MemFlow: Learning Compact Soft Memory for Long-Context Personalization via On-Policy Optimization

Anonymous authors
Paper under double-blind review

Abstract

An assistant that serves the same user over a long horizon has to answer from what that user has revealed: which preferences still hold, which were revised, and which constraints apply now. Retaining that information is not the same as acting on it, and the two are usually optimized as if they were. Keeping the information as text makes the reader's input grow with the retained history, while compressing it into a fixed number of latent vectors bounds the interface but is typically trained to reconstruct text or imitate reference answers, both of which are scored on sequences the reader never produced. We present **MemFlow**, which optimizes a fixed-budget soft memory by the behavior it supports. A query-conditioned textual memory is compressed into K continuous vectors that form the reader's memory interface, and the reader is then trained on its own rollouts under two complementary signals: group-relative rewards for task outcomes, and confidence-gated on-policy distillation in which a frozen textual-memory teacher re-scores the student's sampled tokens under the textual memory. The teacher is never sampled from, so supervision stays on the student's current distribution and adds no autoregressive decoding; at inference it is removed entirely. Across three Qwen backbones, **MemFlow** attains the highest accuracy we measure on PersonaMem-32K and PersonaMem-128K, with margins that widen at the longer history length, and transfers to PrefEval and LongMemEval without target-domain training. Ablations attribute most of the task gain to the reward term and a smaller additional gain to the teacher signal, and memory interventions show that the reader depends on the instance-specific content of its soft memory.



Figure 1: Using Qwen2.5-3B-Instruct on PersonaMem-32K, **MemFlow** improves test accuracy more rapidly during training and achieves higher final performance than the baselines (left), while reaching comparable test accuracy with fewer student rollouts (right), demonstrating both effective optimization and greater sample efficiency. Experimental settings are provided in Appendix B.1.

1 Introduction

A long-running assistant is asked similar questions by different people, and the suitable answer is not the same for each of them. What makes a response suitable is the set of facts, preferences, and constraints that this particular user has revealed in earlier interactions. The same holds for one user across time: a request phrased identically in March and in September can call for different advice

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.

because what the user wants has changed in between. Personalization in this setting is therefore a property of the response rather than of the store. It is the extent to which an answer reflects the information about this user that is in force at the moment the answer is produced.

That last qualification does much of the work. A long history mixes enduring preferences with one-off activities, constraints that held only for a period, explicit revisions, and the reasons behind those revisions. Recency does not resolve them: the most recent mention of an activity is not necessarily a new preference, and an early statement is not necessarily stale. On benchmarks built around evolving user profiles, models recover the course of a preference change and the reasons behind it more reliably than they let the resulting preference constrain a concrete recommendation (Jiang et al., 2025), and they continue to recommend options a user has objected to when the objection was expressed implicitly rather than as an instruction (Zhao et al., 2025; Guo et al., 2026). Evaluations that score forgetting alongside recall find agents still acting on information a later turn has invalidated (Uddin et al., 2026), and condensed histories can retain the facts of an episode while losing the preference signal that made it matter (Wang et al., 2026). Retaining the evidence and acting on it are thus separate requirements, and a memory mechanism has to be judged against both.

Two representations carry that evidence forward. Textual memory keeps it as text, which is readable and editable, but its length varies with the amount retained and it is re-encoded at every turn (Packer et al., 2023; Zhong et al., 2024; Chhikara et al., 2025; Xu et al., 2025; Luo et al., 2026; Liao et al., 2026). Latent memory replaces the text with a fixed number of continuous vectors, giving the reader an interface whose size is independent of how long the interaction has been (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024; Cheng et al., 2024; Li et al., 2026; Zhou & Sang, 2026). That fixed budget is a property of the interface, and it does not by itself determine whether the compressed representation still supports the behavior the text did.

The gap this leaves is one of optimization rather than representation. Compressors are typically trained to reconstruct the source text (Ge et al., 2024; Zhang et al., 2026c) or to align with reference answers under a frozen decoder (Rahman et al., 2026), and both objectives score the model on sequences it did not produce, so neither reaches the errors the reader makes once the text is gone. Recent work closes part of this loop by treating memory as a resource a policy learns to use, optimizing what a memory stores, retrieves, or spends its budget on against a task reward (Fu et al., 2026; Feng et al., 2026; Yu et al., 2026b;a; Ye et al., 2026). A scalar outcome, however, reports only that a response was wrong; it does not indicate which decisions in it failed to use what the memory held. A characteristic failure of this kind is a response that is fluent and topically appropriate, with the preference violation confined to one recommendation among several, which we examine in Section 6. A sequence-level reward charges the entire response for that span.

Our starting point is to judge a compact memory not by whether it returns the text, but by where the reader, given only the compact memory, under-uses what the text would support. This can be measured directly, because the textual and compressed memories can be evaluated on the same response. Given a response sampled by a reader conditioned on soft memory, a frozen copy of that reader conditioned on the corresponding textual memory can score the identical token sequence. The resulting token-level log-probability differences locate where the reader is less confident under compression than under the text, on trajectories the reader actually visits. We use them to weight the reader's own tokens rather than as a target to imitate, so the signal complements the task reward: the reward indicates whether a response is correct, while the comparison indicates which of its tokens the textual reading supports more strongly.

Based on this observation, our contributions are threefold: ❶ Prior compressors are trained on sequences the reader never produces; we propose **`MemFlow`**, which optimizes a fixed-budget soft-memory interface directly by the downstream behavior it supports, on the reader's own rollouts. ❷ We introduce a confidence-gated on-policy distillation objective that re-scores the student's own generations under textual memory; in expectation it acts as a reverse KL toward the textual-memory reader with bounded per-token influence, adding a dense signal to the sequence-level reward without letting the teacher dominate it. ❸ We show that **`MemFlow`** achieves the highest accuracy on all four benchmarks across three backbones, with margins that widen at longer history lengths, and transfers to unseen benchmarks without target-domain training.

2 Related Work

Memory and personalization in long-horizon interaction. Long-horizon assistants are commonly given an explicit textual store, written across sessions and queried at inference (Lewis et al., 2020; Packer et al., 2023; Zhong et al., 2024; Xu et al., 2025; Chhikara et al., 2025; Fang et al., 2026; Luo et al., 2026; Liao et al., 2026), with benchmarks measuring the recall and multi-session reasoning that results (Maharana et al., 2024; Wu et al., 2024; Chang & Chen, 2026). A second line asks whether retained information changes what the model says to a given user, through personalized generation from user profiles (Salemi et al., 2024; Salemi & Zamani, 2025; Zhang et al., 2026b; In et al., 2026), histories in which preferences develop over time (Jiang et al., 2025), preferences expressed implicitly rather than as instructions (Zhao et al., 2025; Guo et al., 2026), and whether information a later turn has invalidated is dropped as well as recalled (Uddin et al., 2026). We adopt the distinction those benchmarks draw: retrieval accuracy over a history and preference-consistent behavior in a response are different quantities. Because the content stays in text, the tokens the reader processes also grow with the amount retained, and the store is optimized separately from the model consuming it.

Compact memory and behavioral optimization. A complementary line shortens the representation, by dropping or rewriting tokens (Jiang et al., 2023; Li et al., 2024) or, building on prompt and prefix tuning (Lester et al., 2021; Li & Liang, 2021), by encoding context into continuous vectors read in the embedding space (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2024; Cheng et al., 2024; Zhang et al., 2026a;c; Li et al., 2026; Zhou & Sang, 2026), often through latent-query resamplers (Jaegle et al., 2022; Alayrac et al., 2022) as ours is. Their objectives, reconstruction or supervised imitation, are scored on sequences the reader did not generate, as is distillation that aligns a compressed context with reference answers under a frozen decoder (Rahman et al., 2026). Reinforcement learning instead trains on the model's own samples (Ouyang et al., 2022; Shao et al., 2024; Guo et al., 2025) and has been used to decide what a memory stores, retrieves, or spends its budget on (Yan et al., 2025; Wang et al., 2025; Yu et al., 2026b;a; Ye et al., 2026; Song et al., 2026; Fu et al., 2026; Feng et al., 2026), whereas we hold the interface fixed and optimize how it is read; on-policy distillation adds the per-token signal a scalar reward lacks (Agarwal et al., 2024; Gu et al., 2024; Zhao et al., 2026) but uses a stronger teacher and a signed gap that pulls the student toward it. Ours is closer to context distillation (Snell et al., 2022): a frozen copy of the student's own initialization, never sampled from, re-scores the student's tokens under the textual memory it has had compressed away, and a non-negative gate replaces the signed gap. The two differ only in their memory, which ties the signal to compression rather than to a more accurate solver.

3 Methodology

`MemFlow` has two parts: a fixed-budget memory interface and an on-policy procedure that optimizes how the reader uses it. A memory writer turns the history visible at query time into query-relevant textual memory, a compressor maps that memory into K soft vectors, and the reader is then trained on its own rollouts under a task reward together with a frozen textual-memory teacher that re-scores those same rollouts (Figure 2). Supervised training is used only to initialize the interface; its data construction and training details are deferred to Appendix A.1.1 and Appendix A.1.2.

3.1 Problem Formulation

A query x arrives at time τ, and the model may condition only on the interaction history C visible before τ. Because no annotation indicates which statements in C remain valid, the model must infer them from the history. All components share a frozen backbone θ_0 and a single LoRA adapter θ. The adapter first produces a query-conditioned textual memory $M = e_\theta(C, x)$ containing the relevant evidence, temporal relations, and derived facts. A compressor $\mathcal{C}_\phi$ then maps it to a fixed-size continuous memory $Z = \mathcal{C}_\phi(E(M)) \in \mathbb{R}^{K \times d}$, where E comprises the first four Transformer blocks of θ_0, K is the memory budget, and d is the reader's input embedding dimension.

The same adapter, acting as the reader, generates $y \sim \pi_\theta(\cdot \mid x, Z)$. During on-policy training, a frozen copy of the initialized reader, $\pi_T(\cdot \mid x, M) := \pi_{\theta_{\text{init}}}(\cdot \mid x, M)$, instead reads the textual memory and serves as a behavioral reference. Teacher and student thus initially share all weights and differ only in their memory inputs. Both score each student-sampled trajectory, and their confidence

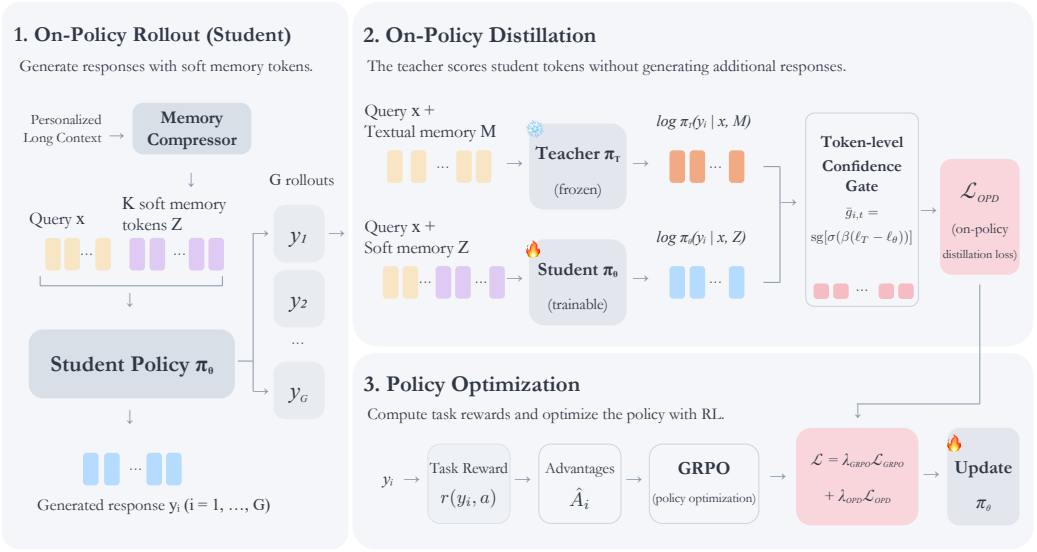


Figure 2: ❶ **On-policy rollout:** The student generates responses conditioned on the query and K compressed soft memory tokens. ❷ **On-policy distillation:** A frozen textual-memory teacher scores the student's tokens, and a detached teacher–student confidence gate $\bar{g}_{i,t}$ weights the student's own tokens. ❸ **Policy optimization:** GRPO task rewards and the gated distillation term jointly update the student, while the teacher and compressor remain frozen.

gap identifies tokens for which the compressed-memory reader is less confident. We use this gap to weight the student's own tokens alongside the task reward, bounding the teacher's influence on each token rather than matching its distribution. At inference, only $\pi_\theta(\cdot \mid x, Z)$ is retained, so the memory interface remains fixed at K vectors regardless of the history length.

3.2 Fixed-Budget Soft Memory

This component supplies the interface that Section 3.3 acts on. Two properties are what the method requires of it: its size does not depend on the length of C, and the reader can read it. We instantiate these requirements with a standard Perceiver-style compressor.

Textual memory. We first transform the long context C into a structured, query-relevant textual memory M, filtering irrelevant history before compression. The adapter is first trained on memories extracted by an external model from the history and query alone, and thereafter writes M itself, so the external model is not needed at inference.

Fixed-budget compression. We encode M with the frozen encoder E, the first four Transformer blocks of the backbone, and a lightweight Perceiver-style compressor aggregates the variable-length sequence with K learned latent queries Q_K:

$$Z = \mathcal{C}_\phi\big(E(M)\big) = P_\phi\big(\mathrm{Comp}_\phi(Q_K, E(M))\big) \in \mathbb{R}^{K\times d},$$

where P_ϕ projects the compressed states into the reader's input embedding space. The reader therefore always receives exactly K memory vectors, regardless of the lengths of C and M.

Initialization. We first pretrain the compressor on raw history prefixes h: the frozen backbone must reconstruct the textual memory from $Z_h = \mathcal{C}_\phi(E(h))$,

$$\mathcal{L}_{\mathrm{rec}} = -\frac{1}{|M|}\sum_{t=1}^{|M|}\log p_{\theta_0}(m_t \mid x, Z_h, m_{<t}), \tag{1}$$

then adapt it to textual-memory inputs, $Z = \mathcal{C}_\phi(E(M))$, and initialize the reader on answer generation from (x, Z). These objectives are scored on sequences the reader did not produce, so they make the interface readable without directly optimizing how the reader uses it on its own generations. On-policy optimization addresses this remaining gap.

Algorithm 1 **MemFlow** on-policy memory optimization

Require: Reader-initialized adapter θ_{init}, compressor C_ϕ, frozen teacher π_T; training examples (C, x, a) with reference answer a and cached on-policy memories $M^{\text{init}} = e_{\theta_{\text{init}}}(C, x)$
Ensure: Trained adapter θ
Initialize $\theta \leftarrow \theta_{\text{init}}$; freeze compressor, projector, and teacher
for each training minibatch of (C, x, a) **do**
  Load cached M^{init} and compute $Z \leftarrow C_\phi(E(M^{\text{init}}))$
  Snapshot rollout policy $\pi_{\text{old}} \leftarrow \text{sg}[\pi_\theta]$
  Sample $y_1, \ldots, y_G \sim \pi_{\text{old}}(\cdot \mid x, Z)$
  Score rewards $r_i = r(y_i, a)$ and compute $\hat{A}_i$
  for each sampled response i and unmasked token t **do**
    $\ell_{\theta,i,t} \leftarrow \log \pi_\theta(y_{i,t} \mid x, Z, y_{i,<t})$
    $\ell_{T,i,t} \leftarrow \log \pi_T(y_{i,t} \mid x, M^{\text{init}}, y_{i,<t})$
    $\bar{g}_{i,t} \leftarrow \text{sg}[\sigma(\beta(\ell_{T,i,t} - \ell_{\theta,i,t}))]$
  end for
  Compute $\mathcal{L}_{\text{GRPO}}$ and $\mathcal{L}_{\text{OPD}}$ over shared response masks
  Update the LoRA adapter θ
end for
return θ; at inference, write $M = e_\theta(C, x)$ and answer from $(x, C_\phi(E(M)))$ without the teacher

3.3 On-Policy Memory Optimization

Initialization leaves two gaps. First, a reader trained on reference answers receives no signal about errors in its own generations. Second, a task reward does score the reader's own samples, but one scalar per response does not identify which tokens failed to use what the textual memory held. **MemFlow** closes the first by optimizing on the student's rollouts, and the second by re-scoring those same rollouts under the textual memory. Algorithm 1 summarizes the resulting loop.

On-policy rollouts. For each query x, the current soft-memory policy samples a group of responses $\{y_i\}_{i=1}^G$: $y_i \sim \pi_\theta(\cdot \mid x, Z)$. Each response receives a task reward r_i, from which we compute the group-relative advantage $\hat{A}_i = \frac{r_i - \mu_r}{\sigma_r + \epsilon}$. Let $\rho_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t}|x,Z,y_{i,<t})}{\pi_{\text{old}}(y_{i,t}|x,Z,y_{i,<t})}$. The group-relative policy objective is

$$\mathcal{L}_{\text{GRPO}} = -\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\Big(\rho_{i,t}\hat{A}_i,\ \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\hat{A}_i\Big)\right]. \tag{2}$$

On-policy memory distillation. GRPO evaluates complete responses but does not reveal where the soft-memory reader under-uses what the memory contains. The textual-memory teacher π_T offers a reference for how the same backbone reads the uncompressed memory: it scores each token sampled by the student,

$$\ell_{T,i,t} = \log \pi_T(y_{i,t} \mid x, M, y_{i,<t}), \qquad \ell_{\theta,i,t} = \log \pi_\theta(y_{i,t} \mid x, Z, y_{i,<t}), \tag{3}$$

and we weight the student's own tokens by a detached gate $\bar{g}_{i,t} = \text{sg}[\sigma(\beta(\ell_{T,i,t} - \ell_{\theta,i,t}))]$:

$$\mathcal{L}_{\text{OPD}} = -\,\mathbb{E}\Big[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\bar{g}_{i,t}\,\ell_{\theta,i,t}\Big]. \tag{4}$$

The objective strengthens the student's sampled tokens in proportion to how much more the textual-memory reading supports them than the soft-memory reading does. Per sample, the gate is largest where the teacher is more confident and approaches zero where it is less confident than the student. The objective never promotes tokens the student did not sample, so it re-ranks the student's own candidates by the teacher's relative confidence rather than importing the teacher's own choices; in expectation, this re-ranking acts as a reverse KL with a bounded per-token coefficient. The bound is deliberate: π_T is a frozen reader of the textual memory, not an accuracy oracle, so no single token on which it is strongly over- or under-confident can dominate the update. Signed feedback on task outcomes comes from GRPO.

Joint objective and training. The two signals address different gaps and are combined as

$$\mathcal{L}_{\text{MemFlow}} = \lambda_{\text{GRPO}}\mathcal{L}_{\text{GRPO}} + \lambda_{\text{OPD}}\mathcal{L}_{\text{OPD}},$$

with the textual-memory teacher and compressor frozen throughout and no additional KL regularization in the final configuration. Because the gate and teacher log-probability are detached, $\mathcal{L}_{\text{OPD}}$ reduces to a gate-weighted likelihood on the student's own samples with no gradient through the teacher branch; its expected update vanishes when the soft and textual readings agree and weakens as they converge (Appendix C). At inference time, the teacher and all distillation computations are removed, and generation conditions only on (x, Z); additional implementation details are in Appendix A.1.3.

4 Experiments

4.1 Experimental Setup

Datasets & Benchmarks. We evaluate on several benchmarks with complementary focuses. PersonaMem-32K and PersonaMem-128K (Jiang et al., 2025) evaluate dynamic user preference tracking under increasingly long conversational histories. We train on PersonaMem-32K and evaluate directly on the implicit persona subset of PrefEval (Zhao et al., 2025) to test cross-dataset personalization generalization. We also train on LoCoMo (Maharana et al., 2024) and evaluate directly on LongMemEval (Wu et al., 2024) without task-specific training to assess generalization to broader long-term memory reasoning tasks.

Backbone Models. We evaluate our method with three backbone models: Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct (Qwen et al., 2025), as well as Qwen3-4B (Yang et al., 2025).

Baselines. We organize the baselines into three categories. For training objectives, we compare with GRPO (Shao et al., 2024) and OPSD (Zhao et al., 2026). For latent-memory and context-compression methods, we include AutoCompressor (Chevalier et al., 2023), MemGen (Zhang et al., 2026a), and xRAG (Cheng et al., 2024). These methods compress textual context into compact continuous representations for downstream inference. Finally, Full Text serves as the uncompressed-context baseline, directly providing the complete textual history to the backbone model.

Evaluation.

We report accuracy (Acc.) and average end-to-end token-equivalent count (#Tok.) per test instance; accounting rules are in Appendix B.2. PrefEval uses PersonaMem-32K checkpoints; predictions on PrefEval and LongMemEval are scored via the Gemini-3.8-flash API (Google DeepMind, 2026) with low thinking, with LongMemEval additionally using structured yes/no outputs. Greedy decoding ($T = 0$) is used by default; ablations report mean accuracy and Pass@16 over 16 samples ($T = 1.0$, top-$p = 0.98$).

4.2 Results

Table 1 reports accuracy and average end-to-end tokens per test instance for every method under a shared backbone and evaluation protocol. **`MemFlow`** obtains the highest accuracy on every benchmark and backbone, tying the strongest baseline in one cell on PrefEval. Two patterns in the table matter more than the individual entries. First, the advantage widens with history length: the margin over the best competing method is larger at PersonaMem-128K than at 32K for all three backbones, and several baselines that are competitive at 32K fall sharply at 128K, whereas **`MemFlow`** does not, even though its reader is given the same K vectors in both settings and only the history behind them grows. Second, the accuracy does not come from spending more at inference: average end-to-end cost stays close to full-context inference and is lower in aggregate for all three backbones. The consistent exception is PrefEval, where the history is too short for reader-side savings to offset memory-construction costs. xRAG uses slightly fewer tokens, but at a substantial accuracy cost. What the table cannot show is where the gain originates, or whether the reader is using the memory at all; we take those up in Sections 5 and 6, respectively.

Table 1: Main results for in-domain and direct cross-dataset evaluation, reporting accuracy and end-to-end token-equivalent counts. Weighted averages use benchmark sample counts. Best and second-best accuracies are highlighted for each backbone and benchmark.

Model	Method	In-Domain				Out-of-Domain					
		PersonaMem-32K		PersonaMem-128K		PrefEval		LongMemEval		Weighted Avg.	
		Acc.↑	#Tok.↓	Acc.↑	#Tok.↓	Acc.↑	#Tok.↓	Acc.↑	#Tok.↓	Acc.↑	#Tok.↓
Qwen2.5-3B Instruct	Full Text	46.0	24,518	21.9	124,317	12.9	1,893	26.6	145,521	18.8	58,803
	xRAG	36.0	28,714	55.8	121,118	8.5	1,693	10.2	115,744	15.9	50,040
	AutoCompressor†	32.0	25,006	30.5	123,999	–	–	–	–	–	–
	MemGen	54.0	49,510	66.1	123,302	13.3	19,386	3.8	208,378	18.7	86,809
	GRPO	68.0	27,297	58.4	123,410	11.3	1,893	26.0	145,521	23.2	58,762
	OPSD	54.0	27,297	29.6	123,410	12.8	1,893	26.2	145,521	19.9	58,762
	MemFlow	**70.0**	24,395	**88.4**	123,229	**19.9**	2,692	**32.4**	122,545	**33.8**	52,662
Qwen2.5-7B Instruct	Full Text	60.0	24,517	24.0	124,317	14.0	1,893	25.4	145,532	19.8	58,806
	xRAG	62.0	28,733	64.9	121,118	10.9	1,703	12.2	115,754	19.8	50,049
	AutoCompressor†	66.0	25,506	30.7	123,999	–	–	9.6	109,030	–	–
	MemGen	76.0	49,510	78.5	123,302	14.1	20,002	11.0	178,477	23.4	78,769
	GRPO	70.0	27,297	62.2	123,411	14.1	1,893	25.0	145,535	25.0	58,766
	OPSD	64.0	27,297	47.2	123,420	13.2	1,895	25.0	145,542	22.4	58,770
	MemFlow	**88.0**	24,851	**94.4**	124,640	14.1	2,192	**36.8**	122,089	**33.0**	52,451
Qwen3-4B	Full Text	56.0	24,500	4.3	124,301	13.6	1,894	27.8	145,506	17.6	58,796
	xRAG	66.0	28,934	65.2	121,146	2.8	1,827	6.6	115,925	13.8	50,176
	AutoCompressor†	58.0	25,519	30.8	123,982	–	–	14.2	107,087	–	–
	MemGen	56.0	49,510	65.7	124,302	12.3	19,717	10.4	112,881	20.0	60,345
	GRPO	62.0	24,500	65.7	123,393	13.8	1,895	27.4	145,527	25.7	58,684
	OPSD	74.0	24,500	39.5	123,441	13.8	1,895	26.0	145,527	22.3	58,691
	MemFlow	**84.0**	24,395	**89.4**	124,155	**15.2**	2,268	**38.6**	123,488	**33.4**	52,810

† AutoCompressor yields no valid outputs in these settings: its PersonaMem-32K-trained checkpoints return a single prediction on PrefEval, while the 3B checkpoint produces malformed LongMemEval outputs.

5 Ablation Study

Table 2 isolates the contributions of memory-interface initialization and the two on-policy objectives. Reader initialization produces the largest initialization gain, indicating that the model must first learn to consume the soft-memory interface before on-policy optimization can be effective. Writer initialization provides an additional gain by improving the textual memory from which the soft representation is constructed. Among the on-policy objectives, GRPO accounts for most of the task improvement, while OPD provides a complementary gain through token-level guidance from the textual-memory teacher when combined with GRPO. Finally, the text-space control performs comparably but does not outperform the full soft-memory model, suggesting that fixed-budget compression is not the primary bottleneck under this training configuration.

Table 2: Ablation of initialization and on-policy objectives. The text-space control uses textual memory, whereas **MemFlow** uses fixed-budget soft memory. ✓ = satisfies, ✗ = does not satisfy, and *N/A* = not applicable.

Method	Initialization		On-Policy Objectives		Memory Interface	PersonaMem-32K		PersonaMem-128K	
	Writer	Reader	OPD	GRPO		Mean ↑	Pass@16 ↑	Mean ↑	Pass@16 ↑
w/o Writer Initialization	✗	✓	✓	✓	Soft	70.8	82.0	80.0	85.7
w/o Reader Initialization	✓	✗	✓	✓	Soft	47.9	76.0	46.2	67.0
w/o OPD	✓	✓	✗	✓	Soft	71.9	84.0	86.9	91.8
w/o GRPO	✓	✓	✓	✗	Soft	58.6	78.0	70.2	82.3
Initialization Only	✓	✓	✗	✗	Soft	60.5	82.0	70.8	85.1
Text-Space Control	✓	*N/A*	✓	✓	Text	69.3	**88.0**	85.5	91.9
MemFlow	✓	✓	✓	✓	Soft	**75.4**	**88.0**	**87.9**	**93.6**

6 Discussion

We examine how the learned interface transfers across datasets, how efficiently it trains, how its budget affects accuracy and cost, and whether the reader uses instance-specific memory. A final example illustrates a failure of personalization in a single response.

Does a soft-memory interface learned on one dataset still work when the dataset, the evaluation format, and the kind of memory the task demands all change?

We consider two OOD settings that target different memory capabilities. For PrefEval, we directly evaluate frozen PersonaMem-32K checkpoints on its implicit-persona subset. This setting preserves the underlying task of modeling user preferences while changing the dataset distribution and evaluation format, thereby measuring cross-dataset personalization transfer. For LongMemEval, we evaluate checkpoints trained and selected only on LoCoMo. This setting tests a broader form of long-context memory generalization involving multi-session reasoning, temporal relations, and knowledge updates. As shown in Table 1, source-domain improvements do not necessarily translate into stronger OOD performance. On PrefEval, GRPO and OPSD provide inconsistent gains and can even underperform the untrained Full Text baseline for some backbones, suggesting that their optimization may specialize to the source-domain supervision and input format. The latent-memory baselines perform poorly on LongMemEval, though not uniformly on PrefEval, where MemGen remains competitive. In contrast, **`MemFlow`** maintains more robust performance across the personalization and long-context OOD settings, indicating that the learned soft-memory interface remains transferable under dataset-distribution shifts. The transfer is not uniform: on PrefEval with Qwen2.5-7B-Instruct, **`MemFlow`** only ties the strongest baselines, and the methods occupy a narrow accuracy range, making this setting less discriminative than the others.

Does the learned memory interface also improve training efficiency?

Figure 1 compares training with Qwen2.5-3B-Instruct on PersonaMem-32K. **`MemFlow`** improves accuracy more rapidly with optimizer updates and reaches comparable accuracy with fewer student rollouts. Its reward and distillation objectives reuse the same responses, while the teacher scores sampled tokens without autoregressive generation. These curves compare complete method configurations, including different inputs and initializations (Appendix B.1). The observed advantage therefore reflects the full training recipe. The curves measure optimization and rollout efficiency; total training cost also includes initialization and teacher forward passes.

How much does the outcome depend on the particular budget K we chose, in accuracy and in what it costs to run?

Sensitivity of accuracy to the budget. Increasing K eightfold from 64 to 512 does not produce monotonic gains (Figure 3). Both curves are single-peaked at $K = 256$ and nearly flat from 128 upward, so only the smallest budget is clearly under-provisioned and the useful range is broad rather than a sharp optimum. The longer histories give a flatter curve, as expected if the writer is already discarding most of the history before compression, so that adding vectors changes what survives less than it changes how much. The decline at 512 is the more informative end. Extra capacity is not free here, and because the compressor and reader are initialized under a fixed budget that we did not retune per K, we read that decline as a property of this training recipe rather than as evidence that more vectors carry less.

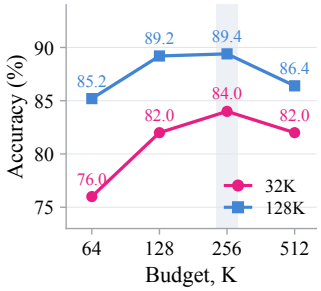


Figure 3: Accuracy across budgets and history lengths (Qwen3-4B).

Cost of the budget. End-to-end token cost barely moves across budgets (Figure 4), varying by less than one percent on the longer benchmark. The change is not monotonic in K: increasing the budget from 64 to 512 adds 448 vectors to the reader's input, roughly a third of the token difference measured on PersonaMem-32K. Most of the variation therefore comes from generated tokens. Every configuration still reads the interaction history once to build the textual memory; on the longer benchmark, that pass accounts for all but a fraction of a percent of the per-instance cost. Enlarging the budget therefore costs little, and shrinking it saves little. A compact memory keeps the reader's interface independent of history length, but does not remove the cost of reading the full interaction history for each instance.

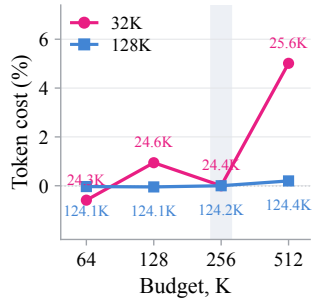


Figure 4: Token cost relative to $K = 256$ (Qwen3-4B); labels show absolute counts.

Is the reader actually using the memory built for this instance, or answering from what the task alone makes likely?

Replacing the matched soft memory with shuffled or null memory causes substantial accuracy drops on both datasets (Figure 5). The model therefore depends on the content of the memory built for this instance, rather than ignoring it or answering from task priors alone. The intervention establishes that the memory is used; it does not establish which parts of it are used, and in particular it does not show that the reader resolves the correct time-dependent version of a preference when the history contains several. The textual-memory teacher is not an accuracy oracle: its task accuracy is well below the final student's. This is consistent with how $\mathcal{L}_{\text{OPD}}$ uses it (Section 3.3). The gate weights only tokens the student has already sampled, and its influence on any single token is bounded, so the teacher can reinforce tokens the student under-reads from its memory without its lower accuracy dominating the update. GRPO drives most of the task-level gain (Section 5).

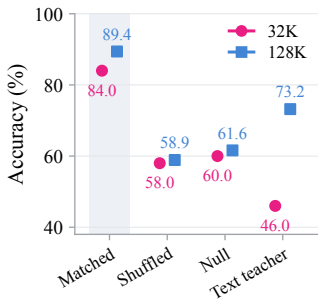


Figure 5: Accuracy across memory conditions and the text teacher (Qwen3-4B).

What does a personalization failure actually look like in a response, when the aggregate scores only say that one was produced?

Aggregate scores do not reveal localized personalization failures. Figure 6 shows one such case: although the GRPO and OPSD responses are generally relevant, both recommend wearable trackers that contradict the user's preference. In contrast, **MemFlow** provides preference-consistent alternatives, including journaling, body measurements, and progress photos. This example illustrates a failure pattern rather than its frequency.

User preference: *"I dislike using wearable technology and fitness trackers."*
Query: *"Can you recommend some effective ways for me to monitor my fitness progress?"*

(a) GRPO	(b) OPSD	(c) MemFlow
Incorrect: violates preference	Incorrect: violates preference	Correct: respects preference
300 answer tokens	300 answer tokens	96 answer tokens
… Keep a record of your weight, body measurements, heart rate, and workout performance. **Use a Fitness Tracker:** Wearables like Fitbit or Apple Watch can monitor steps, calories burned, and sleep quality, providing real-time feedback. … response continues until the generation limit.	… Track your workouts, measure body metrics, take photos, and set SMART goals. **Use Wearable Technology:** Devices like smartwatches or fitness trackers can monitor heart rate, steps, calories burned, and sleep quality. … response continues until the generation limit.	Track your workouts with a fitness app or **journal**, and record your strength and endurance gains. Take regular **body measurements** and use a **progress photo log** to monitor changes in body composition. Set specific, measurable goals, stay consistent, and adjust your plan as needed.

Figure 6: A PrefEval implicit-persona example with Qwen3-4B. The displayed preference summarizes the interaction history and is not given explicitly to the models. GRPO and OPSD recommend wearable trackers, whereas **MemFlow** respects the preference by suggesting non-wearable alternatives. Colored spans mark preference-relevant content. Token counts cover generated answers only; this example illustrates a failure mode, not its frequency or end-to-end efficiency.

7 Conclusion

We presented **MemFlow**, which judges a compact personalized memory by the generations it supports rather than by the text it reconstructs. A query-conditioned textual memory is compressed into K soft vectors forming a fixed-budget reader interface, and the reader is optimized on its own rollouts under two complementary signals: group-relative rewards for task outcomes, and confidence-gated on-policy distillation in which a frozen textual-memory teacher re-scores the student's sampled tokens. Across three backbones it achieves the highest accuracy we measured on PersonaMem-32K and PersonaMem-128K, transfers to PrefEval and LongMemEval without target-domain training, and still depends on instance-specific memory under shuffled- and null-memory interventions.

AI USE STATEMENT

We used generative AI tools to polish the writing of this manuscript, including improving grammar, clarity, and readability. The authors take full responsibility for the final content of the paper, including all claims, analyses, and results.

ETHICS STATEMENT

To the best of our knowledge, this work complies with the ICLR Code of Ethics. Our experiments use existing publicly available benchmarks and do not involve new data collection from human participants.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide detailed descriptions of the model architecture, training objectives, optimization procedure, hyperparameters, and evaluation protocols in the appendix.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 23716–23736, 2022.

Wen-Yu Chang and Yun-Nung Chen. When users don't ask: Benchmarking context-driven memory retrieval in conversational agents. *arXiv preprint arXiv:2609.03467*, 2026.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xrag: Extreme context compression for retrieval-augmented generation with one token. *Advances in Neural Information Processing Systems*, 37:109487–109516, 2024.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846, 2023.

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.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502, 2026.

Tao Feng, Chongrui Ye, Tianyang Luo, Jingjun Xu, Xueqiang Xu, Haozhen Zhang, Ge Liu, and Jiaxuan You. Elasticmem: Latent memory as a learnable resource for llm agents. *arXiv preprint arXiv:2605.30690*, 2026.

Muxin Fu, Xiangyuan Xue, Yafu Li, Zefeng He, Siyuan Huang, Xiaoye Qu, Yu Cheng, and Yang Yang. Latentmem: Customizing latent memory for multi-agent systems. *arXiv preprint arXiv:2602.03036*, 2026.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *International Conference on Learning Representations*, 2024.

Google DeepMind. Gemini 3.8 flash: Our most intelligent flash model for long-horizon engineering and agents, 2026. URL `https://ai.google.dev/gemini-api/docs/generate-content/latest-model`. Accessed: 2026-09-23.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Qianyun Guo, Yibo Li, Yue Liu, and Bryan Hooi. Towards natural personalization: Evaluating long-horizon preference following in personalized user-llm interactions. *arXiv preprint arXiv:2603.04191*, 2026.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Yeonjun In, Wonjoong Kim, Sangwu Park, Kanghoon Yoon, and Chanyoung Park. Personalize-then-store: Benchmarking and learning personalized memory for long-horizon agents. *arXiv preprint arXiv:2605.25535*, 2026.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*, 2022.

Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J Taylor, and Dan Roth. Know me, respond to me: Benchmarking llms for dynamic user profiling and personalized responses at scale. *arXiv preprint arXiv:2504.14225*, 2025.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376, 2023.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, 2021.

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.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pp. 4582–4597, 2021.

Zeju Li, Yizhou Zhou, and Qiang Xu. Latent context compilation: Distilling long context into compact portable memory. *arXiv preprint arXiv:2602.21221*, 2026.

Zongqian Li, Yinhong Liu, Yixuan Su, and Nigel Collier. Prompt compression for large language models: A survey. *arXiv preprint arXiv:2410.12388*, 2024.

Yuxin Liao, Le Wu, Min Hou, Hao Liu, Han Wu, and Zishu Wang. Leanmem: Simple and efficient long-term memory for llm agents. *arXiv preprint arXiv:2608.03463*, 2026.

YuFei Luo, Xiucheng Xu, and Zhen Yang. Memsif: From structured interactions to dual-track fact memory for llm agents. *arXiv preprint arXiv:2608.01742*, 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.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems*, volume 36, pp. 19327–19352, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 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.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

Md Mostafizer Rahman, Md Faizul Ibne Amin, Md Shahajada Mia, Yutaka Watanobe, and Fang Liu. Compressing long context into answer-aligned memory embeddings for llm inference. *arXiv preprint arXiv:2609.25537*, 2026.

Alireza Salemi and Hamed Zamani. Lamp-qa: A benchmark for personalized long-form question answering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1139–1159, 2025.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. Lamp: When large language models meet personalization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7370–7392, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

Yuanyi Song, Yukai Wang, Xinbei Ma, Zhihui Fu, Jianghao Lin, Weiwen Liu, Jun Wang, Huarong Deng, Yong Yu, and Weinan Zhang. Retrieval-driven memory reconsolidation for long-term llm agents. *arXiv preprint arXiv:2609.16053*, 2026.

Md Nayem Uddin, Kumar Shubham, Eduardo Blanco, Chitta Baral, and Gengyu Wang. From recall to forgetting: Benchmarking long-term memory for personalized agents. *arXiv preprint arXiv:2604.20006*, 2026.

Ben Wang, Kang Zhou, Lifan Guo, Feng Chen, and Chi Zhang. Finperma: A theory-informed, event-grounded personalized-memory benchmark for llm agents. *arXiv preprint arXiv:2608.04095*, 2026.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-α: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*, 2025.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 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.

Bowen Ye, Yongchao Xu, Zhijian Li, Xiang Yin, Junkai Ma, and Wenzhao Li. Coevo-mem: Co-evolving retrieval policy and memory bank for llm agents. *arXiv preprint arXiv:2608.01739*, 2026.

Jing Yu, Yibo Zhao, Jiaming Zhang, and Xiang Li. Lazymem: Retrieve broadly, construct selectively for efficient long-term agent memory. *arXiv preprint arXiv:2607.22690*, 2026a.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. *arXiv preprint arXiv:2601.01885*, 2026b.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents. In *International Conference on Learning Representations*, volume 2026, pp. 22555–22588, 2026a.

Weizhi Zhang, Xiaokai Wei, Wei-Chieh Huang, Zheng Hui, Chen Wang, Michelle Gong, and Philip S. Yu. Memorycd: Benchmarking long-context user memory of llm agents for lifelong cross-domain personalization. *arXiv preprint arXiv:2603.25973*, 2026b.

Zeyu Zhang, Rui Li, Xiaoyan Zhao, Yang Zhang, Wenjie Wang, Xu Chen, and Tat-Seng Chua. Nextmem: Towards latent factual memory for llm-based agents. *arXiv preprint arXiv:2603.15634*, 2026c.

Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. Do llms recognize your preferences? evaluating personalized preference following in llms, 2025. URL `https://arxiv.org/abs/2502.09597`.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

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

Zhengze Zhou and Hejian Sang. Latentpress: Context compression beyond text and vision. *arXiv preprint arXiv:2609.01507*, 2026.

A Implementation Details

A.1 MemFlow

A.1.1 Textual Memory Construction

Memory format. For each context–query pair, the external extractor produces a structured memory M consisting of three fields: grounded evidence, temporal relations, and derived facts. Memories are extracted by GLM-5.2 (Zeng et al., 2026) from the history and the question only; the reference answer is never provided to the extractor. We retain only annotations that are grounded in the interaction history and relevant to the corresponding query. The adapter is trained on the resulting context–query–memory examples and subsequently writes M without the external extractor; we denote the adapter after this stage by θ_w.

Example. Figure 7 presents a representative memory template, illustrating how evidence, temporal relations, and derived facts are organized within the structured textual memory.

A Representative Extracted Memory

Evidence

- The user organized a volunteering day at a local shelter for the whole family to help the shelter and show the kids the importance of supporting those in need.
- The user later decided against future family volunteering after observing that the children felt resentful, and instead chose to prioritize activities aligned with their own interests.
- The user started a family photo challenge with weekly themes to encourage creativity and discussion among family members.
- The user enjoys writing a personal finance blog and views writing as a more suitable way to influence and support others.

Temporal Relations

- The user first encouraged structured family volunteering, but later shifted toward child-led activities after observing reduced engagement.
- The user initially participated in a financial literacy volunteer program, but later concluded that blogging better matched their preferred way of helping others.

Derived Facts

- The user currently prioritizes child-led and interest-based activities for maintaining positive family dynamics.
- The user prefers teaching and influencing others through writing and creative projects rather than formal programs.

Figure 7: A representative structured textual memory extracted from a long PersonaMem interaction history. For readability, we show a representative subset of the extracted entries.

A.1.2 Soft-Memory Construction and Initialization

Compressor architecture. We serialize the fields of textual memory M in a fixed order and encode the resulting sequence with a frozen backbone encoder. The sequence is processed in chunks of at most 2,048 tokens, with every 32 consecutive hidden states mean-pooled to form the compressor input H. A two-layer Perceiver-style compressor with K learned queries, a latent dimension of 768, and 12 attention heads aggregates these representations. A backbone-specific projector maps the compressed states into the reader's input embedding space:

$$\mathcal{C}_\phi(H) = P_\phi\left(\mathrm{Comp}_\phi(Q_K, H)\right) \in \mathbb{R}^{K\times d},$$

where $\mathcal{C}_\phi$ denotes the composition of the compressor and projector. Unless otherwise specified, we use $K = 256$ and train a separate compressor for each dataset–backbone configuration.

Hidden-state layer selection. We select the encoder layer using a frozen, probe-free retrieval experiment over 3,347 memory records from 272 sessions. Candidate layers are evaluated on their

Table 3: Hidden-state layer selection. Dev. Score is the mean development AUC over value and update discrimination; the remaining metrics are measured on the held-out update task. Bold indicates the selected layer.

Backbone	Layer	Dev. Score	Update Top-1	Update AUC	Margin
	L1	98.98	96.12	97.55	0.207
	L4	**100.00**	**100.00**	**100.00**	**0.319**
Qwen2.5-3B	L12	94.39	79.61	90.37	0.091
	L24	86.13	67.96	79.04	0.042
	L36 (final)	77.49	37.86	51.76	−0.041
	L1	95.92	93.20	96.35	0.148
	L4	**98.98**	**98.06**	**99.19**	**0.298**
Qwen3-4B	L12	96.94	87.38	90.27	0.151
	L24	84.85	60.19	72.15	0.038
	L36 (final)	77.29	41.75	55.48	−0.021
	L1	97.62	96.12	97.78	0.266
	L4	**98.98**	**100.00**	**100.00**	**0.321**
Qwen2.5-7B	L12	87.96	69.90	80.04	0.078
	L24	76.95	33.98	48.29	−0.037
	L28 (final)	73.69	33.01	47.60	−0.024

ability to distinguish competing attribute values and temporal updates. Selection uses the mean development AUC of the two tasks, while the held-out split is used only for reporting. As shown in Table 3, the fourth Transformer block achieves the highest development score for all three backbones and consistently performs best on held-out update discrimination. We therefore use its output as the compressor input.

Initialization overview. The soft-memory interface is initialized through four sequential procedures: compressor reconstruction, representation warmup, auxiliary reasoning adaptation, and reader initialization. These procedures optimize distinct objectives rather than a single combined initialization objective. Their optimization settings and trainable modules are summarized in Table 5.

For example i, let c_i denote its shared-context identifier, h_i its permitted history prefix, q_i its question, o_i its answer options, m_i its textual memory, s_i its auxiliary reasoning target, and a_i its answer target. During compressor reconstruction and representation warmup, the soft representation is computed from cached history-prefix states:

$$Z_i = \mathcal{C}_\phi(E(h_i)).$$

During auxiliary reasoning adaptation and reader initialization, it is computed from textual-memory states:

$$Z_i = \mathcal{C}_\phi(E(m_i)),$$

where E is the frozen backbone encoder. Cached encoder states are treated as constants, and gradients propagate through $\mathcal{C}_\phi$.

We obtain a context-level unit representation by averaging over the K soft positions and applying ℓ_2 normalization:

$$u_i = \mathrm{norm}_2\left(\frac{1}{K}\sum_{k=1}^{K} Z_{ik}\right).$$

For examples i and j from different shared contexts, the separation loss is

$$\mathcal{L}_{\mathrm{sep}}(i,j) = \left[\sqrt{2-2\kappa} - \|u_i - u_j\|_2\right]_+, \qquad \kappa = 0.8.$$

All cross-entropy terms below are averaged over supervised target positions; soft prefixes, prompts, and padding positions are excluded.

Compressor reconstruction. The frozen decoder reconstructs the textual memory from the compressed history representation:

$$\mathcal{L}_{\text{rec}}(i) = -\frac{1}{|m_i|}\sum_{t=1}^{|m_i|}\log p_{\theta_0}\left(m_{it} \mid Z_i, q_i, m_{i,<t}\right).$$

The complete reconstruction objective is

$$\boxed{\mathcal{L}_{\text{reconstruction}} = \mathbb{E}_i\left[\mathcal{L}_{\text{rec}}(i) + \mathcal{L}_{\text{sep}}(i,j)\right],}$$

where j is sampled from a different shared context. The encoder and decoder remain frozen, while the full compressor and projector are updated. No answer or reasoning supervision is used in this procedure.

Representation warmup. Representation warmup operates on two cached history-state views per context and uses no decoder or answer supervision. It combines separation across different contexts, alignment between views of the same context, and decorrelation among context prototypes:

$$\boxed{\mathcal{L}_{\text{warmup}} = \mathcal{L}_{\text{sep}}^{\text{all}} + 0.1\,\mathcal{L}_{\text{align}} + 0.1\,\mathcal{L}_{\text{Gram}}.}$$

Here, $\mathcal{L}_{\text{sep}}^{\text{all}}$ averages the separation hinge over different-context pairs, $\mathcal{L}_{\text{align}}$ is the mean cosine distance between same-context views, and $\mathcal{L}_{\text{Gram}}$ penalizes squared off-diagonal similarities between normalized context prototypes. The entire compressor and projector are updated.

Auxiliary reasoning adaptation. The third procedure adapts the compressor using textual-memory states and evidence-grounded reasoning targets. Let

$$\ell_i(Z) = -\frac{1}{|s_i|}\sum_{t=1}^{|s_i|}\log p_{\theta_0,\omega}(s_{i,t} \mid Z, q_i, o_i, s_{i,<t})$$

denote the reasoning-target negative log-likelihood. For a mismatched memory from a different context, we define

$$\mathcal{L}_{\text{rank}}(i,j) = \left[\delta + \ell_i(Z_i) - \ell_i(Z_j)\right]_+.$$

The question, options, and reasoning target remain fixed, so only the soft memory is replaced in the negative example. The complete objective is

$$\boxed{\mathcal{L}_{\text{aux}} = \mathbb{E}_i\left[\ell_i(Z_i) + \lambda_{\text{rank}}\mathcal{L}_{\text{rank}}(i,j) + 0.1\,\mathcal{L}_{\text{sep}}(i,j)\right].}$$

We use $(\lambda_{\text{rank}}, \delta) = (0.2, 0.05)$ in the first epoch and $(1.0, 0.1)$ in the remaining epochs. This procedure updates the full compressor and projector together with a temporary rank-8 LoRA ω on the frozen backbone. ω is discarded afterward; only the adapted compressor and projector are transferred to reader initialization.

Reader initialization. The final initialization procedure trains the reader to generate gold answers from self-generated textual memories. Specifically, the writer-initialized adapter θ_w generates and caches $M_i^w = e_{\theta_w}(C_i, q_i)$.

The adapter continues training from θ_w. Let ϕ_0 denote the compressor parameters at the beginning of reader initialization, and retain a frozen reference copy. The trainable and reference soft representations are

$$Z_i = C_\phi(E(M_i^w)), \qquad Z_i^0 = \text{sg}[C_{\phi_0}(E(M_i^w))].$$

The answer objective is

$$\mathcal{L}_{\text{answer}}(i) = -\frac{1}{|a_i|}\sum_{t=1}^{|a_i|}\log \pi_\theta(a_{i,t} \mid Z_i, q_i, o_i, a_{i,<t}).$$

To limit drift in the soft-memory representation, we use the normalized anchoring loss

$$\mathcal{L}_{\text{anchor}}(i) = \frac{\frac{1}{Kd}\,\|Z_i - Z_i^0\|_F^2}{\max\!\left(\frac{1}{Kd}\,\|Z_i^0\|_F^2,\ 10^{-8}\right)}.$$

The complete reader-initialization objective is

$$\mathcal{L}_{\text{reader-init}} = \mathbb{E}_i[\mathcal{L}_{\text{answer}}(i) + 0.1\,\mathcal{L}_{\text{anchor}}(i)]\,.$$

Ranking, auxiliary reasoning supervision, and writer replay are disabled in this procedure. We update the reader LoRA θ, the final resampler layer, the resampler's output normalization layer, and the projector, while keeping the backbone θ_0, the remaining compressor parameters, and the reference compressor frozen. The resulting adapter, θ_{init}, initializes on-policy optimization and, when kept frozen, serves as the textual-memory teacher π_T. Before on-policy optimization begins, it is also used once to generate a new fixed training-memory snapshot, $M_i^{\text{init}} = e_{\theta_{\text{init}}}(C_i, q_i)$,

which is cached and held fixed throughout on-policy training.

A.1.3 On-Policy Training Details

Memory snapshots. For reader initialization, training memories are generated once by the writer-initialized adapter θ_w using greedy decoding and cached. After selecting the reader-initialization checkpoint θ_{init}, we regenerate the training memories once with θ_{init}:

$$M^{\text{init}} = e_{\theta_{\text{init}}}(C, x).$$

These memories and their encoder representations are cached and remain fixed throughout on-policy optimization; they are not refreshed after policy updates. The teacher reads the cached textual memory M^{init}, while the student receives $Z = C_\phi(E(M^{\text{init}}))$ compressed from the same memory. At evaluation, memories are generated by the final adapter θ, so that a single adapter both writes and reads at test time.

OPD implementation. In code, the OPD term is computed as $\bar{g}_{i,t}\,(\ell_{T,i,t} - \ell_{\theta,i,t})$, with both $\ell_{T,i,t}$ and $\bar{g}_{i,t}$ detached. Because $\ell_{T,i,t}$ carries no gradient, this differs from Eq. 4 only by a constant and yields the same gradient, $-\bar{g}_{i,t}\nabla_\theta \ell_{\theta,i,t}$; only the logged loss value differs.

Rewards and advantages. For PersonaMem, $r(y, a) = 1$ if the parsed option of y matches the gold answer a, and 0 otherwise. For LoCoMo, we case-fold the generated and reference answers and extract word units using `[\w]+`, yielding sequences S_y and S_a. Let

$$O = \sum_w \min\big(\text{count}_{S_y}(w),\ \text{count}_{S_a}(w)\big).$$

The LoCoMo reward is

$$r(y, a) = 0.75\,\frac{2O}{|S_y| + |S_a|} + 0.25\mathbb{1}\{S_y = S_a\},$$

and is set to zero if either sequence is empty. No additional format penalty is applied. Group-relative advantages $\hat{A}_i$ follow Section 3.3; a group whose reward variance is at most 10^{-6} is treated as having identical rewards, so all its $\hat{A}_i$ are set to 0. Such a group contributes no GRPO gradient, while the other enabled objectives remain active.

OPD token mask. OPD is computed only over student-generated response positions. The mask excludes the prompt and soft-memory prefix, includes the first EOS token, and excludes padding after EOS; if no EOS is generated, the full response is used.

A.2 Training Configuration

Table 4 summarizes the shared architectural and algorithmic hyperparameters used by **MemFlow**. Unless otherwise specified, we use a fixed soft-memory budget of 256 tokens and extract textual-memory representations from the fourth Transformer block.

Table 4: Shared architecture and optimization hyperparameters used by MemFlow.

Hyperparameter	Setting
Memory representation and compressor	
Soft-memory budget K	256 soft tokens
Mean-pooling window	32 hidden states
Compressor depth	2 layers
Latent dimension	768
Attention heads	12
Training and optimization	
LoRA rank / scaling / dropout	16 / 32 / 0.05
OPD gate scale β	5.0
GRPO clip range ϵ	0.2

Table 5: Training configuration for PersonaMem-32K.

Procedure	Learning Rate	Duration	Updated Modules
Memory-interface initialization			
Memory-writer initialization	1×10^{-5}	3 epochs	Adapter
Compressor reconstruction	3×10^{-4}	25 updates	Compressor, projector
Representation warmup	1×10^{-4}	100 updates	Compressor, projector
Auxiliary reasoning adaptation	$1 \times 10^{-4} \rightarrow$ $5 \times 10^{-5}/3 \times 10^{-5}$	3 epochs	Temporary LoRA ω, compressor, projector
Reader initialization	1×10^{-5}	3 epochs	Adapter, selected compressor modules, projector
On-policy optimization			
Joint OPD–GRPO optimization	3×10^{-7}	3 epochs	Adapter

Table 5 reports the procedure-specific optimization configuration for PersonaMem-32K. All procedures use AdamW with weight decay 0.01 and global gradient clipping at 1.0.

Memory-writer initialization uses cosine learning-rate decay with a 5% warmup, whereas all subsequent procedures use constant learning rates without warmup. Representation warmup combines cross-context separation, alignment, and Gram-matrix regularization, each with a weight of 0.1. During auxiliary reasoning adaptation, the ranking weight increases from 0.2 in the first epoch to 1.0 in the remaining epochs, while the separation weight remains 0.1. The corresponding ranking margins are 0.05 and 0.1, respectively. Reader initialization additionally applies a soft-output anchoring loss with weight 0.1.

Memory-writer initialization, reader initialization, and on-policy optimization train a single LoRA adapter on the query, key, value, output, gate, up, and down projections. Memory-writer initialization updates only this adapter, yielding θ_w. Reader initialization continues training the same adapter together with the final resampler layer, resampler output normalization, and projector, while the backbone and remaining compressor parameters are frozen. The temporary rank-8 LoRA ω used for auxiliary reasoning adaptation is discarded before reader initialization. During on-policy optimization, only the adapter is updated; the backbone, soft-memory compressor, projector, and textual-memory teacher remain frozen.

For PersonaMem, on-policy rollouts use eight responses per query, a temperature of 1.0, top-p of 0.98, and a maximum response length of five tokens. We set $\lambda_{\mathrm{OPD}} = 0.02$ and $\lambda_{\mathrm{GRPO}} = 0.3$. For LoCoMo, we use four responses per query, a temperature of 0.8, top-p of 0.95, and a maximum response length of 64 tokens. Both OPD and GRPO are assigned a coefficient of 1.0. We use no additional reference-policy or KL regularization.

Frozen-backbone computation uses BF16, whereas LoRA parameters and numerically sensitive log-probability and loss computations use FP32. Cached encoder representations are stored in FP16 and converted to BF16 when loaded. Memory-writer initialization uses FlashAttention-2, while reader initialization and on-policy optimization use SDPA. These procedures are trained with dis-

tributed data parallelism on four GPUs, whereas compressor reconstruction, representation warmup, and auxiliary reasoning adaptation use a single GPU. All experiments were conducted on NVIDIA H200 GPUs. In the PersonaMem-32K/Qwen2.5-3B efficiency experiment, on-policy optimization completed 369 updates on four GPUs in approximately 12.9 minutes of training wall-clock time, excluding evaluation and the preceding initialization procedures.

A.3 Datasets & Benchmarks

PersonaMem-32K and PersonaMem-128K. PersonaMem (Jiang et al., 2025) evaluates whether models can track evolving user preferences and answer personalized multiple-choice questions based on long conversational histories. The 32K and 128K variants differ in context length, and our evaluation uses 50 and 233 test questions, respectively.

PrefEval. PrefEval (Zhao et al., 2025) evaluates whether models can infer and apply implicit user preferences from interaction histories; we use its 1,000-example implicit-persona subset to assess cross-dataset personalization generalization. We evaluate the PersonaMem-32K-trained checkpoints and score their predictions using the Gemini API.

LongMemEval. LongMemEval (Wu et al., 2024) evaluates long-term interactive memory across tasks such as knowledge updates, multi-session reasoning, temporal reasoning, and user-preference recall. We use the cleaned 500-question LongMemEval-S split as a held-out benchmark and evaluate answers with `gemini-3.8-flash` using LOW thinking and structured yes/no outputs.

A.4 Baseline Implementation Details

All baselines use the same Qwen backbone, source-domain data splits, visible-history boundary, and evaluation protocol within each experimental setting. Hyperparameters and checkpoints are selected only on the source-domain validation set, and all parameters remain frozen during OOD evaluation. Because the original implementations of xRAG, MemGen, and AutoCompressor do not directly support our Qwen backbones and datasets, we describe them as adapted implementations rather than exact reproductions of official checkpoints.

xRAG. Our xRAG implementation preserves the original frozen-retriever, frozen-language-model, and projector-only training design (Cheng et al., 2024). A frozen GTE-large encoder retrieves individual messages from the visible history, and a two-layer MLP with a GELU activation maps each retrieved vector to one soft token in the Qwen embedding space. We first train the projector to reconstruct source-domain context messages and then fine-tune it on source-domain QA while keeping both the retriever and Qwen backbone frozen. The retrieval query, top-k, and learning rate are selected exclusively using source-domain validation data.

MemGen. Our MemGen implementation uses the official Weaver core (Zhang et al., 2026a) with an interface adapted to Qwen and our data formats. The Weaver generates latent memories that are injected into the reasoner, using eight prompt latents, eight inference latents, and at most five inference-time augmentations. The Weaver, projection parameters, latent parameters, and their associated rank-16 LoRA modules are trained on the source-domain split. Histories exceeding the Weaver's input limit are split into chunks of 28,672 tokens so that every visible message is processed; no history is truncated. The reported PersonaMem and PrefEval checkpoints use the trained Weaver latent memory with the Trigger disabled, and therefore do not reflect MemGen's full Trigger-based closed-loop procedure.

AutoCompressor. We adapt the recurrent summary-token mechanism of AutoCompressor (Chevalier et al., 2023) to the Qwen backbones. Each visible history is divided into 1,536-token segments, with every segment producing 32 learned summary vectors that are carried forward to compress subsequent segments. The final reader receives only the accumulated summaries, system instruction, and question rather than the compressed raw segments. We jointly train the summary embeddings and rank-16 Qwen LoRA modules for one source-domain epoch, using a fixed seed of 42 and without selecting the training duration based on test performance.

OPSD. We implement OPSD (Zhao et al., 2026) using a fixed version of its official trainer as a full-history QA baseline. The student receives the complete history visible before the query time τ together with the question, while the frozen teacher may additionally access the reference answer

for the same source-domain training example, as specified by the method. Training uses generalized Jensen–Shannon divergence with point-wise clipping of 0.05 and no task reward. The reader is trained with rank-64 LoRA, a learning rate of 5×10^{-6}, and temperature 1.1 for 500 steps; reference answers are never available during validation, OOD evaluation, or test-time inference.

Vanilla GRPO. Vanilla GRPO (Shao et al., 2024) is implemented with TRL as a full-history QA baseline without teacher-distribution supervision. For each source-domain question, the policy samples eight completions and computes advantages using group-normalized task rewards without a separate value model. PersonaMem uses strict multiple-choice exact match as the reward, whereas open-ended source-domain QA uses normalized token F1. We set the KL coefficient to zero, use rank-64 LoRA with a learning rate of 5×10^{-6}, and train for 1000 steps at temperature 1.2; evaluation uses the same deterministic decoding protocol as the other methods.

B Evaluation Details

B.1 Efficiency Comparison Setup

Figure 1 compares the training dynamics of five complete method configurations using Qwen2.5-3B-Instruct on PersonaMem-32K. The figure is intended as a system-level comparison of optimization and rollout efficiency, rather than a controlled ablation in which only the loss function changes. SDPO (Hübotter et al., 2026) and GRPO+OPD appear only in this comparison; the latter applies our OPD objective to a full-history student without soft memory.

Table 6: Configurations compared in Figure 1.

Method	Student Input	Initialization	Supervision	Loss
GRPO	Full interaction history	Base model	Task reward	$\mathcal{L}_{\mathrm{GRPO}}$
OPSD	Full interaction history	Base model	On-policy self-distillation	$\mathcal{L}_{\mathrm{OPSD}}$
SDPO	Full interaction history	Base model	Successful-rollout feedback; EMA teacher	$\sum_t D_{\mathrm{KL}}(p_t \| q_t)$
GRPO+OPD	Full interaction history	Base model	Task reward; textual-memory teacher	$\lambda_{\mathrm{GRPO}}\mathcal{L}_{\mathrm{GRPO}} + \lambda_{\mathrm{OPD}}\mathcal{L}_{\mathrm{OPD}}$
MemFlow	$K = 256$ soft-memory vectors	Reader-initialization checkpoint	Task reward; textual-memory teacher	$\lambda_{\mathrm{GRPO}}\mathcal{L}_{\mathrm{GRPO}} + \lambda_{\mathrm{OPD}}\mathcal{L}_{\mathrm{OPD}}$

The GRPO and OPD losses are defined in Eqs. 2 and 4; for GRPO+OPD, the student conditions on the full history in place of Z. $\mathcal{L}_{\mathrm{OPSD}}$ denotes the generalized Jensen–Shannon objective with point-wise clipping described in Appendix A.4. For SDPO, the table shows the defining reverse-KL objective in Hübotter et al. (2026, Eq. (1)), with $p_t = \pi_\theta(\cdot \mid C, x, y_{<t})$ and $q_t = \mathrm{sg}[\pi_{\mathrm{EMA}}(\cdot \mid C, x, f, y_{<t})]$. Here f is successful-rollout feedback, and the teacher scores the student's sampled prefixes with gradients stopped. The corresponding distillation advantage for a candidate token v is $A_t^{\mathrm{SDPO}}(v) = \log q_t(v) - \log p_t(v)$.

Training-step and rollout accounting. A gradient update step denotes one optimizer update after gradient accumulation, rather than one microbatch. Cumulative student rollouts count the number of student responses generated up to each checkpoint. Repeated sampling of the same question contributes a new rollout, whereas teacher forward passes do not. Groups whose rewards are all identical still count toward cumulative rollouts, although they contribute no GRPO gradient. At the final checkpoint of 369 optimizer updates (16 rollouts per update), GRPO, GRPO+OPD, SDPO, and MemFlow each produce 5,904 student rollouts, while OPSD produces 5,860 because its final batch is padded to full size and the padded entries are not counted as rollouts.

Evaluation and plotting. Each method is trained for a fixed 369 optimizer updates and evaluated at steps 37, 74, 111, 148, 185, 222, 259, 296, 333, and 369. Training is not stopped or checkpoint-selected using test performance. Every checkpoint is evaluated on the same 50 PersonaMem-32K test questions using greedy decoding under the same fixed set of 12 deterministic answer-option permutations. The reported curves come from one training run per method; the permutations are repeated evaluations of the same checkpoint rather than independent training seeds. The final curves directly connect the measured checkpoints without smoothing. Where confidence intervals are shown, they are obtained by bootstrapping test questions and therefore do not represent variation across training seeds. For the training rollouts in Figure 1, all five configurations use temperature 1.0 and top-$p = 1.0$.

B.2 End-to-End Token-Equivalent Accounting

Let P and A denote the textual reader prompt and generated answer, respectively, measured using each method's actual prompt and tokenizer. We count every discrete token processed or generated at each inference stage and treat each soft, summary, or latent position as one token equivalent. Repeated processing by different components is counted separately. We exclude padding, training-only annotation and teacher computation, and evaluation-judge tokens; cached computation and memory-construction costs are not amortized across questions or trials. Reported values are averaged over questions within each trial and then over five trials. This metric measures logical token-equivalent processing, not FLOPs or wall-clock latency. Table 7 summarizes the components counted for each method.

Table 7: End-to-end token-equivalent accounting by method.

Method	#Tok.	Accounting Note
Full Text / GRPO / OPSD	$P + A$	P contains the full visible history, query, and instructions.
xRAG	$T_{\text{retriever}} + P + K_{\text{xRAG}} + A$	Counts retriever inputs and projected retrieval positions consumed by the reader.
AutoCompressor	$\sum_s T_{\text{AC}}^{(s)} + P + K_{\text{AC}} + A$	Each compression pass includes processed text and carried summary positions.
MemGen	$T_{\text{Weaver}} + K_{\text{Weaver}} + T_{\text{reasoner}} + K_{\text{reasoner}} + A$	Weaver and reasoner computation are counted separately; repeated history reads are retained.
`MemFlow`	$T_{\text{writer-in}} + T_{\text{memory-out}} + T_{\text{compressor-in}} + P + K + A$	Counts memory generation and subsequent compressor encoding as separate operations.

C Theoretical Analysis of $\mathcal{L}_{\text{OPD}}$

We analyze the sampled-token surrogate introduced in Section 3.3. Let $\Delta_{i,t} = \ell_{T,i,t} - \ell_{\theta,i,t}$ and $\bar{g}_{i,t} = \text{sg}[\sigma(\beta\Delta_{i,t})]$, where $\beta > 0$. During differentiation, sampled trajectories and their masks are held fixed. We distinguish the gradient computed on a sampled batch from its idealized conditional expectation, and state explicitly which properties hold for each.

C.1 Gradient of the Sampled Surrogate

Proposition 1 (Gradient of the sampled surrogate)**.** *Let $\widehat{\mathbb{E}}_{\mathcal{B}}$ denote the masked token average over a fixed rollout batch $\mathcal{B}$. The implemented surrogate is*

$$\widehat{\mathcal{L}}_{\text{OPD}} = \widehat{\mathbb{E}}_{\mathcal{B}}\left[\bar{g}_{i,t}\left(\text{sg}[\ell_{T,i,t}] - \ell_{\theta,i,t}\right)\right], \tag{5}$$

and its gradient is

$$\nabla_\theta \widehat{\mathcal{L}}_{\text{OPD}} = -\widehat{\mathbb{E}}_{\mathcal{B}}\left[\bar{g}_{i,t}\,\nabla_\theta \ell_{\theta,i,t}\right]. \tag{6}$$

Proof. The sampled tokens, masks, gate, and teacher log-probabilities are constant in the backward pass. Differentiating each summand therefore leaves only the student log-probability term. □

The update is thus gradient-equivalent to gate-weighted negative log-likelihood on student-generated tokens, with no gradient through the teacher or the gate. This is a surrogate gradient with fixed samples, rather than a total derivative through the rollout distribution.

Gate semantics. The gate is monotone in the sampled-token confidence gap:

$$\bar{g}_{i,t} \begin{cases} \to 1, & \beta\Delta_{i,t} \to +\infty, \\ = 1/2, & \Delta_{i,t} = 0, \\ \to 0, & \beta\Delta_{i,t} \to -\infty. \end{cases} \tag{7}$$

Tokens assigned substantially higher probability by the teacher receive larger reinforcement weights, equal log-probabilities give weight $1/2$, and tokens assigned substantially higher probability by the student receive weights approaching zero. These regimes describe relative confidence on the sampled token; they do not by themselves establish token correctness or identify the information lost during compression.

C.2 Relationship to Reverse KL

Fix a query and generated prefix, denoted collectively by h, and write

$$p_\theta(a) = \pi_\theta(a \mid h, Z), \qquad q(a) = \pi_T(a \mid h, M), \qquad \Delta(a) = \log q(a) - \log p_\theta(a).$$

Assume a finite vocabulary and strictly positive probabilities. With q fixed, the reverse KL $D_{\mathrm{KL}}(p_\theta \| q) = \mathbb{E}_{a\sim p_\theta}[\log p_\theta(a) - \log q(a)]$ has gradient

$$\nabla_\theta D_{\mathrm{KL}}(p_\theta \| q) = -\mathbb{E}_{a\sim p_\theta}\left[\Delta(a)\, \nabla_\theta \log p_\theta(a)\right], \tag{8}$$

where the term $\mathbb{E}_{a\sim p_\theta}[\nabla_\theta \log p_\theta(a)]$ arising from differentiating $\log p_\theta$ vanishes by the score-function identity. Both objectives can therefore be estimated from student-generated samples. At the level of individual samples, they differ in the coefficient applied to $\nabla_\theta \log p_\theta(a)$: reverse KL uses the unbounded, signed log-ratio $\Delta(a)$, whereas OPD uses the bounded, nonnegative weight $\bar{g}(a) = \sigma(\beta\Delta(a)) \in (0, 1)$. Note, however, that nonnegative sample weights do not imply that every token probability increases: probability normalization and shared parameters couple the updates across tokens. The following results make the relationship precise.

C.3 Expected Update: Stationarity and Bounded Influence

Proposition 2 (Conditional stationarity and local attenuation)**.** *At a fixed prefix h, suppose tokens are sampled exactly from the same distribution p_θ used to compute their log-probabilities, and define the expected surrogate gradient*

$$\mathcal{G}(\theta; h) = -\mathbb{E}_{a\sim p_\theta}\left[\bar{g}(a)\, \nabla_\theta \log p_\theta(a)\right]. \tag{9}$$

Then $p_\theta = q$ implies $\mathcal{G}(\theta; h) = 0$. More generally,

$$\|\mathcal{G}(\theta; h)\| \le \frac{\beta}{4} \sqrt{\mathbb{E}_{a\sim p_\theta}[\Delta(a)^2]} \sqrt{\mathbb{E}_{a\sim p_\theta}\left[\|\nabla_\theta \log p_\theta(a)\|^2\right]}. \tag{10}$$

In particular, if the score second moment is bounded, $\mathbb{E}_{a\sim p_\theta}[\|\nabla_\theta \log p_\theta(a)\|^2] \le S^2$, then $\|\mathcal{G}(\theta; h)\| \le \frac{\beta S}{4}\sqrt{\mathbb{E}_{a\sim p_\theta}[\Delta(a)^2]}$, which vanishes as the mean-squared log-probability gap vanishes.

Proof. By the score-function identity, $\mathbb{E}_{a\sim p_\theta}[\nabla_\theta \log p_\theta(a)] = \sum_a \nabla_\theta p_\theta(a) = \nabla_\theta 1 = 0$. Subtracting the constant $\frac{1}{2}$ from the gate therefore leaves the expectation unchanged:

$$\mathcal{G}(\theta; h) = -\mathbb{E}_{a\sim p_\theta}\left[\left(\bar{g}(a) - \tfrac{1}{2}\right) \nabla_\theta \log p_\theta(a)\right]. \tag{11}$$

If $p_\theta = q$, then $\Delta(a) = 0$ and $\bar{g}(a) = \frac{1}{2}$ for every token, proving stationarity. Since $\sigma'(u) \le \frac{1}{4}$ and $\sigma(0) = \frac{1}{2}$, $|\bar{g}(a) - \frac{1}{2}| \le \frac{\beta}{4}|\Delta(a)|$. Applying the triangle inequality and the Cauchy–Schwarz inequality to Eq. (11) yields Eq. (10). □

Corollary 1 (Expected OPD as a saturated reverse KL)**.** *Under the assumptions of Proposition 2,*

$$\mathcal{G}(\theta; h) = -\frac{1}{2}\, \mathbb{E}_{a\sim p_\theta}\left[\tanh\!\left(\tfrac{\beta\Delta(a)}{2}\right) \nabla_\theta \log p_\theta(a)\right], \tag{12}$$

and

$$\left\|\mathcal{G}(\theta; h) - \tfrac{\beta}{4}\, \nabla_\theta D_{\mathrm{KL}}(p_\theta \| q)\right\| \le \frac{\beta^3}{48}\, \mathbb{E}_{a\sim p_\theta}\left[|\Delta(a)|^3\, \|\nabla_\theta \log p_\theta(a)\|\right]. \tag{13}$$

Proof. Eq. (12) follows from Eq. (11) and the identity $\sigma(u) - \frac{1}{2} = \frac{1}{2}\tanh(u/2)$. By Eq. (8),

$$\mathcal{G}(\theta; h) - \tfrac{\beta}{4}\nabla_\theta D_{\mathrm{KL}}(p_\theta \| q) = -\mathbb{E}_{a\sim p_\theta}\left[\left(\tfrac{1}{2}\tanh\Big(\tfrac{\beta\Delta(a)}{2}\Big) - \tfrac{\beta\Delta(a)}{4}\right)\nabla_\theta \log p_\theta(a)\right].$$

Since $|\tanh(u) - u| \le |u|^3/3$, the scalar coefficient is bounded in absolute value by $\frac{1}{2}\cdot\frac{1}{3}\left|\frac{\beta\Delta(a)}{2}\right|^3 = \frac{\beta^3}{48}|\Delta(a)|^3$. The triangle inequality completes the proof. □

Corollary 1 clarifies what the gate does in expectation. The expected OPD update replaces the unbounded log-ratio $\Delta(a)$ in the reverse-KL gradient with the bounded, sign-preserving coefficient $\frac{1}{2}\tanh(\beta\Delta(a)/2) \in (-\frac{1}{2}, \frac{1}{2})$. When the confidence gap is small, OPD behaves as a reverse KL toward the textual-memory teacher scaled by $\beta/4$. When the gap is large, the influence of any single token is capped, so tokens on which the frozen teacher is strongly over- or under-confident cannot dominate the update. This is consistent with treating π_T as a reference reader of the uncompressed memory rather than as an accuracy oracle. The nonnegativity of $\bar{g}$ is a property of the per-sample weights: once the zero-mean baseline $\frac{1}{2}$ is removed, the effective expected coefficient is signed, and the expected update does move the student toward the teacher, with bounded strength.

Proposition 3 (OPD does not import the teacher's own choices)**.** *Fix a prefix h and treat the logits $z \in \mathbb{R}^{|\mathcal{V}|}$ of $p_\theta(\cdot \mid h, Z) = \mathrm{softmax}(z)$ as free parameters. For tokens $a_1, \ldots, a_n$ sampled at h, the descent direction of $\widehat{\mathcal{L}}_{\mathrm{OPD}}$ with respect to z is*

$$-\nabla_z \widehat{\mathcal{L}}_{\mathrm{OPD}} = \frac{1}{n}\sum_{j=1}^{n} \bar{g}(a_j)\left(e_{a_j} - p_\theta\right), \tag{14}$$

where e_a is the one-hot vector of token a. Consequently: (i) the logit of every token outside $\{a_j\}_{j=1}^n$ decreases or stays unchanged, regardless of the teacher q; and (ii) under exact sampling from p_θ, the expected change in the logit of any token b is

$$p_\theta(b)\Big(\bar{g}(b) - \mathbb{E}_{a\sim p_\theta}[\bar{g}(a)]\Big), \tag{15}$$

whose magnitude is at most $p_\theta(b)$.

Proof. For a softmax policy, $\nabla_z \log p_\theta(a) = e_a - p_\theta$, which gives the descent direction. For $b \notin \{a_j\}$, its b-th component is $-\frac{1}{n}\sum_j \bar{g}(a_j)\, p_\theta(b) \le 0$, proving (i). Taking the expectation over $a \sim p_\theta$, the b-th component becomes $p_\theta(b)\bar{g}(b) - p_\theta(b)\,\mathbb{E}_{a\sim p_\theta}[\bar{g}(a)]$. Since $\bar{g} \in (0, 1)$, the factor in parentheses lies in $(-1, 1)$, proving (ii). □

Proposition 3 separates two senses in which a student can be "pulled toward" a teacher. Imitating the teacher's own choices, as in supervised fine-tuning on teacher outputs or forward-KL distillation, follows $-\nabla_z \mathrm{CE}(q, p_\theta) = q - p_\theta$. Its b-th component $q(b) - p_\theta(b)$ can be large even when $p_\theta(b) \approx 0$, so tokens the teacher prefers are promoted whether or not the student produces them. Under OPD, by contrast, tokens the student did not sample are never promoted, and in expectation every logit change is scaled by the student's own probability. The pull established in Corollary 1 is therefore confined to re-ranking the student's own candidates according to the teacher's relative confidence; the teacher's own choices outside the student's support are never imported. The statement holds for the logits at a fixed prefix; with shared parameters θ, updates are additionally coupled across prefixes, as noted in Section C.2.

C.4 Remarks on the Implemented Estimator

Remark 1. Decomposing Eq. (6) as

$$\nabla_\theta \widehat{\mathcal{L}}_{\mathrm{OPD}} = -\widehat{\mathbb{E}}_{\mathcal{B}}\left[\left(\bar{g}_{i,t} - \tfrac{1}{2}\right)\nabla_\theta \ell_{\theta,i,t}\right] - \tfrac{1}{2}\,\widehat{\mathbb{E}}_{\mathcal{B}}\left[\nabla_\theta \ell_{\theta,i,t}\right]$$

separates the saturated reverse-KL term analyzed above from a baseline term. Under exact sampling, the baseline term has zero mean and contributes only variance. Under tempered or nucleus sampling, however, it becomes a self-imitation term toward the truncated sampling distribution, which tends to sharpen p_θ. Our PersonaMem rollouts use temperature 1.0 and top-$p = 0.98$, close to exact sampling. The LoCoMo rollouts use temperature 0.8 and top-$p = 0.95$, where this effect is larger. Sample-dependent per-sequence normalization $1/|y_i|$ further prevents exact cancellation.

Remark 2. The results above are local, conditional statements about the expected update at a fixed prefix. They do not guarantee that training reaches the regime $p_\theta \approx q$, that attenuation is monotone over training, or that the OPD coefficient λ_{OPD} need not be tuned.

D LIMITATIONS

First, the memory writer is initialized on memories extracted by a stronger external model; although this model is not needed at inference, the quality of the initial textual memory depends on it, and learning the writer without such supervision is left to future work. Second, our experiments cover Qwen backbones of up to 7B parameters; whether the same gains hold for larger models and other model families remains to be verified.