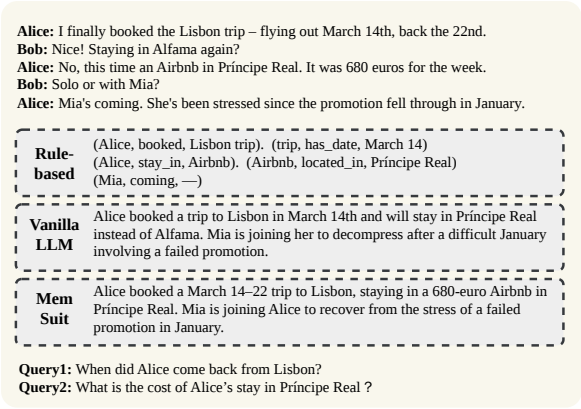
Grounding Memory Summarization in Utility Intent

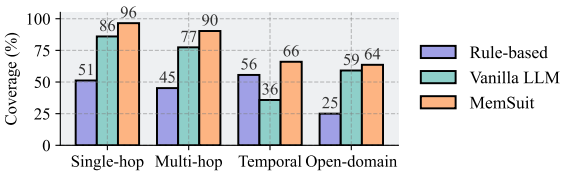
Anonymous ACL submission

Abstract

Existing summarizers for memory systems are typically optimized for human-facing criteria such as faithfulness, which misaligns with their true objective: preserving the evidence needed to support future queries. We show that conditioning summarization on query-answer pairs substantially improves answer quality, and that this utility-aware behavior is transferable across queries. Motivated by these findings, we propose MemSuit, a self-distillation framework in which a teacher summarizer, conditioned on observed query-answer pairs, produces utility-aware memory entries that a student learns to reproduce from the raw conversation alone. To prevent collateral erasure where conditioning on a single query-answer pair discards evidence relevant to other plausible queries, the teacher decomposes each block into multiple self-contained entries that preserve distinct query-relevant facets as independently retrievable units. To align the retriever with the compact, fact-dense style of teacher entries, we further fine-tune the embedding model with a contrastive objective supervised by teacher entries. Across a diverse suite of conversational query types, MemSuit consistently outperforms state-of-the-art baselines, confirming the value of grounding memory in downstream utility.



(a) Summary Case Study



(b) Evidence Coverage

Figure 1: (a) A short conversation summarized by different methods. neither the rule-based nor the vanilla LLM summary retains enough detail to answer the two example queries. (b) Evidence coverage on LoCoMo (Qwen2.5-3B), measured as the fraction of query-supporting facts preserved in the summary. Vanilla LLM summaries lose substantial evidence for temporal and open-domain queries.

1 Introduction

Memory systems have become a standard component of LLM agents deployed in long-horizon interactive settings, storing past interactions and retrieving relevant fragments on demand (Packer et al., 2023; Edge et al., 2024; Zhang et al., 2025). Since storing raw conversational text verbatim is wasteful where the vast majority of utterances are rarely relevant for future queries (Maharana et al., 2024), a substantial body of work has focused on memory summarization: compressing raw history into compact yet informative representations (Wang et al., 2025; Hu et al., 2025). Existing approaches fall into two broad families. Extraction-based methods retain predefined patterns such as entities, events, or temporal markers through handcrafted heuristics, but struggle to adapt across diverse conversational style (Liu et al., 2025; Gupta and Gupta, 2019). LLM-based methods, which have become the de facto solution, instead prompt a language model to retain content it deems important (Zhong et al., 2024; Chhikara et al., 2025). However, as illustrated in Figure 1, these methods are prone to preserving topic-level gist while discarding the concrete details on which downstream demand

most often hinges. As a result, the evidence preserved in memory is often insufficient for downstream answering, particularly for temporal and open-domain queries that depend on fine-grained and implicit information from past interactions.

While prior work has sought to improve memory summarization with various prompts (Xu et al., 2026; Liu et al., 2026), we argue that this shortcoming reflects a more fundamental issue: a misalignment of objectives. LLM summarizers are typically trained against human-written reference summaries, which optimize for properties such as coverage, coherence, and topical faithfulness (Zhang et al., 2024). However, in a memory pipeline, a summary is consumed by a retrieval-and-answer system, and its value is more determined by whether it preserves the information needed to answer future queries, a criterion of downstream utility rather than human-facing fidelity. This diagnosis is corroborated by a simple oracle analysis: when summarizers are granted privileged access to future query-answer pairs and instructed to retain only the evidence needed to support each answer, the resulting memory entries exhibit substantially higher downstream performance. However, such an oracle setting illustrates only the ideal upper bound and is unrealizable in practice, where summaries must be generated and committed to storage hundreds or thousands of turns before future queries are observed (Yang et al., 2026).

In this paper, we demonstrate that utility-aware summarization patterns are transferable across conversations and queries: the criteria a summarizer learns to apply when the query is known generalize to settings where the query is hidden. Motivated by this insight, we propose a downstream-utility-oriented memory summarization framework based on self-distillation. Our approach instantiates two summarization policies within a single pipeline. A teacher model is granted access to observed query-answer pairs and generates tailored summaries, retaining precisely the information required to address the specific question at hand. Conversely, a student model only observes the raw conversation, mirroring the deployment setting where future queries are unknown, and is trained to imitate the teacher across the observed query distribution. Intuitively, the student learns to anticipate what information would be relevant for plausible future queries, thereby internalizing the teacher's utility-aware behavior into a single, query-agnostic summarization policy.

However, operationalizing this principle surfaces two obstacles that a naive student-teacher pipeline cannot overcome. The first we term **collateral erasure**: a single conversational block rarely revolves around a single topic, and the topics it threads together may each anchor details for different queries (Zou et al., 2026). Compressing such a block into one utility-aware summary inevitably privileges the conditioning query-answer pair, causing the details that would have supported other plausible queries to be discarded as incidental. The second is **retrieval mismatch**: teacher-produced summaries are deliberately terse and fact-dense, whereas off-the-shelf embedding models for retrieval are calibrated on the verbose, redundant prose of naturally occurring text (Li et al., 2025). Even when the summaries contain the right information, this gap prevents the retriever from finding them, capping end-to-end performance.

We address both obstacles within a unified framework MemSuit (**Mem**ory **Su**mmarization for Util**it**y). To prevent collateral erasure, MemSuit allows the teacher to emit a variable number of memory entries per block: rather than compressing a predefined block into a single summary, the teacher segments it along semantic boundaries and writes one self-contained entry per topic. Distinct query-relevant aspects of the conversation are thereby preserved in independently retrievable entries, mitigating the interference induced by conditioning on a single query-answer pair. To close the retrieval mismatch, we fine-tune the embedding model with a contrastive objective: teacher-generated entries serve as positives for their conditioning queries and unrelated entries as negatives, aligning the embedding space with our compact summaries without requiring additional annotations. Across a diverse suite of conversational query types, MemSuit consistently outperforms state-of-the-art baselines, confirming the value of grounding memory in downstream utility.

2 Preliminaries

2.1 Problem Formulation

We consider long-horizon conversational memory. Let $\mathcal{C} = (u_1, u_2, \ldots, u_T)$ denote a conversation of T utterances, and let $\mathcal{Q} = \{(q_i, a_i)\}_{i=1}^{N}$ denote a set of query–answer pairs grounded in $\mathcal{C}$. A memory system comprises three components: a *summarizer* f_θ that compresses $\mathcal{C}$ into a set of N memory entries $\mathcal{M} = f_\theta(\mathcal{C}) = \{m_1, \ldots, m_N\}$; a *retriever*

g_ϕ that, given a query q, returns a relevant subset $\mathcal{M}_q \subseteq \mathcal{M}$; and a fixed *reader* π that produces an answer $\hat{a} = \pi(q, \mathcal{M}_q)$. The goal is to design and learn the memory system such that the expected answer quality $\mathbb{E}_{(q,a)\sim\mathcal{Q}}\big[\mathrm{score}(\hat{a}, a)\big]$ is maximized, where $\mathrm{score}(\cdot,\cdot)$ denotes a task-specific metric.

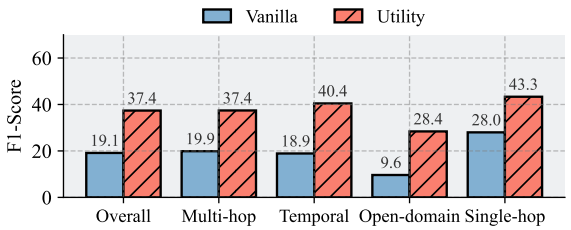
2.2 Preliminary Study

Our approach rests on two empirical claims: (i) conditioning summarization on downstream utility yields memory entries materially more useful for question answering than query-agnostic summaries, and (ii) the underlying utility-aware behavior is transferable, allowing a summarizer to internalize it without access to the target query at storage time. We examine each claim through a simple study on the LoCoMo benchmark (Maharana et al., 2024), using `Qwen2.5-3B-Instruct` as both the summarizer and the downstream reader. Following common practice in conversational memory systems, the summarizer is applied at the granularity of *blocks*: contiguous spans $b \subseteq \mathcal{C}$ of consecutive utterances. All results are F1 scores averaged over 500 randomly sampled queries.

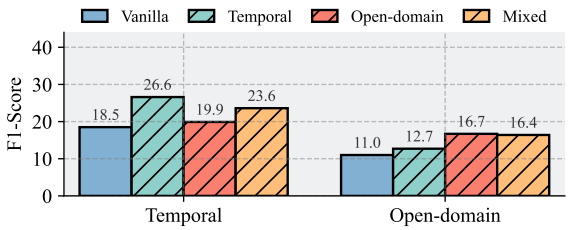
Study 1: Does utility-awareness improve summary quality? We first ask whether exposing the summarizer to a downstream query-answer pair improves the memory entries it produces. For each sampled query q with reference answer a, we locate the contiguous block $b \subseteq \mathcal{C}$ that grounds (q, a), summarize b under one of two conditions, and have the reader answer q from the resulting entry. Under the *vanilla* condition, the summarizer receives only b, mirroring standard practice in prior memory systems. Under the *utility-aware* condition, it additionally observes (q, a) and is instructed to retain the evidence only from b required to support a.

As shown in Figure 2a, utility-aware summaries yield substantial F1 gains across all types of queries, nearly doubling overall performance. This indicates that one of the bottleneck of standard summarization is the objective alignment: once the summarizer knows what the summary is for, it preserves the right evidence. However, this condition is not itself deployable, since it presupposes access to (q, a) at storage time.

Study 2: Does utility-aware behavior transfer across queries? Study 1 establishes an unrealizable upper bound. As a result, we next ask whether the criteria implicit in utility-aware summarization can transfer to a query-agnostic setting



(a) Vanilla vs. utility-aware summarization



(b) Transferability across demonstration types

Figure 2: (a) Granting the summarizer access to the downstream query-answer pair (*Utility*) substantially outperforms the query-agnostic baseline (*Vanilla*). (b) In-context utility-aware demonstrations surpasses the vanilla baseline. Gains peak when demonstration and evaluation types match and persist under mixing, indicating that utility-aware patterns transfer across queries

through demonstration alone. For each sampled query q, we summarize its grounding block b under a demo-conditioned prompt that hides q but provides in-context demonstrations, each a quadruple $(q', a', b', m'_{\text{util}})$ comprising a different query, its answer, the grounding block, and the corresponding utility-aware summary; the reader then answers q from the resulting summary. We vary two factors. The demonstration type (Temporal, Open-domain, or a Mix) controls the kind of utility-aware patterns the model observes, and the evaluation type specifies the category of q at test time.

Figure 2b reveals two findings. First, every demo-conditioned setting outperforms the vanilla baseline, showing that utility-aware pattern is transferable, where the model generalizes the criteria implicit in the demonstrations to unseen queries. Second, transfer is stronger when demonstration and evaluation types match, while mixing in off-category demonstrations largely preserves these in-category gains rather than washing them out. While in-context demonstrations are able to elicit some transfer, this approach is fragile in deployment as it depends on demonstration selection and consumes prompt budget at every storage step. We instead pursue a parametric solution with self-distillation to internalize the utility-aware pattern.

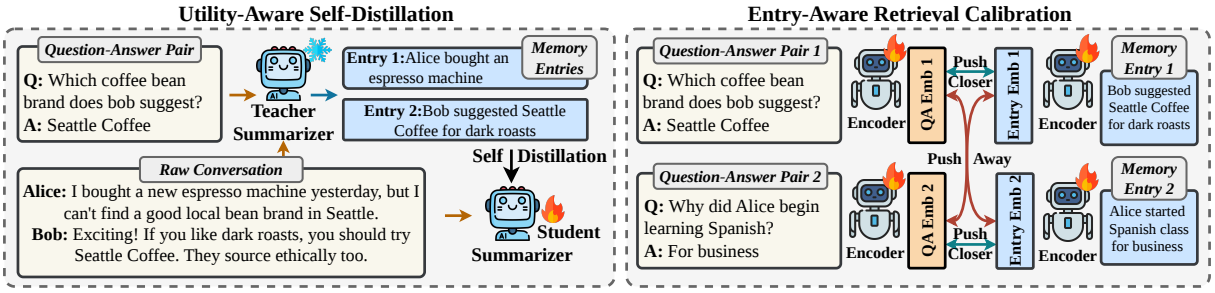


Figure 3: Overview of MemSuit. *Left:* A teacher summarizer conditioned on observed Q-A pairs decomposes each conversation into utility-aware memory entries; a student is distilled to reproduce these entries from the raw conversation alone. *Right:* A contrastive objective calibrates the retriever's embedding space, pulling each query toward its grounding entries and pushing it away from entries grounding other queries.

3 Method

Guided by the findings in Section 2, **MemSuit** operationalizes utility-aware summarization through self-distillation. A utility-aware *teacher*, conditioned on observed query–answer pairs, produces memory entries that retain precisely the evidence required to support each answer. A query-agnostic *student* then learns to imitate the teacher from the raw conversation alone, internalizing its selection criteria into a single deployable summarization policy. MemSuit instantiates this scheme in three stages. We first introduce **Adaptive Entry Decomposition** (Section 3.1), which lets the teacher emit a variable-length sequence of self-contained entries per block, addressing *collateral erasure* at the source of supervision. We then describe **Utility-Aware Self-Distillation** (Section 3.2), which trains the student to reproduce these decomposed entries from the block alone. Finally, **Entry-Aware Retrieval Calibration** (Section 3.3) closes the *retrieval mismatch* by reusing the teacher's entries as contrastive supervision for the embedding model. Figure 3 provides an overview.

3.1 Adaptive Entry Decomposition

We begin by specifying how the teacher constructs memory entries, since these entries serve as the supervisory target for the student summarizer. Let f^{T} denote the teacher summarizer, which is invoked only at training time and conditions on observed query–answer pairs. For a conversation block $b \subseteq \mathcal{C}$, let $\mathcal{Q}_b := \{(q, a) \in \mathcal{Q} \mid (q, a) \text{ is grounded in } b\}$ denote the set of query–answer pairs grounded in b. Given any pair $(q, a) \in \mathcal{Q}_b$, the teacher produces utility-aware content that retains the evidence in b required to support a. A naive instantiation would compress b into a single summary $m = f^{\mathrm{T}}(b, q, a)$. This invites *collateral erasure*: conditioning on (q, a) biases the summary toward evidence for this specific pair, so content supporting other plausible pairs $(q', a') \in \mathcal{Q}_b \setminus \{(q, a)\}$ is discarded as incidental. To mitigate this, we allow the teacher to jointly segment b and produce a variable-length set of self-contained entries in a single pass:

$$\mathcal{M}^{\mathrm{T}}_{b,q} = \{m^{\mathrm{T}}_1, \ldots, m^{\mathrm{T}}_{n_{b,q}}\} = f^{\mathrm{T}}(b, q, a), \quad n_{b,q} \geq 1. \tag{1}$$

where each m^{T}_j summarizes a distinct semantically coherent region of b. Within this set, at least one entry $m^{\mathrm{T}}_{j^\star}$ is anchored to the observed pair (q, a) and retains the evidence required to support a, while the remaining entries $\{m^{\mathrm{T}}_j\}_{j \neq j^\star}$ summarize orthogonal regions of b without query conditioning. This decomposition preserves utility-aware evidence for the observed query at $m^{\mathrm{T}}_{j^\star}$ while shielding the rest of b from query-induced bias, ensuring that information relevant to other potential pairs in $\mathcal{Q}_b$ remains independently retrievable downstream. The prompt used to elicit this joint decomposition is provided in the Appendix.

3.2 Utility-Aware Self-Distillation

With the teacher's decomposed entries in hand, we distill its utility-aware behavior into a student summarizer f^{S}_θ that, unlike the teacher, is conditioned solely on the block b and never observes (q, a). For each training block b and grounded pair $(q, a) \in \mathcal{Q}_b$, we treat the teacher's decomposed entries $\mathcal{M}^{\mathrm{T}}_{b,q}$ as the supervisory target. Concretely, we serialize them into a single delimited sequence,

$$y_{b,q} \;=\; m^{\mathrm{T}}_1 \,\|\, m^{\mathrm{T}}_2 \,\|\, \cdots \,\|\, m^{\mathrm{T}}_{n_{b,q}}, \tag{2}$$

where $\|$ denotes concatenation with a special delimiter token, and optimize the student with a token-

level cross-entropy objective:

$$\mathcal{L}_{\text{distill}}(\theta) = -\,\mathbb{E}_{(b,q)\sim\mathcal{D}_{\text{tr}}}\Big[\sum_{t=1}^{|y_{b,q}|} \log p_\theta\Big(y_{b,q}^{(t)} \,\Big|\, y_{b,q}^{(<t)},\, b\Big)\Big]. \tag{3}$$

where $\mathcal{D}_{\text{tr}} = \{(b, q, a) : b \subseteq \mathcal{C},\ (q, a) \in \mathcal{Q}_b\}$ denotes the training distribution and $y_{b,q}^{(t)}$ is the t-th token of $y_{b,q}$. Crucially, by structuring the target as a delimited sequence of self-contained entries rather than a monolithic summary, the student learns not only *what* content the teacher retains for plausible future queries, but also *how* the teacher partitions a block so that distinct query-relevant facets remain independently retrievable.

3.3 Entry-Aware Retrieval Calibration

A well-trained student is bottlenecked by the retriever: if g_ϕ cannot surface the right entry, the reader never sees it. Off-the-shelf embedding models are calibrated on verbose natural prose and tend to underperform on the compact, fact-dense entries our teacher produces. We close this gap by reusing the teacher's summaries as supervision for the retriever. Concretely, we instantiate g_ϕ as a dense retriever built on an embedding encoder $h_\phi : \mathcal{T} \to \mathbb{R}^d$ that maps any text $t \in \mathcal{T}$ (query or entry) to a d-dimensional vector.

Identifying grounding entries. The teacher decomposes each block b into the entry set $\mathcal{M}_{b,q}^{\text{T}}$, but does not explicitly mark which entries ultimately ground the answer a. To obtain this supervision, we prompt an LLM judge ψ to identify, for each grounded pair $(q, a) \in \mathcal{Q}_b$, the subset of entries containing the evidence that supports a:

$$\mathcal{P}_q \;=\; \psi\big(b, q, a, \mathcal{M}_{b,q}^{\text{T}}\big) \;\subseteq\; \mathcal{M}_{b,q}^{\text{T}}. \tag{4}$$

By construction, $\mathcal{P}_q \neq \emptyset$: the query-anchored entry $m_{j^\star}^{\text{T}}$ from Section 3.1 is guaranteed to support a. The set $\mathcal{P}_q$ thus provides per-query positives that we use to supervise the retriever.

Contrastive objective. Given these positives, we calibrate the retriever so that each query embeds close to the entries that actually support its answer. We measure query–entry affinity via cosine similarity in the embedding space,

$$s_\phi(q, m) \;=\; \frac{\langle h_\phi(q),\, h_\phi(m)\rangle}{\|h_\phi(q)\|\,\|h_\phi(m)\|}, \tag{5}$$

and contrast each query against its positives and a set of negatives drawn from unrelated blocks. Concretely, for each query q in a training mini-batch $\mathcal{Q}^{\text{batch}}$, let $b(q)$ denote the block in which q is grounded; the in-batch negatives for q are then

$$\mathcal{N}_q \;=\; \bigcup_{\substack{q'\in\mathcal{Q}^{\text{batch}}\\ b(q')\neq b(q)}} \mathcal{M}_{b(q'),\,q'}^{\text{T}}, \tag{6}$$

and the full candidate set is $\mathcal{S}_q = \mathcal{P}_q \cup \mathcal{N}_q$. We optimize an InfoNCE objective:

$$\mathcal{L}_{\text{ctr}}(\phi) = -\,\mathbb{E}_{q\sim\mathcal{Q}^{\text{batch}}} \log \frac{\sum_{m^+\in\mathcal{P}_q} \exp\big(s_\phi(q,m^+)/\tau\big)}{\sum_{m\in\mathcal{S}_q} \exp\big(s_\phi(q,m)/\tau\big)}, \tag{7}$$

where $\tau > 0$ is a temperature hyperparameter. This objective pulls queries toward the compact entries that actually support their answers and pushes them away from non-grounding entries, aligning the embedding space with the distributional properties of teacher-generated memory.

3.4 Training and Inference

Training. We train the summarizer and the retriever sequentially. We first run the teacher offline over $\mathcal{D}_{\text{tr}}$ to obtain the entry sets $\mathcal{M}_{b,q}^{\text{T}} = f^{\text{T}}(b, q, a)$ and judge-assigned positives $\mathcal{P}_q$ for each (b, q, a). The student and the retriever are then optimized against the same supervision: $\theta^\star = \arg\min_\theta \mathcal{L}_{\text{distill}}(\theta)$ and $\phi^\star = \arg\min_\phi \mathcal{L}_{\text{ctr}}(\phi)$.

Inference. At deployment, neither the teacher nor any query is available at storage time. For each incoming block b, the student emits entries $\mathcal{M}_b^{\text{S}} = f_{\theta^\star}^{\text{S}}(b)$, which are encoded by $h_{\phi^\star}$ and appended to a growing store $\mathcal{M} = \bigcup_b \mathcal{M}_b^{\text{S}}$. Given a query q, the retriever returns the top-K entries $\mathcal{M}_q = \text{Top-}K_{m\in\mathcal{M}}\, s_{\phi^\star}(q, m)$, and the fixed reader produces the answer $\hat{a} = \pi(q, \mathcal{M}_q)$.

4 Experiments

In this section, we aim to address five research questions: **RQ1:** How does MemSuit compare against state-of-the-art memory systems? **RQ2:** What is the contribution of each component to overall performance? **RQ3:** How does Entry-Aware Retrieval Calibration improve retrieval? **RQ4:** Does MemSuit generalize to unseen datasets with different query distributions? **RQ5:** How token-efficient is MemSuit at storage time?

4.1 Experimental Setup

Dataset. We evaluate on **LoCoMo** (Maharana et al., 2024), a benchmark designed to probe long-term conversational dependencies in LLMs. It

Model	Method	Multi		Temp		Open		Single		Avg	
		F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU
Llama-3.1-8B-Instruct	LoCoMo	21.33	16.42	17.46	14.41	10.60	6.69	30.82	23.41	20.05	15.23
	MemoryBank	24.60	19.46	12.01	9.56	10.47	8.18	38.45	34.34	21.38	17.89
	A-Mem	17.08	13.21	24.00	18.26	11.93	10.48	29.62	21.28	20.66	15.81
	Mem0	28.98	23.35	19.93	15.18	19.73	16.41	**45.76**	**39.79**	28.60	23.68
	SimpleMem	26.92	17.66	35.11	23.58	22.41	16.89	35.92	30.83	30.09	22.24
	MemSuit	**31.18**	**23.55**	**40.77**	**27.94**	**25.42**	**20.83**	44.89	38.61	**35.56**	**27.73**
Qwen2.5-3B-Instruct	LoCoMo	16.38	12.44	16.45	13.31	4.75	2.84	18.57	12.64	14.04	10.31
	MemoryBank	20.00	15.22	15.62	12.22	8.08	6.33	27.22	21.94	17.73	13.93
	A-Mem	18.18	10.93	19.79	15.31	9.60	9.05	29.51	25.05	19.27	15.09
	Mem0	22.24	**16.58**	23.23	18.85	10.02	7.97	29.76	25.67	21.31	17.27
	SimpleMem	18.83	14.59	16.55	11.77	12.44	10.47	26.92	23.23	18.68	15.02
	MemSuit	**22.63**	16.36	**28.66**	**20.40**	**15.40**	**13.15**	**33.48**	**29.96**	**25.04**	**19.97**

Table 1: Main Results. MemSuit achieves the best average performance.

Model	Variant	Multi		Temp		Open		Single		Avg	
		F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU
Llama-3.1-8B-Instruct	w/o Rtr-Cal	26.02	18.36	**41.03**	28.19	21.16	17.29	38.86	33.17	31.77	24.25
	w/o Self-Dist	30.92	21.68	40.24	**28.38**	21.36	16.72	40.92	36.02	33.36	25.70
	w/o Cal & Dist	23.16	15.51	30.88	21.30	17.53	14.50	33.38	28.62	26.24	19.98
	MemSuit	**31.18**	**23.55**	40.77	27.94	**25.42**	**20.83**	**44.89**	**38.61**	**35.56**	**27.73**
Qwen2.5-3B-Instruct	w/o Rtr-Cal	21.23	16.22	26.65	18.57	14.58	11.56	30.09	26.58	23.14	18.23
	w/o Self-Dist	21.51	16.24	23.86	17.85	10.80	8.61	29.84	25.99	21.50	17.17
	w/o Cal & Dist	21.05	15.51	20.02	15.65	10.36	8.79	29.42	25.44	20.21	16.35
	MemSuit	**22.63**	**16.36**	**28.66**	**20.40**	**15.40**	**13.15**	**33.48**	**29.96**	**25.04**	**19.97**

Table 2: Ablation study of MemSuit. Removing either component hurts performance.

contains 10 conversations of 200-400 turns each, featuring complex temporal shifts and interleaved topics, paired with 1,986 evaluation questions spanning four reasoning categories: multi-hop, temporal, open-domain, and single-hop. We use the first two conversations for training, the third for validation, and the remaining seven for evaluation.

Baselines. We compare MemSuit against five representative memory-augmented systems: LoCoMo (Maharana et al., 2024), MemoryBank (Zhong et al., 2024), A-Mem (Xu et al., 2026), Mem0 (Chhikara et al., 2025), and SimpleMem (Liu et al., 2026).

Backbone Models. We instantiate MemSuit with two backbone LLMs: `Llama-3.1-8B-Instruct` (Grattafiori et al., 2024) and `Qwen-2.5-3B-Instruct` (Hui et al., 2024), each serving as both summarizer and reader. The retriever is built on `all-MiniLM-L6-v2` (Wang et al., 2020).

Implementation Details. Conversations are segmented into blocks of 20 turns prior to summarization, and memory entries are indexed in LanceDB. At inference, the retriever returns the top K=10 entries per query. Both the summarizer and the retriever are fine-tuned for 3 epochs, where we use a learning rate of 2×10^{-5} for `Llama-3.1-8B-Instruct` and 2×10^{-4} for `Qwen-2.5-3B-Instruct` as the summarizer, and 2×10^{-4} for the embedding model. All experiments are run on one NVIDIA A100 80GB GPU.

Evaluation Metrics. Following prior work on LoCoMo (Liu et al., 2026), we report **F1** and **BLEU** scores per reasoning category, along with the macro-average across categories.

4.2 Results and Analyses

Main Results. To address **RQ1**, Table 1 compares MemSuit against the five baselines on LoCoMo. We make two following observations. **(1)** Across both backbones, MemSuit attains the best average

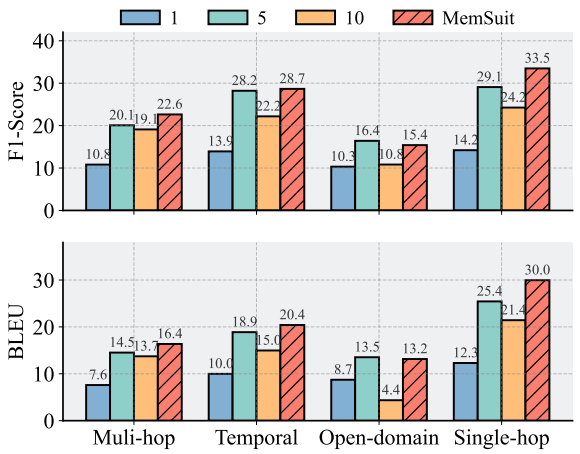


Figure 4: Decomposition granularity ablation on LoCoMo (Qwen2.5-3B). Fixed schemes (1/5/10 turns per entry) underperform MemSuit's adaptive decomposition across all categories of queries.

Model		Recall@1	Recall@5	Recall@10
Llama	Before	25.77	54.64	64.43
	After	35.05	63.40	69.59
Qwen	Before	21.78	44.06	51.49
	After	26.73	48.02	55.94

Table 3: Recall of grounding-evidence entries before and after Entry-Aware Retrieval Calibration.

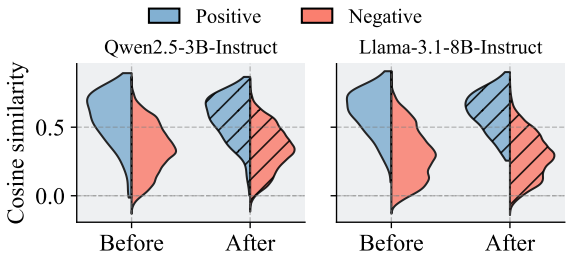


Figure 5: Cosine similarities of positive and negative query-entry pairs before and after calibration. Calibration mainly lifts the tail of the positive distribution.

F1 and BLEU, surpassing the strongest baseline by 4.60 points in F1 and 3.37 points in BLEU, demonstrating its effectiveness. **(2)** The improvement is most pronounced for temporal questions, where MemSuit surpasses the second-best method by 5.54 F1 points. This aligns with our preliminary analysis in Figure 2a, where vanilla summaries discard the fine-grained temporal details that utility-aware distillation successfully recovers.

Ablation Study. To address the **RQ2**, We compare MemSuit against three ablated variants: *w/o Rtr-Cal* removes Entry-Aware Retrieval Calibration and uses the off-the-shelf retriever; *w/o Self-Dist* removes Utility-Aware Self-Distillation; and *w/o Cal & Dist* removes both. From Table 2, we observe that **(1)** MemSuit outperforms all variants on macro-average F1 and BLEU under both backbones, confirming that the two components are complementary. **(2)** Self-distillation contributes more than calibration on temporal and open-domain queries. We attribute this to vanilla summaries omitting the supporting evidence outright, a gap that calibration alone cannot close. We further investigate whether the adaptive decomposition is necessary, where we replace the teacher's variable-length output with a fixed scheme that emits exactly one entry per fixed-size block, sweeping block sizes of 1, 5, and 10 turns. As shown in Figure 4, no fixed setting matches MemSuit, where very fine granularities can fragment evidence and increase retrieval noise, while coarse granularities reintroduce collateral erasure.

Retrieval Calibration Analysis. To address **RQ3**, we measure the recall of the entries containing the grounding evidence on the validation set. Table 3 shows that calibration improves recall at every cutoff, with the largest relative gains concentrated at Recall@1. To localize where this improvement comes from, we further investigate the cosine similarities of positive and negative query-entry pairs before and after calibration in Figure 5. We observe that the negative distribution shifts only slightly, while the positive distribution shifts with the largest movement at its tail. In other words, the pairs that benefit most are those whose similarity was initially low. We attribute this to queries such as open-domain ones that share little surface overlap with their grounding entries, which the off-the-shelf encoder fails to embed close together but calibration pulls into alignment.

Generalization Analysis. To assess whether MemSuit transfers beyond its training distribution (**RQ4**), we train on LoCoMo and evaluate on a 50-example subset of LongMemEval without any additional adaptation. We compare against Mem0 and SimpleMem, the strongest baselines from Table 1. As shown in Table 4, MemSuit reaches 28.16 F1 and 20.84 BLEU, on par with the best-performing SimpleMem and substantially above Mem0. However, the narrowing of MemSuit's advantage is expected, as LongMemEval's query distribution differs from LoCoMo's, leading the teacher to provide a less precisely targeted distillation signal.

Longmemeval	F1	Bleu1
Mem0	8.92	7.01
SimpleMem	**28.91**	**21.40**
MemSuit	28.16	20.84

Table 4: Generalization on a 50-example LongMemEval subset with MemSuit trained on LoCoMo. MemSuit matches the strongest baseline despite distribution shift.

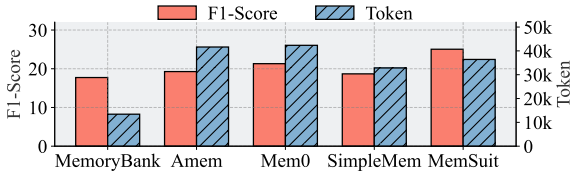


Figure 6: F1 vs. summary tokens per conversation on LoCoMo. MemSuit attains the highest F1 at a competitive token budget.

Storage Efficiency Analysis. To address **RQ5**, we measure the average number of summary tokens produced per conversation. Figure 6 reports token consumption with F1 for each method. We observe that MemSuit produces more compact memory than verbose-summary baselines such as Mem0 while still exceeding them in answer quality, showing that evidence coverage can be improved without a proportional increase in token consumption.

5 Related Works

Memory Summarization for LLM Agents. Modern memory systems compress past interactions into compact entries that can be retrieved on demand (Wu et al., 2025). Early *rule-based* methods retain predefined patterns such as named entities, dialogue acts, or temporal markers via handcrafted heuristics (Liu et al., 2025; Gupta and Gupta, 2019), which are cheap but brittle. Recent systems delegate memory selection to an LLM and adopt a range of heuristic prompting strategies that vary along two axes. The first is what is written to memory: periodic free-form summaries (Zhong et al., 2024; Lu et al., 2023), structured facts governed by explicit lifecycle operations (Chhikara et al., 2025), or minimalist prompted notes (Liu et al., 2026). The second is how memories are organized: as key-value slots (Modarressi et al., 2023), relation-aware graphs (Xu et al., 2026; Li et al., 2024), or temporal scaffolds (Ong et al., 2025). Yet all of their utilized summarizers are optimized, explicitly or implicitly, against human-facing criteria such as faithfulness and coverage (Zhang et al., 2024), leaving a gap between the summarization objective and how summaries are consumed by a retrieval-and-answer pipeline. MemSuit closes this gap by grounding summarization in downstream utility through self-distillation.

Self-Distillation with Privileged Information. Knowledge distillation (Hinton et al., 2015) transfers capabilities from a teacher to a student (Gou et al., 2021; Xu et al., 2024). Self-distillation, where teacher and student share weights or model family, has been applied to compression (Shen et al., 2025), reasoning transfer (Zhang et al., 2019), and self-improvement via teacher-generated data. A critical variant exposes the teacher to auxiliary signals unavailable at inference, which is a form of learning using privileged information (Vapnik and Izmailov, 2015). This paradigm has been demonstrated in online action detection with future frames (Zhao et al., 2022), multimodal recognition with privileged modalities (Chen et al., 2021), and learning-to-rank with privileged features (Sharmanska et al., 2013); and has recently been extended to LLM agents via privileged ground-truth values or reasoning traces (Penaloza et al., 2026; Snell et al., 2022). We are the first to apply privileged-information self-distillation to memory summarization: the teacher observes query-answer pairs to produce utility-aware entries, while the student learns to anticipate plausible queries from the raw conversation alone, thereby internalizing the teacher's selection pattern.

6 Conclusion

In this paper, we recast memory summarization for LLM agents as a problem of preserving downstream utility rather than human-facing fidelity. Building on this perspective, we propose MemSuit, a self-distillation framework in which a teacher conditioned on observed query-answer pairs decomposes each conversational block into self-contained entries, a student internalizes this selection criterion without query access at storage time, and a contrastively calibrated retriever bridges the stylistic gap to the resulting compact entries. Empirically, MemSuit yields consistent gains over strong baselines on LoCoMo and exhibits modest robustness under distribution shift, indicating that aligning memory summarizers with their true downstream consumers constitutes a better productive design principle for memory-augmented agents.

Limitations

Sensitivity to query distribution shift. As our generalization analysis on LongMemEval shows, MemSuit's advantage narrows when the evaluation query distribution diverges from the training distribution. This is an inherent property of any utility-aware approach: the teacher's selection criteria necessarily reflect the queries it was conditioned on. A promising remedy is to augment the training set with synthetic queries generated by an LLM, broadening the coverage of plausible query types and producing a more diverse distillation signal.

Scale of backbones. Our experiments use two open-source backbones (`Llama-3.1-8B-Instruct` and `Qwen-2.5-3B-Instruct`), a choice primarily bounded by available computational resources. Applying MemSuit to larger backbones such as `Llama-3.1-70B-Instruct` or `Qwen-2.5-32B-Instruct`, which are more representative of production-grade memory systems, would help characterize how the benefits of utility-aware distillation scale with model capacity. However, given the consistent gains observed across both backbones in our experiments, we expect MemSuit to remain effective at larger scales.

Simple training strategy. We train the summarizer and retriever sequentially against shared teacher supervision rather than jointly. More sophisticated strategies such as joint optimization could in principle allow the two components to co-adapt their representations, and we leave this as a promising direction for future work.

References

Cheng Chen, Qi Dou, Yueming Jin, Quande Liu, and Pheng Ann Heng. 2021. Learning with privileged multimodal knowledge for unimodal segmentation. *IEEE transactions on medical imaging*, 41(3):621–632.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International journal of computer vision*, 129(6):1789–1819.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Som Gupta and Sanjai Kumar Gupta. 2019. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, 121:49–65.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 2025. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32779–32798.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Minghan Li, Miyang Luo, Tianrui Lv, Yishuai Zhang, Siqi Zhao, Ercong Nie, and Guodong Zhou. 2025. A survey of long-document retrieval in the plm and llm era. *arXiv preprint arXiv:2509.07759*.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, and 1 others. 2024. Graphreader: Building graph-based agent to enhance long-context abilities of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12758–12786.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. 2026. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*.

Siru Liu, Allison B McCoy, Qingyu Chen, and Adam Wright. 2025. Integrating rule-based nlp and large language models for statin information extraction from clinical notes. *International journal of medical informatics*, page 106104.

Junru Lu, Siyu An, Mingbao Lin, Gabriele Pergola, Yulan He, Di Yin, Xing Sun, and Yunsheng Wu. 2023. Memochat: Tuning llms to use memos for consistent long-range open-domain conversation. *arXiv preprint arXiv:2308.08239*.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 13851–13870.

Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. 2023. Ret-llm: Towards a general read-write memory for large language models. *arXiv preprint arXiv:2305.14322*.

Kai Tzu-iunn Ong, Namyoung Kim, Minju Gwak, Hyungjoo Chae, Taeyoon Kwon, Yohan Jo, Seungwon Hwang, Dongha Lee, and Jinyoung Yeo. 2025. Towards lifelong dialogue agents via timeline-based memory management. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8631–8661.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. 2023. Memgpt: towards llms as operating systems.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. 2026. Privileged information distillation for language models. *arXiv preprint arXiv:2602.04942*.

Viktoriia Sharmanska, Novi Quadrianto, and Christoph H Lampert. 2013. Learning to rank using privileged information. In *Proceedings of the IEEE international conference on computer vision*, pages 825–832.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. 2025. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 677–693.

Charlie Snell, Dan Klein, and Ruiqi Zhong. 2022. Learning by distilling context. *arXiv preprint arXiv:2209.15189*.

Vladimir Vapnik and Rauf Izmailov. 2015. Learning using privileged information: similarity control and knowledge transfer. *The Journal of Machine Learning Research*, 16(1):2023–2049.

Qingyue Wang, Yanhe Fu, Yanan Cao, Shuai Wang, Zhiliang Tian, and Liang Ding. 2025. Recursively summarizing enables long-term dialogue memory in large language models. *Neurocomputing*, 639:130193.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788.

Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. 2025. From human memory to ai memory: A survey on memory mechanisms in the era of llms. *arXiv preprint arXiv:2504.15965*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2026. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*.

Chengyuan Yang, Zequn Sun, Wei Wei, and Wei Hu. 2026. Beyond static summarization: Proactive memory extraction for llm agents. *arXiv preprint arXiv:2601.04463*.

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. 2019. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3713–3722.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. 2024. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47.

Peisen Zhao, Lingxi Xie, Jiajie Wang, Ya Zhang, and Qi Tian. 2022. Progressive privileged knowledge distillation for online action detection. *Pattern Recognition*, 129:108741.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 19724–19731.

Huhai Zou, Tianhao Sun, Chuanjiang He, Yu Tian, Zhenyang Li, Li Jin, Nayu Liu, Jiang Zhong, and Kaiwen Wei. 2026. Es-mem: Event segmentation-based memory for long-term dialogue agents. *arXiv preprint arXiv:2601.07582*.

A Experimental Details

A.1 Dataset and Splits

We partition the ten LoCoMo conversations at the conversation level to ensure that no block, speaker pair, or topical thread leaks across splits. Conversations `0`–`1` are used for training, conversation `2` for validation, and conversations `3`–`9` for testing. The evidence-coverage analysis in Figure 2a is conducted on conversations `2` and `4`.

A.2 Rule-Based Summarizer Baseline

The rule-based baseline shown in Figure 1(a) and Figure 2a performs two extraction passes per turn. The first pass extracts atomic facts via regular expressions for dates, times, numerals, and relative-time expressions, complemented by named-entity recognition; each match yields a single entry. The second pass extracts subject-verb-object triples through spaCy dependency parsing, producing one proposition-level entry per triple. The triple pass elevates this baseline beyond a bag-of-tokens extractor: rather than emitting isolated entity entries such as `[ENTITY] Maria` and `[ENTITY] car`, it emits bound propositions such as `(Maria, donate, old car)`. When spaCy or its English model is unavailable, the system falls back to closed-vocabulary regular expressions matching simple `Subject-Verb-Object` templates.

A.3 Evidence Coverage Protocol

Evidence coverage measures, for each query, whether the memory entries produced by a summarizer contain sufficient information to support the gold answer. For each query, we run the summarizer over its source conversation, collect the resulting entries, and prompt `Qwen2.5-3B-Instruct` as an LLM judge to issue a binary YES/NO verdict on whether the entries jointly support the gold answer. The judge is explicitly instructed to allow surface-form variation (e.g., date normalization), multi-hop composition across entries, and reasonable inference, while rejecting cases requiring external world knowledge. Coverage is reported per category as the fraction of queries receiving a YES verdict. The full judge prompt is given below.

Evidence Coverage Judge Prompt

```
You are given a question, its gold answer, and a list
of memory notes extracted from a conversation. Decide
whether the notes contain enough information to
support the gold answer.

Memory notes:
{formatted_entries}

Question:    {question}
Gold answer: {gold_answer}

Decision rules:
- Answer YES if the notes - taken together, with
  reasonable inference and composition - establish the
  gold answer.
- Answer YES even when surface forms differ. For
  example:
    * "May 7, 2023", "7 May 2023", "2023-05-07" all
      match the same date.
    * "$680" matches "680 dollars" or "680 USD".
    * "Caroline Smith" matches "Caroline" when context
      makes it unambiguous.
- Answer YES if the gold answer can be obtained by
  composing several notes (multi-hop reasoning over
  the notes).
- Answer YES if the gold answer is a reasonable
  inference grounded in the notes (e.g., the question
  asks what someone "might" do, and the notes describe
  their relevant interests, skills, or stated plans).
- Answer NO only if the specific fact required by the
  gold answer is missing from the notes, or if
  establishing the gold answer would require
  information not present in the notes.
- For gold answers with multiple components (e.g.,
  "A, B"), answer YES if the notes support each
  component.
- Do NOT use general world knowledge that is unrelated
  to the conversation.

First, write 1-3 short sentences of reasoning citing
the relevant note numbers. Then end your reply with
exactly one of these two lines:
Verdict: YES
Verdict: NO
```

A.4 Code Availability

Our anonymous code for MemSuit is available at https://anonymous.4open.science/r/MemSuit-4BCD/README.md.

B Training and Inference Configuration

Self-Distillation. We fine-tune the student summarizer with LoRA (rank r=16, α=32, dropout 0.05), a per-device batch size of 2 with gradient accumulation of 8 (effective batch size 16), and `bf16` mixed precision. We use a warmup ratio of 0.03 followed by linear decay over 3 epochs. The learning rate is 2×10^{-5} for `Llama-3.1-8B-Instruct` and 2×10^{-4} for `Qwen2.5-3B-Instruct`.

Retrieval Calibration. We fine-tune `all-MiniLM-L6-v2` with a batch size of 32 under `bf16` mixed precision, a learning rate of 2×10^{-4}, and the same 0.03 warmup ratio for 3 epochs.

Inference. All inference, including teacher generation, student generation, and reader QA, is served through vLLM. The retriever returns the top K=10 entries per query, and all memory entries are indexed in LanceDB.

C Prompts

This section provides the complete set of prompts used by MemSuit's pipeline components. We group them by lifecycle stage: *training-time* prompts that produce teacher and judge supervision (Appendix C.1), and *inference-time* prompts for the deployed student summarizer and the fixed reader (Appendix C.2). The evaluation-time prompt for evidence-coverage judging is co-located with its protocol in Appendix A.3.

C.1 Training-Time Prompts

C.1.1 Teacher: Utility-Aware Decomposition

The teacher receives the dialogue block together with an observed query–answer pair. The prompt is deliberately structured so that the query–answer pair acts purely as a post-hoc coverage check, applied after a comprehensive set of entries has already been drafted. This design discourages the teacher from collapsing the output around the conditioning query, mitigating the collateral-erasure problem identified in Section 3.1.

Teacher Prompt — Utility-Aware Decomposition

```
[Internal Coverage Check - do not mention in output,
do not let it shape your focus]
A downstream system will later be asked the following
question against your memory entries. This block exists
ONLY to verify coverage after you have already drafted
a comprehensive set of entries. It is NOT the topic of
this extraction and must NOT cause you to drop,
shorten, merge, reorder, or de-prioritize any other
entry.

  Question:    {question}
  Gold answer: {answer}
  Evidence utterance (must be preserved verbatim in
  at least one entry):
    {evidence_text}

Procedure:
1. First, draft your memory entries normally, covering
   ALL information in the dialogue at the level of
   detail required by the main Requirements above
   (about one entry per dialogue turn). Do this as if
   this block did not exist.
2. Then, verify that the evidence utterance above
   appears, with its original wording preserved (key
   nouns, verbs, and named entities unchanged), inside
   the lossless_restatement of at least one entry -
   and that the gold answer is recoverable from that
   entry using only the original words from the
   dialogue.
3. If the check already passes, change nothing. If it
   does not, add or minimally revise exactly ONE entry
   to satisfy it, and leave every other entry
   untouched.

Hard constraints:
- Do NOT mention the question, the gold answer, the
  evidence, or this check in your output.
- Do NOT phrase any entry as an answer to the question;
  entries are statements about the dialogue, not
  responses.
- Do NOT use the gold answer's wording if it does not
  appear in the dialogue - use the original word(s)
  from the raw text.
- Coverage of unrelated topics (small talk, emotions,
  plans, image descriptions, etc.) must remain
  identical to what you would produce without this
  block. Fewer entries than the dialogue's richness
  warrants is a failure of this task.
```

C.1.2 LLM Judge: Grounding-Entry Identification

Given the teacher's decomposed entries for a block, the judge ψ selects the single entry that most directly supports the gold answer. The selected entry is used as the positive example in the contrastive objective of Section 3.3.

LLM Judge Prompt — Grounding-Entry Identification

```
We are building a retrieval-augmented memory system.
At inference time, the system will see only the
question and must retrieve the most relevant memory
entry from the store. We need to label which entry is
the correct retrieval target.

[Question]
{question}

[Gold Answer]
{answer}

[Evidence Turn - the dialogue line this question is
grounded in]
{evidence_text}

[Candidate Memory Entries]
{numbered_entries}

[Your Task]
Pick the index (0-based) of the SINGLE entry that:
  (a) directly contains the fact that grounds the gold
      answer, AND
  (b) would be the most useful entry to retrieve given
      ONLY the question (without seeing the evidence
      or answer).

If two entries both ground the answer, prefer the one
whose phrasing is more self-contained and more likely
to match a query that asks the question.

[Output Format]
Return ONLY a JSON object of the form:
  {"index": <int>}
where <int> is in the range [0, {n_entries_minus_1}].
No prose, no markdown.
```

C.2 Inference-Time Prompts

C.2.1 Student: Memory Entry Generation

At inference time, the student receives the dialogue block alone, without any query conditioning, and must emit a self-contained set of memory entries in JSON form.

Student Prompt — Memory Entry Generation

```
Your task is to extract all valuable information from
the following dialogues and convert them into
structured memory entries.

[Current Window Dialogues]
{dialogue_text}

[Requirements]
1. Complete Coverage: Generate enough memory entries
   to ensure ALL information in the dialogues is
   captured.
2. Force Disambiguation: Absolutely PROHIBIT using
   pronouns (he, she, it, they, this, that) and
```

```
   relative time (yesterday, today, last week,
   tomorrow). Use full names and absolute ISO 8601
   timestamps inline.
3. Lossless Information: Each entry's
   lossless_restatement must be a complete,
   independent, understandable sentence that includes
   all relevant subjects, objects, time, and location
   inline.

[Output Format]
Return a JSON array. Each element is a memory entry
with a single field:

[
  {
    "lossless_restatement": "Complete unambiguous
    restatement (must include all subjects, objects,
    time, location, etc.)"
  },
  ...
]

Now process the above dialogues. Return ONLY the JSON
array, no other explanations.
```

C.2.2 Reader: Question Answering

The fixed reader π receives the query and the top-K retrieved entries, and produces a concise answer together with a brief reasoning chain in JSON form.

Reader Prompt — Question Answering

```
Answer the user's question based on the provided
context.

User Question: {query}

Relevant Context:
{context_str}

Requirements:
1. First, think through the reasoning process.
2. Then provide a very CONCISE answer (short phrase
   about core information).
3. Answer must be based ONLY on the provided context.
4. All dates in the response must be formatted as
   'DD Month YYYY' but you can output more or less
   details if needed.
5. Return your response in JSON format.

Output Format:
{
  "reasoning": "Brief explanation of your thought
               process",
  "answer":    "Concise answer in a short phrase"
}

Now answer the question. Return ONLY the JSON, no
other text.
```