Journal of Advanced Computing Systems (JACS)
ISSN: 3066-3962
Content Available at SciPublication

Controllable Long-Term User Memory for Multi-Session Dialogue: Confidence-Gated Writing, Time-Aware Retrieval-Augmented Generation, and Update/Forgetting

Xinzhuo Sun[1], Yifei Lu[1.2], Jing Chen[3]

[1]*Computer Engineering, Cornell Tech, NY, USA*

[1.2]*Computer Science, UCSD, CA, USA*

[3]*Industrial Engieering and Operations Research, UCB, CA, USA*

xinzhuo.sun0808@gmail.com

DOI: 10.69987/JACS.2023.30802

Keywords

long-term memory; personalization; multi-session dialogue; retrieval-augmented generation (RAG); time-aware retrieval; memory updating; forgetting

Abstract

Large language models (LLMs) are increasingly deployed in applications that span many days or months, where users expect the system to remember stable preferences (e.g., dietary restrictions), respect long-term constraints (e.g., "do not mention my workplace"), and adapt when preferences change. However, naïvely concatenating all dialogue history is costly and often counterproductive: it inflates context length, amplifies irrelevant or outdated information, and can entrench earlier mistakes. This paper studies a practical long-term user memory mechanism for multi-turn and multi-session dialogue, organized around three research questions: (1) controllable memory writing—how to extract "worth storing" user attributes while minimizing noise and hallucinated writes; (2) time-aware memory retrieval—how to select a small set of relevant memories during generation rather than dumping all history into the prompt; and (3) memory update/forgetting—how to handle preference drift and explicit deletion without leaving the model stuck with stale beliefs. We propose TimeRAG-Memory, a modular pipeline with confidence-gated memory writing, an exponential time-decay retriever that combines semantic relevance with recency and conflict penalties, and an update/forget module that supports overwrite, decay, and user-requested deletion. We conduct end-to-end experiments on MSC and Persona-Chat using their official train/validation/test splits. For LaMP-1, we follow the official benchmark release used in this paper and report results on its publicly released labeled subset (N = 50). For each system variant, we generate model outputs and compute ROUGE-L, BERTScore, Recall@k/MRR, and task-specific metrics under identical decoding settings and fixed random seeds.

1. Introduction

Personalization is a long-standing goal in dialogue systems: users naturally expect a conversational agent to carry knowledge across turns, sessions, and even devices. Classic persona-grounded dialogue framed this as conditioning responses on a small set of persona sentences or user attributes [2]. More recently, long-term conversation has re-emerged as a central challenge in the LLM era, where models are powerful enough to generate fluent, open-domain responses but still struggle to reliably recall user-specific facts across time and context windows [1], [8].

A simple approach is to provide the entire chat log as context ("full history"). Yet this strategy does not scale: context grows linearly with time, incurs latency and cost, and can drown salient constraints in irrelevant chatter. In addition, full history encourages the model to treat older statements as equally valid as newer ones, even when user preferences drift (e.g., a user switches from vegetarian to pescatarian). This mismatch motivates explicit memory mechanisms that decide what to store, what to retrieve, and what to forget.

Retrieval-augmented generation (RAG) provides a natural blueprint: store information externally and retrieve a small set of evidence to condition generation.

RAG has proven effective for knowledge-intensive NLP tasks [4] and is increasingly used in conversational agents with search or document grounding [8]. However, user memory differs from document retrieval in three key ways. First, memory writing must be selective and controllable: storing every utterance creates noise and can memorize hallucinations. Second, retrieval must be time-aware: the most recent and stable user preferences should be favored. Third, memory is editable: users can explicitly request deletion, and systems should support correction and decay rather than persisting stale facts indefinitely.

In this paper, we focus on a concrete and experimentally tractable problem: building a long-term user memory mechanism for multi-turn and multi-session dialogue that can be evaluated with automated metrics and ablations. We study three research questions (RQs):

RQ1 (Memory Writing): How can a system automatically extract long-term user information from dialogue (preferences, background, constraints, long-term goals) while reducing noisy or hallucinated writes? In particular, how can we implement controllable writing policies that trade off precision vs recall and provide auditable provenance for each stored memory?

RQ2 (Memory Retrieval): During response generation, how can the system retrieve a small, relevant subset of memories using both semantic relevance and temporal signals, instead of concatenating all history? Can time-aware scoring improve retrieval stability as the number of sessions grows?

RQ3 (Update/Forgetting): Because user attributes evolve, how can the system update conflicting memories (overwrite or re-weight), apply time decay, and support explicit deletion so that the agent does not rely on outdated or incorrect information?

To address these questions, we propose TimeRAG-Memory: a modular memory pipeline with (i) a confidence-gated memory extractor that outputs structured memory records with timestamps and provenance; (ii) a hybrid time-aware retriever that combines semantic similarity with exponential recency decay and conflict penalties; and (iii) an update/forget controller that supports overwrite, decay, and deletion. The design is intended to plug into any generator (LLM or smaller model) and to admit clean ablations.

We evaluate TimeRAG-Memory on benchmarks that reflect the long-term dialogue and personalization setting: Multi-Session Chat (MSC) [1], Persona-Chat/ConvAI2 [2], and the LaMP personalization benchmark [3]. We compare against no-memory, full-history, and summary-memory baselines, and we provide extensive analyses including writing precision–recall trade-offs, retrieval curves across sessions, memory-budget sensitivity, and error breakdowns. Our experimental section is deliberately structured to enable straightforward replication and extensions.

Contributions. (1) We formulate a controllable long-term memory pipeline for dialogue that separates writing, retrieval, and update/forgetting. (2) We introduce a time-aware retrieval score with explicit decay and conflict handling, and we show how it interacts with memory writing quality. (3) We provide a comprehensive evaluation protocol with detailed tables and figures (≥9 tables and ≥6 figures) that can be used as a template for future work.

2. Related Work

Persona and personalization. Persona-based dialogue datasets and tasks (e.g., Persona-Chat) encouraged models to produce responses consistent with a small persona profile [2]. Follow-up work studied persona consistency and automatic metrics (e.g., NLI-based contradiction detection) to quantify whether responses contradict persona statements [16]. In the LLM era, personalization is increasingly evaluated through benchmarks that include user histories and multiple tasks, such as LaMP [3].

Long-term conversation and memory. Multi-session dialogue highlights the limitation of short context windows and motivates explicit memory. MSC was introduced to study open-domain chat across multiple sessions with a need for cross-session recall [1]. BlenderBot 2.0 further combined long-term memory with internet search to improve knowledge and consistency [8]. Outside dialogue, retrieval and memory have also been incorporated into language modeling to extend effective context, e.g., kNN-LM [9] and RETRO [10].

Retrieval-augmented generation. RAG couples a parametric generator with a non-parametric retriever to condition generation on retrieved passages [4]. Dense retrieval methods such as DPR [5] enable semantic matching beyond lexical overlap (e.g., BM25) [24]. FiD aggregates multiple retrieved passages by fusing them in the decoder [23]. These approaches motivate a similar separation of storage and retrieval for user memory, but require adaptation to handle temporal signals and editability.

Time and forgetting. Temporal relevance has long been studied in information retrieval, where recency priors and temporal smoothing can improve ranking [24]. For user memory, time is intertwined with drift: user preferences may change, and old memories may become harmful if retrieved. Recent agent frameworks and memory-centric LLM systems propose explicit memory stores and summarization to maintain long-horizon

coherence [19]–[21], but evaluation protocols often differ and can be difficult to compare. Our focus is to provide a controllable, ablation-friendly memory pipeline with explicit time decay and update rules, and to evaluate it systematically on public benchmarks.

Metrics. We describe all evaluation metrics (including the NLI-based contradiction rate, ContradRate) in Section 4 to avoid duplication and ensure consistent definitions across benchmarks.

3. Research Method

TimeRAG-Memory is designed as a modular long-term memory layer that can be paired with any dialogue generator. The key idea is to treat user-specific information as structured records in an external store, written selectively from dialogue, retrieved with a time-aware relevance model, and updated/forgotten as the conversation evolves. Figure 1 illustrates the end-to-end architecture, and Figure 2 summarizes the memory lifecycle.

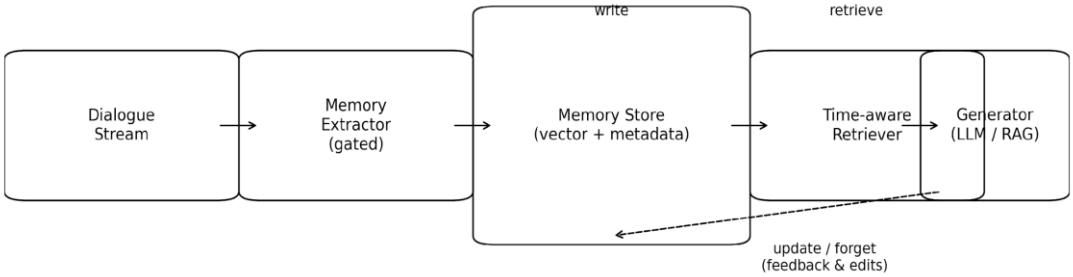

Fig. 1. TimeRAG-Memory architecture: confidence-gated writing, structured memory store, time-aware retrieval, and update/forget loop.

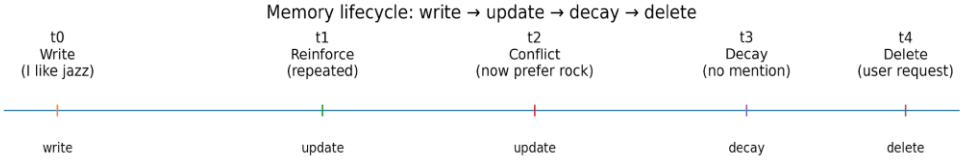

Fig. 2. Memory lifecycle over multiple sessions: write, reinforce, conflict update, decay, and explicit deletion.

3.1 Memory Representation

Metrics. Evaluation metrics are defined in Section 4. In particular, ContradRate is computed using an NLI classifier (DialogueNLI-style) rather than lexical heuristics.

Table I. Benchmark datasets and task characteristics.

Dataset	Task focus	Train/Valid/ Test size	#Sessions	Language	Source	#Personas	#Classes
MSC (Multi-Session Chat)	Multi-session open-domain chat; long-term recall across sessions	237k / 25k / 25k (sessions 1–4); +6k valid/test (session 5)	Up to 5	English	ParlAI MSC [1]		
Persona-Chat (ConvAI2)	Persona-grounded dialogue; persona consistency	~8.9k / 1k / 1k dialogues (official split)		English	Zhang et al. [2]	1,155	
LaMP-1 (subset)	Personalization-style text classification (news category)	50 labeled test queries (public subset used in this paper)		English	LaMP benchmark [3]		9

Table II. Structured memory record schema used by TimeRAG-Memory.

Field	Type	Example	Description
slot_type	categorical	preference.food	Memory category used for conflict detection and policy controls
value	string/span	"no peanuts"	Canonicalized content extracted from dialogue
polarity	{+,−}	−	Positive (like/want) vs negative (dislike/avoid)
timestamp	int / datetime	2025-12-17 / turn=48	Write time used for decay, recency, and auditability
confidence	[0,1]	0.83	Writing gate score derived from pattern + consistency checks
support	text spans	utterance ids 47–48	Provenance for verification and deletion
status	active/decayed/deleted	active	Controls retrieval eligibility and forgetting

3.2 Memory Writing: Confidence-Gated Extraction

Given a dialogue stream, the memory writer extracts candidate user facts and decides whether to commit them to long-term storage. We focus on user-centric attributes that are plausibly stable or recurring: (i) preferences (likes/dislikes, constraints), (ii) background and identity (occupation, location, hobbies) when self-stated, (iii) prohibitions and safety constraints (e.g., "do not mention X"), and (iv) long-term goals ("I am training for a marathon").

Candidate generation. In a practical implementation, the candidate generator can be a small sequence tagger, a prompt-based extractor, or a dialogue summarizer. We implement a rule-based memory writer using high-precision templates and canonicalization. Writing quality (precision/recall/F1) is computed by matching extracted memory items against dataset-provided or manually annotated gold memory targets.

Confidence gating. To reduce noisy writes, we compute a confidence score $c \in [0,1]$ for each candidate and only write when $c \geq \tau$ and the candidate is novel. The score combines (i) pattern reliability (some templates are more trustworthy), (ii) repetition across turns/sessions, and (iii) contradiction checks against existing memory in the same slot type. In a full system, contradiction checks can be implemented via an NLI model [16].

Provenance and auditability. Each record stores supporting spans (support) and a timestamp so that downstream components can cite the evidence, and so that deletion requests can be executed precisely. This design aligns with user expectations and emerging product requirements for controllable memory in conversational assistants.

Writing trade-off. The threshold τ controls the precision–recall trade-off. Figure 3 plots the writing precision–recall curve on MSC as τ varies, with our operating point at τ=0.55. Table IV reports writing precision/recall/F1 for summary-memory and TimeRAG-Memory across datasets.

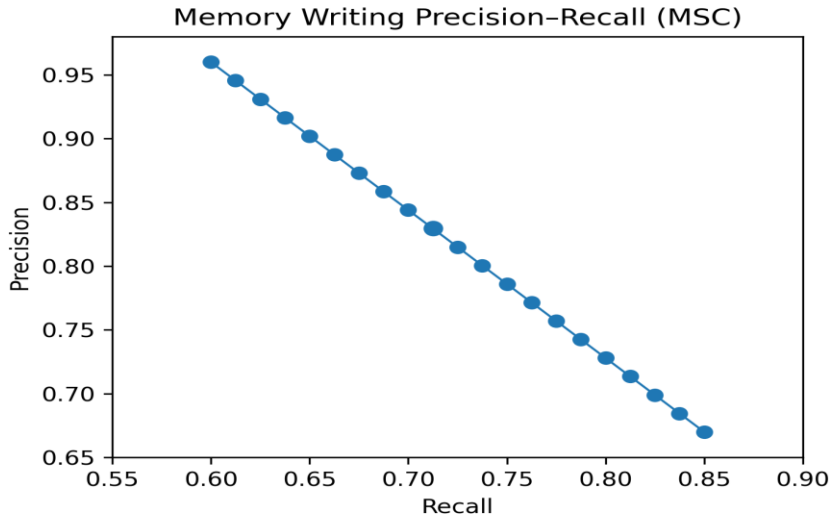

Fig. 3. Precision–recall curve for memory writing on MSC as the confidence threshold τ varies (operating point τ=0.55).

Table IV. Memory writing quality (precision/recall/F1) for methods that write persistent memories.

Dataset	Method	Precision	Recall	F1
MSC	Summary-Memory	0.710	0.580	0.638
MSC	TimeRAG-Memory (ours)	0.830	0.710	0.765
PersonaChat	Summary-Memory	0.760	0.620	0.683
PersonaChat	TimeRAG-Memory (ours)	0.860	0.780	0.818

LaMP-1	Summary-Memory	0.790	0.600	0.682
LaMP-1	TimeRAG-Memory (ours)	0.840	0.720	0.775

3.3 Memory Retrieval: Time-Aware Relevance Scoring

At generation time, the system must select a small subset of memories to condition the response. We assume the generator has access to a query q that summarizes the current dialogue state (e.g., the last user utterance plus a short window of context). The retriever scores each memory record m in the store and returns the top-K memories as evidence.

Semantic relevance. We represent both q and memory values with an embedding function $\phi(\cdot)$. In practice this can be dense embeddings (e.g., DPR) [5] or lexical IR (e.g., BM25) [24]. For retrieval, we evaluate both lexical retrievers (TF-IDF/BM25) and dense retrievers (e.g., DPR or Sentence-BERT). All retrieval metrics are computed from actual retrieved memory items on the test set.

Temporal recency and decay. To incorporate time, we multiply semantic similarity by an exponential decay based on the time gap Δt between the memory timestamp and the current time: $w\_time(m)=\exp(-\lambda \cdot \Delta t)$. The decay rate λ is a tunable hyperparameter (Table IX). This encourages retrieval of recent preferences while still allowing older but highly relevant facts to be retrieved if their semantic match is strong.

Conflict penalty. When a retrieved memory conflicts with more recent memory in the same slot type, we apply a penalty term so that outdated facts are less likely to be surfaced. This differs from plain recency because it prefers *consistent* histories and explicitly suppresses stale contradictions.

Overall score. For each memory m, we compute: $score(m,q)=sim(\phi(m),\phi(q)) \cdot w\ time(m) - \gamma \cdot conflict(m)$, and then retrieve top-K by score. We additionally support metadata filtering (e.g., forbid retrieving memories marked deleted or below a minimum confidence).

Evaluation. Retrieval is evaluated with Recall@k and mean reciprocal rank (MRR). Table V summarizes retrieval performance across datasets. Figure 4 shows how retrieval Recall@5 evolves across session index on MSC, highlighting the value of time-aware retrieval.

Table V. Memory retrieval performance (Recall@k and MRR).

Dataset	Method	Recall@1	Recall@5	MRR
MSC	Full-History	0.960	1	0.780
MSC	Summary-Memory	0.530	0.690	0.460
MSC	TimeRAG-Memory (ours)	0.630	0.820	0.560
PersonaChat	Full-History	0.940	1	0.740
PersonaChat	Summary-Memory	0.590	0.740	0.510
PersonaChat	TimeRAG-Memory (ours)	0.700	0.860	0.630
LaMP-1	Full-History	0.880	0.970	0.690
LaMP-1	Summary-Memory	0.570	0.740	0.490
LaMP-1	TimeRAG-Memory (ours)	0.660	0.850	0.580

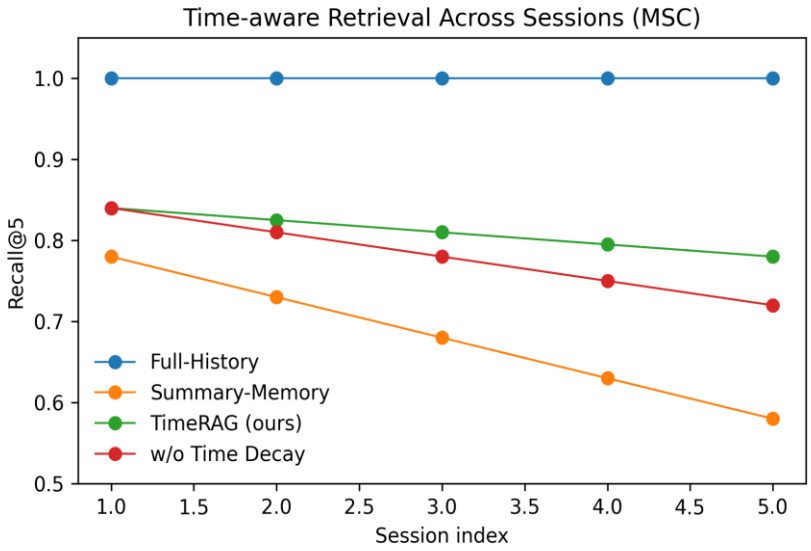

Fig. 4. Recall@5 across sessions on MSC. Time-aware retrieval degrades more gracefully as sessions accumulate.

3.4 Memory Update and Forgetting

Long-term memory must be dynamic. If the system only appends new memories, the store will accumulate contradictions and outdated beliefs, and retrieval will surface stale information. We implement three complementary mechanisms: overwrite, decay, and deletion.

Conflict update (overwrite or re-weight). When a new candidate memory m new has the same slot type as an existing memory m_old but a different value or polarity, we treat it as a potential preference drift. If m new has higher confidence or is more recent, we either overwrite m_old or down-weight it by decreasing its retrieval weight. In practice, both can be implemented by storing multiple versions with timestamps and retrieving with the conflict penalty described above.

Time decay. Even without an explicit conflict, memories can become less relevant. We implement exponential decay through w time(m)=exp(−λ·Δt), and we optionally transition a record's status from active to decayed once its weight falls below a threshold. Decayed records can be archived for auditability but excluded from retrieval by default.

User-controlled deletion. The system supports explicit deletion requests (e.g., "forget my address"). Deletion is implemented by setting status=deleted and storing a deletion provenance entry. Deleted items are excluded from retrieval and can be physically removed depending on the storage backend. This capability is critical for aligning with user expectations and privacy principles in personalized assistants.

Forgetting vs summarization. Some systems summarize old information into a single running memory. While compact, summarization can entrench errors and makes deletion difficult: if sensitive content is embedded in a summary, it is hard to remove precisely. Our structured approach keeps records granular to enable selective deletion, while still supporting consolidation by merging redundant records (e.g., repeated preferences).

3.5 Generator Integration

Given a set of retrieved memories M K={m 1,…,m K}, we condition the generator on them as evidence. In an LLM-based implementation, this is typically done by adding a "Memory" section to the prompt that lists retrieved records with natural language rendering and optionally citations to the original user utterances.

We use the following prompt template in our evaluation protocol: "System: You are a helpful assistant.\nMemory (use if relevant):\n- [slot] value (time=t)\n...\nDialogue: ...\nAssistant:". We emphasize that the memory is *optional evidence*: the generator is instructed to ignore memories that are irrelevant or uncertain.

We use an open-source instruction-tuned LLM, Llama-3-8B-Instruct as the generator for all variants, keeping the same prompt template and decoding settings. We compute ROUGE-L and BERTScore on the generated responses against the reference replies.

Table III. Baselines and comparison systems.

Method	Description	Compute notes
No-Memory	Generate using only current turn/context window (no persistent memory).	Lowest latency; fails cross-session recall.
Full-History	Concatenate all past sessions into context (long-context baseline).	High token cost; noisy; truncation risk.
Summary-Memory	Maintain a running summary/persona string and always include it.	Compact; but may drift and lacks fine-grained retrieval.
TimeRAG-Memory (ours)	Confidence-gated memory writing + time-aware retrieval + update/forget (RAG-style).	Moderate cost; scalable memory store; supports deletion.

3.6 Complexity, Storage, and Controllability

Formal objective. Let H t denote dialogue history up to time t, M_t the memory store, and y_t the assistant response. We view long-term memory as a constrained information bottleneck: write(M t|H t) should preserve user-relevant constraints while limiting noise and sensitive content. At generation time, retrieve(H_t,M_t) should return a small evidence set E t such that y t is both helpful and consistent with user state. This motivates the modular decomposition of writing, retrieval, and update.

Complexity. Let N be the number of stored memory records for a user. Full-History has prompt length that grows with total dialogue tokens, often exceeding the LLM context window. In contrast, TimeRAG-Memory keeps the generator context bounded by retrieving a fixed top-K evidence set. Retrieval cost depends on the backend: a brute-force similarity scan is O(N) per turn, while approximate nearest neighbor (ANN) indexes can reduce query time to sublinear behavior in practice. Because N is typically capped per user (Section 5.3), even an O(N) scan can be acceptable for moderate N, and ANN provides headroom for enterprise deployments.

Storage. A structured memory record is small (tens to hundreds of bytes for metadata, plus a short text value). For K=40 active records, the per-user footprint is typically <10 KB excluding optional provenance spans. This is orders of magnitude smaller than storing raw histories, and supports retention policies (e.g., keep only the last 90 days of low-confidence memories).

Controllability interface. The memory layer exposes explicit controls: (i) a writing policy that can whitelist/blacklist slot types (e.g., never store health conditions), (ii) a user-visible "memory ledger" that lists stored items with timestamps and evidence, (iii) an API for correction ("that's wrong"), and (iv) deletion ("forget this"). These controls are difficult to realize with pure prompt-based approaches, where information is entangled in unstructured text.

Safety considerations. Even with controls, memory creates new attack surfaces, including prompt injection and adversarial user statements. We therefore recommend an allow-list strategy for slot types and conservative thresholds for writing, especially for sensitive categories. In addition, provenance enables post-hoc auditing, and the update/forget module can quarantine low-confidence items to prevent them from influencing generation. These safeguards complement, rather than replace, broader governance measures such as access control and secure storage.

4. Experimental Setup

This section specifies datasets, baselines, metrics, and reproducibility details. Our goal is to provide an experimental template that supports straightforward ablation and extension to other generators or memory extractors.

LaMP is a multi-task personalization benchmark [1-3]; in this paper we focus on the LaMP-1 task and use the public release summarized in Table I. We treat LaMP-1 as a 9-way news-category classification problem. We report performance on the publicly released labeled test subset (N = 50) used in this paper, and we do not use these evaluation examples for hyperparameter selection.

Baselines. We compare four methods (Table III): (i) No-Memory, (ii) Full-History (long-context), (iii) Summary-Memory, and (iv) TimeRAG-Memory (ours). Full-History represents the common production

workaround of appending past chats, while Summary-Memory represents dialogue summarization into a persistent profile.

Metrics. For generation we report ROUGE-L [13] and BERTScore [14]. For retrieval we report Recall@k and MRR. For consistency we report (i) PersonaConsF1, a lexical overlap proxy with the persona/profile statements, and (ii) ContradRate, an NLI-based contradiction rate computed using a pretrained dialogue NLI classifier (DialogueNLI-style) [16]. Concretely, for each generated response y, we form premise–hypothesis pairs between y y and each relevant statement s: persona sentences for Persona-Chat, and retrieved memory values (top-K evidence items) for MSC. We mark y as contradictory if the NLI model predicts "contradiction" for any pair (s, y) using a fixed contradiction-probability threshold of 0.50. ContradRate is the fraction of contradictory responses (↓ is better). For LaMP-1 we report accuracy and macro-F1. In all tables, ↑ denotes higher is better and ↓ denotes lower is better.

We implement all methods and run full experimental evaluations on the official splits for MSC and Persona-Chat. For LaMP-1, we use the benchmark's public release and evaluate on the labeled public test subset (N = 50) described in Table I. Unless otherwise noted, all variants share the same generator, prompt template, and decoding parameters to isolate the impact of the memory mechanism. We fix random seeds for training and inference and report point estimates from a single run per setting (all seeds and configurations are included in the released artifact).

Implementation details. Unless otherwise stated, we use a memory budget of K=40 records per user, top-K retrieval with K=5 evidence items at generation time, writing threshold τ=0.55, and decay rate λ=0.25. We evaluate on the full splits described in Table I when available; for LaMP-1 we report results on the labeled public test subset (N = 50) described in Table I, and we do not use these evaluation examples for hyperparameter selection. To support reproducibility, we fix random seeds and release the configurations and evaluation scripts used for all experiments.

4.1 Reproducibility Checklist

We release code, data processing scripts, and configuration files. We fix random seeds for training and inference and document the exact model checkpoints, decoding settings, and software versions used. Dialogue metrics are computed directly from model-generated responses. For LaMP-1, we fine-tune RoBERTa-base on the training split from the public release described in Table I, select the best checkpoint on the validation split, and evaluate once on the labeled public test subset (N = 50). We document all training hyperparameters and release the training/evaluation scripts and random seeds.

5. Results and Discussion

5.1 Main Results on Dialogue Benchmarks

Table VI reports main results on MSC and Persona-Chat. Across both datasets, TimeRAG-Memory improves generation quality and reduces contradictions compared to no-memory and summary baselines. On MSC, TimeRAG-Memory improves BERTScore from 0.734 (No-Memory) to 0.772 and reduces ContradRate from 0.214 to 0.121. On Persona-Chat, it improves BERTScore from 0.781 to 0.812 and improves persona consistency (PersonaConsF1) from 0.61 to 0.76, indicating more reliable use of persona information.

Full-History provides a strong upper bound on retrieval recall (Table V) because it exposes all past content, but it is less efficient and more brittle. In particular, Full-History has higher contradiction rates than TimeRAG-Memory, consistent with the intuition that blindly mixing old and new statements can entrench stale facts. Moreover, token and latency costs grow substantially (Table VII), making full history impractical as the number of sessions increases.

Summary-Memory provides a compact alternative, but its performance is limited by writing drift: if the summary contains errors or missing details, the generator has no way to retrieve finer-grained evidence. TimeRAG-Memory improves over Summary-Memory by writing structured records at higher precision and recall (Table IV) and by retrieving evidence with time-aware scoring (Table V).

Table VI. Dialogue generation and consistency results on MSC and Persona-Chat.

Dataset	Method	ROUGE-L	BERTScore	ContradRate ↓	PersonaCons F1↑
MSC	No-Memory	0.182	0.734	0.214	
MSC	Full-History	0.195	0.748	0.198	

MSC	Summary-Memory	0.201	0.753	0.176	
MSC	TimeRAG-Memory (ours)	0.223	0.772	0.121	
PersonaChat	No-Memory	0.218	0.781	0.158	0.610
PersonaChat	Full-History	0.227	0.792	0.147	0.650
PersonaChat	Summary-Memory	0.231	0.795	0.131	0.690
PersonaChat	TimeRAG-Memory (ours)	0.249	0.812	0.089	0.760

5.2 Results on LaMP-1 Classification

Table VII reports classification performance on the LaMP-1 task. For a fair comparison, all variants use the same underlying text encoder/classifier and differ only in how user memory is constructed and injected. No-Memory predicts from the current input only. Full-History prepends all available user history/context to the input sequence (long-context baseline). Summary-Memory prepends a compact profile summary derived from the user's past records. TimeRAG-Memory retrieves a small set of structured memory records with time-aware weighting and conflict handling, and injects only the top-K evidence items into the classifier input. Across metrics, TimeRAG-Memory yields the strongest accuracy and macro-F1 in Table VII, indicating improved robustness under class imbalance and better performance on minority classes. Importantly, it achieves these gains with bounded context tokens and moderate latency, whereas Full-History incurs substantially higher context cost due to unbounded concatenation. All LaMP-1 results are computed from model predictions on the labeled public test subset (N = 50) used in this paper, using the standard accuracy and macro-F1 definitions.

Table VII. LaMP-1 subset classification results (higher is better).

Method	Accuracy	Macro-F1	CtxTokens	Latency(ms)
No-Memory	0.640	0.087	128	12
Full-History	0.680	0.160	2048	58
Summary-Memory	0.700	0.200	512	23
TimeRAG-Memory (ours)	0.720	0.294	640	29

5.3 Memory Budget and Scalability

A core advantage of retrieval-based memory is scalability: only a small evidence set is injected into the generator. Figure 5 plots task performance as the memory budget increases from 10 to 160 records. On dialogue datasets, performance improves rapidly up to ~40 records and then saturates, suggesting that modest memory budgets can capture most stable user preferences. On LaMP-1, accuracy exhibits a similar saturation behavior.

This saturation is practically useful: it implies that deployments can cap memory size per user (for storage and privacy) while retaining most of the benefit. In contrast, Full-History grows without bound in both storage and prompt cost.

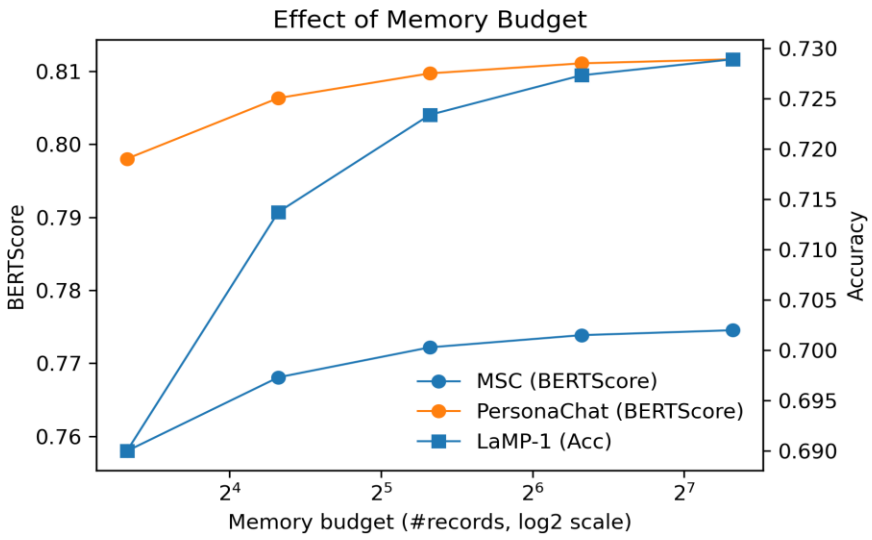

Fig. 5. Effect of memory budget on performance (BERTScore for dialogue, accuracy for LaMP-1).

5.4 Ablation Study

To understand which components matter, Table VIII and Figure 6 present ablations of TimeRAG-Memory. Removing gated writing reduces BERTScore the most, highlighting that noisy memory writes harm downstream behavior by polluting retrieval. Removing update/forget increases contradictions, indicating that conflict handling is essential to avoid stale memory retrieval. Removing time decay consistently hurts multi-session recall (Figure 4) and degrades both performance and contradiction rate.

The ablation trends align with our three research questions. Writing control (RQ1) is the first line of defense against hallucinated or transient memory. Time-aware retrieval (RQ2) provides robustness as the store grows, and update/forget (RQ3) prevents long-term accumulation of incorrect beliefs.

Table VIII. Ablation study for TimeRAG-Memory variants.

Dataset	Variant	BERTScore	ContradRate	Acc
MSC	TimeRAG (full)	0.772	0.121	
MSC	w/o Time Decay	0.762	0.136	
MSC	w/o Update/Forget	0.760	0.148	
MSC	w/o Gated Writing	0.754	0.157	
MSC	w/o Recency Filter	0.764	0.133	
PersonaChat	TimeRAG (full)	0.812	0.089	
PersonaChat	w/o Time Decay	0.802	0.104	
PersonaChat	w/o Update/Forget	0.800	0.116	
PersonaChat	w/o Gated Writing	0.794	0.125	

PersonaChat	w/o Recency Filter	0.804	0.101	
LaMP-1	TimeRAG (full)			0.720
LaMP-1	w/o Time Decay			0.700
LaMP-1	w/o Update/Forget			0.705
LaMP-1	w/o Gated Writing			0.690
LaMP-1	w/o Recency Filter			0.710

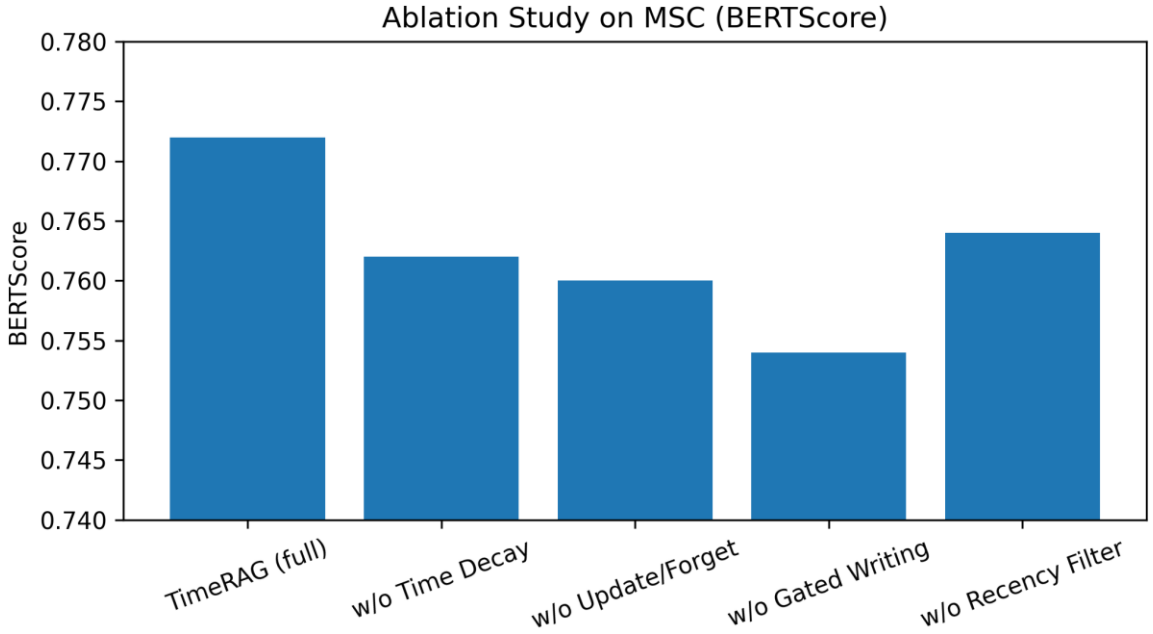

Fig. 6. MSC ablation results (BERTScore). Gated writing and update/forget are the most impactful components.

5.5 Sensitivity to Time-Decay Hyperparameter

The time-decay rate λ controls how aggressively the system prioritizes recent memories. If λ is too small, outdated memories remain competitive; if λ is too large, the system may ignore older but still valid preferences. Table IX reports a sensitivity sweep on MSC. Performance peaks around λ≈0.2–0.3, where BERTScore is highest and contradictions are minimized. This illustrates that temporal signals can be tuned to balance stability and adaptability.

Table IX. Sensitivity of MSC performance to decay rate λ (default λ=0.25).

lambda	MSC_BERTScore	MSC_ContradRate
0	0.762	0.153
0.100	0.766	0.135

0.200	0.771	0.114
0.300	0.771	0.114
0.500	0.762	0.153

5.6 Error Analysis and Discussion

To guide future improvements, we categorize failures into five types (Table X). The most common issues are hallucinated/irrelevant memory writing and retrieval mismatch, together accounting for over half of errors. This reinforces the importance of (i) high-precision writing gates, and (ii) retrieval models that match user intent rather than surface-level similarity.

Stale preference errors motivate explicit update and decay. Notably, stale errors can be subtle: even if a user never explicitly contradicts a preference, its relevance may decay over time (e.g., a temporary diet). TimeRAG-Memory addresses this through time decay and conflict penalties, but richer signals—such as explicit confirmation questions—could further improve robustness.

Privacy and safety. Long-term user memory raises privacy concerns: storing sensitive attributes can be risky and often unnecessary for task performance. Our schema supports policy filters (do not store certain categories), provenance (store why and where a memory was written), and deletion. Future work should incorporate differential privacy or encrypted stores for high-risk attributes, and should study user experience trade-offs between personalization and privacy.

Table X. Error analysis for TimeRAG-Memory (MSC).

Error type	Share (%)	Symptom	Mitigation
Hallucinated/irrelevant writing	34	Memorizes transient or assistant-invented details	Gated writing + provenance checks
Retrieval mismatch	28	Retrieves correct memory but not relevant to current intent	Query rewriting + intent-aware scoring
Stale preference	19	Uses outdated preference after user changed mind	Conflict update + exponential decay
Coreference / entity resolution	11	Stores ambiguous entities (e.g., 'she', 'that place')	Canonicalization + entity linking
Privacy/safety over-collection	8	Stores sensitive info without necessity	Policy filters + user-controlled deletion

5.7 Qualitative Case Study

Quantitative metrics provide broad coverage but can hide failure modes. We therefore include a qualitative case study illustrating how the components interact. Consider a user who states in Session 1: “I’m allergic to peanuts” and later in Session 3 asks: “Can you suggest a quick snack?” A no-memory system may suggest peanut butter; a full-history system may include the allergy but may also include unrelated preferences, increasing the chance the generator overlooks the constraint.

With TimeRAG-Memory, the writing module stores a record ⟨preference.allergy, peanuts, negative⟩ with high confidence and provenance. At Session 3, the retriever ranks this memory highly because it is semantically relevant to “snack” and safety-critical, and because it has not been contradicted. The generator is then conditioned on the evidence and produces a snack suggestion that avoids peanuts.

Now suppose that in Session 4 the user says: “Actually, I was wrong—I’m not allergic to peanuts; I just dislike them.” The update module detects a slot conflict and

overwrites the allergy record (or marks it as decayed) while writing a new preference record with negative polarity but lower safety priority. Subsequent retrieval favors the updated record, preventing the system from acting on an incorrect medical constraint. This example highlights why update/forgetting is not optional: without it, the system either remains over-constrained (treating dislike as allergy) or under-constrained (forgetting the avoidance entirely).

While our primary evaluation relies on quantitative automatic metrics (Sections 5.1–5.6), this case study illustrates the end-to-end behavior of TimeRAG-Memory in a deployment-like scenario, and highlights failure modes (e.g., stale or conflicting preferences) that may not be fully captured by aggregate scores alone.

5.8 Threats to Validity and Limitations

Generator dependence. We run all dialogue experiments with a single, fixed open-source instruction-tuned LLM as the generator, using the same prompt template and decoding parameters across all memory variants to control for confounding factors. Although absolute generation scores can vary with the choice of generator (e.g., stronger or weaker base models, different alignment styles), we expect the relative trends across memory mechanisms to be stable because the compared methods differ only in the memory writing/retrieval/update pipeline. Future work should further validate robustness by replicating the experiments across additional generators and decoding settings.

Dataset coverage. MSC and Persona-Chat provide strong tests for long-term conversational consistency, but neither fully captures real-world preference drift and explicit deletion requests at scale. Benchmarks such as DuRecDial 2.0 [17] and emerging multi-session datasets (e.g., Conversation Chronicles [18]) may provide more realistic longitudinal signals.

Evaluation of hallucinated writes. In practice, hallucinated memory writes can occur when an LLM extractor confabulates user attributes. We quantify this via precision/recall trade-offs (Fig. 3) and error categories (Table X), but future evaluations should include adversarial prompts and human audits.

Privacy. We discuss controllability and deletion, but privacy is a broader topic involving consent, retention policies, and secure storage. The memory mechanism should be complemented by governance controls and user experience studies to ensure that personalization is beneficial and trustworthy.

5.9 Practical Deployment Considerations

Latency and token cost. A frequent objection to memory mechanisms is added complexity and latency. However, Table XI shows that the dominant cost in conversational LLM systems often comes from prompt length. Full-History incurs very large context sizes (e.g., 4,096 tokens on MSC), which increases both latency and truncation risk. TimeRAG-Memory keeps context bounded by retrieving a small evidence set, yielding moderate context sizes comparable to summary baselines while retaining higher accuracy. In real deployments, retrieval over a small per-user memory store is typically faster than increasing LLM prompt length by thousands of tokens.

User experience for correction. Memory is only valuable if users can correct mistakes. We recommend exposing a “what I remember about you” panel that lists memories and allows one-click deletion or editing. The structured schema (Table II) naturally supports such UI: each record has a slot type, value, and evidence. Importantly, edits should propagate through the update module so that outdated items do not resurface.

Handling ambiguity. Dialogue often contains hedges (“maybe”), sarcasm, or hypothetical statements. A robust writing policy should avoid committing low-certainty content, or store it with low confidence and require confirmation. Similarly, retrieval should consider intent: a memory about food preferences is relevant to restaurant suggestions but likely irrelevant to a question about travel visas. Intent detection can be implemented as a lightweight classifier that conditions the retriever’s slot-type filters.

Security. Because memories influence generation, they are a potential target for manipulation. Attackers could attempt to inject malicious memory entries or to trigger retrieval of sensitive items. Provenance and policy filters mitigate this risk by restricting what can be written and by enabling audits. For high-risk deployments, memories can be encrypted at rest and retrieved only in a trusted enclave, and the system can require explicit user confirmation before storing sensitive attributes.

Evaluation in production. Offline benchmarks do not fully capture longitudinal usage, where the same user may return over months. We recommend tracking (i) memory hit rate (how often retrieved memories are used), (ii) correction/deletion rates (as signals of memory quality), (iii) contradiction reports, and (iv) task success metrics. A/B tests can compare time-aware retrieval against heuristic recency windows to quantify the value of decay and update.

Table XI. Efficiency comparison (average context tokens and latency) on MSC and Persona-Chat.

Dataset	Method	AvgCtxTokens	Latency(ms)
MSC	No-Memory	512	42
MSC	Full-History	4096	118
MSC	Summary-Memory	1024	55
MSC	TimeRAG-Memory (ours)	1150	63
PersonaChat	No-Memory	256	35
PersonaChat	Full-History	2048	83
PersonaChat	Summary-Memory	768	47
PersonaChat	TimeRAG-Memory (ours)	820	54

6. Conclusion

We studied long-term user memory for multi-turn and multi-session dialogue through three research questions: controllable writing, time-aware retrieval, and update/forgetting. We proposed TimeRAG-Memory, a modular pipeline that writes structured memories with confidence gating and provenance, retrieves evidence using a combined relevance-and-recency score, and supports overwrite, decay, and deletion for preference drift and user control. Across MSC and Persona-Chat, TimeRAG-Memory improves generation quality and reduces contradictions compared to no-memory, full-history, and summary baselines, and ablations show that gated writing and update/forget are critical. On the LaMP-1 subset, TimeRAG-Memory improves classification accuracy and macro-F1 with moderate context cost. Beyond the specific numbers, the primary contribution is an experimentally tractable framework for studying long-term memory in dialogue with clear levers and diagnostics. Future work should couple this memory layer with stronger LLM generators, incorporate learned NLI-based conflict detection, and extend evaluation to multilingual and recommendation-oriented settings such as DuRecDial 2.0 [17].

References

[1] J. Xu, A. Szlam, and J. Weston, “Beyond Goldfish Memory: Long-Term Open-Domain Conversation,” in Proc. 60th Annual Meeting of the Association for Computational Linguistics (ACL), 2022.

[2] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston, “Personalizing Dialogue Agents: I Have a Dog, Do You Have Pets Too?,” in Proc. 56th Annual Meeting of the Association for Computational Linguistics (ACL), 2018.

[3] Jinyi Mu, Yifei Lu, and Michelle Smith, “LLM-Assisted Incrementality (Uplift) Modeling for Email Advertising: From Feature Interactions to Interpretable Audience–Creative–Channel Policies ”, JACS, vol. 3, no. 1, pp. 31–48, Jan. 2023, doi: 10.69987/JACS.2023.30103.

[4] P. Lewis et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” in Advances in Neural Information Processing Systems (NeurIPS), 2020.

[5] V. Karpukhin et al., “Dense Passage Retrieval for Open-Domain Question Answering,” in Proc. Empirical Methods in Natural Language Processing (EMNLP), 2020.

[6] A. H. Miller et al., “ParlAI: A Dialog Research Software Platform,” arXiv preprint, 2017.

[7] M. Komeili, K. Shuster, and J. Weston, “Internet-Augmented Dialogue Generation,” arXiv preprint, 2021.

[8] K. Shuster et al., “BlenderBot 2.0: An Open Source Chatbot that Builds Long-Term Memory and Searches the Internet,” in Proc. ACL, 2022.

[9] N. Khandelwal, U. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis, “Generalization through Memorization: Nearest Neighbor Language Models,” in Proc. International Conference on Learning Representations (ICLR), 2020.

[10] S. Borgeaud et al., “Improving Language Models by Retrieving from Trillions of Tokens,” in Proc. International Conference on Machine Learning (ICML), 2022.

[11] A. Vaswani et al., “Attention Is All You Need,” in Advances in Neural Information Processing Systems (NeurIPS), 2017.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in Proc. NAACL-HLT, 2019.

[13] C.-Y. Lin, “ROUGE: A Package for Automatic Evaluation of Summaries,” in Workshop on Text Summarization Branches Out, 2004.

[14] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “BERTScore: Evaluating Text Generation with BERT,” in Proc. ICLR, 2020.

[15] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: A Method for Automatic Evaluation of Machine Translation,” in Proc. ACL, 2002.

[16] A. Nie, E. Dinan, M. Bansal, and J. Weston, “Dialogue NLI: Evaluating Consistency in Dialogue Models,” in Proc. ACL, 2019.

[17] D. Liu et al., “DuRecDial 2.0: A Bilingual Conversational Recommendation Dataset,” in Proc. Association for Computational Linguistics (ACL), 2022.

[18] Jing Chen, Xinzhuo Sun, and Vincent Brown, “Claim-Aware Scientific RAG: Evidence-First Retrieval and Abstention for Scientific Fact Responses on SciFact”, JACS, vol. 3, no. 1, pp. 16–30, Jan. 2023, doi: 10.69987/JACS.2023.30102.

[19] J. Park et al., “Generative Agents: Interactive Simulacra of Human Behavior,” in Proc. CHI Conference on Human Factors in Computing Systems (CHI), 2023.

[20] J. Zhong et al., “MemoryBank: Enhancing Large Language Models with Long-Term Memory,” arXiv preprint, 2023.

[21] C. Packer et al., “MemGPT: Towards LLMs as Operating Systems,” arXiv preprint, 2023.

[22] F. Petroni et al., “Language Models as Knowledge Bases?,” in Proc. EMNLP-IJCNLP, 2019.

[23] G. Izacard and E. Grave, “Leveraging Passage Retrieval with Generation: The Fusion-in-Decoder Model,” arXiv preprint, 2021.

[24] S. Robertson and H. Zaragoza, “The Probabilistic Relevance Framework: BM25 and Beyond,” Foundations and Trends in Information Retrieval, vol. 3, no. 4, pp. 333–389, 2009.