TEACHING LLMS TO REMEMBER: PARAMETRIC MEMORY VIA PERSONALIZED ADAPTER WEIGHTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-term personalization requires AI assistants to recall personal facts across sessions. Existing long-term memory systems mostly rely on token memory: facts are written as text and retrieved by searching among external embeddings. Although this design is practical and easy to maintain, retrieval is decoupled from answer utility, and the memory store grows with user history. Motivated by these limitations and the way LLMs compress knowledge into parameters, we explore whether personal memory can instead be represented at the parameter level. In this paper, we introduce **ParaMem**, a parametric memory framework for personal LLM assistants, where a user's history is compressed into a compact per-user LoRA adapter and recalled by generating explicit memory contexts from the learned weights. This formulation shifts memory retrieval from external search to learned, personalized generation, while introducing two distinctive challenges: parametric recall must cover sufficient encoded history for a given query, and the generated memory should actually help answer the query rather than merely be related to it. To address recall coverage, we propose *ParamRecall*, an inference-time strategy that combines input-side query probing with output-side memory sampling, providing a parametric counterpart to top-K retrieval in token-memory systems. To study answer helpfulness, we further propose *memory-aware preference learning* and show that the quality of the negative construction is critical: different-subject hard negatives, which contrast memories about different people sharing the same fact type, provide a meaningful signal for distinguishing helpful memories from plausible but insufficient ones. Finally, we show that parametric and token memory are complementary: grounding the adapter's recalled memories in a stored fact bank corrects attribution errors and yields the best overall system. Experiments on real-world benchmarks show that ParaMem outperforms strong token-memory baselines across most question types, suggesting that parameter space is a promising design direction for long-term memory in AI assistants.

1 INTRODUCTION

Modern large language models (LLMs) achieve powerful language understanding by compressing vast knowledge into their parameters through large-scale training (Roberts et al., 2020; Hu et al., 2024; Chang et al., 2024). Deploying them as long-term personal AI assistants demands one further capability that training alone fails to provide: *memory*, i.e., the ability to recall specific information from prior conversations on demand (Hu et al., 2025; Wu et al., 2025; Zhang et al., 2025c). As a user interacts across chat sessions, they accumulate a personal history that the assistant draws on precisely, e.g., what they do for work, who their close friends are, and when they participated in a recent event. Existing systems usually implement memory at the token level: facts are extracted from conversations and stored as text snippets in an external memory bank; at inference time, a retrieval model fetches relevant snippets using lexical, embedding, or learned retrieval mechanisms and appends them to the context window of an LLM (Packer et al., 2023; Zhong et al., 2024; Zhang et al., 2025b; Wang & Chen, 2025). This design is practical as memories are explicit, inspectable, editable, and easy to append. However, token memory also has several structural limitations. First, memory retrieval is often handled by a module isolated from the LLM that ultimately answers the user query, creating a mismatch between retrieval relevance and answer utility. Second, as retrieval is performed over external text embeddings, it may underuse the rich semantic knowledge and reasoning

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

behavior encoded in the backbone LLM. Third, token memory ties storage and retrieval cost to the size of the external memory bank: as user histories grow, the system must maintain, index, and select from an increasing number of text records. This scaling pressure can be mitigated by compression, summarization, or structured memory organization, but the memory substrate remains record-based. These limitations motivate a complementary direction: can personal memory be compressed into a fixed-size parametric substrate and recalled through generation?

We propose **ParaMem**, a parametric memory framework that compresses each user's personal history into the weights of a per-user LoRA adapter (Hu et al., 2022). Given a query about the user, the adapter generates an explicit memory context, which is then supplied to the backbone LLM for answering. This preserves the useful interface of token-memory systems: the final answer is still grounded in a textual memory context, while moving the memory substrate from external tokens to learned weights. The motivation is closely aligned with how LLMs store general knowledge: large-scale training compresses broad corpora into a fixed parameter space that can later be accessed through inference. ParaMem studies whether an analogous mechanism can support personal memory at the user level. As a beginning step toward this paradigm, we do not argue that parametric memory should replace token memory. Instead, we study it as a complementary memory substrate that is compact, personalized, and directly optimizable for answer-useful recall.

However, parametric memory introduces further unique challenges. In token memory, recall is naturally one-to-many: a retriever can search an external bank and return multiple candidate records. In ParaMem, recall is generative, so a single query followed by a single generation may activate only one narrow region of the encoded memory, especially when the question uses a different surface form or requires multiple facts. To address this ***recall coverage*** challenge, we introduce **ParamRecall**, an inference-time strategy that serves as the counterpart of top-K retrieval (Manning et al., 2008). It probes the adapter from the input side by paraphrasing the query into multiple retrieval cues, and from the output side by sampling multiple memory contexts for each cue. The resulting candidates are deduplicated and concatenated, turning brittle one-query/one-generation recall into broader one-to-many parametric recall without external search or additional training. The second challenge is ***answer helpfulness***: the adapter may recall a real memory about the correct user, entity, and topic that is still insufficient for answering the question. For example, when asked "when did Tom attend summer school?", recalling that "Tom learned programming at summer school" is related to the right person and event, but does not provide the date. We therefore study **memory-aware preference learning** with different-subject hard negatives, inspired by negative sampling in recommender systems. Our analysis shows that the negative signal helps teach the adapter to distinguish helpful memories from plausible but unhelpful ones. Moreover, we find that parametric memory and token memory are *complementary rather than competing*: by grounding the adapter's recalled memories in a stored fact bank, the system inherits both the semantic routing of parametric recall and the faithful surface form of token memory. We evaluate ParaMem on LoCoMo, a long-context conversational memory benchmark, across multiple backbone scales and question categories. ParaMem outperforms strong token-memory baselines overall, suggesting that parameter-level memory is not a replacement for token memory, but a promising complement: compact, personalized, and directly optimizable for answer-useful recall. Our contributions are summarized as follows:

- We propose **ParaMem**, a pioneering parametric memory framework for personal LLM assistants, where each user's history is encoded into per-user LoRA adapter weights and recalled as explicit memory context by generation.
- We introduce **ParamRecall**, a parametric recall strategy combining input-side query probing and output-side memory sampling, as a natural counterpart to top-K retrieval in token-memory systems, and investigate **memory-aware preference learning**, indicating that different-subject hard negatives teach the adapter to prefer answer-helpful memories over merely related ones.
- We demonstrate that parametric and token memory are *complementary*: grounding the adapter's recalled memories in the teacher-extracted fact store corrects attribution errors and achieves the best overall system, outperforming strong token-memory baselines on real-world benchmarks.

2 PARAMETRIC MEMORY

Problem Setup. Let $\mathcal{H}_s = \{T_1, T_2, \ldots, T_n\}$ denote the conversational history of user s, where each session transcript T_i is associated with a session date d_i. Given a query q about the user's past

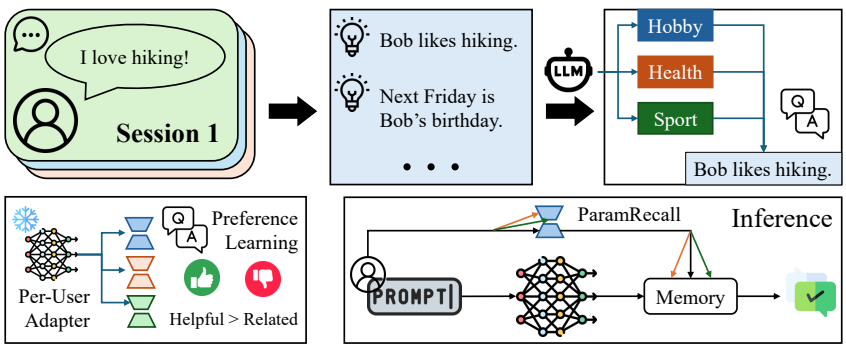


Figure 1: An overview of the training and inference pipeline of ParaMem. ParaMem first extracts structured facts from user sessions, then generates retrieval QA data for training. Per-user adapters are trained on a frozen backbone with SFT, then with preference learning that favors helpful over merely related answers. At inference, ParaMem leverages paraphrased query probes to the adapted model and samples multiple memories per probe. Deduplicated memories are passed as context to the backbone LLM, which produces the answer.

interactions, the goal is to produce an answer a that is factually grounded in $\mathcal{H}_s$. Token-memory systems address this problem by constructing an explicit memory bank $\mathcal{M}_s$ from $\mathcal{H}_s$, such as extracted facts, summaries, or structured memory records (Xu et al., 2025; Liu et al., 2026). At inference time, a retriever selects a subset of memory entries $C \subseteq \mathcal{M}_s$, and the final answer is generated by conditioning the backbone LLM on the retrieved context: $a = f_b(\cdot \mid q, C)$, where f_b denotes the backbone model generation.

In contrast, ParaMem explores whether the user's memory can be encoded into user-specific parameters. For each user s, we train a lightweight LoRA adapter θ_s with supervision derived from $\mathcal{H}_s$. Throughout, the adapter θ_s refers only to the low-rank weight updates; we write f_{θ_s} for the frozen backbone with θ_s applied, which we call the adapted model. The adapter is then used to support answering queries about that user's history. This general formulation can be instantiated in two ways: $a = f_{\theta_s}(\cdot \mid q)$ (direct answer) and $C = f_{\theta_s}(\cdot \mid q)$, $a = f_b(\cdot \mid q, C)$ (context generation). The direct-answer manner treats memory as an implicit latent state inside the adapter and directly decodes the answer. The context-generation manner asks the adapter to explicitly generate the memory context, which is then used by the backbone LLM for final query answering. The context-generation manner aligns well with the workflow of token memory systems. This formulation raises three questions for design: (i) How should raw conversations be converted into supervision for training user-specific parametric memory? (ii) As recall is performed through generation rather than external search, how can the adapter generate sufficient memory contexts at inference time? (iii) How can the adapter learn to prefer memories that actually answer the query over memories that are merely related? We address these questions in the remaining part of this section. An overview of ParaMem is shown in Figure 1.

2.1 Training Parametric Memory from User Conversations

Grounded fact extraction. Raw conversations are not directly suitable as training data for parametric memory. They contain irrelevant dialogue, implicit references, pronouns, and relative time expressions whose meaning depends on the session context. We therefore first convert each user's history into grounded user-related facts. Given a session transcript T_i and its date d_i, we extract factual statements about the user and related entities, covering attributes, events, preferences, opinions, and relationships. During this process, we normalize facts into self-contained memory units: pronouns are resolved to explicit entity names, and relative temporal expressions are converted into absolute dates whenever possible. For example, a statement such as "he went there last Tuesday" is rewritten with the relevant subject, location, and date anchored by d_i. This preprocessing reduces the mismatch between conversational surface form and later memory queries, especially for temporal questions. For each grounded fact, we also record lightweight metadata used by later training stages. The `fact_id` identifies the underlying memory unit, the `subject` denotes the main person or entity discussed by the fact, and the `facet` describes the answer-bearing dimension of the fact, such as temporal, location, event, preference, opinion, or relation. After extracting grounded facts, we use a teacher

LLM such as GPT-4o (Achiam et al., 2023) to synthesize query-target supervision from each fact, following the common practice of using capable LLMs to generate supervision for instruction tuning, distillation, and memory construction (Taori et al., 2023; Wang et al., 2023c; Xu et al., 2025). This step converts passive memory records into active recall tasks: the adapter is trained not merely to store a fact, but to produce the appropriate answer or memory context when queried in different ways. Detailed prompts for fact extraction and data synthesis are provided in Appendix A.

Two supervision formats. Given a grounded fact f, there are two natural ways to supervise a user-specific adapter as parametric memory. The first is **direct-answer mode**. Given a query q, the adapter directly generates the answer a as $a \sim f_{\theta_s}(\cdot \mid q)$. This formulation treats the relevant memory context as a latent variable internalized in the adapter weights. It provides a compact learning signal because the target is the answer value, such as a date, location, person, or preference. However, it entangles memory recall with final answer generation, making it harder to inspect what memory the adapter used. The second is **memory-context mode**. Given the same query q, the adapter generates an explicit memory context c as $c \sim f_{\theta_s}(\cdot \mid q)$, $a = f_b(\cdot \mid q, c)$. This formulation keeps the adapter's role closer to retrieval: it exposes the relevant memory as text, while the backbone LLM performs final answer generation. It also makes the system more interpretable because the recalled memory can be inspected before answering.

We synthesize both forms of supervision from each grounded fact using a fact-first prompting procedure. The synthesized queries cover diverse canonical categories, including *what*, *who*, *when*, *where*, *how*, *why*, *paraphrase*, *summary*, and *yes/no* questions. Each query is paired with both a direct-answer target and a memory-context target. The resulting training set for the user s is $\mathcal{D}_s = \mathcal{D}_s^d \cup \mathcal{D}_s^c$, and the adapter is trained with the supervised fine-tuning (SFT) objective

$$\mathcal{L}_{\mathrm{SFT}}(\theta_s) = -\mathbb{E}_{(q,y)\sim\mathcal{D}_s}\left[\log f_{\theta_s}(y \mid q)\right], \tag{1}$$

where y is either the direct answer a or the memory context c depending on the supervision format. At inference time, ParaMem can switch between the two interfaces through the system prompt. The direct-answer interface is suitable for simple single-hop questions or latency-sensitive settings, where the adapter can answer directly from its encoded memory without an additional backbone LLM call. The memory-context interface is preferable when the downstream task requires complex reasoning or interpretability: the adapter first exposes relevant memory as textual context, and the backbone LLM then generates the final answer. Training on both formats provides *complementary supervision*: direct-answer rows emphasize the answer-bearing content to be stored in the adapter (*what to remember*), while context rows teach the adapter how to recall that content as usable evidence (*how to use*). In the remainder of the paper, we focus on the memory-context interface because it perfectly aligns with the retrieve-then-answer workflow of token-memory systems, exposes inspectable evidence, and better supports complex conversational question answering (QA) in our experiments.

Per-user adapter training. We train one LoRA adapter θ_s per user. Each adapter is initialized from the same backbone LLM and trained only on that user's synthesized memory data. For a fixed LoRA configuration, the adapter provides a fixed-size parametric memory substrate for that user: unlike token-memory systems, the active memory representation does not grow with the number of extracted memory records, although memory capacity is bounded by the adapter size, and total storage still scales moderately with the number of users. At inference time, the system selects the adapter corresponding to the queried user and asks it to generate either a direct answer or an explicit memory context, depending on the system prompt.

2.2 ParamRecall: Broad Recall from Parametric Memory

In retrieval-augmented generation and QA, recall is typically improved by retrieving multiple top-ranked passages, expanding or reformulating the query, and aggregating evidence before generation Karpukhin et al. (2020); Lewis et al. (2020); Gao et al. (2023); Wang et al. (2023a). ParamRecall transfers this principle to parametric memory: instead of expanding a query to search an external index, we expand and sample around the query to probe different regions of the adapter's encoded memory. ParamRecall addresses the *recall coverage* challenge by probing parametric memory from both the input side and the output side. The intuition is that if memory is encoded in weights rather than stored as explicit records, then recall should not rely on one exact query string and one deterministic generation. Instead, we should query the adapter from multiple semantic directions and sample multiple possible memory contexts from each direction.

Input-side query probing. Given a test query q, we first generate k_q paraphrases as $\mathcal{Q}(q) = \{q_1, q_2, \ldots, q_{k_q}\}$. The paraphrases are constrained to preserve the original query type, so a *when* question remains a *when* question and a *who* question remains a *who* question. These paraphrases provide multiple retrieval cues for the same information need. Since the adapter has learned memory through different synthesized question forms, different paraphrases can activate different regions of the encoded memory.

Output-side memory sampling. For each paraphrased query q_i, we generate k_s memory contexts from the adapter using stochastic decoding: $c_{i,j} \sim p_{\theta_s}(\cdot \mid q_i)$, for $i \in [k_q]$ and $j \in [k_s]$. Sampling explores alternative generations near each query probe, recovering candidate memories that greedy decoding may suppress. The full candidate pool is then deduplicated and concatenated into a multi-memory context C, which is supplied to the backbone LLM for answer generation.

ParamRecall is the parametric-memory counterpart of top-K retrieval in token-memory systems. Token memory broadens recall by searching more records; ParamRecall broadens recall by probing the adapter with multiple input cues and sampling multiple output contexts. Importantly, it requires no retraining and no external memory search.

2.3 Grounding: Bridging Parametric and Token Memory

ParamRecall broadens the scope of recalled memories, but the recalled text is generated by the adapter and may contain attribution errors. The dominant failure mode is on adversarial questions (C5): the adapter copies the subject named in the query into the generated memory, producing a fluent but wrongly attributed response. For example, when asked about an event that belongs to a different speaker, the adapter still generates a plausible memory attributed to the queried speaker because it has been trained to answer every query with a memory.

We address this by *grounding* the adapter's recalled memories in the same teacher-extracted fact store used during training (Section 2.1). The grounding step has two components:

- **Fact snapping.** Each recalled memory line is replaced by its nearest stored fact from the per-user fact store, ranked by BM25 similarity. This corrects attribution and anchors the generated memory to a verified factual record with the correct subject, date, and event detail.
- **Query-side retrieval.** In addition to snapping the adapter's recalled memories, we retrieve the top-k facts from the same fact store by BM25 similarity to the original question. This provides a complementary token-level recall channel that can surface facts the adapter did not generate.

Grounding hybridizes parametric and token memory rather than replacing one with the other. The adapter decides *what to recall*: its generative recall acts as a learned semantic router that selects which region of the user's fact space is relevant to the query. The fact store then supplies *faithful surface form*: the actual stored text, with correct attribution and resolved dates. This design preserves the adapter's advantage in semantic recall while eliminating the attribution errors that arise from purely generative memory.

2.4 Memory-Aware Preference Learning with Cross-Fact Negatives

ParamRecall improves recall coverage, but a second challenge remains: *answer helpfulness*. Standard SFT provides limited signals for the distinction between merely relevant memory and actually helpful memory. It teaches the adapter to generate memory contexts for each query, but it does not explicitly contrast answer-helpful memories with plausible but unhelpful alternatives (Rendle et al., 2009; Xiong et al., 2021). To provide this supervision, we introduce a lightweight memory-aware preference-learning stage based on Direct Preference Optimization (DPO) (Rafailov et al., 2023). Each preference pair contains a query q, a chosen memory context c^+ that answers the query, and a rejected memory context c^- that is related but less helpful. We optimize $\mathcal{L}_{\mathrm{DPO}}(\tilde{\theta}_s) = -\mathbb{E}_{(q,c^+,c^-)}\Big[\log \sigma\big(\beta\ (r(c^+) - r(c^-))\big)\Big]$, $r(c) = \log \frac{f_{\tilde{\theta}_s}(c|q)}{f_{\theta_s}(c|q)}$, where σ is the sigmoid function. The loss increases the likelihood of the helpful memory c^+ relative to the merely related c^-. $\beta > 0$ is the inverse temperature of this preference: a larger β keeps the adapter closer to its SFT memories, while a smaller β lets the preference reshape it more aggressively. We use a small β with a small learning rate so that DPO refines which memory is surfaced without overwriting stored facts.

The key design question is how to choose c^-. A useful negative should satisfy two conditions: it should be close enough to the query to represent a realistic retrieval error, but it should differ from c^+ in the answer-bearing information. We study two structured negative construction strategies, both using the fact metadata introduced in Section 2.1:

Cross-fact negatives. Given a positive memory c^+ about subject s and fact f, we select c^- from the same user's history that shares the same `subject` s but has a different `fact_id` or `facet`. For instance, if c^+ states that "Tom attended summer school in July 2023", a valid cross-fact negative might state that "Tom learned programming at summer school" or that "the experience helped him later". This negative is topically plausible because it concerns the same person and event, but it does not provide the date.

Different-subject negatives. An alternative strategy selects c^- from a *different* subject that shares the same fact type or facet. For instance, if c^+ is a temporal fact about Tom, the rejected memory might be a temporal fact about a different person: "Sarah got a promotion in March 2023". This tests whether the adapter can distinguish between subjects, not just between facts about the same subject. Different-subject negatives are structurally harder than cross-fact negatives because the rejected memory has the right answer type but concerns the wrong person, making it a realistic confusion case for adversarial questions.

We further filter candidate negatives to avoid same-fact paraphrases and near-duplicate contexts. This safeguard is important because synthesized training data contains multiple variants of query and context for the same underlying fact; treating these variants as rejected memories would create false-negative preference pairs.

3 Experiments

3.1 Experimental Setup

Dataset and backbone models. We evaluate on LoCoMo (Maharana et al., 2024), a benchmark of long-term conversational histories with up to 35 sessions per person and 1,986 human-annotated questions. The questions cover five complementary categories: multi-hop reasoning (C1, n=282), temporal reasoning (C2, n=321), open-domain knowledge (C3, n=96), single-hop recall (C4, n=841), and adversarial questions (C5, n=446). Following (Xu et al., 2025), we report token-level F1 against ground-truth answers. Experiments are conducted on teacher models DeepSeek-V3.1 (Liu et al., 2024) and Qwen3-235B-A22B (Yang et al., 2025), and backbone models Qwen2.5-3B-Instruct (Bai et al., 2023) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024).

Baselines. We compare against seven token-memory systems that represent different points in the external-memory design space. **FullContext** places the entire conversational history in the context window without memory extraction. **Summarization** compresses each session into a summary and retrieves summaries by embedding similarity. **LangMem** (LangChain, 2025) provides general memory operations such as extraction and consolidation. **Mem0** (Chhikara et al., 2025) maintains a dynamic memory store with extraction and update operations. **ReadAgent** (Lee et al., 2024) represents long histories as paged gist memories. **A-Mem** (Xu et al., 2025) builds an associative memory graph from extracted facts. **SimpleMem** (Liu et al., 2026) stores semantically compressed memory entries. Together, these baselines test whether parametric memory can compete with strong token-memory pipelines, not just simple retrieval. To isolate what makes weight-encoded memory effective, we compare against two parametric baselines. **CLM** trains a per-user adapter on raw conversation text with a standard next-token prediction loss. This baseline tests whether passively compressing conversations into adapter weights is sufficient. **Param A-Mem** trains adapters using A-Mem-extracted memory units as memory-context targets instead of our synthesized query-memory pairs; it uses the same inference interface as ParaMem-context. This baseline isolates the role of retrieval-oriented training supervision. We build ParaMem incrementally. **ParaMem-direct** uses the direct-answer interface, where the adapter answers the query directly from its encoded memory. **ParaMem-context** uses memory-context SFT and greedy generation. **+ParamRecall** applies the inference-time recall strategy with input-side query probing and output-side memory sampling (Section 2.2). **+ParamRecall+grounding** additionally grounds the adapter's recalled memories in

Table 1: Token-level F1 on LoCoMo across question categories, with DeepSeek-V3.1 as the teacher model for all methods. C1 multi-hop, C2 temporal, C3 open-domain, C4 single-hop, C5 adversarial. The best result per column is **bold**; the second best is underlined.

	Qwen2.5-3B-Instruct						Llama-3.1-8B-Instruct					
Method	Avg	C1	C2	C3	C4	C5	Avg	C1	C2	C3	C4	C5
Token Baselines												
FullContext	0.233	0.055	0.042	0.071	0.100	0.767	0.323	0.117	0.161	0.091	0.201	0.852
Summarization	0.333	0.219	0.123	0.093	0.286	0.697	0.424	0.260	0.254	0.128	0.348	0.859
LangMem	0.217	0.099	0.023	0.117	0.101	0.670	0.226	0.033	0.014	0.057	0.058	0.855
Mem0	0.259	0.141	0.017	0.089	0.201	0.652	0.302	0.179	0.070	0.111	0.224	0.732
ReadAgent	0.292	0.131	0.122	0.065	0.198	0.740	0.383	0.218	0.192	0.112	0.269	0.899
SimpleMem	0.310	0.195	0.048	0.112	0.219	**0.788**	0.348	0.211	0.102	0.107	0.220	**0.903**
A-Mem	0.347	0.259	0.172	0.124	0.262	0.737	0.394	0.274	0.247	0.131	0.298	0.810
Parametric Baselines												
CLM	0.036	0.028	0.012	0.065	0.036	0.053	0.036	0.025	0.011	0.047	0.041	0.048
Param A-Mem	0.264	0.125	0.147	0.125	0.163	0.659	0.259	0.122	0.093	0.090	0.171	0.665
ParaMem												
ParaMem-direct	0.300	0.277	**0.430**	0.159	**0.418**	0.028	0.313	**0.308**	0.430	**0.201**	**0.430**	0.037
ParaMem-context	0.329	0.219	0.283	0.205	0.359	0.400	0.387	0.258	0.432	0.150	0.392	0.479
+ParamRecall	0.346	0.266	0.345	**0.210**	0.374	0.375	0.393	0.275	0.433	0.186	0.402	0.467
+ParamRecall+grounding	**0.453**	**0.307**	0.393	0.107	0.403	0.757	**0.495**	0.277	**0.469**	0.135	0.407	0.896

the teacher-extracted fact store (Section 2.3), combining parametric recall with token-level retrieval. Memory-aware preference learning (DPO) is studied as a separate ablation in Section 3.3. Except for FullContext, all baselines build their memory with an LLM extraction or compression step using the same teacher model as ParaMem.

Implementation details. All adapters use LoRA (r=8, α=32) on the attention and MLP parameters and are trained on a 50/50 mixture of direct-answer and memory-context supervision. ParamRecall uses k_q=2 query probes and k_s=3 memory samples per probe at temperature T=0.4, with each backbone's default nucleus-sampling settings, and deduplicates candidates before concatenation. DPO uses β=0.1 with the SFT adapter frozen as the reference. Full hyperparameters and data-synthesis details are provided in Appendix A.

3.2 Main Results

Table 1 compares ParaMem against token-memory and parametric baselines on LoCoMo. The results provide positive evidence for the central premise of this paper: personal memory can be encoded into user-specific parameters and recalled effectively by generation. On Qwen2.5-3B, ParaMem-context already outperforms most token-memory baselines before applying ParamRecall or DPO, showing that a retrieval-oriented adapter objective is sufficient to make parametric memory competitive. The full ParaMem system achieves the best overall performance, improving over the strongest token-memory baseline. On Llama-3.1-8B, the advantage becomes larger, suggesting that parametric memory benefits from stronger backbone models: as the base model becomes more capable at interpreting generated memory context, the value of learned semantic recall increases.

Despite its impressive performance on answerable questions (C1-C4), pure parametric memory's weakness is adversarial questions (C5). The cause is structural: the adapter is trained to answer every query with a memory, so a question about a common event among different users still yields a fluent but wrongly attributed memory. To address it, we introduce grounding to incorporate the advantages of token memory into parametric memory. Each recalled memory is replaced by its *nearest stored fact*, which carries the correct attribution. Adding token retrieval channels over the same facts yields the best performance, ahead of every baseline. Parametric recall and token memory are therefore complementary: the adapter decides *what to recall*, and the fact store supplies *faithful memory content*.

The parametric baselines show that not all weight-encoded memory is useful memory. CLM indicates that passively training on raw conversation text does not teach the adapter how to retrieve specific facts in response to user queries. Param A-Mem, trained on the memory notes that an A-Mem pipeline builds from the same conversations with the same teacher, performs better than CLM but remains below ParaMem. In particular, competitive parametric memory requires a retrieval-oriented objective:

the adapter should be trained not merely to absorb user history, but to expose answer-relevant memory when queried. These results support the methodological choices behind ParaMem: grounded fact construction, query-conditioned memory-context supervision, and an explicit recall interface.

3.3 Ablation Study

ParamRecall improves generative recall coverage. Figure 2 studies the two recall dimensions of ParamRecall: k_q, the number of input-side query probes, and k_s, the number of output samples per probe. Increasing either dimension improves over greedy generation, confirming the central motivation of ParamRecall: a single query and a single decoding path undercovers the memory encoded in adapter weights. The two dimensions provide complementary forms of exploration. Increasing k_q changes the surface form of the information need and can activate different regions of the adapter's encoded memory. Increasing k_s samples multiple memory contexts from the same prompt and explores nearby decoding paths. At equal inference budget, query probing provides a larger marginal gain than output sampling, suggesting that the main bottleneck is not only decoding stochasticity but also how the query activates memory inside the adapter. This supports our view of ParamRecall as the parametric counterpart of multi-candidate retrieval: token-memory systems broaden recall by retrieving more records, while ParamRecall broadens recall by probing and sampling the parameterized memory from multiple directions. In practice, k_q and k_s expose a latency-accuracy tradeoff. Query probing is more effective when the main uncertainty is how to express the information need, while output sampling is useful when the adapter can generate several plausible memory contexts from the same cue.

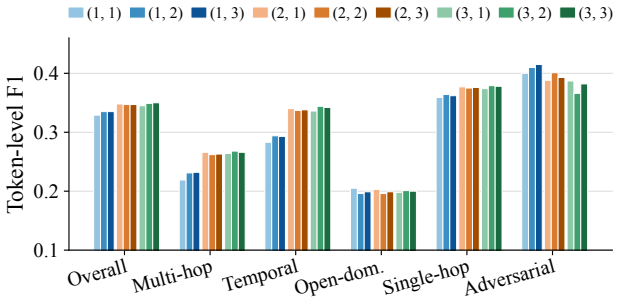


Figure 2: ParamRecall hyperparameter study (Qwen2.5-3B, DeepSeek teacher, SFT adapter). Each cell shows overall F1 for a (k_q, k_s) configuration.

Training with both interfaces improves context-mode recall. Figure 3 compares direct-answer and memory-context supervision. Context-only training substantially outperforms direct-only training, confirming why the main ParaMem pipeline uses the memory-context interface: it separates recall from final reasoning, exposes inspectable evidence, and better supports complex conversational QA. However, direct-answer supervision is still useful when mixed with context supervision. The 50/50 mixture further improves context-mode performance, especially on temporal reasoning (C2) and single-hop recall (C4), where the answer often depends on precise atomic values such as dates, entities, or relations. Direct-answer targets provide a compact gradient signal for these values, while context targets teach the adapter how to express them as usable evidence. The exception is open-domain knowledge (C3), where context-only training remains strong, as richer memory contexts provide better grounding for the base model's reasoning. Importantly, dual-format training does not collapse the direct-answer interface: ParaMem can still answer directly when needed, while achieving its best performance in the memory-context workflow.

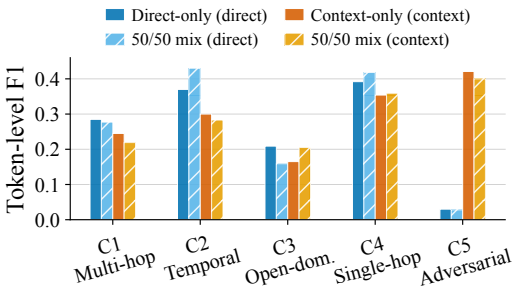


Figure 3: Per-category F1 for four training configurations.

Across the experiments, three conclusions emerge. First, parametric memory is effective only when the adapter is trained with an explicit retrieval-oriented objective. Second, a single query exposes only a narrow slice of the encoded memory; ParamRecall recovers broader coverage through query probing and memory sampling. Third, parametric and token memories are complementary: the adapter provides a learned scalable retrieval signal, and token-memory grounding supplies faithful memory content.

4 Related Work

Token-level memory. Long-term memory has become a central component of LLM-based agents, especially for applications that require consistency, personalization, and reasoning over past interactions. A recent survey categorizes agent memory by its representational form into token-level, parametric, and latent memory, with token-level memory referring to systems that store past information as explicit text records and retrieve them at inference time (Hu et al., 2025). Most existing personal and agentic memory systems fall into this category. Early agent architectures such as Generative Agents maintain a natural-language memory stream, retrieve relevant observations, and synthesize higher-level reflections for planning and interaction (Park et al., 2023). LongMem augments language models with a long-term memory bank and a decoupled retrieval-and-reading architecture for reusing long histories (Wang et al., 2023b). ReadAgent similarly compresses long inputs into gist memories and looks up relevant passages when needed (Lee et al., 2024). More recent long-term conversational memory systems, including Mem0 and A-Mem, focus on extracting salient facts from conversations, organizing them into structured or graph-like memory records, and retrieving them for downstream question answering (Chhikara et al., 2025; Xu et al., 2025). LangMem and SimpleMem further emphasize practical memory operations such as extraction, consolidation, semantic compression, and query-aware retrieval (LangChain, 2025; Liu et al., 2026). These methods make memory explicit, editable, and easy to update, but retrieval is usually performed by an external module whose relevance signal is separated from the answer-generation objective.

Parametric memory. Parametric memory broadly refers to information stored in model weights rather than in an external text store. This idea appears in several related but distinct research lines. Work on model editing studies how factual associations are localized and updated inside transformer parameters (Mitchell et al., 2022; Hartvigsen et al., 2023; Zhang et al., 2025a). ROME edits individual factual associations through rank-one updates, while MEMIT scales this idea to mass editing of many memories (Meng et al., 2022; 2023). WISE further introduces a dual-memory scheme for lifelong model editing, separating pretrained knowledge from edited knowledge through a side memory and router (Wang et al., 2024). Another line studies parameter-efficient adaptation as a way to store task, domain, or behavioral knowledge (Xu et al., 2024; Lei et al., 2026; Chen et al., 2026). LoRA provides a practical mechanism for low-rank weight adaptation (Hu et al., 2022), and recent work has explored LoRA-style modules as knowledge memories or domain-adaptive parametric components. For example, ParamAgent uses a domain-adaptive parametric memory to internalize cross-sample regularities for language-agent reasoning tasks, while hierarchical-memory pretraining stores long-tail world knowledge in large parametric memory banks accessed by smaller language models (Yao et al., 2025; Pouransari et al., 2025). These works show that parameters can serve as a memory substrate, but they mainly focus on world knowledge, domain-level regularities, model editing, or task adaptation. In contrast, ParaMem encodes each user's long-term personal history into a lightweight LoRA adapter, and retrieves memory by generating explicit answer-relevant context. This makes parametric memory directly comparable to token-level personal memory systems, while exposing new challenges that do not arise in the same form in prior parametric memory work.

5 Conclusion

We introduced ParaMem, a parametric memory framework that compresses each user's conversational history into a lightweight LoRA adapter and recalls relevant memory by generation, providing a complementary design point to token-memory systems. Our study shows that effective parametric memory requires retrieval-oriented supervision from grounded user facts, an inference-time recall strategy (ParamRecall) that probes the adapter with multiple query forms and samples multiple memory contexts, and memory-aware preference learning that teaches the adapter to prefer answer-bearing evidence over plausible distractors. Furthermore, grounding parametric recall against a stored fact bank addresses the attribution weakness of pure generative recall, showing that parametric and token memory are complementary: the adapter decides what to recall, while the fact store supplies faithful content. Experiments on real-world benchmarks show that ParaMem outperforms strong token-memory baselines on most question types, with consistent gains across backbone scales and teacher models. These results suggest that personal AI memory need not be confined to external token stores: it can also be encoded, recalled, and optimized in parameter space, combining the transparency of textual evidence with the semantic capacity of learned parameters.

AI USE STATEMENT

In this work, we used generative AI tools in two capacities: as part of the research methodology and as development aids. Teacher LLMs were used to synthesize training data for ParaMem, including grounded fact extraction from conversations, query generation, and short-answer construction. These uses are integral to the proposed method and are fully described in Section 2 and Appendix A.1. We used LLMs to help write and debug experiment scripts, evaluation pipelines, and visualization code. We also used LLMs to assist with drafting and revising portions of the manuscript text. All AI-generated code was reviewed, tested, and verified for correctness by the authors. All AI-assisted text was reviewed, revised, and validated against experimental evidence by the authors. We have not used generative AI tools for research ideation, experimental design, or literature review. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

ParaMem encodes personal conversational history into model parameters, which raises privacy considerations. Unlike token-memory systems where individual records can be inspected, edited, or deleted, parametric memory makes selective modification of stored information less straightforward. We discuss these concerns and the need for user control mechanisms in Section C. All experiments use the publicly available benchmarks, which contains synthetic conversations with no real user data. No human subjects were involved in this study.

REPRODUCIBILITY STATEMENT

We provide several resources to support reproducibility. Our code and data synthesis pipelines are available at the anonymous repository linked in Appendix A. Section 2 fully specifies the ParaMem framework, including the data synthesis pipeline, adapter training procedure, ParamRecall inference strategy, and DPO preference learning setup. Appendix A provides complete hyperparameters, training configurations, prompts, and data volumes.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Hoyeon Chang, Jinho Park, Seonghyeon Ye, Sohee Yang, Youngkyung Seo, Du-Seong Chang, and Minjoon Seo. How do large language models acquire factual knowledge during pretraining? *Advances in Neural Information Processing Systems*, 37, 2024.

Shigeng Chen, Linhao Luo, Zhangchi Qiu, Yanan Cao, Carl Yang, and Shirui Pan. Beyond memorization: A rigorous evaluation framework for medical knowledge editing. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4727–4751, 2026.

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

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1762–1777, 2023.

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

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36:47934–47959, 2023.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Xuming Hu, Junzhe Chen, Xiaochuan Li, Yufei Guo, Lijie Wen, S Yu Philip, and Zhijiang Guo. Towards understanding factual knowledge of large language models. In *International Conference on Learning Representations*, 2024.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*, 2025.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020.

LangChain. Langmem: Long-term memory in llm applications. `https://langchain-ai.github.io/langmem/`, 2025. Software documentation.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts. In *International Conference on Machine Learning*, pp. 26396–26415, 2024.

Zhenyu Lei, Patrick Soga, Yaochen Zhu, Yinhan He, Yushun Dong, and Jundong Li. Moledit: Knowledge editing for multimodal molecule language models. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*, pp. 334–344, 2026.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

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

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 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)*, pp. 13851–13870, 2024.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*, 2023.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning*, pp. 15817–15831, 2022.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Hadi Pouransari, David Grangier, Christopher Thomas, Michael Kirchhof, and Oncel Tuzel. Pre-training with hierarchical memories: separating long-tail and common knowledge. *arXiv preprint arXiv:2510.02375*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 452–461, 2009.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5418–5426, 2020.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9414–9423, 2023a.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Advances in Neural Information Processing Systems*, 37, 2024.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36:74530–74543, 2023b.

Yile Wang, Peng Li, Maosong Sun, and Yang Liu. Self-knowledge guided retrieval augmentation for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10303–10315, 2023c.

Yu Wang and Xi Chen. Mirix: Multi-agent memory system for llm-based agents. *arXiv preprint arXiv:2507.07957*, 2025.

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

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2021.

Derong Xu, Ziheng Zhang, Zhihong Zhu, Zhenxi Lin, Qidong Liu, Xian Wu, Tong Xu, Wanyu Wang, Yuyang Ye, Xiangyu Zhao, Enhong Chen, and Yefeng Zheng. Editing factual knowledge and explanatory ability of medical large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 2660–2670, 2024.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. In *Advances in Neural Information Processing Systems*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Tianjun Yao, Yongqiang Chen, Yujia Zheng, Zhiqiang Shen, Pan Li, and Kun Zhang. Augmenting language agents with parametric memory. *NeurIPS 2025 Workshop on Socially Responsible and Trustworthy Foundation Models*, 2025.

Binchi Zhang, Zhengzhang Chen, Zaiyi Zheng, Jundong Li, and Haifeng Chen. Resolving editing-unlearning conflicts: A knowledge codebook framework for large language model updating. *arXiv preprint arXiv:2502.00158*, 2025a.

Guibin Zhang, Muxin Fu, Kun Wang, Guancheng Wan, Miao Yu, and Shuicheng YAN. G-memory: Tracing hierarchical memory for multi-agent systems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

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

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. *Proceedings of the AAAI conference on artificial intelligence*, 38(17):19724–19731, 2024.

A Implementation Details

This appendix provides reproducibility details for ParaMem. Our code is available at `https://anonymous.4open.science/r/paramem-F189`. We first describe how raw conversations are converted into grounded facts and query-supervision pairs, then give adapter training, inference, ParamRecall, and DPO implementation details.

A.1 Data Synthesis Pipeline

Data synthesis converts raw dated conversations into retrieval-oriented supervision for parametric memory. It proceeds in three stages: grounded fact extraction, query generation, and short-answer construction. All LLM calls use the teacher LLM with JSON-mode outputs and are filtered to remove empty or malformed generations. For the open-weight teachers, JSON is extracted from the completion.

Grounded fact extraction. For each dated conversation session, GPT-4o extracts atomic long-term memory facts. The extraction prompt asks for facts that are explicitly supported by the session, written as concise third-person subject-predicate-value statements. Relative temporal expressions are resolved against the session date, and pronouns are rewritten into explicit entities when possible. The prompt encourages broad coverage across stable attributes, temporal events, relations, preferences, updates, and plans. Each extracted fact is returned as a JSON record with `fact_id`, `fact_text`, `facet`, `subject`, and `evidence`. The `fact_id` identifies the underlying memory unit, `subject` denotes the main person or entity, and `facet` describes the answer-bearing dimension of the fact, such as temporal, location, event, preference, opinion, or relation.

Disambiguation. When multiple facts from the same session are likely to be confused, we optionally apply a disambiguation prompt. The prompt asks GPT-4o to append a short parenthetical distinction only to confusable facts while leaving unique facts unchanged. This preserves the original fact inventory while making near-neighbor memories easier to separate during retrieval-oriented training.

Query generation. For each grounded fact, GPT-4o generates diverse natural-language queries that can be answered from that fact alone. The prompt covers multiple query forms, including *what*, *who*, *when*, *where*, *how*, *why*, paraphrase, summary, and implicit questions. Queries that require information not contained in the grounded fact are disallowed. Each query is returned with a `query_id`, `query_text`, and `query_type`.

Answer construction. For each generated query-fact pair, GPT-4o produces the shortest correct answer using only the grounded fact. The short answer is used for direct-answer SFT rows, while the grounded fact text is used as the target for memory-context SFT rows. Thus, the same fact supports both ParaMem interfaces: direct answering and explicit memory-context generation.

Data volume and example. Across the ten LoCoMo users, the GPT-4o teacher extracts 2,435 grounded facts from 272 sessions (about 244 per user). Query generation yields 9.41 queries per fact (22,902 queries), each paired with one memory-context target and one direct-answer target, for 45,804 SFT rows (about 4,580 per user). For example, the fact "Caroline attended a LGBTQ support group yesterday" (session dated 8 May 2023) is resolved to "Caroline attended a LGBTQ support group 7 May, 2023" and produces, among others, the memory-context row *Query: Who attended a LGBTQ support group 7 May, 2023?* $\rightarrow$ *Caroline attended a LGBTQ support group 7 May, 2023.* and the direct-answer row $\rightarrow$ *Caroline*.

A.2 Adapter Training Details

All ParaMem adapters are trained with LoRA on the attention projection matrices (`q_proj`, `k_proj`, `v_proj`, `o_proj`) and MLP projection matrices (`gate_proj`, `up_proj`, `down_proj`).

Unless otherwise specified, we use rank $r = 8$, LoRA $\alpha = 32$, dropout 0.05, and apply LoRA to all transformer layers. The main Qwen2.5-3B and Llama-3.1-8B per-user adapters are trained separately for each LoCoMo user.

SFT training uses a chat-template causal language modeling objective with loss masked on the prompt tokens and applied only to the assistant target. Each training row follows one of two formats: a *memory-context* row, where the assistant target is the relevant grounded memory context, or a *direct-answer* row, where the target is the short answer. The main system uses a 50/50 mixture of these two row types. We train for 5 epochs with AdamW, learning rate 2×10^{-4}, cosine scheduling, warmup ratio 0.03, bfloat16 precision, and effective batch size 32.

A.3 Inference and ParamRecall Details

At inference time, ParaMem can be prompted in either direct-answer mode or memory-context mode. The main experiments use memory-context mode, where the adapter first generates memory evidence under the instruction:

> *You are a memory retrieval module. Return only the stored memory relevant to the query. Do not answer the query directly. If there is no relevant memory, return NO_RELEVANT_MEMORY.*

The generated memory context is then passed to the backbone model for answer generation. We use category-aware short-answer prompts: temporal questions ask for an approximate date when possible; opinion/preference questions are cast as a two-choice selection between the gold answer and "Not mentioned in the conversation"; all other categories use a short-phrase answer prompt.

ParamRecall expands the memory-context interface with input-side query probing and output-side memory sampling. For query probing, we generate type-preserving paraphrases of the test query using:

> *Rephrase the following question in a different way while preserving its exact meaning and question type. Output only the rephrased question.*

For each query probe, we sample multiple memory contexts from the adapter. The resulting candidate memories are deduplicated and concatenated as structured `[Memory 1]`, `[Memory 2]`, ... blocks before the final QA call.

Query paraphrases are generated by the QA backbone at temperature 0.7. With k_s=1, memory contexts are decoded greedily; with k_s>1 they are sampled at temperature T using the backbone's default nucleus settings (Qwen2.5: top-p 0.8, top-k 20, repetition penalty 1.05; Llama-3.1: top-p 0.9), up to 128 new tokens each. Candidates are deduplicated by exact match after lower-casing. If no memory is recalled (`NO_RELEVANT_MEMORY`), ParaMem falls back to the direct-answer interface. The final QA call uses temperature 0.7.

A.4 DPO Negative Construction

DPO pairs are built only from memory-context rows. For each query, the chosen completion c^+ is the correct memory context. The rejected completion c^- is selected from the same user's memory bank to be related but answer-unhelpful.

We mine candidates with BM25 and then filter out same-fact paraphrases to avoid false negatives. The preferred rejected sample is a cross-fact candidate with the same `subject` but a different `facet` or `fact_id`. If no same-subject cross-fact candidate exists, the miner falls back to another cross-fact candidate ranked by BM25. Candidates with a very small normalized edit distance from the chosen context are discarded. This construction encourages DPO to compare answer-bearing memories against plausible but insufficient same-user memories, rather than unrelated noise.

An alternative strategy selects the rejected memory from a different subject who shares the same fact type. Given a positive memory about subject s with facet t (e.g., a temporal event), we select c^- from a different subject $s' \neq s$ with the same facet t. This produces negatives that have the correct answer structure (e.g., a date for a temporal question) but concern the wrong person. We compare cross-fact and different-subject strategies in the ablation study (Table 3).

For DPO, the trainable policy is initialized from the corresponding SFT adapter, while the frozen reference is the same SFT adapter. The main DPO configuration uses learning rate 5×10^{-6}, 0.25 epochs, $\beta = 0.1$, batch size 1, gradient accumulation 16, and maximum sequence length 1024. We

Table 2: Token-level F1 on LoCoMo across question categories, with Qwen3-235B-A22B-Instruct-2507 as the teacher model for all methods.

	Qwen2.5-3B-Instruct						Llama-3.1-8B-Instruct					
Method	Avg	C1	C2	C3	C4	C5	Avg	C1	C2	C3	C4	C5
Token Baselines												
FullContext	0.233	0.055	0.042	0.071	0.100	0.767	0.323	0.117	0.161	0.091	0.201	0.852
Summarization	0.332	0.242	0.108	0.109	0.298	0.662	0.443	0.303	0.340	0.143	0.357	0.832
LangMem	0.214	0.098	0.016	0.142	0.105	0.651	0.225	0.027	0.013	0.063	0.054	0.862
Mem0	0.259	0.157	0.018	0.116	0.178	0.681	0.322	0.209	0.082	0.133	0.225	0.788
ReadAgent	0.287	0.144	0.092	0.046	0.214	0.706	0.378	0.217	0.175	0.131	0.266	0.890
SimpleMem	0.354	0.242	0.067	0.120	0.307	**0.768**	0.398	0.260	0.163	0.119	0.318	0.863
A-Mem	0.339	0.235	0.145	0.141	0.249	0.757	0.377	0.256	0.212	0.111	0.278	0.814
Parametric Baselines												
CLM	0.036	0.028	0.012	0.065	0.036	0.053	0.036	0.025	0.011	0.047	0.041	0.048
Param A-Mem	0.277	0.160	0.145	0.132	0.165	0.686	0.261	0.134	0.154	0.101	0.170	0.627
ParaMem												
ParaMem-direct	0.291	0.287	0.373	**0.209**	0.406	0.038	0.302	**0.317**	0.380	**0.182**	0.424	0.034
ParaMem-context	0.339	0.248	0.243	0.179	0.373	0.436	0.388	0.257	0.396	0.138	0.411	0.476
+ParamRecall	0.361	0.269	0.319	0.197	0.385	0.440	0.399	0.257	0.414	0.168	**0.429**	0.468
+ParamRecall+grounding	**0.457**	**0.301**	**0.399**	0.125	**0.416**	0.747	**0.492**	0.291	**0.457**	0.158	0.395	**0.898**

Table 3: DPO negative selection (Qwen2.5-3B, DeepSeek teacher). All rows start from the same SFT adapter and use the same ParamRecall setup (k_q=2, k_s=3) at inference.

System	Avg	C1	C2	C3	C4	C5
SFT (no preference learning)	0.347	0.267	0.338	0.193	0.370	0.393
DPO + BM25-hard negatives	0.348	**0.273**	0.336	0.174	0.368	**0.402**
DPO + cross-fact negatives	0.337	0.263	0.333	0.187	0.363	0.370
DPO + different-subject negatives	**0.349**	0.265	**0.341**	**0.199**	**0.372**	0.398

use this small update because larger preference updates can damage the adapter's already-learned recall behavior.

B Supplementary Experimental Results

B.1 Negative construction makes preference learning helpful.

Table 3 compares three negative construction strategies for DPO: **BM25-hard** selects the highest BM25-scoring memory from the same user's memory bank (excluding the positive); **cross-fact** selects a same-subject memory with a different fact identity or semantic facet; **different-subject** selects a memory about a different person with the same fact type. Different-subject negatives produce the best result. Cross-fact negatives *hurt* performance: same-subject distractors are too close to the positive, creating false-negative pairs that disrupt both answerable recall and adversarial robustness. BM25-hard negatives are nearly neutral: in synthesized memory data, the highest BM25-scoring alternative is often a paraphrase of the positive, and penalizing valid memory expressions provides little useful signal. The results show that preference learning is useful for parametric memory only when negatives are constructed at the right semantic granularity. Different-subject negatives provide the clearest signal because they share the answer type but differ in the subject, forcing the adapter to attend to who the memory is about, not just what type of information it contains.

B.2 Controlled Study on Retrieval Methods

Because ParaMem's facts are themselves teacher-extracted, we isolate the contribution of the parametric substrate with two controls that reuse the same fact store: token retrieval over the facts with no adapter, and letting the backbone generate a hypothetical memory that is then grounded against the store. Both controls perform competitively, confirming that the grounded fact store accounts for the majority of the absolute gain over token-memory baselines. However, the adapter's contribution becomes measurable at scale: on Llama-3.1-8B, ParaMem + grounding consistently outperforms both controls, with the largest margin on multi-hop questions where generative recall can surface cross-session associations that lexical retrieval misses. On the smaller Qwen2.5-3B backbone, the

Table 4: Controls that reuse ParaMem's own teacher-extracted fact store without the per-user adapter (DeepSeek-V3.1 teacher). "Facts + BM25" retrieves facts with the question only; "no-adapter generation" lets the backbone write a hypothetical memory that is then grounded exactly as ParaMem's recalled memories are. Grounding denotes snapping each recalled memory to its nearest stored fact *and* adding BM25(question) facts; the ablation row drops the latter.

	Qwen2.5-3B-Instruct						Llama-3.1-8B-Instruct					
Method	Avg	C1	C2	C3	C4	C5	Avg	C1	C2	C3	C4	C5
Facts + BM25, no adapter	0.437	0.273	**0.394**	0.083	0.376	0.761	0.474	0.256	0.442	0.138	0.388	0.868
No-adapter gen + grounding	0.450	0.298	0.392	**0.138**	0.392	**0.764**	0.478	**0.292**	0.428	**0.157**	0.386	0.873
ParaMem + snapped facts only	0.417	0.300	0.350	0.123	0.379	0.674	0.489	0.256	0.459	0.156	**0.407**	0.884
ParaMem + grounding	**0.453**	**0.307**	0.393	0.107	**0.403**	0.757	**0.495**	0.277	**0.469**	0.135	**0.407**	**0.896**

Table 5: Storage-accuracy tradeoff across token-memory and parametric-memory systems on LoCoMo-10. *MB/user* denotes inference-time storage per user. For token-memory systems, storage includes text memory records and retrieval embeddings when used. For ParaMem, storage includes LoRA adapter weights in bfloat16; shared adapters amortize one adapter across 10 users, grouped adapters amortize three adapters across 10 users, and per-user adapters allocate one adapter per user.

System	**Memory granularity**	**MB/user**	**F1**	**Δ vs. A-Mem**
Token-memory baselines				
LangMem	extracted text memories	0.01	0.068	−0.210
Summarization	session summaries	0.02	0.206	−0.072
A-Mem	extracted facts + vectors	0.07	0.278	0.000
FullContext	raw history text	0.13	0.092	−0.186
Mem0	dynamic memories + vectors	0.32	0.168	−0.110
SimpleMem	compressed memories + vectors	0.77	0.231	−0.047
ParaMem (ours): shared or grouped adapters				
r=2, l=5	shared adapter (×1)	0.05	0.205	−0.073
r=4, l=5	shared adapter (×1)	0.10	0.219	−0.068
r=4, l=5	grouped adapters (×3)	0.31	0.225	−0.062
r=4, l=all	shared adapter (×1)	0.74	0.284	+0.006
ParaMem (ours): per-user adapters				
r=2, l=5	per-user adapters (×10)	0.52	0.235	−0.043
r=4, l=5	per-user adapters (×10)	1.03	0.239	−0.039
r=8, l=5	per-user adapters (×10)	2.05	0.267	−0.022
r=8, l=10	per-user adapters (×10)	4.11	0.299	+0.021
r=8, l=all	per-user adapters (×10)	14.78	0.303	+0.025
r=8, l=all (8B)	per-user adapters (×10)	27.30	0.354	+0.076

controls are statistically indistinguishable from ParaMem, suggesting that the parametric recall advantage depends on sufficient backbone capacity to exploit the adapter's learned retrieval signal. These results position parametric and token memory as complementary: the fact store provides faithful, attributable content, while the adapter adds a learned retrieval channel whose value grows with model scale.

B.3 Storage-Accuracy Tradeoff

Table 5 studies the storage-accuracy tradeoff of token-memory and parametric-memory systems on LoCoMo-10. This analysis complements the main results by asking how memory representation affects not only F1, but also the cost of maintaining long-term personal memory at inference time. The goal is not to show that token memory is uniformly storage-inefficient. Indeed, compact token-memory systems can be highly effective when records are well extracted, compressed, and organized. Rather, this study highlights a difference in scaling mechanisms. Token-memory systems control storage by deciding what text records to keep, compress, merge, or index; their storage and retrieval costs remain tied to the external memory inventory. ParaMem instead stores memory in LoRA adapter weights. For a fixed adapter configuration, the active memory representation is fixed-size with respect

to the number of extracted memory records, while its capacity is bounded by the adapter size, and its total cost depends on whether adapters are shared, grouped, or assigned per user.

Three observations emerge. First, storage alone does not determine token-memory performance. Among token-memory baselines, A-Mem achieves the strongest F1 with a compact graph-like memory store, while FullContext stores more raw text but performs worse. This suggests that memory organization and retrieval quality matter more than simply storing more text. Second, ParaMem exposes a distinct accuracy-capacity tradeoff. Small shared adapters provide low amortized per-user storage, while larger per-user adapters improve accuracy by allocating more personalized capacity to each user. This tradeoff differs from token memory: increasing capacity means increasing adapter rank, depth, or routing granularity, rather than appending more text records. Third, the parametric-memory ceiling improves with backbone scale. At the same LoRA configuration, switching from Qwen2.5-3B to Llama-3.1-8B substantially improves F1, suggesting that stronger base models can better interpret and use memory recalled from adapter weights.

Overall, the storage results refine rather than weaken the motivation for ParaMem. Token memory offers transparent, incrementally editable records and can be storage-efficient when those records are carefully compressed or structured, but its memory substrate remains an external record store. ParaMem replaces record-level growth with an adapter-capacity tradeoff, suggesting a complementary design axis for long-term memory systems.

C Limitations and Broader Impact

Limitations. This work studies an early formulation of parametric memory, and several limitations remain. First, ParamRecall addresses the coverage problem at inference time by probing the adapter with multiple query forms and sampling multiple memory contexts. An alternative direction is to address coverage during training by directly supervising the adapter to generate multiple relevant memories for a query, or by training it with set-valued memory targets. Such objectives may reduce the need for inference-time sampling, but they also require reliable construction of multi-memory supervision and mechanisms for controlling redundancy and ordering. Second, ParaMem relies on a teacher LLM to synthesize grounded facts, queries, and answer targets. This follows common practice in synthetic instruction and memory construction, but it introduces additional token cost and potential dependence on the quality of the teacher model. Future work should study cheaper synthesis pipelines, self-training, incremental data construction, and human-verifiable memory extraction. Third, while a fixed LoRA adapter avoids growth with the number of extracted memory records for a chosen configuration, it does not make the memory free. Capacity is bounded by adapter size, and total storage depends on whether adapters are shared, grouped, or assigned per user. Our storage study shows that parametric memory defines a different accuracy-capacity tradeoff rather than uniformly dominating token memory. Finally, ParaMem is currently evaluated on conversational QA benchmarks. Real assistants must handle continual updates, memory deletion, privacy controls, user correction, and safety-sensitive personalization. These requirements are natural next steps for making parametric memory practical in deployed systems.

Broader impact. ParaMem suggests that long-term AI memory need not be restricted to external token stores. By showing that personal history can be encoded into user-specific adapter weights and recalled as explicit textual evidence, this work opens a complementary direction for agent memory, LLM memory, and memory architecture research. For agent-memory systems, our results translate familiar retrieval principles into parameter space: ParamRecall acts as a parametric analogue of multi-candidate retrieval, while cross-fact preference learning adapts hard-negative ranking ideas to generated memory contexts. For LLM memory research, ParaMem provides a concrete setting where parametric knowledge is not only pretraining-scale world knowledge, but also dynamic, user-specific, and query-conditioned. For memory architectures more broadly, the findings highlight a new design axis between editable token memories and compact learned memories, where systems may combine external records for transparency and fast updates with parametric memories for semantic recall and fixed-size personalization. At the same time, personal memory is sensitive. Encoding user history into parameters could make memory harder to inspect, edit, or delete than external text records if not designed carefully. We therefore view ParaMem as a step toward hybrid memory systems that preserve explicit evidence and user control while exploring the semantic capacity of learned parameters.