EVOUSER: FROM INTERACTION HISTORY TO EVOLVING USER STATE FOR LONG-TERM LLM PERSONALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Personalized large language models aim to generate responses tailored to individual users by leveraging their historical interactions or user profiles. However, existing approaches either represent users through textual memories, which are effective for explicit facts but struggle to capture weak and distributed signals emerging across interactions, or rely on user-specific model adaptation, which incurs substantial computational cost.We propose EVOUSER, a long-term personalization framework that combines a fixed-length, continuously evolving user state with query-driven textual retrieval. The framework uses the evolving user state to capture implicit information accumulated across interactions, while leveraging query-relevant textual retrieval to preserve explicit and traceable contextual evidence. We jointly train the state interface and response model through RAG-conditioned supervised learning and GRPO optimization. The complete system achieves accuracies of 82.68% and 67.53% on PersonaMem-v2 and PrefEval, respectively. Further controlled analyses demonstrate that even when the user states are frozen, they retain decodable user-specific attributes and preferences, validating that the continuous user state learns and stores effective personalized representations.

1 INTRODUCTION

As large language models (LLMs) are increasingly deployed as long-term assistants, personalization has become important for adapting responses to individual users based on information accumulated across past interactions (Salemi et al., 2024; Zhao et al., 2025; Tseng et al., 2024). Effective long-term personalization requires more than retrieving isolated historical facts. User information is heterogeneous and gradually revealed: some facts, constraints, and preferences are explicit, while others emerge through repeated choices, corrections, or behavioral patterns distributed across interactions (Salemi et al., 2024; Zhao et al., 2025; Jiang et al., 2025). A central challenge is therefore to maintain user representations that continuously incorporate new evidence while preserving information useful for future interactions (Maharana et al., 2024; Wu et al., 2025).

Existing approaches capture user information through complementary mechanisms. **Context-based methods** expose historical information to the LLM through retrieved interactions, textual profiles, or consolidated memories (Salemi et al., 2024; Zhong et al., 2023; Packer et al., 2023; Chhikara et al., 2025; Xu et al., 2025; Jiang et al., 2025; Kang et al., 2025). These representations are interpretable and preserve explicit facts, events, and preferences, but weak or distributed signals may remain fragmented across entries, increasing redundancy and retrieval noise, or be compressed into summaries that discard fine-grained distinctions. **Parameter-based methods** instead internalize personalization through model adaptation: OPPU learns a separate parameter-efficient module for each user (Tan et al., 2024b), while Profile-to-PEFT (P2P) trains a shared hypernetwork to generate personalized adapter parameters from encoded histories or profiles, avoiding per-user optimization at deployment (Tan et al., 2026). These methods provide powerful implicit personalization, but represent user-specific behavior through model parameters rather than a persistent state updated sequentially across interactions.

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.

A complementary line represents users directly in continuous latent spaces. USER-LLM encodes user timelines into continuous embeddings that capture latent behavioral patterns and temporal evolution (Ning et al., 2025), while TAP-PER shows that compact user-state prefixes can support scalable personalization with temporal information (Cao et al., 2026). Together, these studies establish the value of continuous user representations, but leave open a distinct question for long-term conversational personalization: can a persistent user state be *incrementally maintained* through a shared recurrent transition that updates the state whenever a new interaction arrives, rather than reconstructing it from accumulated history or encoding personalization in separately adapted parameters?

We introduce EVOUSER, a long-term personalization framework centered on a persistent, fixed-size *evolving user state*. Rather than reconstructing the user representation from accumulated history at each query, EVOUSER learns a shared recurrent updater that maps the previous state and latest interaction to an updated state. The updater is shared across users, while each user stores only the resulting continuous state. New evidence is thus incorporated as interactions unfold, allowing information distributed across interactions to accumulate progressively without user-specific model parameters or per-user optimization.

The evolving state complements rather than replaces explicit memory. EVOUSER combines retrieved textual memories, which provide direct access to explicit facts, events, and preferences that can be expressed and verified in language, with a recurrent state that compactly accumulates distributed latent information. At inference, the projected user state is provided to the response LLM with the current query, while retrieved memories supply explicit historical evidence. This design separates latent-signal accumulation from explicit-information retrieval, allowing the two mechanisms to play complementary roles.

For each target query, the shared updater processes preceding interactions chronologically to obtain the current state. Training proceeds in two stages. First, with the response LLM frozen, next-token supervision trains the updater and readout module, establishing an interface between the continuous state and language model. Second, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) jointly optimizes the updater, readout, and response LLM using task-specific rewards. The recurrent state is therefore trained end-to-end to retain and update user information insofar as it improves downstream personalized responses.

We evaluate EVOUSER on PersonaMem-v2 and PrefEval, which emphasize implicit user understanding and long-context preference following, respectively (Jiang et al., 2025; Zhao et al., 2025). Across Qwen3-1.7B, 4B, and 8B backbones, the complete EVOUSER system, comprising the evolving state and RAG, achieves accuracies of 57.42%, 72.16%, and 82.68% on PersonaMem-v2, and 49.77%, 65.07%, and 67.53% on PrefEval. Controlled analyses additionally show that frozen states contain decodable user-specific profile information.

Our contributions are threefold:

- We study long-term LLM personalization through *incrementally maintained user representations*, where a shared recurrent transition continuously updates a persistent, fixed-size state as interactions unfold.
- We propose EVOUSER, combining an evolving continuous user state with retrieved textual memory. The recurrent state is designed to integrate distributed user signals, while textual retrieval preserves explicit historical evidence, without user-specific model parameters or per-user optimization.
- We optimize the recurrent state and response model for downstream personalization through RAG-conditioned supervised interface learning followed by joint GRPO training, and demonstrate consistent improvements of the complete system on PersonaMem-v2 and PrefEval.

2 RELATED WORK

2.1 LONG-TERM PERSONALIZATION FOR LLMS

Long-term LLM personalization commonly incorporates user histories through textual context or model adaptation (Salemi et al., 2024; Tan et al., 2024b; Tseng et al., 2024). Retrieval- and memory-

based approaches expose historical information to the LLM as retrieved interactions, textual profiles, or explicitly maintained memories (Salemi et al., 2024; Chhikara et al., 2025; Xu et al., 2025; Jiang et al., 2025). These methods preserve explicit user information in an interpretable form, but their personalization ultimately depends on representing relevant history in natural language context. A complementary direction internalizes user information into model parameters. OPPU trains a separate parameter-efficient module for each user, while Per-Pcs improves scalability by sharing and assembling pieces of personalized PEFT modules across users (Tan et al., 2024b;a). Profile-to-PEFT (P2P) further eliminates per-user training at deployment by using a shared hypernetwork to generate personalized adapter parameters from encoded user profiles (Tan et al., 2026). In contrast, EVOUSER maintains personalization as a persistent activation-space state: model parameters are shared across users, while user-specific information is carried forward through a recurrently updated continuous representation. We additionally retain textual retrieval as a complementary mechanism for explicit historical evidence rather than treating latent state and textual memory as mutually exclusive alternatives.

2.2 LATENT USER REPRESENTATIONS

Recent work has explored continuous user representations as an alternative to serializing entire histories into prompts. USER-LLM encodes user timelines into continuous embeddings that capture latent behavior and its temporal evolution (Ning et al., 2025). PPlug constructs a user-specific embedding from historical contexts, PERSOMA compresses free-form histories into personalized soft prompts (Liu et al., 2025; Hebert et al., 2024), and CURP extracts multidimensional user traits through a prototype codebook (Wang et al., 2026). More closely related, TAP-PER combines a learnable user-state prefix with query-conditioned history aggregation and temporal biases (Cao et al., 2026), while Latent Personal Memory maintains persistent user-specific latent slots mapped to input-conditioned soft prompts (Das et al., 2026). Although these methods demonstrate the effectiveness of continuous user representations, they primarily encode available histories or maintain user-specific representations. In contrast, EVOUSER uses a shared recurrent transition to update the previous state with each new interaction, accumulating persistent user information without user-specific trainable parameters or reconstructing the representation from the full history for every query.

2.3 CONTINUAL AND RECURRENT LATENT MEMORY

Our formulation is also related to work on persistent and recurrent latent memory beyond personalization (Rae et al., 2019; Bulatov et al., 2022; Chevalier et al., 2023; Wang et al., 2023). MEMORYLLM introduces a fixed-size latent memory pool that can be repeatedly updated to incorporate new knowledge into an LLM (Wang et al., 2024), while LiveMem studies fixed-capacity memory-state continuity when the active context turns over during long-running inference (Liu et al., 2026). Their primary objective is general knowledge retention or inference continuity rather than user-specific personalization.

Rec2PM applies recurrent preference tokens to item-sequence recommendation, whereas PUMA uses action-conditioned latent states for dialogue-policy selection (Chen et al., 2026; Luo et al., 2026). EVOUSER introduces a different formulation for general LLM personalization: a shared LLM-based transition maintains a fixed-capacity user state online, a separate retrieval channel supplies explicit episodic evidence, and both channels are optimized with the response model for future personalized generation. This combination supports open-ended user histories without user-indexed parameters or replaying the full history through the continuous branch.

3 METHOD

Figure 1 provides an overview of EVOUSER. The framework maintains a fixed-size continuous state for each user and updates it recurrently as new interactions are completed. The state is maintained by a shared LLM-based updater and is projected into continuous prefix embeddings for personalized generation. In this way, the evolving state provides a compact and persistent representation of user-specific information accumulated across interactions.

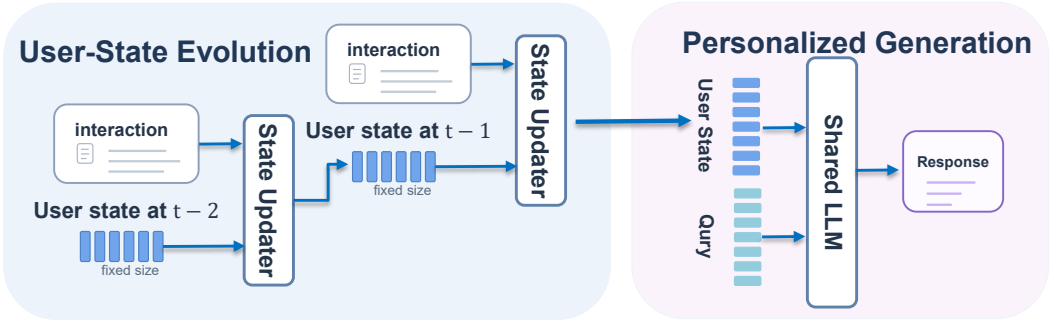


Figure 1: **EvoUser architecture.** At the core of EvoUser is an evolving continuous user state that is persistently carried across interactions and recurrently updated as new evidence arrives. For a new query, the current state is projected into a short continuous prefix and consumed by a shared response LLM. A retrieval-augmented generation (RAG) module additionally supplies explicit query-relevant evidence from past interactions.

3.1 Problem Setup and Temporal Protocol

Consider a user u with a chronological sequence of completed interactions $I_1^u, \ldots, I_t^u$, where each interaction contains the user input and the corresponding system response. Before the next query q_{t+1}^u arrives, EvoUser maintains two representations of the preceding history: a continuous user state

$$h_t^u \in \mathbb{R}^{K \times d_h},$$

and a textual interaction archive $\mathcal{A}_t^u$. Here, K denotes the number of latent state tokens and d_h is the hidden dimension of the state updater. The continuous state has fixed capacity and persists across interactions. It is not reconstructed from the full history when a new query arrives. Instead, each completed interaction modifies the state carried forward from the previous step. In contrast, $\mathcal{A}_t^u$ retains textual records that can be retrieved when the current query requires explicit historical evidence. Importantly, both representations obey the same temporal boundary. When generating the response to q_{t+1}^u, the model has access only to interactions completed before that query. The resulting interaction is incorporated into the state and textual archive only after the response has been generated.

3.2 Recurrent User-State Updater

The core of EvoUser is a shared recurrent state updater F_θ. We implement F_θ using a pretrained decoder-only language model with its language-modeling head removed. The updater therefore produces hidden representations rather than token probabilities.

The initial state consists of K learned latent query tokens,

$$h_0 = Q \in \mathbb{R}^{K \times d_h},$$

which are shared as the initialization for all users. After interaction I_t^u is completed, its token embeddings are provided to the updater together with the previous user state. Concretely, we form

$$Z_t^u = \left[E_\theta(I_t^u); h_{t-1}^u \right],$$

where $E_\theta(I_t^u)$ denotes the embedded interaction sequence. The previous state occupies K designated latent query positions following the interaction tokens. Passing the sequence through the headless language model gives

$$H_t^u = \mathrm{LM}_\theta^{\text{no-head}}\left(Z_t^u\right).$$

The hidden representations corresponding to the K latent query positions are then extracted as the updated state:

$$h_t^u = \mathrm{Select}_K\left(H_t^u\right) = F_\theta\left(h_{t-1}^u, I_t^u\right).$$

This construction turns the latent query tokens into a fixed-capacity carrier of information across interactions. At every step, the updater observes only the previous state and the newly completed interaction; earlier interactions are not replayed to the updater. Because the same parameters θ are shared across users, personalization is represented by the evolving activations h_t^u rather than by user-specific model parameters. The fixed number of state tokens also prevents memory growth through simple accumulation. As new evidence arrives, the updater must transform the existing state, implicitly learning which information should be preserved, revised, or overwritten.

3.3 State-Conditioned Generation with Textual Retrieval

The hidden space used by the state updater need not coincide with the embedding space of the response LLM. We therefore introduce a lightweight readout network G_ϕ that maps each latent state token into the response-model embedding space. Specifically, G_ϕ is a two-layer feed-forward network,

$$G_\phi(h) = W_2\,\sigma(W_1 h + b_1) + b_2,$$

applied token-wise to the K latent states. The resulting continuous prefix is

$$P_t^u = G_\phi(h_t^u) \in \mathbb{R}^{K \times d_m},$$

where d_m is the embedding dimension of the response LLM. Continuous state is complemented by textual retrieval. Given the current query, we retrieve query-relevant evidence from the interaction archive:

$$R_t^u(q_{t+1}^u) = \mathrm{Retrieve}\left(q_{t+1}^u, \mathcal{A}_t^u\right).$$

Let $\mathcal{M} * \psi$ denote the response LLM. The response $y * t + 1^u = (y_1, \ldots, y_L)$ is generated according to

$$p_{\theta,\phi,\psi}\left(y_{t+1}^u \mid q_{t+1}^u, \mathcal{A}_t^u\right) = \prod_{j=1}^{L} p_\psi\left(y_j \mid P_t^u, R_t^u(q_{t+1}^u), q_{t+1}^u, y_{<j}\right).$$

The two conditioning channels are intentionally complementary. The recurrent state is constructed before the current query is observed and carries information accumulated across earlier interactions, whereas RAG retrieves explicit records conditioned on the current query. We do not impose an artificial requirement that the two channels contain disjoint information. Unlike a train-test setup in which retrieval is introduced only at evaluation time, we use the same RAG-conditioned generation interface during training and inference. Both supervised warm-up and subsequent GRPO optimization therefore expose the response model to the combination of continuous state, retrieved textual evidence, and the current query. After generating the response to q_{t+1}^u, the completed interaction I_{t+1}^u is used to obtain

$$h_{t+1}^u = F_\theta\left(h_t^u, I_{t+1}^u\right),$$

and the same interaction is appended to $\mathcal{A}_t^u$ for future retrieval. Thus, the target response never enters either personalization channel before it is generated.

3.4 LEARNING THE EVOLVING USER STATE

The recurrent architecture specifies how information can persist, but not what information the state should retain. Rather than supervising the state toward a predefined textual profile or requiring it to reconstruct previous interactions, we optimize it according to its utility for future personalized responses. For each training target, interactions preceding the target query are processed chronologically to construct the current user state. Training consists of two stages with different optimization roles.

Stage 1: State Interface Warm-up. We first freeze the response LLM at pretrained parameters ψ_0 and optimize the recurrent updater F_θ together with the readout network G_ϕ using next-token supervision. During this stage, recurrent states are detached between consecutive update steps. Specifically, the forward rollout follows

$$h_t^u = F_\theta\left(\mathrm{sg}(h_{t-1}^u), I_t^u\right),$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator. The updater therefore still observes states produced by preceding interactions during the forward pass, but the target loss does not backpropagate through the entire historical rollout. This provides a stable warm-up stage for learning the state transition and the interface between latent state and the frozen response model. Using the same RAG conditioning employed at inference time, the supervised objective is

$$\mathcal{L}_{\mathrm{SFT}} = -\sum_{(u,q,y)} \sum_{j=1}^{|y|} \log p_{\psi_0}\left(y_j \mid G_\phi(h_T^u), R_T^u(q), q, y_{<j}\right).$$

Only the updater and readout are optimized in this stage; the response LLM remains frozen.

Stage 2: End-to-End Recurrent Optimization with GRPO. After the state interface has been initialized, we jointly optimize the state updater, readout, and response LLM using Group Relative Policy Optimization (GRPO) (Shao et al., 2024). In contrast to Stage 1, recurrent states are no longer detached. Gradients from the downstream personalization objective therefore propagate through the readout and through the sequence of recurrent state transitions preceding the target query. For each training instance, the model receives the evolved state prefix, retrieved textual evidence, and current query, and samples a group of G candidate responses. Because our training tasks are multiple-choice personalization problems, we use an outcome reward based on answer correctness:

$$r_i = \mathbb{I}\left[\mathrm{Ans}(y_i) = y^\star\right],$$

where $y^\star$ is the gold answer and $\mathrm{Ans}(\cdot)$ extracts the predicted option. Incorrect or invalid answers receive zero reward. Rewards are normalized within each sampled group to obtain relative advantages,

$$A_i = \frac{r_i - \mathrm{mean}(r)}{\mathrm{std}(r) + \epsilon_A}.$$

We use the standard clipped GRPO objective with KL regularization. Let $\rho_{i,j}$ denote the token-level probability ratio between the current policy and the behavior policy used to generate the sampled response. The optimization objective takes the form

$$\mathcal{L}_{\mathrm{GRPO}} = -\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{j=1}^{|y_i|}\min\left(\rho_{i,j}A_i,\ \mathrm{clip}(\rho_{i,j}, 1-\epsilon, 1+\epsilon)A_i\right) - \beta D_{\mathrm{KL}}\left(\pi_{\theta,\phi,\psi}\|\pi_{\mathrm{ref}}\right)\right],$$

where ϵ is the clipping coefficient, β controls KL regularization, and π_{ref} is a fixed reference policy. Because the response probabilities depend on the continuous prefix $G_\phi(h_T^u)$, and h_T^u is obtained

through the recurrent updater, the Stage 2 objective jointly adapts the response policy, the state-to-prefix interface, and the sequence of state transitions. The evolving state is therefore optimized according to the information it preserves for future personalization rather than according to an externally specified representation target.

3.5 Bounded State Maintenance and Deployment

The continuous-state branch stores K latent vectors per user, requiring $\mathcal{O}(Kd_h)$ space independent of history length. Each update processes only the previous state and latest interaction, with input length $\mathcal{O}(K + |I_t^u|)$, while generation adds a fixed number of prefix tokens. Because all parameters are shared, new users require only forward state updates, without user-specific optimization. These bounds apply only to the continuous branch; the auxiliary RAG archive may grow with interaction history.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate on two complementary long-term personalization benchmarks. **PersonaMem-v2** Jiang et al. (2025) tests whether a model can infer implicit user characteristics from evidence distributed across long interaction histories; we follow its multiple-choice protocol. **PrefEval** Zhao et al. (2025) evaluates whether a model can recover and follow user preferences despite substantial conversational interference. We include its `explicit`, `implicit_choice`, and `implicit_persona` settings, with 1,000 examples each, and insert 88 complete distractor interactions into every history. We use the classification formulation for PrefEval and report accuracy on both benchmarks. Full history construction and evaluation settings are provided in Appendix B.1.

Baselines. We compare against representative context-, memory-, and parameter-based personalization methods. **No Memory** receives only the current query and measures performance without user history, whereas **Full History** directly places the raw interaction history in the response-model context. For textual memory, **LightMem** Fang et al. (2025) extracts and retrieves compact memories, **Inside Out / MemListener** Zhao et al. (2026) maintains structured user-centric memory, and **Mem-α** Wang et al. (2025) uses a learned policy for memory construction. We also include continuous and parametric user representations: **PPlug** Liu et al. (2025) encodes historical interactions into a user embedding that conditions generation, while **OPPU** Tan et al. (2024b) fits user-specific parameter-efficient modules. All baselines requiring task-specific training use only the PersonaMem-v2 training split; the resulting models are evaluated on both benchmarks without additional training on PrefEval.

Evaluation Settings. All methods use the same Qwen3-1.7B or Qwen3-4B response backbone and are evaluated on the same test instances. We use greedy decoding with temperature $= 0$ and randomize answer-option orders with a fixed seed of 41. All memories use only interactions preceding the target query; no test labels are used during memory construction or retrieval. Appendix B.1 provides the complete context, serving, and retrieval settings for each method.

4.2 Overall Results

As presented in Table 1, the complete EvoUser framework (combining the continuous evolving state with query-driven RAG) achieves superior performance across all model scales (Qwen3-1.7B, 4B, and 8B) on both benchmarks.

To understand the fundamental limitations of existing personalization strategies, we categorize the baselines into three prevailing paradigms:

- **Passive Context Concatenation (No Memory & Full History).** Exposing raw interaction histories directly to the response LLM leads to severe context dilution and high inference cost. As shown in Table 1, extending the accessible context window (Full History 32K) offers marginal or even negative gains over No Memory (e.g., 32.36% vs.

Table 1: **Accuracy (%) with the same Qwen3 response backbones.** Full EVOUSER (state + RAG) results are bold; strongest baselines are underlined. Dashes denote unavailable results. PPlug uses $\mathrm{lr} = 10^{-6}$ for its Qwen3-8B run.

Method	PersonaMem-v2			PREFEVAL		
	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B
No Memory	24.82	27.70	27.86	29.00	41.80	35.80
Full History 32K	24.22	33.36	32.36	30.27	44.00	39.17
LightMem	28.68	25.94	28.62	<u>39.53</u>	45.50	<u>54.10</u>
PPlug	28.16	<u>38.18</u>	26.92	22.63	44.87	36.27
InsideOut / MemListener	<u>28.86</u>	33.84	29.80	36.57	<u>48.07</u>	44.20
OPPU	24.82	26.82	<u>33.30</u>	23.07	33.67	41.33
Mem-α	26.04	24.52	24.78	36.97	45.73	43.33
EVOUSER (State + RAG)	**57.42**	**72.16**	**82.68**	**49.77**	**65.07**	**67.53**

Table 2: **Component ablation on PrefEval.** We vary whether the recurrent state updater and response LLM are trained and whether textual retrieval is available. *Untrained State* receives no interaction history and uses an untrained state input. Accuracy (%) is reported.

Setting	**Updater**	**Response LLM**	**RAG**	**Qwen3-1.7B**	**Qwen3-4B**
Untrained State	Untrained	Trained	×	25.73	46.07
Evolving State	Trained	Trained	×	39.00	54.47
EvoUser	**Trained**	**Trained**	✓	**49.77**	**65.07**

27.86% on PERSONAMEM-V2 8B). This indicates that simply expanding context capacity cannot substitute for active user state modeling, as uncurated raw interactions introduce heavy attention noise and force the LLM to reconstruct user traits on the fly.

- **Discrete Textual Memory (LIGHTMEM & MEM-α).** Memory-augmented methods reduce context noise by extracting and retrieving explicit symbolic records. However, their reliance on discretization creates an expressiveness bottleneck: weak preference signals or subtle behavioral habits distributed across turns often fail to trigger explicit memory extraction rules and are lost prior to retrieval. Consequently, their peak accuracy remains capped across scales (e.g., 28.62% on PERSONAMEM-V2 8B and 54.10% on PREFEVAL 8B).
- **Static or Parameterized Representations (PPLUG, INSIDEOUT, & OPPU).** Methods that compress interaction streams into soft prompts or parameter-efficient modules capture sub-symbolic user signals better than discrete text. Nevertheless, they lack an online recurrent update mechanism. Reconstructing representations per query or adapting user-isolated parameter modules fails to maintain a persistent state that evolves chronologically as new evidence arrives. As a result, these approaches reach at most 38.18% on PERSONAMEM-V2 and 48.07% on PREFEVAL.

Scaling Trajectory and Dual-Channel Synergy. EVOUSER breaks through these limitations, achieving state-of-the-art accuracies of **82.68%** on PERSONAMEM-V2 and **67.53%** on PREFEVAL with Qwen3-8B. Crucially, EVOUSER's advantage expands dramatically as the backbone scale grows. On PERSONAMEM-V2, the performance margin over the strongest baseline widens from +28.56% at 1.7B to +33.98% at 4B, and culminates at +49.38% at 8B. On PREFEVAL, it delivers robust gains of +10.24%, +17.00%, and +13.43% across the respective backbones. This widening performance gap suggests that larger base models possess superior reasoning capacity to decode, ground, and leverage the dense continuous user state prefix alongside retrieved explicit evidence, validating the strong scaling synergy of our dual-channel design.

4.3 ABLATION STUDY

Table2 examines the contributions of recurrent user states and textual retrieval on PrefEval. Without retrieval, the evolving state improves over the history-free untrained-state reference by 13.27 and 8.40 percentage points for Qwen3-1.7B and Qwen3-4B, respectively. These gains indicate that the continuous state carries useful historical information for personalization, even without direct access to historical text at response time. Nevertheless, adding RAG yields further gains of 10.77 and 10.60

points, suggesting that the fixed-capacity state does not retain all task-relevant evidence accessible through retrieval. The similar retrieval gains across both model sizes further suggest that increasing backbone capacity alone does not remove the benefit of textual access in these settings. Overall, the results support combining recurrent state maintenance with query-relevant retrieval. The first comparison evaluates history access and learned state updating jointly, rather than isolating updater training.

4.4 Effect of the Number of Latent State Tokens

We next study how the capacity of the evolving user state affects personalization performance. We vary the number of latent state tokens K within the full state-and-RAG configuration while keeping retrieval and the remaining evaluation pipeline unchanged.

Table 3 shows that increasing state capacity consistently improves accuracy across all three backbones. The largest gain occurs when increasing K from 1 to 5, yielding improvements of 19.70, 19.00, and 24.20 percentage points for the 1.7B, 4B, and 8B models, respectively. This indicates that a single latent token forms a restrictive bottleneck for representing heterogeneous user information. Performance continues to improve with 10 and 20 tokens, but the marginal gains gradually decrease. Increasing K from 20 to 40 adds only 0.62, 1.36, and 1.08 points, suggesting diminishing returns once the state has sufficient capacity. We therefore use $K = 40$ in the main experiments: it provides the strongest accuracy in this sweep while retaining a fixed state size independent of the length of the interaction history.

Table 3: **Effect of latent-state capacity.** Accuracy (%) as K increases with the RAG configuration fixed.

# Latent State Tokens (K)	1.7B	4B	8B
1	32.40	45.30	51.20
5	52.10	64.30	75.40
10	55.20	68.00	79.20
20	56.80	70.80	81.60
40	**57.42**	**72.16**	**82.68**

4.5 Profile-Related Information in the Continuous State

We test whether profile-derived user information is linearly readable from the learned states. With the state models frozen, we fit ridge probes on 683 users and evaluate on 294 users with disjoint IDs and normalized profiles. Figure 2 shows that aligning Rolling and Direct states from the same training users yields 0.9203 mean AUROC on two preference axes, compared with 0.5295 after shuffling user correspondence. This result shows that user correspondence matters for transferring the profile-label readout. Because the labels describe global user attributes, this experiment is scoped to user-specific profile-label readout; temporal state behavior is analyzed separately in the appendix. Full protocols and controls are provided there.

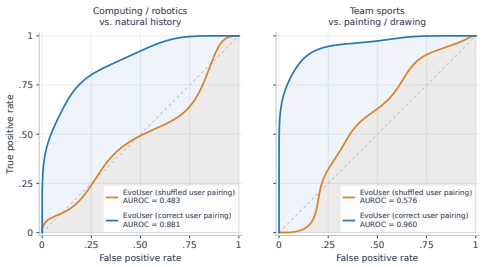


Figure 2: **Profile-label readout depends on user correspondence.** A fixed profile-label probe transfers more successfully after aligning Rolling and Direct states from the same training users (blue) than after shuffling their correspondence (orange).

5 Conclusion

We presented EvoUser, a recurrent state framework that combines a bounded, shared-update user representation with query-conditioned textual retrieval. The complete state-and-RAG system achieves 57.42%/72.16%/82.68% on PersonaMem-v2 and 49.77%/65.07%/67.53% on PrefEval with Qwen3-1.7B/4B/8B. Component ablations show measurable gains from training the recurrent updater and further gains from retrieval, while frozen state analyses identify decodable user-specific profile information. Future work will explore broader real-world personalization scenarios, more adaptive state update mechanisms, and improved analysis of state interpretability and controllability.

LIMITATIONS

Our empirical evaluation is conducted on existing personalization benchmarks, and future work could further investigate the effectiveness of EVOUSER in more diverse real-world assistant scenarios, including open-ended dialogue and interactive applications. In addition, the current implementation relies on a shared recurrent updater based on a pretrained language model, and exploring more efficient architectures for state updating may further improve scalability. Finally, our experiments focus on evaluating personalization performance, while a more comprehensive analysis of user privacy, controllability, and state interpretability remains an interesting direction for future research.

AI USE STATEMENT

Generative AI tools were used to assist with language polishing and presentation. All technical content, experimental design, analysis, and conclusions were developed and verified by the authors.

ETHICS STATEMENT

Our work studies long-term user modeling and personalization, which may involve privacy risks when applied to real-world user data. The experiments in this work rely on existing benchmarks and controlled evaluation settings without collecting new sensitive user data. In practical deployment, appropriate consent, data protection, and user control over stored information should be considered.

REPRODUCIBILITY STATEMENT

We will release the implementation and necessary configurations to facilitate reproduction of the reported results.

REFERENCES

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/47e288629a6996a17ce50b90a056a0e1-Abstract.html.

Heng Cao, Fan Zhang, Jian Yao, Yujie Zheng, Changlin Zhao, Lu Hao, Yuxuan Wei, Wangze Ni, Huaiyu Fu, Yuqian Sun, and Xuyan Mo. Beyond retrieval: Learning compact user representations for scalable llm personalization. *arXiv preprint arXiv:2606.04547*, 2026.

Yixiao Chen, Yuan Wang, Yue Liu, Qiyao Wang, Ke Cheng, Xin Xu, Juntong Yan, Shuojin Yang, Menghao Guo, Jun Zhang, Huan Yu, and Jie Jiang. Recurrent preference memory for efficient long-sequence generative recommendation. *arXiv preprint arXiv:2602.11605*, 2026.

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. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.232. URL https://aclanthology.org/2023.emnlp-main.232/.

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.

Debrup Das, Avinash Amballa, Yashas Malur Saidutta, Vijay Srinivasan, Vivek Kulkarni, and Srinivas Chappidi. Latent personal memory: Represent personal memory as dynamic soft prompts. *arXiv preprint arXiv:2606.20911*, 2026.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. LightMem: Lightweight and efficient memory-augmented generation. *arXiv preprint arXiv:2510.18866*, 2025. URL https://arxiv.org/abs/2510.18866.

Liam Hebert, Krishna Sayana, Ambarish Jash, Alexandros Karatzoglou, Sukhdeep Sodhi, Sumanth Doddapaneni, Yanli Cai, and Dima Kuzmin. Persoma: Personalized soft prompt adapter architecture for personalized language prompting. *arXiv preprint arXiv:2408.00960*, 2024.

Bowen Jiang, Yuan Yuan, Maohao Shen, Zhuoqun Hao, Zhangchen Xu, Zichen Chen, Ziyi Liu, Anvesh Rao Vijjini, Jiashu He, Hanchao Yu, Radha Poovendran, Gregory Wornell, Lyle Ungar, Dan Roth, Sihao Chen, and Camillo Jose Taylor. PersonaMem-v2: Towards personalized intelligence via learning implicit user personas and agentic memory. *arXiv preprint arXiv:2512.06688*, 2025.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25961–25970, 2025. doi: 10.18653/v1/2025.emnlp-main.1318. URL https://aclanthology.org/2025.emnlp-main.1318/.

Jiongnan Liu, Yutao Zhu, Shuting Wang, Xiaochi Wei, Erxue Min, Yu Lu, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. Llms + persona-plug = personalized llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9373–9385. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.461.

Zhichen Liu, Ruihan Sun, Hengjie Yang, Zipeng Wu, Zhaohan Chen, Xiaofan Zhang, and Yang Xu. Livemem: Maintaining memory state continuity in long-running llm inference. *arXiv preprint arXiv:2608.02515*, 2026.

Jiani Luo, Xiaoyan Zhao, Yang Zhang, Shuyi Miao, Bingbing Xu, Stefan Konigorski, and Tat-Seng Chua. Know you before you speak: User-state modeling for llm personalization in multi-turn conversation. *arXiv preprint arXiv:2605.24647*, 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. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Lin Ning, Luyang Liu, Jiaxing Wu, Neo Wu, Devora Berlowitz, Sushant Prakash, Bradley Green, Shawn O'Banion, and Jun Xie. User-llm: Efficient llm contextualization with user embeddings. In *Companion Proceedings of the ACM Web Conference 2025*, pp. 1219–1223, 2025. doi: 10.1145/3701716.3715463.

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.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling, 2019. URL https://arxiv.org/abs/1911.05507.

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, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Zhaoxuan Tan, Zheyuan Liu, and Meng Jiang. Personalized pieces: Efficient personalized large language models through collaborative efforts. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 6459–6475. Association for Computational Linguistics, 2024a. doi: 10.18653/v1/2024.emnlp-main.371.

Zhaoxuan Tan, Qingkai Zeng, Yijun Tian, Zheyuan Liu, Bing Yin, and Meng Jiang. Democratizing large language models via personalized parameter-efficient fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2024b. doi: 10.18653/v1/2024.emnlp-main.372.

Zhaoxuan Tan, Zixuan Zhang, Haoyang Wen, Zheng Li, Rongzhi Zhang, Pei Chen, Fengran Mo, Zheyuan Liu, Qingkai Zeng, Qingyu Yin, and Meng Jiang. Instant personalized large language model adaptation via hypernetwork. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23557–23580. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1081.

Yu-Min Tseng, Yu-Chao Huang, Teng-Yun Hsiao, Wei-Lin Chen, Chao-Wei Huang, Yu Meng, and Yun-Nung Chen. Two tales of persona in LLMs: A survey of role-playing and personalization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 16612–16631, 2024. doi: 10.18653/v1/2024.findings-emnlp.969. URL https://aclanthology.org/2024.findings-emnlp.969/.

Liang Wang, Xinyi Mou, Xiaoyou Liu, Xuanjing Huang, and Zhongyu Wei. Curp: Codebook-based continuous user representation for personalized generation with llms. *arXiv preprint arXiv:2602.00742*, 2026.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/ebd82705f44793b6f9ade5a669d0f0bf-Abstract-Conference.html.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian McAuley. Memoryllm: Towards self-updatable large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 50453–50466. PMLR, 2024.

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. URL https://arxiv.org/abs/2509.25911.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/d813d324dbf0598bbdc9c8e79740ed01-Abstract-Conference.html.

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.

Jihao Zhao, Ding Chen, Zhaoxin Fan, Kerun Xu, Mengting Hu, Bo Tang, Feiyu Xiong, and Zhiyu Li. Inside out: Evolving user-centric core memory trees for long-term personalized dialogue systems. *arXiv preprint arXiv:2601.05171*, 2026. URL https://arxiv.org/abs/2601.05171.

Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. Do llms recognize your preferences? evaluating personalized preference following in llms. *arXiv preprint arXiv:2502.09597*, 2025.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. *arXiv preprint arXiv:2305.10250*, 2023.

A Supplementary Response Evaluations

A.1 Additional and Heterogeneous Comparisons

PrefEval results. InsideOut / MemListener is the strongest Qwen3-4B baseline at 48.07%, while LightMem is strongest with Qwen3-1.7B and Qwen3-8B at 39.53% and 54.10%, respectively. The full EvoUser system (state + RAG) reaches 49.77%, 65.07%, and 67.53% with Qwen3-1.7B, 4B, and 8B, exceeding the strongest respective baselines by 10.24, 17.00, and 13.43 percentage points. All results use the same 3,000-query denominator.

Official heterogeneous checkpoints. P2P, MemAgent, and MEM1 obtain 28.42%, 25.60%, and 24.24% with their heterogeneous official 7B checkpoints and are not scale-matched comparisons.

A.2 Baseline Audit: Contribution of PPlug Profiles

Removing the profile from PPlug while retaining its remaining graph reduces accuracy from 38.18% to 36.54% (4B) and from 28.16% to 26.40% (1.7B). Table 1 uses the stronger profile-enabled variant.

A.3 Control Inputs for the Continuous Prefix

We replace the learned Qwen3-4B prefix with dimension-matched Gaussian noise or zero vectors while leaving the remaining pipeline unchanged.

Table 4: **Sensitivity to replacing the learned prefix.** Qwen3-4B accuracy (%).

State-prefix input	Accuracy	Δ vs. learned
Gaussian noise	25.76	-46.40
Zero vectors	26.82	-45.34
Learned recurrent state	**72.16**	–

Replacing the learned prefix with Gaussian noise or zero vectors reduces accuracy by 46.40 and 45.34 points, respectively. These results show sensitivity to the learned continuous input: extra prefix positions alone do not preserve performance under matched-size replacements. The profile-label and user-correspondence controls separately assess the user-specific content of the learned representation.

B Additional Experimental Details

B.1 Detailed Evaluation Settings

For PrefEval, we use the `explicit`, `implicit_choice`, and `implicit_persona` subsets, each containing 1,000 examples. We insert 88 complete user–assistant distractor interactions per instance, resulting in 89, 90, and 91–95 turns, respectively. This gives 3,000 classification instances in total.

For **Full History**, we set `max_prompt_tokens=32768` and `max_model_len=33792`. For the rolling state in EvoUser, we use $K = 40$ latent state tokens, with `max_prompt_tokens=1024` and `max_model_len=4096`. For EvoUser+RAG, we additionally perform dense retrieval using all-MiniLM-L6-v2 and retrieve the Top-10 historical items. **No Memory** uses no interaction history, with `max_prompt_tokens=1000` and `max_model_len=2048`. For **LightMem**, we retrieve the Top-10 memories for each user, use a 40K serving context, and disable thinking.

B.2 Training Objectives and Procedure

Training preserves the temporal order of each user's interactions. Starting from h_0, we apply the recurrent state update sequentially so that the state used for a training example is produced by the

same rolling process used at deployment. For a target response y_t^u to query q_t^u, the model is conditioned on the current state prefix $G_\phi(h_t^u)$, retrieved evidence $R_t^u(q_t^u)$, and the query. The same RAG-conditioned generation interface is used in both training stages and at inference. We use two stages because a randomly initialized continuous interface is difficult for the response model and updater to learn simultaneously.

Stage 1: state warm-up with a frozen LLM. We first freeze the response-model parameters, $\psi = \psi_0$, and optimize only the recurrent updater F_θ and state-to-prefix readout G_ϕ with supervised fine-tuning (SFT):

$$\mathcal{L}_{\mathrm{SFT}}^{(1)} = -\sum_{u,t} \log p_{\psi_0}\big(y_t^u \mid G_\phi(h_t^u), R_t^u(q_t^u), q_t^u\big). \tag{1}$$

This warm-up establishes the state-prefix interface before response-model adaptation.

Stage 2: joint GRPO. After supervised warm-up, we jointly update θ, ϕ, and ψ on the task training set $\mathcal{D}_{\mathrm{train}}$ using Group Relative Policy Optimization (GRPO):

$$\mathcal{L}^{(2)} = \mathcal{L}_{\mathrm{GRPO}}(\mathcal{D}_{\mathrm{train}}), \tag{2}$$

For each training prompt, the model receives the evolved state prefix, retrieved textual evidence, and current query. GRPO samples a group of candidate outputs, scores them with the task-specific reward, and updates the model using their relative performance within the group.

The state is optimized for personalization task performance, not history reconstruction or profile matching. All learned parameters are shared; deployment uses forward state updates and retrieval operations without per-user gradient optimization.

B.3 Profile-Label Readout Analysis Protocol

Frozen representations and inputs. The analyses use the original Qwen3-0.6B base with the released Direct LoRA/latent queries and Rolling updater, without retraining these components. Both adapters have rank 16 and LoRA scaling parameter 32; dropout is disabled in evaluation mode. Direct appends 40 query embeddings to the final 32,728 chat-templated history tokens, for a maximum total length of 32,768. The initial system profile is excluded when present as the first message. States are taken before the response projector and averaged over their 40 slots to obtain 1,024-dimensional features.

Labels and splitting. The fixed split contains 683 probe-training and 294 probe-evaluation users, with no shared user IDs or normalized persona profiles across these sets. All probes predict the same 26 existing binary labels. For the primary endpoint, each preference axis uses the difference between its two label scores; the two axis AUROCs are averaged equally. Computing/robotics versus natural history contributes 27 observations and team sports versus painting 45, with 70 distinct users in total because the axes overlap. The secondary endpoint averages one-versus-rest AUROC over the 21 labels with at least 20 positive and 20 negative training examples and at least 10 of each in evaluation. The eligible label set is fixed across conditions.

Ridge readout. Each ordinary condition fits its own deterministic multi-output ridge probe. Let X_c and Y_c be training-centered feature and binary-label matrices, with $n = 683$ users and feature dimension d. Define

$$G = \frac{X_c X_c^\top}{d}, \qquad \alpha = \max\left(0.1\frac{\mathrm{tr}(G)}{n}, 10^{-12}\right), \qquad C = (G + \alpha I_n)^{-1} Y_c. \tag{3}$$

For evaluation features X_{eval}, the scores are

$$\widehat{Y}_{\mathrm{eval}} = \frac{(X_{\mathrm{eval}} - \mu_X)X_c^\top}{d} C + \mu_Y, \tag{4}$$

where both means use training users only. No sigmoid, class reweighting, or hyperparameter search is used. Equal probe budgets refer to identical users and regularization rules, not identical numerical values of α.

Matched controls. The LoRA ablation disables only the Direct adapter while preserving learned queries and exact tokenized histories. The structure control standardizes 14 dialogue length, turn-count, and formatting variables using training statistics. For concatenation, the centered state and standardized structure blocks are each divided by their training root-mean-square vector energy. The 14-dimensional control and 1,038-dimensional concatenation test incremental prediction, not an equal-dimensional random-feature comparison. For alignment, a ridge map is fitted from centered Rolling to centered Direct training states with the same regularization rule. Its control permutes the Direct target rows once (seed 61911), retaining all other fitting choices. Both mapped evaluation states use the same fixed Direct-trained probe; evaluation users are never used to fit the maps. The shuffled-correspondence control uses the fixed seed 61911 for reproducibility.

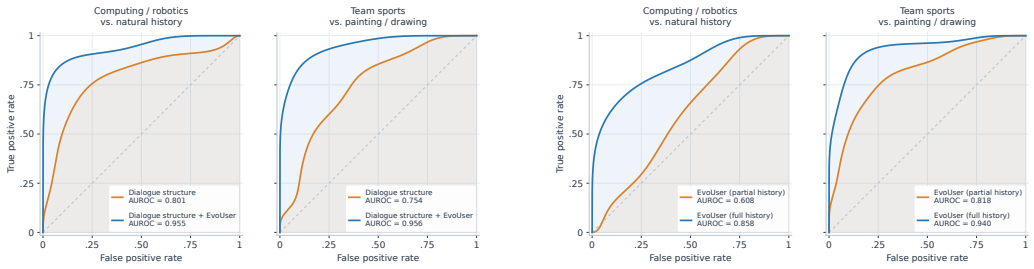
History stages. Rolling processes the retained history chronologically in chunks of at most 5,000 tokens, starting from its learned initial state and applying the checkpoint's residual update with scale 0.1. With K chunks, the nominal fraction $f \in \{0.25, 0.5, 0.75\}$ selects the state after chunk $\lceil fK \rceil$; full history selects chunk K. Fractions are therefore not exact token quantiles, and stages can coincide for short histories. Separate ridge probes are trained and evaluated at each matching stage. Only final minus early is the formal growth contrast; adjacent changes are descriptive.

Uncertainty and interpretation. We use 2,000 bootstrap draws of evaluation persona-profile blocks (seed 20260911), shared across conditions. Within each endpoint, the 95th percentile of the maximum absolute standardized bootstrap deviation over the original 34 numerical contrasts determines simultaneous intervals. Selecting four analyses does not reduce this correction family. Intervals condition on the fitted probes and frozen representations. This protocol targets cross-user representation readout, while end-to-end response quality is measured separately by the main PersonaMem-v2 and PrefEval evaluations.

C Additional Results

C.1 Supplementary Profile-Label Readout Analyses

The main text focuses on learned adaptation and user-matched transfer. Here we report two additional analyses that test dialogue-structure confounds and the sensitivity of profile-label readout to the amount of observed history.



(a) Structure control (b) History-stage comparison

Figure 3: **Supplementary profile-label readout analyses.** ROC curves compare (a) dialogue-structure features with and without state and (b) Rolling states after the first 25% versus the full history.

Controlling for measured dialogue structure. Adding state to 14 length, turn-count, and formatting features raises primary AUROC from 0.7776 to 0.9553 (Figure 3a), an observed gain of 0.1777. After simultaneous correction, we report this contrast as descriptive and use the user-correspondence controls below as the stronger test of user-specific structure.

Readout at different history stages. Separate equal-budget, stage-matched Rolling probes yield primary AUROCs 0.7130, 0.8491, 0.8741, and 0.8990 at nominal 25%, 50%, 75%, and full history (Figure 3b). The final-minus-early gain is 0.1860. The point estimates increase across stages; under the simultaneous correction, this pattern is reported as a descriptive history-stage trend.

C.2 Robustness Across Probe-Training Budgets

Learned adaptation improves profile-label readout consistently across training budgets and user selections. With the full probe-training set and the base, histories, and queries fixed, LoRA raises primary AUROC from 0.6767 to 0.9546: gain 0.2779, simultaneous 95% CI $[0.0004, 0.5555]$.

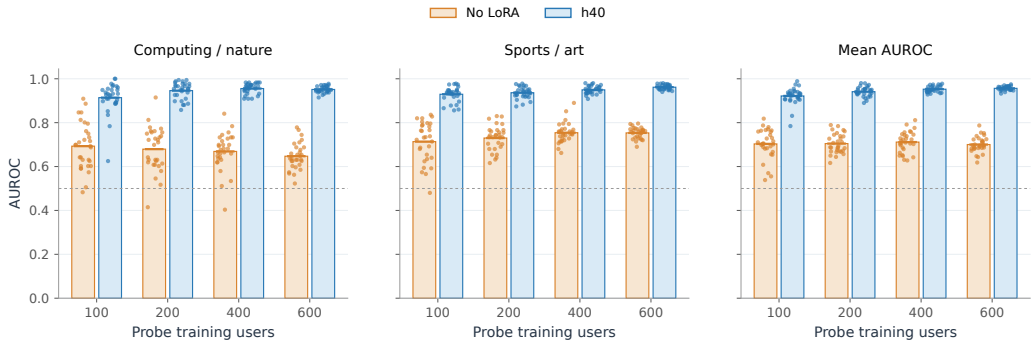


Figure 4: **Readout across probe-training budgets.** Bars show mean AUROC over 30 matched training-user subsets per budget; dots show individual fits for both preference axes and their mean. The learned state achieves higher mean AUROC in all 120 matched fits.

For each of 30 fixed seeds, we construct nested subsets of 100, 200, 400, and 600 training users and fit matched learned-state and no-LoRA probes. Figure 4 shows learned-state mean AUROCs of 0.9217, 0.9413, 0.9525, and 0.9565 across these budgets, compared with 0.7033, 0.7047, 0.7116, and 0.7003 without LoRA. All 120 paired comparisons have positive gains; mean gains by budget are 0.2184, 0.2366, 0.2409, and 0.2562. The paired scatter view appears in Figure 6. These paired repetitions quantify sensitivity to training-user selection on a fixed evaluation cohort and consistently favor the learned representation.

C.3 Additional Controls: Robustness of Profile-Label Readout

We add training-label permutation and progressively corrupted Direct/Rolling user correspondence. The same frozen states, 683-user training pool, and 294-user evaluation set are reused; the two axes still cover 70 distinct evaluation users. Only probes or linear maps are refitted, keeping encoders, updaters, and the evaluation cohort fixed to isolate each control. Subset and corruption distributions characterize robustness, while the label-permutation tests use a separate two-test Holm correction, not the original 34-contrast bootstrap correction. Additional settings appear in Appendix C.4; Figure 6 provides an alternative paired scatter view of the preceding budget analysis.

Profile-block training-label permutation. We jointly permute the four labels underlying the two axes across training profile blocks of equal size, sharing 499 permutations between representations and keeping evaluation labels fixed (Figure 5). The plot shows axis-specific AUROCs with marginal histograms (bin width 0.05) and observed-score reference lines. The learned state's true-label two-axis AUROC is 0.9546, versus a permutation mean of 0.4985; no-LoRA scores are 0.6767 and 0.5068, respectively. One-sided plus-one permutation p-values are 0.002 and 0.010, or 0.004 and 0.010 after Holm correction over the two mean-AUROC tests. Both representations therefore show training-label-associated readout under the assumed exchangeability of equal-size profile blocks. These tests isolate label-associated readout within each representation, while the paired analyses above quantify the learned-state advantage directly.

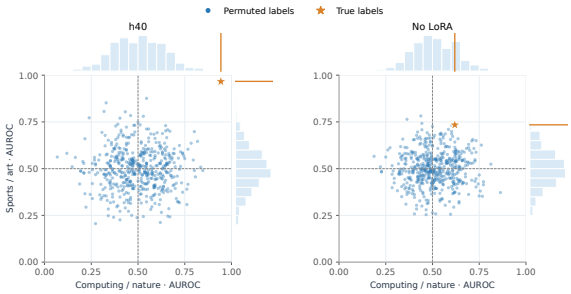


Figure 5: **Training-label permutation controls.** Dots show AUROCs from 499 profile-block permutations; stars mark true-label fits.

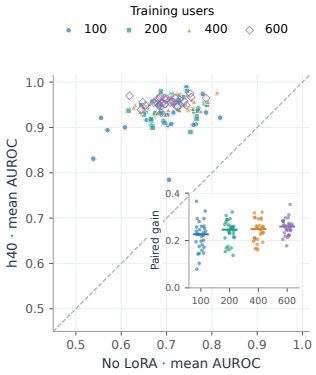


Figure 6: **Paired subset readout.** Each point compares learned and no-LoRA states on the same training subset; all 120 lie above equality.

Sensitivity to correct training-user correspondence. We retain nominally 0%, 25%, 50%, 75%, or 100% of correct Rolling-to-Direct training-user pairs and derange the remaining assignments, preventing accidental correct matches. Each corrupted level uses 30 maps; the fully correct map is fitted once. Applying the same fixed Direct probe, mean two-axis AUROC is 0.4899, 0.7736, 0.8743, 0.9127, and 0.9203, respectively (Figure 7). The rising means show sensitivity to the amount of correct user correspondence, extending the earlier single shuffled-pair control. The aggregate pattern provides a graded correspondence test rather than relying on a single shuffled endpoint.

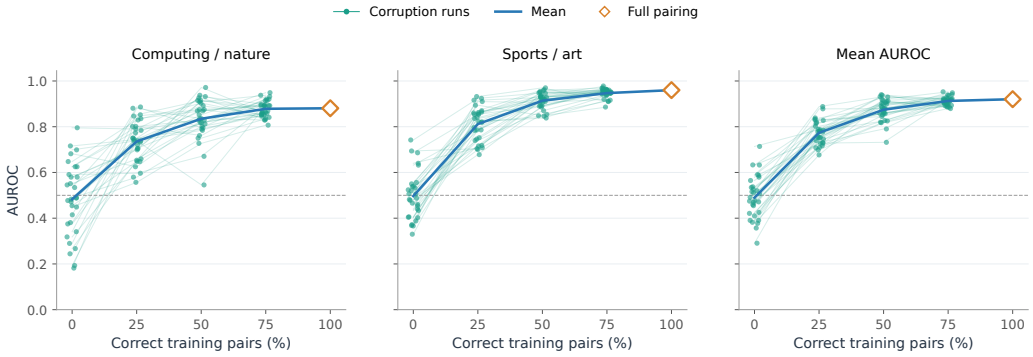


Figure 7: **Readout under user-pairing corruption.** AUROC across 30 maps per corruption level, with lines connecting matched runs and means. Diamonds mark the deterministic fully paired fit.

Together, these controls strengthen the evidence for readable profile-label information across variations in training-user selection, label assignment, and cross-representation user correspondence.

C.4 Additional Frozen-State Controls and Paired Readout

Shared inputs and fitting. The three new controls reuse existing matched-history Direct, no-LoRA, and Rolling states, each mean-pooled to 1,024 dimensions. No histories, states, or encoder/updater training are added. The 683 training users form 676 persona-profile groups: 669 singletons and seven two-user groups. The two evaluation axes contain 27 observations (11 computing/robotics, 16 natural history) and 45 observations (25 team sports, 20 painting/drawing), covering 70 distinct users. Only the four relevant output labels are needed because the multi-output ridge rule fits targets independently. All centering and regularization use training data only, following Appendix B.3. Full-data reference predictions reproduce the cached readouts to numerical precision.

Nested training-user subsets. Each of 30 fixed-seed permutations supplies nested subsets of 100, 200, 400, and 600 users, shared across representations. Feature and label centering and ridge regularization are refitted for each subset and representation. Sampling is at the user level, not the profile-block level. All 240 perturbed fits are retained, including small subsets with no positive examples for a selected label. The full 683-user reference is fitted once per representation. Figure 6 shows the same 120 paired comparisons as the training-budget experiment in a paired scatter view.

Profile-block label permutations. For 499 draws (seed base 2026091203), singleton profile blocks are permuted among themselves and two-user blocks among themselves. Each block's four labels move jointly with fixed within-block user order. Both representations share permutations; features, regularization, and evaluation labels remain fixed. A cached linear operator is algebraically equivalent to refitting ridge for each label matrix, giving 998 perturbed fits. The one-sided p-value is $(1 + N_{\text{null}\geq\text{observed}})/500$, where the count compares mean two-axis AUROC. Holm correction covers only the two representation-specific mean-AUROC tests; individual-axis tests are descriptive. Minimum raw p is 0.002. Inference assumes exchangeability within equal-size profile strata; paired representation comparisons are reported in the training-budget analysis.

Correspondence corruption. Nominal correct fractions of 0%, 25%, 50%, 75%, and 100% retain 0, 171, 342, 512, and 683 correct pairs. Within each of 30 seeds, correct sets are nested and the remaining assignment is a bijection with no fixed points. Each centered Rolling-to-Direct map uses the same ridge rule and is evaluated through a fixed, label-supervised Direct probe. Projecting the targets through this probe before solving is algebraically equivalent to fitting the full 1024×1024 map. There are 30 draws at each corrupted level and one full-pairing reference, totaling 121 maps. Maps use paired identities but not preference labels; the probe remains supervised. Replicate spread is sensitivity to corruption, not a population confidence interval. These controls use their stated descriptive or permutation-based inference procedures.