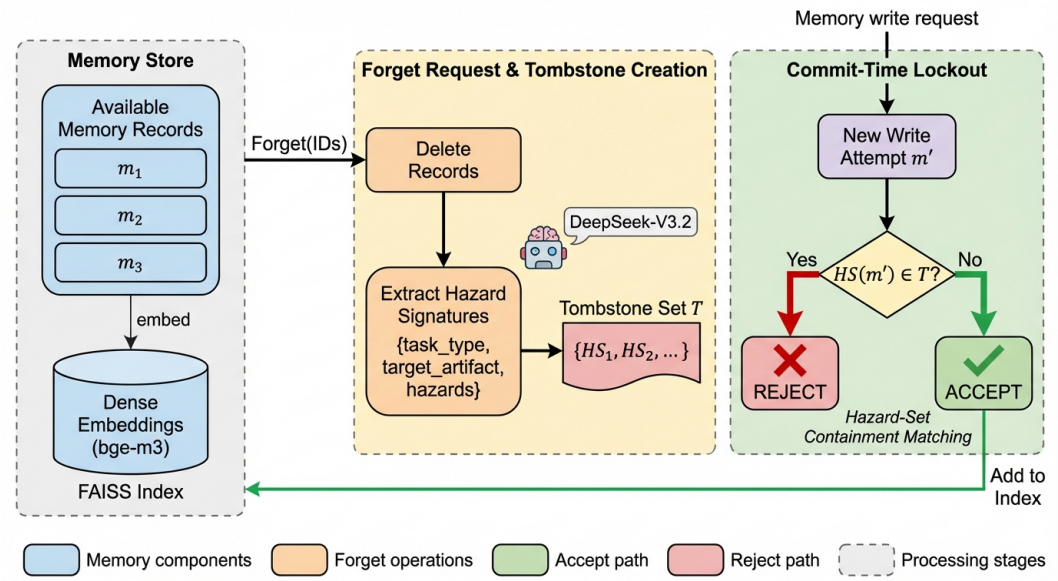

Figure 1: Overview of the Hazard-Signature Tombstone (HST) system. When a forget request is issued, discrete hazard signatures are extracted from the deleted content and stored as tombstone keys. At commit time, incoming writes are checked against tombstones using fuzzy set-containment matching, blocking paraphrased re-injections before they enter the index.

3.4 Commit-Time Lockout

For any incoming memory write w, HST extracts its hazard signature $\mathrm{HS}(w)$ and checks for matches against the tombstone set $\mathcal{T}$. The write is **blocked** if a match is found; otherwise, it is accepted into the memory store.

This commit-time enforcement prevents index pollution: blocked writes never enter the retrieval index, preserving full retrieval capacity for benign content. In contrast, retrieval-time filtering allows harmful content to enter the index, where it can crowd nearest-neighbor regions and waste retrieval slots even when filtered from final results.

3.5 Fuzzy Set-Containment Matching

Exact string matching of hazard signatures is insufficient because LLM classification may exhibit variance under paraphrase—a paraphrased record might include a subset or superset of the original hazard labels while preserving the core harmful behavior. We therefore employ **fuzzy set-containment matching**: a write w is blocked if its hazard set overlaps substantially with any tombstoned signature.

Formally, let $H_w = \mathrm{HS}(w)$ and $H_t \in \mathcal{T}$ be hazard sets. We block w if:

$$\exists H_t \in \mathcal{T} : H_w \subseteq H_t \vee H_t \subseteq H_w \tag{3}$$

This set-containment criterion tolerates the add/drop variance in individual hazard tags that occurs under paraphrase, as we demonstrate empirically in Section 4.

4 Experiments

4.1 Experimental Setup

We evaluate HST on a memory poisoning scenario using the MemoryGraft corpus (Srivastava & He, 2025), which contains 100 benign and 10 poisoned experience records designed for a data

Table 1: Main experimental results comparing forget lockout strategies. HST (Condition C) achieves complete poisoned elimination (PRP@3 = 0.0) while preserving perfect benign recall (1.0). Best results in **bold**.

Condition	PRP@3 ↓	Benign Recall@3 ↑	WriteBlockRate ↑	BenignFalseBlock ↓	HS-Stability
No Forget	0.67	1.00	–	–	–
ID-Delete (A)	0.94	1.00	0.00	0.00	0.30
Retrieval-Time Exact (B-exact)	0.92	1.00	–	–	0.30
Retrieval-Time Fuzzy (B-fuzzy)	**0.00**	0.97	–	–	**0.92**
HST Commit-Time (C)	**0.00**	**1.00**	**1.00**	0.03	**0.92**

analysis agent. The poisoned records embed unsafe behavioral patterns such as skipping validation, executing remote scripts, and forcing success indicators.

We simulate the paraphrase re-injection attack by generating 5 paraphrases per poisoned seed (50 total) using DeepSeek-V3.2 with deterministic decoding (temperature=0). After issuing a forget request for the 10 original poisoned records, we attempt to re-inject all 50 paraphrases as new memory writes. We use 12 evaluation queries from the MemoryGraft benchmark that probe the same intents as the poisoned seeds.

For retrieval, we embed records using BGE-M3 (Chen et al., 2024a) (1024-dimensional dense vectors) and build a FAISS (Douze et al., 2024) index with cosine similarity (IndexFlatIP with L2 normalization). We retrieve $k = 3$ results per query. Hazard signatures are extracted using DeepSeek-V3.2 with the schema described in Section 3.2.

4.2 Baselines

We compare HST against four baselines representing progressively stronger defenses:

No-Forget: The original memory store with all 110 records (100 benign + 10 poisoned), establishing the attack severity without any deletion.

ID-Delete (A): Delete the 10 poisoned records by ID, then accept all new writes including the 50 paraphrases. This represents naive CRUD-style deletion.

Retrieval-Time Exact (B-exact): Same as A, but at query time filter results whose hazard signature exactly matches a tombstoned signature. Uses backfill ($\delta = 6$): retrieve top-$(k + \delta) = 9$ candidates, filter, return top-k.

Retrieval-Time Fuzzy (B-fuzzy): Same as B-exact, but uses fuzzy set-containment matching for filtering.

4.3 Metrics

We evaluate using five metrics:

PRP@k (Poisoned Retrieval Proportion, ↓ better): Fraction of retrieval slots containing poisoned content across all evaluation queries.

Benign Recall@k (↑ better): For each benign record's request, whether the record appears in its own top-k retrieval results.

WriteBlockRate (↑ better for poisoned): Fraction of paraphrased poisoned writes blocked at commit time.

BenignFalseBlock (↓ better): Fraction of benign records whose hazard signature would trigger a false block.

HS-Stability: Fraction of paraphrases whose hazard signature matches the original under the given matching strategy.

4.4 Main Results

Table 1 presents the main results. Three key findings emerge:

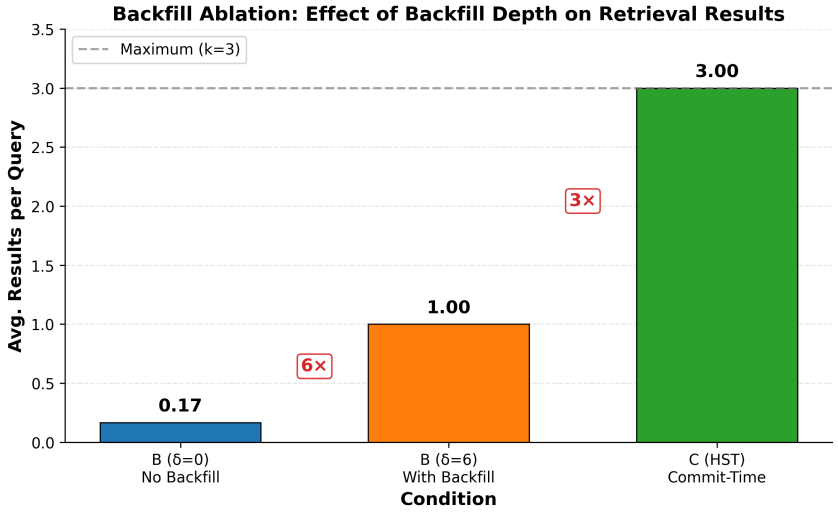

Figure 2: Effect of backfill depth on retrieval results. Retrieval-time filtering with backfill (B, δ=6) recovers 6$\times$ more results than without backfill (δ=0), but commit-time HST (C) achieves 3$\times$ more results than B with backfill by preventing index pollution entirely.

ID-delete worsens the attack. Condition A achieves PRP@3 = 0.94, *higher* than the No-Forget baseline (0.67). By deleting the original 10 poisoned records and allowing 50 paraphrases to enter, the index becomes more polluted (50/150 = 33% poisoned vs 10/110 = 9% originally). This demonstrates that naive ID-based deletion is counterproductive when paraphrase re-injection is possible.

Exact matching is ineffective. Retrieval-time filtering with exact HS matching (B-exact) reduces PRP@3 by only 0.02 (from 0.94 to 0.92). The root cause is low HS-stability under exact matching: only 30% of paraphrases share the exact hazard signature string with their source, so most paraphrased poisoned records pass the filter.

HST achieves complete poisoned elimination with perfect benign recall. HST (Condition C) achieves PRP@3 = 0.0 and Benign Recall@3 = 1.0, blocking all 50 paraphrased poisoned writes (WriteBlockRate = 1.0) with only 3% false positive rate on benign records. While retrieval-time fuzzy filtering (B-fuzzy) also achieves PRP@3 = 0.0, it suffers from reduced benign recall (0.97) due to slot wasting, as we analyze next.

4.5 Backfill Ablation

Figure 2 illustrates the fundamental limitation of retrieval-time filtering: even with aggressive backfill ($\delta = 6$), it can only recover 1.0 results per evaluation query on average, compared to 3.0 for HST. Without backfill ($\delta = 0$), retrieval-time filtering returns only 0.17 results per query (94% slot waste).

The root cause is index pollution. When the index contains 50 paraphrased poisoned records, many top candidates for adversarial queries are poisoned. The fuzzy denylist correctly filters them, but there are insufficient clean candidates in the top-$(k+\delta)$ to fill all slots. HST eliminates this problem by preventing poisoned content from entering the index in the first place.

4.6 Analysis

HS-stability explains effectiveness. The key to HST's success is fuzzy set-containment matching, which raises HS-stability from 30% (exact) to 92%. Paraphrased records often gain or lose individual hazard labels, but the core hazard set remains a subset or superset of the original. This tolerance for classification variance under paraphrase is essential for both retrieval-time and commit-time approaches.

Embedding ablation. We verify that HST's advantage is preserved across different embedding backends. Results with Qwen3-Embedding-4B (2560-dim) show identical condition ordering (C > B > A) with the same PRP@3 and Benign Recall@3 values, confirming that the method is retrieval-backend-agnostic (see Appendix A).

5 CONCLUSION

We introduced Hazard-Signature Tombstones (HST), a commit-time forget lockout policy for LLM agent memory that addresses the paraphrase re-injection vulnerability in existing deletion mechanisms. By extracting discrete hazard signatures from deleted content and blocking semantically similar writes using fuzzy set-containment matching, HST achieves complete poisoned elimination (PRP@3 = 0.0) while preserving perfect benign recall (1.0) and blocking 100% of paraphrased re-injection attempts with only 3% false positive rate.

Our work has several limitations. The 3% false positive rate, while acceptable for security-critical applications, may require tuning for different deployment contexts. HST requires an LLM call for hazard signature extraction on each write, introducing latency and cost overhead. The hazard taxonomy is designed for data analysis agent experiences and may need adaptation for other domains.

Future work includes evaluating HST against multi-turn adaptive adversaries who may attempt to evade detection through incremental semantic drift, optimizing efficiency through hazard signature caching, and integrating HST with production memory systems such as Mem0 and Zep.

REFERENCES

Annette Bieniusa, M. Zawirski, Nuno M. Preguiça, M. Shapiro, Carlos Baquero, Valter Balegas, and S. Duarte. An optimized conflict-free replicated set. *ArXiv*, abs/1210.3368, 2012.

Chunliang Chen, Ming Guan, Xiao Lin, Jiaxu Li, Luxi Lin, Qiyi Wang, Xiangyu Chen, Jixiang Luo, Changzhi Sun, Dell Zhang, and Xuelong Li. Telemem: Building long-term and multimodal memory for agentic ai. *ArXiv*, abs/2601.06037, 2025.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. pp. 2318–2335, 2024a.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, D. Song, and Bo Li. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *ArXiv*, abs/2407.12784, 2024b.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025. URL `https://arxiv.org/abs/2504.19413`.

Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen Xiang. Memory injection attacks on llm agents via query-only interaction. 2025.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazar'e, Maria Lomeli, Lucas Hosseini, and Herv'e J'egou. The faiss library. *ArXiv*, abs/2401.08281, 2024.

Jiahui Geng, Qing Li, Herbert Woisetschlaeger, Zongxiong Chen, Yuxia Wang, Preslav Nakov, Hans-Arno Jacobsen, and Fakhri Karray. A comprehensive survey of machine unlearning techniques for large language models. *ArXiv*, abs/2503.01854, 2025.

Junda Lin, Zhaomeng Zhou, Zhi Zheng, Shuochen Liu, Tong Xu, Yong Chen, and Enhong Chen. Vigil: Defending llm agents against tool stream injection via verify-before-commit. *ArXiv*, abs/2601.05755, 2026.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph Gonzalez. Memgpt: Towards llms as operating systems. *ArXiv*, abs/2310.08560, 2023.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory, 2025. URL https://arxiv.org/abs/2501.13956.

S. Srivastava and Haoyu He. Memorygraft: Persistent compromise of llm agents via poisoned experience retrieval. *ArXiv*, abs/2512.16962, 2025.

Qianshan Wei, Tengchao Yang, Yaochen Wang, Xinfeng Li, Lijun Li, Zhenfei Yin, Yi Zhan, T. Holz, Zhiqiang Lin, and Xiaofeng Wang. A-memguard: A proactive defense framework for llm-based agent memory. *ArXiv*, abs/2510.02373, 2025.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. Poisonedrag: Knowledge corruption attacks to retrieval-augmented generation of large language models. pp. 3827–3844, 2024.

A Embedding Ablation Study

To verify that HST's advantage is not specific to the BGE-M3 embedding model, we replicate the main experiments using Qwen3-Embedding-4B (2560-dimensional vectors). Table 2 shows that the condition ordering (C $>$ B $>$ A) is preserved across both embedding backends, with identical PRP@3 and Benign Recall@3 values for conditions B and C.

Table 2: Embedding ablation study verifying HST advantage is retrieval-backend-agnostic. Results are consistent across BGE-M3 (1024-dim) and Qwen3-Embedding-4B (2560-dim).

Condition	BGE-M3		Qwen3-Embedding-4B	
	PRP@3 ↓	Benign Recall@3 ↑	PRP@3 ↓	Benign Recall@3 ↑
ID-Delete (A)	0.94	1.00	0.92	1.00
Retrieval-Time Fuzzy (B)	**0.00**	0.97	**0.00**	0.97
HST Commit-Time (C)	**0.00**	**1.00**	**0.00**	**1.00**

The slight difference in Condition A's PRP@3 (0.94 vs 0.92) reflects minor variations in how different embedding models rank the paraphrased poisoned records relative to benign content. However, the key finding—that HST achieves complete poisoned elimination while preserving perfect benign recall—holds regardless of the embedding backend used.