DOES EVERY USER NEED A PRIVATE LORA? DECOUPLING PERSONALIZATION FROM PER-USER ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Personalized large language models often require a complete adaptation state for each user. However, this paradigm scales poorly as the user population grows. We revisit this design through the lens of *personalization capacity allocation*: how much adaptation capacity can be shared across users, how that shared capacity should be composed, and how much adaptation must remain user-specific. We answer these questions through three complementary empirical analyses. We find that independent user adapters contain substantial cross-user reusable structure, that the utility of reusable directions reflects both user relevance and variation across queries, and that user histories provide transferable signals for compact individual correction. Motivated by these findings, we propose **LINEUP**. It learns a bank of reusable low-rank personalization factors, composes them through user-conditioned recall and query-dependent calibration, and restricts target-user adaptation to a tiny user code over a shared correction space. This design decouples expressive personalization capacity from per-user trainable state. Each target user optimizes only eight scalars, while all shared components remain fixed. By comparison, the evaluated private-LoRA configuration uses 4.19 million per-user parameters. Our theoretical analysis gives a finite-step, finite-history risk bound and sufficient conditions for user-code refinement to improve on history initialization. Across six tasks spanning personalized classification, prediction, and generation, LINEUP leads on all 12 metrics, each averaged over three independent runs (e.g., reducing LaMP-3 RMSE by 11.4% relative to the strongest baseline). It maintains advantages under limited history. These results show that rich personalization can be supported primarily by reusable, conditionally composed shared capacity, while independent user adaptation remains confined to a tiny correction state.

1 INTRODUCTION

Large language models (LLMs) can acquire task-specific knowledge and behaviors through adaptation (Chung et al., 2024; Wang et al., 2025). However, such adaptation is typically learned from data pooled across users and produces a single model shared by the population. It captures what users have in common within a task, but not how their preferences differ. For example, the same input may require different outputs for different users. Personalized Large Language Models (PLLMs) therefore seek to adapt shared models to individual preferences inferred from user histories or feedback (Liu et al., 2025). Recently, PLLMs have received attention in scenarios across text generation, conversation, multimodal generation, and recommender systems (Trajanovski et al., 2021; Zhao et al., 2025b; Xu et al., 2025; Huang et al., 2026).

Existing PLLM methods broadly differ in how user information is converted into personalized behavior. Tuning-free approaches incorporate user histories through summaries, retrieval, or inference-time steering (Richardson et al., 2023; Salemi et al., 2024a; Lv et al., 2026). Their effectiveness depends on the quality of history-derived signals. Learned adaptation-based methods instead encode personalization into parameters or representations. One direction performs **individualized adaptation construction**: OPPU directly optimizes a user-specific PEFT module (Tan et al., 2024b), PerFit retains a personalized representation intervention alongside a collective shift (Liu et al., 2026a), and

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.

P2P uses a population-trained hypernetwork to generate complete personalized adapters from user information (Tan et al., 2026). Another direction improves reuse through **user-derived reusable components**, such as pieces, group-level modules, or anchor-derived PEFT modules (Tan et al., 2024a; Zhang et al., 2025b; Li et al., 2026). These methods improve how individualized adaptation is obtained or reused, but largely retain user-associated adaptation modules as the starting point for organizing personalization. A detailed review of related work is provided in Appendix A.

We instead view personalized adaptation through the lens of *personalization capacity allocation*. Rather than starting from a complete individualized adaptation state, we ask what personalization capacity can be learned directly for cross-user reuse, how this shared capacity should be composed for each request, and how much still must be independently estimated for a target user. This leads to the central question of our work: **How much of a personalized adapter is actually personal?**

We investigate this question through three complementary empirical analyses. **1.** Independently learned user adapters contain substantial cross-user reusable structure (Figure 1(a)). **2.** The functional utility of reusable directions exhibits both persistent user association (Figure 1(b)) and substantial variation across queries from the same user. **3.** User histories provide transferable individual correction signals (Figure 1(c)) that can be preferentially retained in compact shared spaces. Together, these findings suggest organizing personalization around shared capacity, conditional composition, and compact individual correction.

Motivated by these findings, we propose **LINEUP**, which decouples personalization capacity from per-user adaptation. LINEUP directly learns fine-grained low-rank personalization factors across all source-training users as reusable population-level capacity, rather than deriving reusable components from independently learned user adapters. A target user's history recalls a set of relevant factors. Each query from the same user then calibrates their sparse composition. Remaining individual variation is handled by shared correction maps controlled by a single eight-dimensional user code. The code is initialized from the user's history and refined using history-derived support examples while all shared components remain fixed. Therefore, rich personalization capacity is learned predominantly through reusable shared structure, while target-user optimization is confined to **only eight scalars**. By comparison, the evaluated OPPU configuration uses 4.19 million per-user parameters. To characterize this compact adaptation, we derive a finite-step, finite-history risk bound for user-code refinement, linking shared-space approximation and history initialization to fitting progress and support–query disagreement.

Across six tasks spanning personalized classification, prediction, and generation, LINEUP leads on all 12 reported metrics, with each reported score averaged over three independent runs. Under cold-start settings, LINEUP maintains clear advantages as the available user history is limited. The scalability analysis shows low incremental per-user storage as the user population grows.

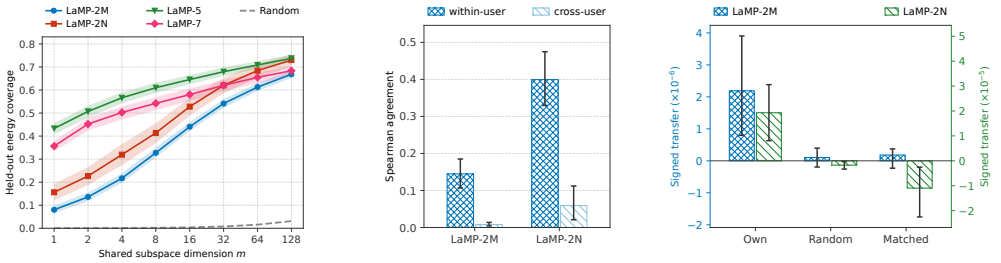
In summary, our main contributions are as follows:

- We systematically study personalized adaptation from a personalization capacity allocation perspective and identify three properties: cross-user reuse, user- and query-dependent composition, and transferable individual correction.
- We derive a finite-step, finite-history risk bound for regularized user-code refinement and establish sufficient conditions for improvement over history initialization.
- We propose **LINEUP**, which decouples personalization capacity from per-user adaptation. It achieves the best mean performance on various personalization tasks while reducing parameter and storage growth, with further benefits under cold-start settings.

2 Problem Setup and Preliminaries

2.1 Personalization Problem Setup

We study each personalized task $\mathcal{T}$ independently, with input space $\mathcal{X}_\mathcal{T}$ and output space $\mathcal{Y}_\mathcal{T}$. Let $\mathcal{U}_{\text{src}}$ and $\mathcal{U}_{\text{tgt}}$ denote disjoint sets of source and target users. Source users provide data for learning population-shared personalization components, while target users are unseen during shared training and are personalized only from their own available history.



(a) Shared-subspace coverage (b) Rank agreement (c) Support-to-query transfer

Figure 1: Empirical evidence for personalization capacity allocation. (a) Small shared subspaces capture substantial held-out adapter-update energy. (b) Direction-utility rankings are substantially more consistent within users than across users. (c) A user's own history yields positive transferable correction, unlike random or label-matched donor histories (separate y-axis scales).

Each user u has an available history $\mathcal{H}_u = \{h_{u,i}\}_{i=1}^{n_u}$ and held-out queries $\mathcal{Q}_u = \{(x_{u,j}, y_{u,j})\}_{j=1}^{m_u}$. Following LaMP and OPPU (Salemi et al., 2024b; Tan et al., 2024b), history records are input–output pairs. Personalization may use $\mathcal{H}_u$ and the current query x, but never its target output y.

Let $\theta_{\mathcal{T}}$ denote the task-adapted LLM backbone, $\phi_{\mathcal{T}}$ the population-shared personalization parameters, and ω_u the target-user-specific adaptation state. We write personalized prediction abstractly as $p_u(y \mid x, \mathcal{H}_u) = p_{\theta_{\mathcal{T}},\phi_{\mathcal{T}}}(y \mid x, \mathcal{H}_u, \omega_u)$. This notation separates shared personalization capacity from the state that must be independently adapted for a target user, while allowing the available history and current query to directly condition prediction.

When target-user optimization is required, task-specific support examples $\mathcal{S}_u$ are constructed from the available history. We denote by $n_u^{\text{sup}} = |\mathcal{S}_u|$ the number of support examples used for fitting. It need not equal the number of history records used for history conditioning. Under limited-history evaluation, all target-user-dependent operations only use the currently available history, including history conditioning and support construction. The held-out evaluation queries remain fixed.

2.2 Adaptation Preliminaries

Parameter-space adaptation. LoRA (Hu et al., 2021) augments a frozen linear transformation with a low-rank update. At an adapted site $i \in \mathcal{I}$, a user-specific LoRA induces the effective update $D_{u,i} = s_i B_{u,i} A_{u,i}$, where $A_{u,i} \in \mathbb{R}^{r_{\text{L}} \times d_{\text{in},i}}$, $B_{u,i} \in \mathbb{R}^{d_{\text{out},i} \times r_{\text{L}}}$, $r_{\text{L}} \ll \min(d_{\text{in},i}, d_{\text{out},i})$, and s_i is a scaling coefficient. OPPU independently fits these user-specific updates on a common task-adapted LLM backbone, after which the same fitted update is reused across all that user's queries. Our parameter-space analyses operate on the effective updates $D_{u,i}$, rather than on a particular factorization into $A_{u,i}$ and $B_{u,i}$.

Representation-space correction. We also study personalization directly in representation space. Let $h_{\ell,t} \in \mathbb{R}^d$ denote the hidden representation at layer ℓ and token position t. An additive intervention takes the form $h'_{\ell,t} = h_{\ell,t} + a_{\ell,t} v_\ell$, where $v_\ell \in \mathbb{R}^d$ is a correction vector and $a_{\ell,t}$ specifies its positional scope. For a fixed reference model and example z, let $\mathcal{L}_{\mathcal{T}}(z; v_\ell)$ denote the negative log-likelihood under this intervention, and define the first-order loss-reducing direction as $g_z^\ell = -\nabla_{v_\ell} \mathcal{L}_{\mathcal{T}}(z; v_\ell)|_{v_\ell=0}$. Because v_ℓ is unrestricted, this view provides a diagnostic of the representation-space correction indicated by the example's loss gradient, before imposing a low-dimensional user parameterization. Section 3 specifies the concrete intervention and normalization used in the corresponding empirical analysis.

3 Empirical Analysis of Personalized Adaptation

We examine how personalization capacity can be shared, composed, and individually corrected. RQ1 and RQ2 analyze independently optimized OPPU adapters (Tan et al., 2024b) on a fixed LLM backbone; RQ3 studies representation corrections without user adapters. Unless stated otherwise, intervals are 95% user-level bootstrap confidence intervals. See Appendix B for detailed mathemat-

ical definitions and supporting evidence. Dataset details are provided in Section 5.1. The design choices informed by these analyses are evaluated through ablations in Section 5.5.

3.1 RQ1: How Much Structure Is Reusable across User Adapters?

> **Observation 1 (Sharing).** Independently trained user adapters share substantial structure: a small subspace learned from other users captures much of held-out update energy.

We here learn common output directions from one group of users and test how much of other users' adapter-update energy they capture. We compare these directions with random subspaces of the same dimension on LaMP-2M, LaMP-2N, LaMP-5, and LaMP-7 datasets.

Figure 1(a) shows that 32 directions (0.8% of the output dimension) capture 54.1%–67.9% of held-out update energy, versus 0.78% for random subspaces. Removing task-adapter output directions barely changes coverage. Size-matched random-partition bases also capture substantial energy, but genuine-user bases cover 14.3–21.0 percentage points more on the same updates (Appendix B.1). These findings motivate jointly learning reusable personalization factors rather than duplicating shared structure in user adapters.

3.2 RQ2: How Does Direction Utility Vary across Users and Queries?

> **Observation 2 (Composing).** The utility of reusable directions exhibits persistent user-associated structure and substantial variation across queries from the same user.

Are reusable directions equally useful for every request? On LaMP-2M and LaMP-2N, we keep each user's adapter fixed and measure the first-order change in query loss induced by strengthening each of eight shared directions. Comparing these signed responses within and across users tests whether their utility reflects recurring user patterns, request-specific needs, or both.

Figure 1(b) shows higher within-user than cross-user Spearman agreement: 0.145 versus 0.008 on LaMP-2M and 0.400 versus 0.059 on LaMP-2N. This within-user advantage holds across all analyzed layers and projections (Appendix B.2.2). Yet differences between a user's queries still account for 82.7% and 61.3% of response variation after controlling for overall sensitivity scale (Appendix B.2.3). Direction-removal checks further show that the sensitivities closely track actual loss changes (Appendix B.2.4). Together, these results suggest a hierarchical technique where history-conditioned candidate recall is followed by query-dependent calibration.

3.3 RQ3: Can User History Guide Compact Individual Correction?

> **Observation 3 (Correcting).** User histories provide transferable correction after removing average label-response directions, and compact shared spaces preferentially retain it.

We now ask whether correction inferred from a user's history transfers to new queries, measured by alignment with their first-order loss-reducing directions. On LaMP-2M and LaMP-2N, we control shared label structure by removing the span of average label responses from both history and query correction signals. We compare own history with random and label-matched donor histories and evaluate compact spaces learned only from other users' histories (Appendix B.3).

Figure 1(c) shows positive own-history transfer, with intervals excluding zero and means above those of both donor conditions. Own-history point estimates are positive at all eight analyzed layers on both tasks (Appendix B.3.2). First-order intervention checks support the loss-based interpretation of this alignment (Appendix B.3.4). An eight-dimensional space per layer retains 34.7% and 29.4% of full-space own-history transfer on LaMP-2M and LaMP-2N, respectively, while capturing only 12.9% and 17.0% of residual query-correction energy (Figure 6(a)). Compact spaces therefore preserve a larger fraction of useful transfer than of correction energy. This motivates shared correction maps with compact, history-fitted user coordinates, complementing conditional factor composition.

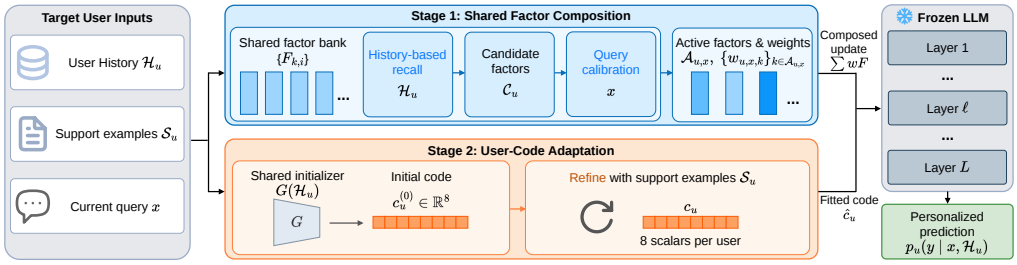


Figure 2: Overview of LINEUP. Stage 1 combines shared personalization factors (Section 4.1) through history-conditioned recall and query-dependent calibration (Section 4.2). Stage 2 initializes an eight-dimensional user code from history and refines only this code on support examples, with Stage 1 active and all shared components fixed (Section 4.3).

4 METHODOLOGY: LINEUP

Motivated by findings in Section 3, we propose **LINEUP** (**LI**ghtweight perso**N**alization via r**EU**sable factors and tiny **P**er-user codes), which decouples personalization capacity from per-user adaptation. LINEUP comprises two stages (S1 and S2). Figure 2 illustrates the architecture.

LINEUP **[Algorithm 1]**

Stage 1 (S1): Shared-factor composition. User history recalls candidate factors, and the current query calibrates their sparse composition. Let $\mathcal{A}_{u,x}$ denote the active factors and $w_{u,x,k}$ their mixture weights. At each adapted linear map i, shared factors $F_{k,i}$ yield

$$D_{u,x,i} = \sum_{k\in\mathcal{A}_{u,x}} w_{u,x,k} F_{k,i}. \tag{1}$$

Stage 2 (S2): Tiny user-code adaptation. We refine a history-initialized code on support examples from the user history, with Stage 1 active and all shared parameters fixed. The fitted code $\hat{c}_u$ controls shared correction maps V_ℓ through

$$h'_{\ell,t} = h_{\ell,t} + a_{\ell,t} V_\ell \hat{c}_u. \tag{2}$$

Here $h_{\ell,t}$ is the hidden state at layer ℓ and token t, and $a_{\ell,t}$ selects correction positions. One eight-dimensional code is reused across correction layers and the user's queries.

4.1 FINE-GRAINED SHARED PERSONALIZATION FACTORS

Observation 1 motivates changing the unit of personalization from complete user adapters to reusable factors. LINEUP jointly learns its factor bank across all source-training users under conditional composition, rather than extracting components from separately trained user adapters. Each factor can serve multiple users without representing any one user in full. This allows factors to specialize as complementary personalization units, while overlapping subsets produce distinct personalized updates (Appendix C.2).

4.2 HIERARCHICAL HISTORY–QUERY COMPOSITION

Observation 2 suggests separating user relevance from request-specific utility. Let $\tilde{e}_u, \tilde{e}_x \in \mathbb{R}^{d_f}$ denote fixed features of the available history and current query, obtained from frozen token embeddings (Appendix C.2). For a bank of K factors, learned shared maps $W_H, W_Q \in \mathbb{R}^{K\times d_f}$ implement history-conditioned recall followed by query calibration:

$$\begin{aligned} s_u &= W_H \tilde{e}_u, & \mathcal{C}_u &= \mathrm{TopK}(s_u, M_c), \\ z_{u,x} &= s_u + \lambda W_Q \tilde{e}_x, & \mathcal{A}_{u,x} &= \mathrm{TopK}_{\mathcal{C}_u}(z_{u,x}, M_a). \end{aligned} \tag{3}$$

Here M_c and M_a are the candidate and active set sizes, with $M_a \leq M_c < K$, and λ controls query strength. TopK returns factor indices, and its subscript restricts selection to $\mathcal{C}_u$. History scores guide both candidate recall and query-calibrated composition. Softmax over active scores gives the mixture weights $w_{u,x,k}$ in Equation 1, with zero weights outside $\mathcal{A}_{u,x}$. When $M_a = M_c$, query calibration reweights the recalled factors; when $M_a < M_c$, it can also change the active subset. S1 provides request-dependent personalization for unseen users without target-user gradient updates.

4.3 History-Initialized Adaptation with Tiny User Codes

Different users can draw on the same personalization factors while requiring different combinations of them. LINEUP therefore places its main personalization capacity in shared factor composition, while a tiny user code addresses individual differences that this composition does not fully capture. Observation 3 motivates sharing correction capabilities while adapting only their user-specific coordinates. Shared maps learn how to correct representations. A single eight-dimensional code specifies the user's correction. It is initialized from a population-informed starting point, and support fitting adjusts it using individual evidence. The same fitted code coordinates corrections across layers and is reused across queries, rather than introducing independently learned user matrices (Appendix C.3).

Learning shared capacity. We first train S1 across source users, then freeze it and learn the shared correction maps and initializer using initialized codes. S1 remains active during both shared S2 training and target-user code fitting. At personalization time, only the code is updated. The LLM backbone remains frozen throughout (Algorithm 1).

4.4 Theoretical Analysis of Compact User Adaptation

We analyze S2's regularized, finite-step Adam fitting with S1 active and all shared parameters fixed. Under the conditions in Appendix D, our risk bound relates shared-space approximation and history-initialization error to fitting progress and support–query disagreement. It gives a sufficient condition for support fitting to improve on initialization, while accounting for history reuse (Corollary 1).

A deterministic bound on code displacement supports this finite-step analysis without requiring convergence (Lemma 1). The risk bound separates shared-space approximation from code-space estimation, while accounting for shared geometry and stability. A complementary quadratic analysis explains when remaining correction need outweighs estimation noise (Corollary 2). Together, these results support learning rich correction capabilities jointly while keeping individual fitting compact.

5 Experiments

5.1 Experimental Setup

Datasets and metrics. We evaluate LINEUP on six tasks of the LaMP personalization benchmark (Salemi et al., 2024b), spanning classification, rating prediction, and generation. We follow OPPU's data processing and target-user protocol, keeping source and target users disjoint. We report Accuracy and Macro-F1 for LaMP-2M/2N, mean absolute error (MAE) and root mean square error (RMSE) for LaMP-3, and ROUGE-1/ROUGE-L for generation. Lower is better for MAE and RMSE; higher is better otherwise. Dataset and metric details appear in Appendix E.1–E.2.

Baselines. We compare LINEUP against eight baselines. **Non** uses the pretrained LLM backbone without task adaptation or personalization. **Basic** applies task adaptation without per-user personalization. **RAG** augments the task-adapted model's input with retrieved user history. **OPPU** (Tan et al., 2024b) learns a private adapter for each user, while **PerFit** (Liu et al., 2026a) learns collective and user-specific representation shifts. **Per-Pcs** (Tan et al., 2024a) assembles reusable pieces of user adapters. **P2P** (Tan et al., 2026) trains a shared hypernetwork across source users to generate adapters from user profiles and retrieved history. **MTA** (Li et al., 2026) merges anchor-user adapters before individual refinement. Details appear in Appendix E.3.

Implementation details. We use Llama-2-7B as the base model to align with most baselines' released code. LINEUP builds on OPPU's non-personalized task-adapted backbone and learns shared components separately. We report mean $\pm$ standard deviation across three seeds. Implementation details and hyperparameters are provided in Appendix E.4–E.5.

Table 1: Performance on LaMP-2M/2N/3. **Acc** = Accuracy, **F1** = Macro-F1, **MAE** = mean absolute error, **RMSE** = root mean square error. Underline and **bold** denote the second-best and best mean results. Δ reports absolute improvements over the strongest baseline, with error reductions shown as positive. LINEUP leads on all metrics across classification and prediction.

	LaMP-2M		LaMP-2N		LaMP-3	
Method	**Acc** ↑	**F1** ↑	**Acc** ↑	**F1** ↑	**MAE** ↓	**RMSE** ↓
Non	.032 ± .001	.035 ± .002	.355 ± .002	.120 ± .006	1.342 ± .016	2.005 ± .022
Basic	.470 ± .001	.423 ± .001	.783 ± .002	.536 ± .002	.221 ± .009	.491 ± .009
RAG	.552 ± .000	.479 ± .002	.817 ± .001	.594 ± .001	.216 ± .009	.495 ± .009
OPPU	.619 ± .000	.515 ± .001	.801 ± .001	.556 ± .002	.181 ± .009	.441 ± .011
PerFit	.632 ± .000	.519 ± .001	.818 ± .000	.589 ± .002	.172 ± .000	.440 ± .000
Per-Pcs	.503 ± .001	.458 ± .002	NA	NA	.217 ± .000	.490 ± .000
P2P	.521 ± .005	.472 ± .006	.804 ± .003	.563 ± .004	.215 ± .005	.488 ± .004
MTA	.526 ± .005	.478 ± .005	NA	NA	.204 ± .005	.468 ± .004
LINEUP	**.645** ± .003	**.539** ± .002	**.822** ± .005	**.621** ± .003	**.152** ± .003	**.390** ± .003
Δ	**+0.013**	**+0.020**	**+0.004**	**+0.027**	**+0.020**	**+0.050**

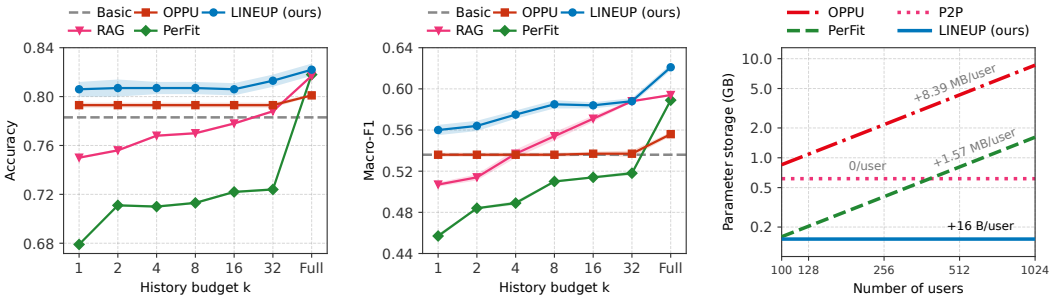
Table 2: Performance on LaMP-4/5/7. **R-1** = ROUGE-1, **R-L** = ROUGE-L. Underline and **bold** denote the second-best and best mean results. Δ reports absolute improvements over the strongest baseline. LINEUP leads on all metrics across generation tasks.

	LaMP-4		LaMP-5		LaMP-7	
Method	**R-1** ↑	**R-L** ↑	**R-1** ↑	**R-L** ↑	**R-1** ↑	**R-L** ↑
Non	.137 ± .000	.120 ± .000	.156 ± .015	.128 ± .012	.131 ± .001	.127 ± .001
Basic	.186 ± .000	.161 ± .001	.477 ± .006	.413 ± .007	.520 ± .004	.471 ± .006
RAG	.198 ± .001	.175 ± .000	.504 ± .006	.444 ± .004	.561 ± .004	.514 ± .002
OPPU	.216 ± .000	.192 ± .001	.535 ± .011	.466 ± .010	.564 ± .003	.522 ± .003
PerFit	.207 ± .001	.183 ± .001	.524 ± .001	.446 ± .002	.526 ± .007	.474 ± .008
Per-Pcs	.187 ± .001	.174 ± .000	.482 ± .004	.427 ± .005	.522 ± .001	.473 ± .001
P2P	.213 ± .001	.190 ± .001	.518 ± .009	.453 ± .009	.548 ± .005	.499 ± .002
MTA	.218 ± .001	.195 ± .000	.537 ± .007	.475 ± .005	NA	NA
LINEUP	**.222** ± .001	**.197** ± .001	**.564** ± .003	**.494** ± .004	**.594** ± .004	**.551** ± .006
Δ	**+0.004**	**+0.002**	**+0.027**	**+0.019**	**+0.030**	**+0.029**

5.2 Main Results on LaMP

Tables 1 and 2 show that **LINEUP achieves the best mean performance on all 12 metrics across six tasks**. Representative gains include a 0.020 F1 improvement over PerFit on LaMP-2M, a 0.050 RMSE reduction on LaMP-3, and a 0.029 improvement in ROUGE-L over OPPU on LaMP-7. While the strongest baseline varies across tasks, LINEUP consistently leads across tasks.

Effectively organizing shared capacity around reusable factors. OPPU and PerFit retain individual adaptation states, while Per-Pcs and MTA reuse components derived from user adapters. Our comparison also includes cross-user, history-conditioned learning through P2P's population-trained hypernetwork. RQ 1 indicates that separate states repeatedly devote limited capacity to common structure. LINEUP jointly learns factors for user- and query-conditioned composition, allowing overlapping subsets to serve different users and requests without retraining. Its distinction lies in organizing shared capacity around reusable factors while restricting independent adaptation to compact correction. The consistent gains support this factor-centric capacity-allocation strategy as a whole, combining shared learning, conditional composition, and compact individual refinement.



(a) LaMP-2N Accuracy (b) LaMP-2N Macro-F1 (c) LaMP-2N storage

Figure 3: Cold-start personalization and adaptation-storage scaling. (a) LINEUP maintains high classification accuracy. (b) Additional history improves Macro-F1 from an already strong cold-start result. (c) LINEUP has the lowest total adaptation storage over the displayed user range.

Reserving individual adaptation for compact correction. RQ 3 reveals that user histories provide transferable correction signals that compact shared spaces can preferentially retain. Section 4.4 shows how finite-step fitting can improve history initialization when its progress outweighs support–query disagreement, with statistical complexity controlled in the user-code space. LINEUP therefore places most modeling capacity in shared structures and reserves a lightweight code for targeted individual correction. The results support this division of labor: shared capacity and compact individual correction together yield stronger personalization without a full private adaptation state per user.

5.3 Cold-Start Personalization

Cold-start scenarios are common in real-world PLLM deployment, where personalization must begin with limited user history. We therefore evaluate how effectively LINEUP personalizes unseen target users as the amount of available history is restricted. We vary the available history of unseen target users while keeping users and evaluation queries fixed. On LaMP-2N, LINEUP achieves the best mean Accuracy and the best or tied-best reported Macro-F1 across history budgets (Figure 3(a–b)). With only one history record, it reaches 0.806 Accuracy and 0.560 Macro-F1, compared with OPPU's $0.793/0.536$ at the same budget. Its Macro-F1 increases to 0.621 with full history, showing that strong cold-start performance still benefits from additional user evidence. The detailed protocol and complementary generation task results on LaMP-4 appear in Appendix F.2.

Rich shared capacity from minimal history. Our factor bank learns reusable personalization patterns across source users before target-user adaptation. Available history provides cues for recalling relevant factors, rather than requiring these patterns to be relearned for each user. It indicates that our shared factors already supply much of the needed personalization. The tiny user code then addresses residual individual differences not fully captured by factor composition. Concentrating independent estimation on this correction also aligns with the finite-history risk analysis in Section 4.4.

5.4 Efficiency and Scalability

Most benchmark tasks in our evaluation involve about 100 target users, whereas real-world PLLM services may serve far larger populations. Adaptation storage thereby becomes a vital constraint for multi-adapter deployment (Abdelzaher et al., 2025). Figure 3(c) shows analytical storage scaling for the evaluated LaMP-2N configurations, normalized to bf16: LINEUP adds only 16 B per user. LINEUP's total adaptation storage becomes lower than OPPU's at 17 users and PerFit's at 95 users. Both thresholds are very small for large-scale services, where LINEUP's storage savings over these baselines grow with the user population. At 1,024 users, the projected totals, including shared parameters, are approximately 151.2 MB for LINEUP, 1.61 GB for PerFit, and 8.60 GB for OPPU. Accounting details and comparisons with Per-Pcs and MTA appear in Appendix F.3.

Amortizing shared capacity across users. In OPPU, each private adapter must accommodate both reusable patterns and individual variation, while overlapping structure is repeatedly learned and stored across users (RQ 1). LINEUP instead invests in a larger factor bank jointly learned across source users, providing more capacity to capture diverse reusable personalization patterns. Only

Table 3: Ablation results on LaMP-2M and LaMP-5. **Bold** denotes the best mean.

	LaMP-2M		LaMP-5	
Method	**Acc** ↑	**F1** ↑	**R-1** ↑	**R-L** ↑
Full LINEUP (S1 + S2)	**.645** ± .003	**.539** ± .002	**.564** ± .003	**.494** ± .004
S1 + initialized code	.639 ± .002	.533 ± .001	.555 ± .000	.490 ± .001
S1 only	.634 ± .005	.530 ± .002	.549 ± .003	.485 ± .001
S1 w/o query calibration	.585 ± .008	.494 ± .007	.523 ± .002	.461 ± .002
S1 w/ random candidates	.497 ± .023	.419 ± .030	.492 ± .012	.436 ± .011

residual individual correction requires an eight-dimensional code. This organization explains how richer personalization and lower storage can coexist. Capacity is thus expanded where it can be shared, rather than replicated in every user module. MTA also reuses components but retains large individualized adapters, leaving substantial per-user storage growth (Figure 7(c)).

Compact per-request activation. History-conditioned recall and query calibration allow each request to draw on only a small part of the shared bank. Each request activates four rank-4 factors from the 32-factor bank, yielding a combined update of rank at most 16 at each adapted projection. The larger bank therefore expands the range of reusable personalization patterns without requiring its full capacity to be activated for every request.

5.5 Ablation Studies

We conduct an ablation study across representative tasks (LaMP-2M and LaMP-5) to validate our three design choices of LINEUP: history-guided factor retrieval, query-conditioned composition, and the tiny user-specific correction. The ablation protocol is provided in Appendix F.4. For the two routing ablations, S1-only is the reference. *Random candidates* replaces history-guided recall with a fixed random subset of the same size for each user, while preserving history- and query-based scoring within the candidate set. *No query calibration* removes the query contribution during training and inference while retaining history-conditioned recall. To examine S2, we consider three variants. *S1 only* omits the entire correction mechanism, while *S1 + initialized code* applies history-initialized correction without support fitting. Full LINEUP uses the same shared checkpoints and further refines the code on history-derived support examples.

Table 3 shows that random candidates cause the largest degradation relative to S1-only, while removing query calibration also reduces performance on both tasks. Full LINEUP outperforms S1-only, supporting S2 as a whole. On all four metrics, shared correction maps with history-based initialization improve mean performance over S1-only, and support fitting adds further gains. These results support both routing choices within the factor bank and refinement beyond initialized correction. A code-dimension sweep further shows that increasing r_c from 2 to 8 improves mean performance, while larger codes yield no consistent additional gains across metrics (Appendix F.5).

6 Conclusion and Future Work

We revisited personalized adaptation through the lens of personalization capacity allocation and conducted three empirical analyses. Our findings motivated LINEUP, which directly learns fine-grained reusable factors across users, composes them using user history and the current query, and confines user-specific fitting to a compact residual correction. Our theoretical analysis bounds the risk of finite-step code refinement under limited history. Across six LaMP tasks, LINEUP achieves the best mean performance while optimizing only eight scalars per target user, with clear advantages under limited history and low incremental parameter and storage costs. These results show that rich personalization need not require a full private adapter for every user. Shared learning and conditional composition provide rich personalization capacity, while independent user adaptation is reserved for compact individual correction. Future work will extend this task-isolated formulation to cross-task personalization, capturing stable user preferences through reusable factors.

AI USE STATEMENT

In this work, we used generative AI tools, including ChatGPT (https://chatgpt.com/) and Claude (https://claude.ai/), to assist with polishing the writing. Specifically, we used these tools to polish text, check grammar, and correct formatting errors throughout the writing process. Other tasks requiring disclosure, such as generating synthetic datasets or assisting with translation, are not applicable to this work, and we did not use generative AI tools for any other disclosure-required tasks. We have carefully reviewed all AI-assisted work to ensure there is no falsehood, plagiarism, or misrepresentation. We take responsibility for the final content of this work, including any text, claims or artifacts produced with the aid of generative AI. This use of LLMs complies with the ICLR 2027 Author Guidelines (https://iclr.cc/Conferences/2027/AuthorGuidelines) and AI Policy for Authors (https://iclr.cc/Conferences/2027/AIPolicyForAuthors).

ETHICS STATEMENT

Personalized large language models raise privacy, bias, and resource concerns because they rely on user-specific histories and adaptation states. First, LINEUP replaces complete private adapters with eight-dimensional user codes, substantially reducing the persistent user-specific model state that must be stored and protected. This compact design can simplify personalized-parameter management. User histories, codes, and shared components still require appropriate access controls and privacy safeguards. Second, history-based personalization may reflect biases or manipulation in user data. LINEUP's separation of shared components and individual codes offers a basis for targeted auditing. Output review and transparent user controls can help address undesirable personalization and support responsible use. Third, LLM training and inference entail computational and environmental costs. By reusing shared personalization capacity rather than maintaining a complete adapter for every user, LINEUP substantially reduces per-user parameter and storage requirements, supporting more resource-efficient personalization as the user population grows.

REPRODUCIBILITY STATEMENT

To ensure that our work is reproducible, we provide comprehensive details in our submission. We provide our method formulation and training procedure in Appendix C, with pseudocode in Algorithm 1. Mathematical definitions and supporting protocols for the empirical analyses are given in Appendix B, and the theoretical results and proofs are provided in Appendix D. We use the LaMP and LongLaMP benchmarks and describe data preparation, user splits, evaluation metrics, baseline configurations, and hyperparameters in Appendix E, with LongLaMP-specific settings in Appendix F.1. The full code will be released after de-anonymization.

REFERENCES

Tarek F. Abdelzaher, Yigong Hu, Denizhan Kara, Tomoyoshi Kimura, Ashitabh Misra, Vishakha Ramani, Olivier Tardieu, Tianshi Wang, Maggie B. Wigness, and Alaa Youssef. The bottlenecks of AI: challenges for embedded and real-time research in a data-centric age. *Real Time Syst.*, 61(2):185–236, 2025. doi: 10.1007/S11241-025-09452-W. URL https://doi.org/10.1007/s11241-025-09452-w.

Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities - A Nonasymptotic Theory of Independence*. Oxford University Press, 2013. ISBN 978-0-19-953525-5. doi: 10.1093/ACPROF:OSO/9780199535255.001.0001. URL https://doi.org/10.1093/acprof:oso/9780199535255.001.0001.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*, 37(7):3896–3915, 2025. doi: 10.1109/TKDE.2025.3554028.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie

Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024. URL http://jmlr.org/papers/v25/23-0870.html.

Damai Dai, Chengqi Deng, Chenggang Zhao, R.X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1280–1297, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.70. URL https://aclanthology.org/2024.acl-long.70/.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. GLaM: Efficient scaling of language models with mixture-of-experts. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5547–5569. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/du22c.html.

Yu Feng, Yangli-ao Geng, Yifan Zhu, Zongfu Han, Xie Yu, Kaiwen Xue, Haoran Luo, Mengyang Sun, Guangwei Zhang, and Meina Song. Pm-moe: Mixture of experts on private model parameters for personalized federated learning. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, pp. 134–146, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400712746. doi: 10.1145/3696410.3714561. URL https://doi.org/10.1145/3696410.3714561.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Jiani Huang, Shijie Wang, Liangbo Ning, Wenqi Fan, Shuaiqiang Wang, Dawei Yin, and Qing Li. Towards next-generation recommender systems: A benchmark for personalized recommendation assistant with llms. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*, WSDM '26, pp. 217–226, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400722929. doi: 10.1145/3773966.3777954. URL https://doi.org/10.1145/3773966.3777954.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Ishita Kumar, Snigdha Viswanathan, Sushrita Yerra, Alireza Salemi, Ryan A. Rossi, Franck Dernoncourt, Hanieh Deilamsalehy, Xiang Chen, Ruiyi Zhang, Shubham Agarwal, Nedim Lipka, Chien Van Nguyen, Thien Huu Nguyen, and Hamed Zamani. Longlamp: A benchmark for personalized long-form text generation, 2024. URL https://arxiv.org/abs/2407.11016.

Xiaopeng Li, Yuanjin Zheng, Wanyu Wang, Wenlin Zhang, Pengyue Jia, Yingyi Zhang, Haiying He, Mengyang Ma, Yiqi Wang, Maolin Wang, Xuetao Wei, and Xiangyu Zhao. MTA: A merge-then-adapt framework for personalized large language models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 44078–44091, San Diego, California, United States, July 2026. Association for Computational Lin-

guistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.2040. URL https://aclanthology.org/2026.acl-long.2040/.

Jiahong Liu, Zexuan Qiu, Zhongyang Li, Quanyu Dai, Wenhao Yu, Jieming Zhu, Minda Hu, Menglin Yang, Tat-Seng Chua, and Irwin King. A survey of personalized large language models: Progress and future directions, 2025. URL https://arxiv.org/abs/2502.11528.

Jiahong Liu, Wenhao Yu, Quanyu Dai, Zhongyang Li, Jieming Zhu, Menglin Yang, Tat-Seng Chua, and Irwin King. Perfit: Exploring personalization shifts in representation space of llms. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 75361–75396, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/7a1ac10ea9d98178a389c7fbb3575567-Paper-Conference.pdf.

Jiaxing Liu, Qi Qi, Haifeng Sun, Dunjun Li, Zirui Zhuang, Bo He, Xiang Yang, Cong Liu, Jianxin Liao, and Jingyu Wang. ModularMoE: Fast LLM customization with parameter-sharing mixture-of-experts for low-resource settings. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 3562–3575, San Diego, California, United States, July 2026b. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.174. URL https://aclanthology.org/2026.findings-acl.174/.

Hang Lv, Sheng Liang, Hao Wang, Hongchao Gu, Yaxiong Wu, Wei Guo, Defu Lian, Yong Liu, and Enhong Chen. Costeer: Collaborative decoding-time personalization via local delta steering, 2026. URL https://arxiv.org/abs/2507.04756.

Yilun Qiu, Xiaoyan Zhao, Yang Zhang, Yimeng Bai, Wenjie Wang, Hong Cheng, Fuli Feng, and Tat-Seng Chua. Measuring what makes you unique: Difference-aware user modeling for enhancing LLM personalization. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 21258–21277, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1095. URL https://aclanthology.org/2025.findings-acl.1095/.

Alexandre Rame, Guillaume Couairon, Corentin Dancette, Jean-Baptiste Gaya, Mustafa Shukor, Laure Soulier, and Matthieu Cord. Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 71095–71134. Curran Associates, Inc., 2023. doi: 10.52202/075280-3114. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/e12a3b98b67e8395f639fde4c2b03168-Paper-Conference.pdf.

Chris Richardson, Yao Zhang, Kellen Gillespie, Sudipta Kar, Arshdeep Singh, Zeynab Raeesy, Omar Zia Khan, and Abhinav Sethy. Integrating summarization and retrieval for enhanced personalization via large language models, 2023. URL https://arxiv.org/abs/2310.20081.

Fahad Sabah, Yuwen Chen, Zhen Yang, Muhammad Azam, Nadeem Ahmad, and Raheem Sarwar. Model optimization techniques in personalized federated learning: A survey. *Expert Syst. Appl.*, 243(C), June 2024. ISSN 0957-4174. doi: 10.1016/j.eswa.2023.122874. URL https://doi.org/10.1016/j.eswa.2023.122874.

Alireza Salemi, Surya Kallumadi, and Hamed Zamani. Optimization methods for personalizing large language models through retrieval augmentation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pp. 752–762, New York, NY, USA, 2024a. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657783. URL https://doi.org/10.1145/3626772.3657783.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. LaMP: When large language models meet personalization. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar

(eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7370–7392, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.399. URL https://aclanthology.org/2024.acl-long.399/.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Ruizhe Shi, Yifang Chen, Yushi Hu, Alisa Liu, Hannaneh Hajishirzi, Noah A. Smith, and Simon S. Du. Decoding-time language model alignment with multiple objectives. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 48875–48920. Curran Associates, Inc., 2024. doi: 10.52202/079017-1549. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/57c89126d60c209f48d0e6395c766bb3-Paper-Conference.pdf.

Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):9587–9603, 2023. doi: 10.1109/TNNLS.2022.3160699.

Zhaoxuan Tan, Zheyuan Liu, and Meng Jiang. Personalized pieces: Efficient personalized large language models through collaborative efforts. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 6459–6475, Miami, Florida, USA, November 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.371. URL https://aclanthology.org/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 Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 6476–6491, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.372. URL https://aclanthology.org/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 Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23557–23580, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1081. URL https://aclanthology.org/2026.acl-long.1081/.

Stojan Trajanovski, Chad Atalla, Kunho Kim, Vipul Agarwal, Milad Shokouhi, and Chris Quirk. When does text prediction benefit from additional context? an exploration of contextual signals for chat and email messages. In Young-bum Kim, Yunyao Li, and Owen Rambow (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*, pp. 1–9, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-industry.1. URL https://aclanthology.org/2021.naacl-industry.1/.

Luping Wang, Sheng Chen, Linnan Jiang, Shu Pan, Runze Cai, Sen Yang, and Fei Yang. Parameter-efficient fine-tuning in large language models: a survey of methodologies. *Artificial Intelligence Review*, 58(8):227, 2025.

Yiyan Xu, Wenjie Wang, Yang Zhang, Biao Tang, Peng Yan, Fuli Feng, and Xiangnan He. Personalized image generation with large multimodal models. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, pp. 264–274, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400712746. doi: 10.1145/3696410.3714843. URL https://doi.org/10.1145/3696410.3714843.

Liping Yi, Han Yu, Gang Wang, Xiaoguang Liu, and Qinghua Hu. pfedmoe: Data-level personalization with mixture of experts in model-heterogeneous personalized federated learning. *IEEE Transactions on Knowledge and Data Engineering*, 38(3):1905–1918, 2026. doi: 10.1109/TKDE.2026.3656194.

Saber Zerhoudi and Michael Granitzer. Personarag: Enhancing retrieval-augmented generation systems with user-centric agents, 2026. URL https://arxiv.org/abs/2407.09394.

Jinghao Zhang, Yuting Liu, Wenjie Wang, Qiang Liu, Shu Wu, Liang Wang, and Tat-Seng Chua. Personalized text generation with contrastive activation steering. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7128–7141, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.353. URL https://aclanthology.org/2025.acl-long.353/.

Linhai Zhang, Jialong Wu, Deyu Zhou, and Yulan He. PROPER: A progressive learning framework for personalized large language models with group-level adaptation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16399–16411, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.800. URL https://aclanthology.org/2025.acl-long.800/.

Ze Yu Zhang, Bolin Ding, and Bryan Kian Hsiang Low. Pc-moe: memory-efficient and privacy-preserving collaborative training for mixture-of-experts llms. *Machine Learning*, 115(2):31, 2026.

Weixiang Zhao, Xingyu Sui, Yulin Hu, Jiahe Guo, Haixiao Liu, Biye Li, Yanyan Zhao, Bing Qin, and Ting Liu. Teaching language models to evolve with users: Dynamic profile modeling for personalized alignment. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 33162–33199. Curran Associates, Inc., 2025a. doi: 10.52202/085713-1115. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/2fbc1ea60e215275e823c86338417162-Paper-Conference.pdf.

Xiaoyan Zhao, Juntao Jun, Yang Zhang, Wenjie Wang, Hong Cheng, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Nextquill: Causal preference modeling for enhancing llm personalization. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 118745–118770, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/c0ae487420ebc8d0ed7c541b4e3f09d4-Paper-Conference.pdf.

Zheng Zhao, Clara Vania, Subhradeep Kayal, Naila Khan, Shay B. Cohen, and Emine Yilmaz. PersonaLens: A benchmark for personalization evaluation in conversational AI assistants. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18023–18055, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.927. URL https://aclanthology.org/2025.findings-acl.927/.

Appendix Contents

A Related Work **16**
A.1 Personalized Large Language Models 16
A.2 Mixture of Experts 17

B Additional Empirical Analysis **17**
B.1 RQ1: Cross-User Reuse 18
B.2 RQ2: User- and Query-Dependent Utility 19
B.3 RQ3: Compact History-Based Correction 21

C Methodology Details **22**
C.1 Algorithm of LINEUP 23
C.2 Shared Factors and Hierarchical Composition 23
C.3 Tiny User Codes and Support Adaptation 24

D Theoretical Analysis and Proofs **25**
D.1 History-Initialized Fitting and Query Risk 25
D.2 Finite History and Shared Conditioning 26
D.3 Finite-Step Adam and the Combined Bound 26
D.4 Correction Geometry and Transfer 27

E Supplementary Materials for Experimental Setup **28**
E.1 Datasets 28
E.2 Evaluation Metrics 28
E.3 Baselines 29
E.4 Implementation 30
E.5 Hyperparameters 30

F Supplementary Experimental Results **30**
F.1 Long-Form Generation on LongLaMP 30
F.2 Additional Cold-Start Results 31
F.3 Additional Efficiency and Scalability Results 32
F.4 Ablation Protocol 33
F.5 User-Code Dimension Sensitivity 33

A Related Work

A.1 Personalized Large Language Models

Personalized large language models adapt model behavior to individual users based on their histories, profiles, or preference feedback (Liu et al., 2025). Existing methods can be broadly grouped into tuning-free, learned adaptation-based, and alignment approaches.

Tuning-free personalization. Tuning-free methods typically summarize or retrieve user histories to construct personalized prompts or augment the model context (Richardson et al., 2023; Qiu et al., 2025; Salemi et al., 2024a; Zerhoudi & Granitzer, 2026). Other approaches steer generation at inference time by contrasting personalized and non-personalized model states or combining decoding signals (Zhang et al., 2025a; Lv et al., 2026). These methods avoid target-user parameter optimization and can readily incorporate newly available histories. However, their effectiveness depends on the quality of historical data preprocessing.

Learned adaptation-based personalization. This is a complementary family that encodes personalization into learned parameters or representation-space interventions. We organize them according to the basic adaptation unit from which target-user personalization is constructed.

The first direction performs **individualized adaptation construction**. OPPU directly optimizes an independent PEFT module for each user by using their history (Tan et al., 2024b). PerFit decomposes representation-space personalization into collective and personalized shifts while retaining a user-specific intervention for individual variation (Liu et al., 2026a). P2P changes how such individualized adaptation is obtained by introducing a population-trained hypernetwork. It maps a user profile gained during preprocessing and the user's query-retrieved history to a complete set of personalized LoRA parameters. P2P thereby avoids iterative per-user fine-tuning at deployment (Tan et al., 2026). Despite these differences, the primary personalization object remains an individualized adaptation module that can be optimized directly, instantiated in representation space, or generated from user information. Methods that fit and retain such modules incur user-specific optimization and persistent-state costs, while hypernetwork generation shifts part of this burden to preprocessing stages. Thus, these approaches mainly improve how an individualized adaptation state is obtained, rather than changing the granularity at which personalization capacity itself is organized.

The second direction improves reuse through **user-derived reusable components**. Per-Pcs learns personalized PEFT modules for selected users, decomposes them into reusable pieces, and assembles pieces for target users (Tan et al., 2024a). PROPER introduces intermediate group-level adaptation and combines group LoRAs with individual user LoRAs (Zhang et al., 2025b). MTA constructs a bank of anchor-derived Meta-LoRAs, retrieves and merges relevant modules for a target user, and additionally fits an individual LoRA for few-shot refinement (Li et al., 2026). These approaches demonstrate the value of cross-user reuse. Nevertheless, their reusable units are obtained from adaptation modules associated with particular users. Their emphasis is on reusing such modules, rather than systematically characterizing what structure is reusable. This leaves open a different organization of personalization capacity: whether reusable units can instead be learned *directly* as cross-user personalization factors and composed according to the current request.

Personalized alignment. Personalized alignment forms a complementary line of research that typically models personalization through heterogeneous objectives or preference signals. User preferences can be incorporated through personalized reward modeling, parameter merging, or decoding-time policy combination (Liu et al., 2025; Rame et al., 2023; Shi et al., 2024; Zhao et al., 2025a; 2026). These methods usually assume explicit preference feedback or a predefined preference space. By contrast, our setting focuses on task-specific personalization inferred from heterogeneous behavioral histories, where user-specific behavior must be recovered from prior interactions rather than supplied as an explicit alignment objective.

Positioning of our work. We take *personalization capacity allocation* as the organizing principle. The central distinction is not whether parameters are shared across users, but *what is shared, how reusable capacity is learned, and what must still be independently adapted for a target user*.

LINEUP differs from individualized adaptation methods by changing the basic personalization unit. Rather than constructing a complete adaptation state for each user or request, it directly learns fine-grained low-rank personalization factors across source users as reusable population-level capacity.

Different users can reuse overlapping factors, while the same user can employ different compositions for different requests. LINEUP also differs from collaborative adapter-reuse methods in the origin of this reusable capacity, since its factor bank is learned jointly for cross-user composition from the outset, rather than being obtained by first learning user-, anchor-, or group-associated adapters and subsequently decomposing, selecting, or merging them.

Our composition mechanism further separates persistent user relevance from request-specific utility. A user's history first recalls a compact candidate set of factors, and the current query then calibrates their sparse selection and mixture. LINEUP further confines target-user optimization to an eight-dimensional code over shared correction maps. Therefore, most personalization capacity resides in reusable shared structures, while only a tiny coordinate state must be independently estimated for a new target user. This factor-centric design is motivated by our empirical findings.

Taken together, LINEUP *decouples personalization capacity from per-user adaptation*: expressive personalization is learned predominantly through reusable, conditionally composed shared capacity, whereas independent target-user fitting is reserved for a compact residual correction. This organization simultaneously targets strong personalization, adaptation from limited user history, and low incremental parameter and storage cost as the user population grows.

A.2 Mixture of Experts

Mixture-of-Experts (MoE) models combine specialized expert modules through learned routing mechanisms (Jacobs et al., 1991; Cai et al., 2025). Sparse MoE activates only a subset of experts for each input, enabling specialized and compositional model capacity without proportional computation (Shazeer et al., 2017; Du et al., 2022; Dai et al., 2024). Beyond general-purpose model scaling, MoE-style architectures have also been studied in decentralized collaborative training and personalized federated learning (Liu et al., 2026b; Zhang et al., 2026; Yi et al., 2026; Feng et al., 2025). In much of this literature, personalization is defined at the client level, with shared or private experts introduced to accommodate heterogeneity in local data distributions, model architectures, or computational resources (Tan et al., 2023; Sabah et al., 2024). Personalized LLMs focus on a different problem where the model must infer latent individual preferences from behavioral histories and tailor its behavior jointly to the user and the current request.

LINEUP adopts the compositional principle of sparse expert models but uses it for a different granularity and purpose. Its experts are fine-grained low-rank personalization factors jointly learned as reusable cross-user capacity, rather than independently meaningful user or client models. No single factor is intended to represent a complete user. Instead, different users can reuse overlapping factors, and personalization emerges from their conditional composition. Routing is explicitly hierarchical, where user history first narrows the shared factor bank to a candidate set, after which the current query calibrates candidate factors. This separates persistent user relevance from request-specific utility rather than collapsing both into a single routing decision. LINEUP therefore uses expert-style composition as part of a broader personalization capacity allocation strategy, rather than as a conventional mechanism for model scaling or client specialization.

B Additional Empirical Analysis

This appendix provides definitions and supporting evidence for Section 3. The numbered subsections below provide direct access to each diagnostic definition, comparison, and validation.

RQ1: Cross-user reuse. Shared subspaces and update coverage (Appendix B.1.1); task-adapter control (Appendix B.1.3).

RQ2: Direction utility. Setup and utility definition (Appendix B.2.1); within-user and cross-user agreement (Appendix B.2.2); scale-controlled query variation (Appendix B.2.3); direction-removal validation (Appendix B.2.4).

RQ3: History-based correction. Correction signals and label controls (Appendix B.3.1); support-to-query transfer (Appendix B.3.2); compact correction spaces (Appendix B.3.3); first-order intervention validation (Appendix B.3.4).

Diagnostic reference. RQ1 and RQ2 use independently fitted OPPU user LoRAs (Tan et al., 2024b) on a common task-adapted backbone. We analyze their effective updates $D_{u,i} = s_i B_{u,i} A_{u,i}$ rather than the individual LoRA factors. RQ3 probes representation corrections in the task-adapted backbone without user-specific adapters. Diagnostic user folds and learned bases are separate from LINEUP's shared-training procedure and are not used to train LINEUP.

Uncertainty and aggregation. We use fixed checkpoints. Unless stated otherwise, intervals are pointwise 95% percentile confidence intervals from 2,000 user-level bootstrap draws, with checkpoints, diagnostic folds, and learned bases held fixed. These intervals quantify user-sampling uncertainty rather than variation across training seeds. Aggregation is specified below; RQ2 also accounts for dependence between user pairs.

B.1 RQ1: Cross-User Reuse

B.1.1 Shared Subspaces and Update Coverage

Diagnostic setup. We study LaMP-2M, LaMP-2N, LaMP-5, and LaMP-7, covering classification and generation. Each user contributes 64 adaptation sites: the query and value projections in all 32 layers. Every site has output dimension 4096. Evaluated users do not participate in learning the shared directions.

Shared-subspace construction. For diagnostic fold f, let $\mathcal{V}_{-f}$ denote the users outside the held-out fold. At site i, we construct the normalized output covariance

$$C_i^{(-f)} = \frac{1}{|\mathcal{V}_{-f}|} \sum_{u \in \mathcal{V}_{-f}} \frac{D_{u,i} D_{u,i}^\top}{\|D_{u,i}\|_F^2}. \tag{4}$$

The columns of $Q_{m,i}^{(-f)}$ are its leading m eigenvectors. Normalization gives users equal weight regardless of update magnitude. Each held-out user is evaluated using a basis learned outside that user's fold.

Coverage and random reference. For a held-out user, coverage measures the fraction of update energy captured by the shared output subspace:

$$\text{Coverage}_{u,i}(m) = \frac{\|(Q_{m,i}^{(-f)})^\top D_{u,i}\|_F^2}{\|D_{u,i}\|_F^2}. \tag{5}$$

This measures reuse of output directions while allowing their coefficients to differ across users. We average over held-out user–site pairs and evaluate $m \in \{1, 2, 4, 8, 16, 32, 64, 128\}$. The random reference in Figure 1(a) averages five Haar-random subspace draws, with coverage approximately 0.0078 at $m = 32$ and 0.0312 at $m = 128$. Coverage at $m = 32$ is also reported in Table 4 under genuine-user basis.

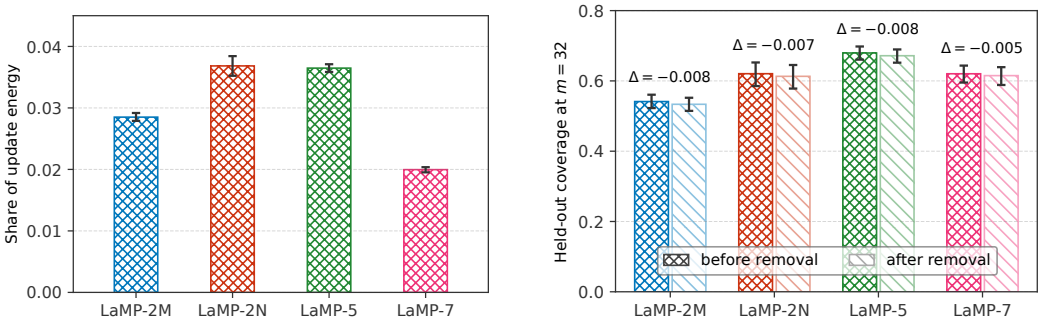
B.1.2 Random-Partition Control

Protocol. We retain the diagnostic cohorts of 100, 49, 101, and 100 users on LaMP-2M/2N/5/7, respectively. Within each diagnostic fold, we pool the supervised examples used to fit the genuine-user adapters and randomly repartition them into pseudo-users, preserving every input–output pair and the original group-size sequence. Each pseudo-user adapter is independently fitted using the same task-adapted backbone, LoRA configuration, and optimization schedule as its genuine-user counterpart. We use one fixed random partition per task. Partitioning within folds keeps held-out users' examples out of basis construction.

Matched coverage comparison. We construct each basis from either genuine-user or random-partition adapters outside the held-out fold, using Equation 4. Both bases are evaluated on the same held-out genuine-user updates using Equation 5. We denote the genuine-minus-random coverage difference by Δ_{group}. Its 95% confidence interval uses 2,000 paired user-level bootstrap draws after averaging sitewise differences within each user. Checkpoints, diagnostic folds, learned bases, and the random partition remain fixed.

Table 4: Held-out update-energy coverage at $m = 32$ for genuine-user and size-matched random-partition bases. Genuine-user grouping improves coverage, including after task-span removal.

Task	Genuine-user basis	Random-partition basis	Δ_{group}	95% CI	Post-removal Δ_{group}
LaMP-2M	.541	.377	**+.164**	[.159, .171]	+.164
LaMP-2N	.621	.411	**+.210**	[.195, .226]	+.212
LaMP-5	.679	.536	**+.143**	[.141, .147]	+.144
LaMP-7	.620	.462	**+.158**	[.155, .163]	+.158



(a) Task-span overlap (b) Task-span removal

Figure 4: Task-adapter controls for RQ 1. (a) Only a small fraction of user-update energy lies in the task-LoRA output span. (b) Removing this span leaves shared-subspace coverage nearly unchanged.

Results. At $m = 32$, genuine-user bases provide higher coverage on all four tasks, with absolute advantages of 0.143–0.210 and paired confidence intervals excluding zero (Table 4); reported differences are computed from unrounded coverage values. Removing the task-adapter output span from both adapter populations and repeating basis construction and evaluation (Appendix B.1.3) leaves these gaps nearly unchanged. Thus, preserving genuine user grouping yields shared directions that better capture held-out genuine-user updates, strengthening the cross-user reuse evidence in Observation 1.

B.1.3 Task-Adapter Control

To test whether the observed coverage is concentrated in the trained task adapter's output span, let $Q_{\text{task},i}$ be an orthonormal basis of the task-LoRA output span. Its overlap with a user update is $\|Q_{\text{task},i}^{\top} D_{u,i}\|_F^2 / \|D_{u,i}\|_F^2$. Mean overlap is 2.85%, 3.68%, 3.65%, and 2.00% on LaMP-2M/2N/5/7, respectively (Figure 4(a)).

We remove this span,

$$D_{u,i}^{\perp} = \left(I - Q_{\text{task},i} Q_{\text{task},i}^{\top}\right) D_{u,i}, \tag{6}$$

and repeat shared-basis construction and coverage evaluation on the residual updates. At $m = 32$, residual coverage is 0.533, 0.613, 0.672, and 0.615 on LaMP-2M/2N/5/7, respectively (Figure 4(b)). Coverage remains nearly unchanged, supporting the reusable structure in Observation 1 beyond the task adapter's output span.

B.2 RQ2: User- and Query-Dependent Utility

B.2.1 Diagnostic Setup and Direction Utility

Diagnostic setup. We use LaMP-2M and LaMP-2N because they provide repeated held-out queries per user, allowing utility to be compared while the fitted adapter remains fixed. The diagnostic uses 100 users with 3,302 queries on LaMP-2M and 49 users with 6,033 queries on LaMP-2N.

At each of the 64 sites, we use the first eight shared directions learned outside the evaluated user's fold.

Direction-utility definition. Let $q_{i,k}$ denote shared direction k at site i. Its component in user u's fixed update is $E_{u,i,k} = q_{i,k} q_{i,k}^\top D_{u,i}$. We introduce diagnostic gates through

$$D_{u,i}(\boldsymbol{\alpha}_i) = D_{u,i} + \sum_{k=1}^{8} (\alpha_{i,k} - 1) E_{u,i,k}. \tag{7}$$

All gates equal to one recover the original adapter. Let $\mathcal{L}_{u,j}(\boldsymbol{\alpha})$ denote the query loss under the gated model, where $\boldsymbol{\alpha}$ collects gates across sites. The signed sensitivity is

$$G_{u,j,i,k} = \left. \frac{\partial \mathcal{L}_{u,j}(\boldsymbol{\alpha})}{\partial \alpha_{i,k}} \right|_{\boldsymbol{\alpha}=\mathbf{1}}. \tag{8}$$

The sign indicates whether strengthening a component raises or lowers loss to first order. The user adapter and diagnostic directions remain fixed across queries; the gates are diagnostic variables, not a trained router.

B.2.2 Within-User and Cross-User Agreement

For each user, we use the first eight held-out queries. Within-user agreement averages Spearman correlation over all $\binom{8}{2} = 28$ query pairs, then over users and sites. For cross-user agreement, let $z_{u,j,i}$ denote the centered, unit-length rank vector of the eight signed sensitivities, and define $\bar{z}_{u,i} = \frac{1}{8} \sum_{j=1}^{8} z_{u,j,i}$. The mean query-pair agreement between distinct users u and v is

$$\frac{1}{8^2} \sum_{j=1}^{8} \sum_{j'=1}^{8} z_{u,j,i}^\top z_{v,j',i} = \bar{z}_{u,i}^\top \bar{z}_{v,i}. \tag{9}$$

We average these values over distinct user pairs and sites. Within-user intervals resample users as clusters; cross-user intervals use a user-vertex bootstrap to preserve dependence between pairs sharing a user.

Within- versus cross-user agreement is $0.145\,[0.107, 0.185]$ versus $0.008\,[0.004, 0.014]$ on LaMP-2M and $0.400\,[0.330, 0.475]$ versus $0.059\,[0.021, 0.112]$ on LaMP-2N (Figure 1(b)). The within-user value is higher at all 64 sites on both tasks (Figure 5(b)).

B.2.3 Scale-Controlled Query Variation

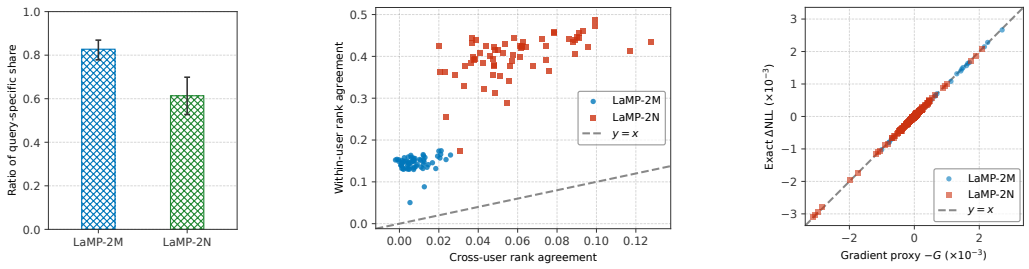
To distinguish changes in relative directional responses from overall sensitivity scale, we normalize each eight-dimensional sensitivity vector as $X_{u,j,i} = G_{u,j,i} / \|G_{u,j,i}\|_2$. Let $SS_{\mathrm{W},i}$ and $SS_{\mathrm{B},i}$ denote the within-user and between-user sums of squares of these normalized responses at site i. We report

$$V_{\mathrm{query}} = \frac{\sum_{i \in \mathcal{I}} SS_{\mathrm{W},i}}{\sum_{i \in \mathcal{I}} (SS_{\mathrm{W},i} + SS_{\mathrm{B},i})}. \tag{10}$$

This fraction is $0.827\,[0.778, 0.869]$ on LaMP-2M and $0.613\,[0.527, 0.699]$ on LaMP-2N (Figure 5(a)). Substantial query-dependent variation therefore remains after controlling for overall sensitivity scale.

B.2.4 Exact Direction-Removal Validation

We compare the first-order removal prediction $-G_{u,j,i,k}$ with the exact query-loss change when one gate is set from one to zero while all other gates remain unchanged. Each task contributes 512 interventions: two users, eight queries per user, eight directions, and four sites. Pooled Pearson correlations are 0.998955 on LaMP-2M and 0.999295 on LaMP-2N (Figure 5(c)). This agreement supports the functional interpretation of the signed sensitivities in Observation 2.



(a) Query-specific variation (b) Sitewise rank agreement (c) Direction-removal validation

Figure 5: Additional evidence for user- and query-dependent utility in RQ 2. (a) Substantial query variation remains after sensitivity-scale control. (b) Within-user rank agreement exceeds cross-user agreement at every analyzed site. (c) First-order predictions closely match exact direction-removal loss changes.

B.3 RQ3: Compact History-Based Correction

B.3.1 Correction Signals and Label Controls

Diagnostic setup and correction signals. We use LaMP-2M and LaMP-2N, whose finite categorical label spaces allow label-based controls. The task-adapted backbone is frozen and contains no user-specific adapters. At zero-indexed layers $\mathcal{L}_{\text{diag}} = \{0, 4, 9, 13, 18, 22, 27, 31\}$, we add a vector $v_\ell \in \mathbb{R}^{4096}$ to all T_z non-padding token positions of example z. Let $\bar{\mathcal{L}}_{\mathcal{T}}(z; v_\ell)$ denote the mean target-token NLL under this intervention. Following the representation-space view in Section 2.2, the length-rescaled correction signal is

$$\gamma_z^\ell = -\frac{1}{T_z} \nabla_{v_\ell} \bar{\mathcal{L}}_{\mathcal{T}}(z; v_\ell)\big|_{v_\ell=0}. \tag{11}$$

This rescales the loss-reducing gradient by the number of positions receiving the intervention; it is not unit-normalized.

Diagnostic support $\mathcal{S}_u^{\text{diag}} \subseteq \mathcal{H}_u$ contains the first $\min(32, |\mathcal{H}_u|)$ available history records. Their mean correction direction is

$$\bar{s}_u^\ell = \frac{1}{|\mathcal{S}_u^{\text{diag}}|} \sum_{z \in \mathcal{S}_u^{\text{diag}}} \gamma_z^\ell. \tag{12}$$

Label controls and donor histories. For each diagnostic fold and layer, we compute the record-weighted mean correction signal m_c^ℓ for each label c using only diagnostic support from analysis-training users. Let Q_{label}^ℓ be an orthonormal basis of $\text{span}\{m_1^\ell, \ldots, m_C^\ell\}$. Both tasks have $C = 15$ labels, and the span has dimension 15 at every analyzed fold and layer. We remove this span from query and support signals:

$$\begin{aligned} \gamma_z^{\ell,\perp} &= \left(I - Q_{\text{label}}^\ell (Q_{\text{label}}^\ell)^\top\right) \gamma_z^\ell, \\ s_u^{\ell,\perp} &= \left(I - Q_{\text{label}}^\ell (Q_{\text{label}}^\ell)^\top\right) \bar{s}_u^\ell. \end{aligned} \tag{13}$$

Held-out query signals are used only for evaluation, not to construct the label span.

We compare three history sources. *Own* uses the target user's history. *Random* uses a randomly selected donor from another diagnostic fold. *Matched* uses an out-of-fold donor with the nearest historical label distribution in L_1 distance. All support directions follow the same construction and label-span removal.

B.3.2 Support-to-Query Transfer

For history source a, let $s_{a(u)}^{\ell,\perp}$ denote its support direction. Signed transfer to held-out query j of user u is

$$T_{u,j,\ell}^{(a)} = \left(\gamma_{u,j}^{\ell,\perp}\right)^\top s_{a(u)}^{\ell,\perp}. \tag{14}$$

A positive value indicates alignment between the history-derived correction and a first-order loss-reducing query direction. The statistic retains signal magnitudes; token count and intervention step enter the corresponding first-order NLL prediction (Equation 18). We average over held-out query records within each layer, then average the eight layerwise means equally.

Own-history transfer is 2.190×10^{-6} with interval $[0.798, 3.906] \times 10^{-6}$ on LaMP-2M and 1.930×10^{-5} with interval $[0.808, 3.058] \times 10^{-5}$ on LaMP-2N (Figure 1(c)). Own-history point estimates are positive at all eight layers; intervals exclude zero at seven layers on LaMP-2M and all eight on LaMP-2N. Random and matched donor intervals cross zero on LaMP-2M, while both donor means are negative on LaMP-2N. This consistent own-history alignment supports individual correction beyond the removed label-mean span (Observation 3).

B.3.3 Compact Correction Spaces

For each layer and diagnostic fold, we learn an orthonormal correction basis $R_{r,\ell}$ using only support-side signals from analysis-training users outside the evaluated fold. Held-out query signals are used only for evaluation. Let $P_{r,\ell} = R_{r,\ell} R_{r,\ell}^\top$. We measure both retained correction energy and retained support-to-query transfer. Residual query-gradient coverage is

$$\mathrm{Coverage}^{\mathrm{corr}}_{u,j,\ell}(r) = \frac{\|R_{r,\ell}^\top \gamma_{u,j}^{\ell,\perp}\|_2^2}{\|\gamma_{u,j}^{\ell,\perp}\|_2^2}. \tag{15}$$

Writing $T^{\mathrm{own}}_{u,j,\ell}$ for the full-space transfer under *Own*, its projected counterpart is

$$T^{\mathrm{own}}_{u,j,\ell}(r) = \left(\gamma_{u,j}^{\ell,\perp}\right)^\top P_{r,\ell} s_u^{\ell,\perp}. \tag{16}$$

For the set $\mathcal{J}$ of held-out user–query records, transfer retention is

$$\mathrm{Retention}(r) = \frac{1}{|\mathcal{L}_{\mathrm{diag}}|} \sum_{\ell \in \mathcal{L}_{\mathrm{diag}}} \frac{\sum_{(u,j)\in\mathcal{J}} T^{\mathrm{own}}_{u,j,\ell}(r)}{\sum_{(u,j)\in\mathcal{J}} T^{\mathrm{own}}_{u,j,\ell}}. \tag{17}$$

The transfer ratio is computed within each layer before averaging equally across layers. Coverage and retention use different references: residual query-gradient energy and full-space, label-controlled support-to-query transfer, respectively.

At $r = 8$, energy coverage is $0.129\,[0.123, 0.136]$ on LaMP-2M and $0.170\,[0.160, 0.181]$ on LaMP-2N, whereas transfer retention is $0.347\,[0.222, 0.479]$ and $0.294\,[0.198, 0.413]$, respectively (Figure 6(a)). Figures 6(b,c) show the corresponding trends over $r \in \{1, 2, 4, 8, 16\}$. At $r = 16$, coverage reaches 0.168 and 0.244, while retention reaches 0.425 and 0.416. These comparisons support the preferential retention of transferable correction in Observation 3. The diagnostic sweep is not used to select LINEUP's user-code dimension.

B.3.4 First-Order Intervention Validation

We compare first-order predictions with exact loss changes after injecting the own-history direction projected into the $r = 8$ diagnostic space. Because Equation 11 includes $1/T_z$, the predicted change in mean target-token NLL is

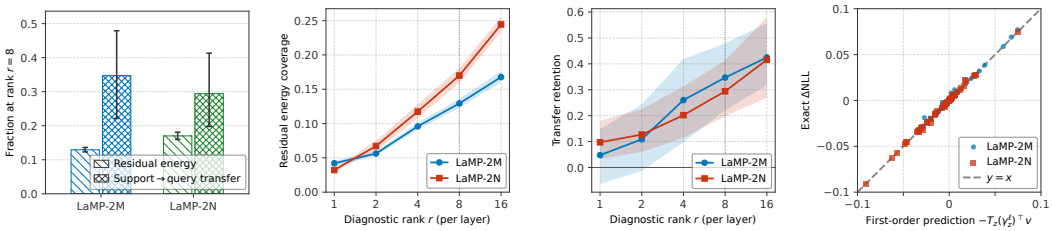
$$\Delta\mathrm{NLL}_z \approx -T_z \left(\gamma_z^\ell\right)^\top v, \tag{18}$$

where v is the projected support direction scaled by a small step size calibrated on analysis-training users.

Each task contributes 64 interventions: two users, four queries per user, and eight layers. Median layerwise Pearson correlations are 0.9998 on LaMP-2M and 0.9997 on LaMP-2N (Figure 6(d)).

C Methodology Details

This appendix defines the shared factors, conditional composition, and user-code adaptation in Section 4. Algorithm 1 summarizes shared training and target-user personalization. Task-specific hyperparameters and model-selection settings are reported with the experimental configuration.



(a) Transfer vs. energy (b) Residual-energy coverage (c) Transfer retention (d) First-order intervention validation

Figure 6: Additional evidence for compact history-based correction in RQ 3. (a) At rank eight, transfer retention exceeds residual-energy coverage. (b) Residual-energy coverage increases with diagnostic rank. (c) Higher ranks retain more support-to-query transfer. (d) First-order predictions track exact intervention loss changes in most tested cases.

C.1 Algorithm of LINEUP

Shared parameters and user state. The S1 and S2 shared parameters are

$$\phi_1 = \{A_{k,i}, B_{k,i}\}_{k,i} \cup \{W_H, W_Q\}, \qquad \phi_2 = \{V_\ell\}_{\ell \in \mathcal{L}_{\text{corr}}} \cup \{G\}.$$

Together, $\phi_{\mathcal{T}} = \phi_1 \cup \phi_2$ instantiates the shared personalization parameters in Section 2.1. The full method's only independently adapted target-user state is $\omega_u = \hat{c}_u \in \mathbb{R}^{r_c}$. The task-adapted backbone $\theta_{\mathcal{T}}$ and feature extraction remain fixed throughout.

Training episodes and prediction loss. Each source-user episode contains a history $\mathcal{H}_u^{\text{ep}}$ and supervised examples $\mathcal{B}_u$. Depending on the task, supervision comes from history records held out from the episode history or from the source user's task queries. For a predictor f conditioned on the episode history, define

$$\mathcal{L}_{\text{tok}}(\mathcal{B}; f) = -\frac{1}{N(\mathcal{B})} \sum_{(x,y)\in\mathcal{B}} \sum_{t=1}^{|y|} \log f(y_t \mid x, y_{<t}), \tag{19}$$

where $N(\mathcal{B}) = \sum_{(x,y)\in\mathcal{B}} |y|$ counts supervised output tokens, excluding prompt and padding tokens. Write f_1 for the S1-only predictor and $f_2(c)$ for the predictor with S1 and code-controlled correction. Both use the history encoding and each example's input for routing.

Shared training. Stage 1 optimizes ϕ_1 using f_1. Stage 2 freezes S1 and optimizes ϕ_2 using $f_2(G\tilde{e}_u)$ directly. The correction maps and initializer are therefore learned from predictions at initialized codes, before target-user support fitting. All shared parameters are then fixed for target-user personalization.

C.2 Shared Factors and Hierarchical Composition

Factor parameterization. At site i, factor k contributes $F_{k,i} = s_i B_{k,i} A_{k,i}$, where $A_{k,i} \in \mathbb{R}^{r_L \times d_{\text{in},i}}$, $B_{k,i} \in \mathbb{R}^{d_{\text{out},i} \times r_L}$, and s_i is the LoRA scaling coefficient. Factor index k identifies coordinated low-rank updates across the query and value projections throughout the transformer. These factors are learned directly through source-user prediction under their conditional composition, rather than obtained from the diagnostic adapters in Section 3.

History and query features. Let E be the frozen input-embedding table and $\bar{e}(s) = |s|^{-1} \sum_t E[s_t]$ the mean of the non-padding token embeddings of text s. Historical records are rendered using their available inputs and outputs. With separate fixed random projections $P_H, P_Q \in \mathbb{R}^{d_f \times d}$, we compute

$$\begin{aligned} e_u &= P_H \frac{1}{|\mathcal{H}|} \sum_{h\in\mathcal{H}} \bar{e}(\text{render}(h)), & \tilde{e}_u &= \text{norm}(e_u - \mu_H), \\ e_x &= P_Q \bar{e}(\text{prompt}(x)), & \tilde{e}_x &= \text{norm}(e_x - \mu_Q), \end{aligned} \tag{20}$$

Algorithm 1 LINEUP

Require: Frozen task model $\theta_{\mathcal{T}}$, source users, target-user histories and query inputs, code-update budget T

Construct fixed feature maps and source-training centering statistics
Initialize ϕ_1 and ϕ_2
Stage 1: Learn shared factor composition
for each S1 training episode **do**
    Sample $(\mathcal{H}_u^{\mathrm{ep}}, \mathcal{B}_u)$ from a source user
    Encode the episode history; route each input as in Appendix C.2
    Update ϕ_1 on $\mathcal{L}_{\mathrm{tok}}(\mathcal{B}_u; f_1)$
end for
Freeze ϕ_1
Stage 2: Learn shared corrections and initialization
for each S2 training episode **do**
    Sample $(\mathcal{H}_u^{\mathrm{ep}}, \mathcal{B}_u)$; compute $\tilde{e}_u$
    Set $c_u^{(0)} \leftarrow G\tilde{e}_u$ and keep S1 active
    Update ϕ_2 on $\mathcal{L}_{\mathrm{tok}}(\mathcal{B}_u; f_2(c_u^{(0)}))$
end for
Freeze all shared parameters
Target-user personalization and prediction
for each target user u **do**
    Encode available $\mathcal{H}_u$; recall $\mathcal{C}_u$; construct $\mathcal{S}_u$
    Set $c_u^{(0)} \leftarrow G\tilde{e}_u$, $c \leftarrow c_u^{(0)}$; initialize fresh Adam state
    for $t = 1, \ldots, T$ **do**
        Route support inputs using the fixed history encoding; evaluate $\mathcal{J}_u(c)$
        Take one Adam update of c only
    end for
    Store $\hat{c}_u \leftarrow c$
    for each query input x of user u **do**
        Compute $\mathcal{A}_{u,x}$ and $w_{u,x}$ from the history encoding and current query
        Predict using the shared-factor mixture and correction from $\hat{c}_u$
    end for
end for

where $\mathrm{norm}(v) = v/\|v\|_2$. Here $\mathcal{H}$ is the episode history during shared training and the available history during target-user personalization. The centering vectors μ_H, μ_Q are estimated from source-training histories and query prompts and then fixed. Each history record is pooled separately before averaging across records; query encoding uses only the current input prompt. Historical outputs provide personalization evidence, but evaluation targets never enter the features. S1 recall and the S2 initializer use the same history encoding.

Candidate recall and query calibration. The candidate set $\mathcal{C}_u$, calibrated scores $z_{u,x}$, and active set $\mathcal{A}_{u,x}$ are defined in Equation 3. Softmax over active scores gives

$$w_{u,x,k} = \begin{cases} \dfrac{\exp(z_{u,x,k})}{\sum_{j \in \mathcal{A}_{u,x}} \exp(z_{u,x,j})}, & k \in \mathcal{A}_{u,x}, \\ 0, & k \notin \mathcal{A}_{u,x}. \end{cases} \tag{21}$$

These weights yield the update in Equation 1, replacing $W_i h$ by $(W_i + D_{u,x,i})h$. One factor mixture applies across all adapted sites and every prompt or generated token of a request. When $M_a = M_c$, query calibration reweights the recalled factors; when $M_a < M_c$, it can also change the active subset. Prediction gradients update active factors and routing scores through the selected-set softmax; TopK indices are treated as discrete choices.

C.3 Tiny User Codes and Support Adaptation

Shared correction maps. Equation 2 uses a shared map $V_\ell \in \mathbb{R}^{d \times r_c}$ at each selected block output $\ell \in \mathcal{L}_{\mathrm{corr}}$ and one user code shared across these layers. Any fixed multiplicative scaling is absorbed

into V_ℓ, so V_ℓ denotes the effective shared correction map. For a general forward pass, replace $\hat{c}_u$ by $c_u^{(0)}$ during shared S2 training, c during support fitting, and $\hat{c}_u$ during target-query prediction. The binary mask $a_{\ell,t}$ selects correction positions. In multi-token passes, it selects configured boundary windows of the non-padding sequence: prompt plus output for teacher forcing, and prompt only for generation prefill. Cached decoding applies the correction to each newly processed token. Layer choices and window sizes are specified with task configurations.

History-informed initialization. The shared linear map $G \in \mathbb{R}^{r_c \times d_f}$ produces $c_u^{(0)} = G\tilde{e}_u$ from the same history encoding used by S1. Source-user training learns G and $\{V_\ell\}_{\ell \in \mathcal{L}_{\text{corr}}}$ through prediction loss at these initialized codes. Target-user support fitting refines this starting point while all shared maps remain fixed.

Support construction. A task-specific construction maps selected records from the available history to supervised input–output pairs $\mathcal{S}_u$. History encoding uses all available records, independently of how many are selected for fitting. Under limited-history evaluation, both conditioning and support construction use only the available history.

Code refinement. Write $\widehat{\mathcal{L}}_{\mathcal{S}_u}(c) = \mathcal{L}_{\text{tok}}(\mathcal{S}_u; f_2(c))$. The fitting objective is

$$\mathcal{J}_u(c) = \widehat{\mathcal{L}}_{\mathcal{S}_u}(c) + \lambda_{\text{reg}} \|c - c_u^{(0)}\|_2^2. \tag{22}$$

Cross-entropy is averaged over supervised output tokens pooled across support examples, rather than averaging example losses. The regularizer penalizes squared deviation from the fixed history-based initialization.

S1 remains active: each support input receives its own query-calibrated factor mixture using the unchanged history encoding. Fitting uses the same routing and correction configuration as target-query prediction. A fresh Adam state is initialized for each user, and T full-support updates are taken from $c_u^{(0)}$. Only the code is updated; the backbone and all shared parameters remain fixed. The final iterate $\hat{c}_u$ is reused across subsequent queries, while S1 recomputes their mixtures from the same history and their respective inputs. We use $r_c = 8$. S1-only evaluation omits correction and code fitting; full LINEUP includes history-based initialization followed by support refinement.

D Theoretical Analysis and Proofs

We analyze the finite-step S2 estimator conditional on all source-learned parameters; a quadratic specialization explains the learned geometry.

D.1 History-Initialized Fitting and Query Risk

Setup. Fix a user with N history records $\mathcal{H}$ and n support examples. Stack the shared correction maps into $V \in \mathbb{R}^{D \times r}$, where $D = |\mathcal{L}_{\text{corr}}|d$ and $r = r_c$. One code controls all selected layers, fixed scaling is included in V, and S1 remains history–query conditioned. For the token-pooled support NLL $\widehat{L}_{\mathcal{H}}$ in Equation 22, write

$$c_0 = G\tilde{e}(\mathcal{H}), \qquad F_{\mathcal{H}}(c) = \widehat{L}_{\mathcal{H}}(c) + \frac{\rho}{2}\|c - c_0\|_2^2, \qquad \rho = 2\lambda_{\text{reg}} \geq 0. \tag{23}$$

Let c_T be the final code and $d_T = c_T - c_0$. For expected query NLL write $\mathcal{R}_{\mathcal{H}}(c) = \mathcal{R}_{\mathcal{H}}^{\text{rep}}(Vc)$ and $\mathcal{R}_{\mathcal{H}}^\star = \inf_v \mathcal{R}_{\mathcal{H}}^{\text{rep}}(v)$: v has the same correction scope, not arbitrary backbone updates. The query distribution and loss weighting are fixed and may differ from support. With finite risks, define

$$\mathcal{A}_{\mathcal{H}} = \inf_c \mathcal{R}_{\mathcal{H}}(c) - \mathcal{R}_{\mathcal{H}}^\star, \qquad \mathcal{I}_{\mathcal{H}} = \mathcal{R}_{\mathcal{H}}(c_0) - \inf_c \mathcal{R}_{\mathcal{H}}(c). \tag{24}$$

These are shared-space approximation and initialization errors. Let $\Delta_T = F_{\mathcal{H}}(c_0) - F_{\mathcal{H}}(c_T)$ and

$$\epsilon_{\mathcal{H}}(R) = \sup_{\|d\|_2 \leq R} \|\nabla \mathcal{R}_{\mathcal{H}}(c_0 + d) - \nabla \widehat{L}_{\mathcal{H}}(c_0 + d)\|_2. \tag{25}$$

Proposition 1 (Finite-step comparison)**.** *Suppose the losses are continuously differentiable on the segment from c_0 to c_T, $\|d_T\|_2 \le R$, and $\rho > 0$. Then*

$$\mathcal{R}_{\mathcal{H}}(c_T) - \mathcal{R}^\star_{\mathcal{H}} \le \mathcal{A}_{\mathcal{H}} + \mathcal{I}_{\mathcal{H}} - \Delta_T + \frac{\epsilon_{\mathcal{H}}(R)^2}{2\rho}. \tag{26}$$

Proof. Integrating $E_{\mathcal{H}} = \mathcal{R}_{\mathcal{H}} - \widehat{L}_{\mathcal{H}}$ along the segment yields $E_{\mathcal{H}}(c_T) - E_{\mathcal{H}}(c_0) \le \epsilon_{\mathcal{H}}(R)\|d_T\|_2$. Also, $\widehat{L}_{\mathcal{H}}(c_T) - \widehat{L}_{\mathcal{H}}(c_0) = -\Delta_T - \rho\|d_T\|_2^2/2$. Combine these relations with $\mathcal{R}_{\mathcal{H}}(c_0) - \mathcal{R}^\star_{\mathcal{H}} = \mathcal{A}_{\mathcal{H}} + \mathcal{I}_{\mathcal{H}}$, then use $-\rho x^2/2 + \epsilon x \le \epsilon^2/(2\rho)$. □

D.2 Finite History and Shared Conditioning

Define $Z_{\mathcal{H}}(d) = \nabla\mathcal{R}_{\mathcal{H}}(c_0(\mathcal{H}) + d) - \nabla\widehat{L}_{\mathcal{H}}(c_0(\mathcal{H}) + d)$. Record replacement acts on conditioning, initialization, and support jointly; initialization is not assumed independent of support.

Proposition 2 (Uniform gradient discrepancy)**.** *Condition on the user and shared parameters. Assume independent records and square-integrable, L_Z-Lipschitz $Z_{\mathcal{H}}$ on the radius-R ball. Uniformly over this ball and permitted record replacements, assume*

$$\|\mathbb{E}Z_{\mathcal{H}}(d)\|_2 \le b, \qquad \|Z_{\mathcal{H}}(d) - Z_{\mathcal{H}^{(i)}}(d)\|_2 \le a_i. \tag{27}$$

Here $\mathcal{H}^{(i)}$ replaces record i, and all bounds are deterministic. Set $S_a = \sum_i a_i^2$. For fixed $s > 0$ and $\delta \in (0,1)$, with probability at least $1-\delta$,

$$\epsilon_{\mathcal{H}}(R) \le \bar{\epsilon}(R,s,\delta) := b + \sqrt{S_a/2} + L_Z s + \sqrt{\frac{S_a}{2}\left[r\log\left(1 + \frac{2R}{s}\right) + \log\frac{1}{\delta}\right]}. \tag{28}$$

Proof. For fixed d, vector Efron–Stein bounds the expected centered norm by $\sqrt{S_a/2}$; McDiarmid gives tail probability $\exp(-2t^2/S_a)$ at t above this mean (Boucheron et al., 2013). Apply a union bound over an s-net of size at most $(1+2R/s)^r$, add b, and extend to the ball using L_Z-Lipschitzness. The case $S_a = 0$ is deterministic. □

Record-level scaling. For i.i.d. records, a data-independent subset of $n \le N$ support records, bounded output-length ratios, per-example code gradients bounded by B, and gradient sensitivity $O(\kappa/N)$ to replacing a record in conditioning and initialization, a replace-one argument yields $S_a = O(\kappa^2/N + B^2/n)$ and $b = b_{\text{shift}} + O(\kappa/N + B/\sqrt{n})$, where b_{shift} denotes population gradient mismatch.

Routing stability. With bounded record features and centered encoding norms bounded away from zero, history averaging changes scores by at most $\Delta_s = O(1/N)$ under record replacement, and TopK sets are preserved when each nontrivial selection gap exceeds $2\Delta_s$. Otherwise routing jumps must enter the influence constants rather than assuming the same $O(1/N)$ sensitivity across changes of selected indices.

D.3 Finite-Step Adam and the Combined Bound

Updates. For full-support Adam with fresh moments (Kingma & Ba, 2015), $g_t = \nabla F_{\mathcal{H}}(c_{t-1})$ and

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)g_t, & s_t &= \beta_2 s_{t-1} + (1-\beta_2)g_t^{\odot 2}, \\ \hat{m}_t &= m_t/(1-\beta_1^t), & \hat{s}_t &= s_t/(1-\beta_2^t), \\ M_t &= \operatorname{diag}((\sqrt{\hat{s}_t} + \varepsilon_{\mathrm{A}})^{-1}), & c_t &= c_{t-1} - \alpha_t M_t \hat{m}_t. \end{aligned} \tag{29}$$

Here $m_0 = s_0 = 0$, $0 < \beta_1, \beta_2 < 1$, and $\varepsilon_{\mathrm{A}} > 0$; elementwise operations are understood. Learning rates $\alpha_t > 0$ are fixed before target-user fitting.

Lemma 1 (Deterministic displacement bound). *For every $t \le T$,*

$$\|c_t - c_0\|_2 \le R_T := \sqrt{r} \sum_{j=1}^{T} \alpha_j \sqrt{\chi_j},$$
$$\chi_t := \frac{(1-\beta_1)^2(1-\beta_2^t)}{(1-\beta_1^t)^2(1-\beta_2)} \sum_{j=0}^{t-1} \left(\frac{\beta_1^2}{\beta_2}\right)^j. \tag{30}$$

Each step also satisfies $\|c_t - c_{t-1}\|_2 \le \alpha_t \sqrt{r\chi_t}$.

Proof. Apply weighted Cauchy–Schwarz to the bias-corrected moment sums in Equation 29: for each coordinate k, $\hat{m}_{t,k}^2 \le \chi_t \hat{s}_{t,k}$. The normalized step in that coordinate is therefore at most $\alpha_t\sqrt{\chi_t}$, including the zero-moment case. Sum squared coordinate bounds and then step lengths. □

Numerical scale and observables. For illustration, $(\beta_1, \beta_2) = (0.9, 0.999)$, $r = 8$, and $T = 10$ give $1 \le \sqrt{\chi_t} < 1.044$ for $1 \le t \le 10$, and $R_{10} \simeq 28.752\alpha$ for constant learning rate. The reported code learning rates 0.001–0.008 (Table 7) yield bounds 0.029–0.230 under these moment settings. Actual $\|d_T\|_2$ and Δ_T are observable, but query improvement still requires control of $\epsilon_{\mathcal{H}}(R_T)$; the numerical bounds are not measurements.

Proposition 3 (Progress along Adam). *If $F_{\mathcal{H}}$ has an L_F-Lipschitz gradient on the trajectory ball, then*

$$\Delta_T \ge \Gamma_T := \sum_{t=1}^{T} \left[\alpha_t g_t^\top M_t \hat{m}_t - \frac{L_F \alpha_t^2}{2} \|M_t \hat{m}_t\|_2^2\right]. \tag{31}$$

Proof. Apply the smoothness inequality to each update in Equation 29 and sum. □

Momentum remark. If $\alpha_t L_F \|M_t\|_{\mathrm{op}} \le 1$, the per-step decrease is at least $\alpha_t(\|g_t\|_{M_t}^2 - \|e_t\|_{M_t}^2)/2$, where $e_t = \hat{m}_t - g_t$ and $\|x\|_M^2 = x^\top M x$. As e_t is a convex combination of previous gradient differences, smoothness and Lemma 1 bound its norm through preceding step lengths.

Corollary 1 (Finite-history risk of S2). *Under Propositions 2 and 3 with $R = R_T$, set $\bar{\epsilon}_T = \bar{\epsilon}(R_T, s, \delta)$. For $\rho > 0$, with probability at least $1 - \delta$,*

$$\mathcal{R}_{\mathcal{H}}(c_T) - \mathcal{R}_{\mathcal{H}}^\star \le \mathcal{A}_{\mathcal{H}} + \mathcal{I}_{\mathcal{H}} - \Gamma_T + \frac{\bar{\epsilon}_T^2}{2\rho}. \tag{32}$$

The sharper bound with Δ_T replacing Γ_T also holds; in particular, fitting improves on initialization if $\Delta_T + \rho\|d_T\|_2^2/2 > \bar{\epsilon}_T\|d_T\|_2$. For $\rho = 0$, the last term in Equation 32 becomes $\bar{\epsilon}_T R_T$.

Proof. Lemma 1 supplies a deterministic ball, so the uniform discrepancy bound applies to its history-dependent output. Use the proof of Proposition 1 and $\Delta_T \ge \Gamma_T$. □

D.4 Correction Geometry and Transfer

Quadratic specialization. Let $q(v) = q(v^\star) + \|v - v^\star\|_H^2/2$ with $H \succ 0$, and fix $Q = V^\top H V$. The optimal displacement from initialization is $d^\star = Q^\dagger V^\top H(v^\star - V c_0)$, where $\dagger$ denotes the Moore–Penrose pseudoinverse, and define $\mathcal{A}_q(V) = \inf_c q(Vc) - q(v^\star)$. Consider a support model with the same curvature, $\widehat{q}(d) = \mathrm{const} - (Qd^\star + \xi)^\top d + d^\top Q d/2$, and its exact regularized solution $\bar{d} = \arg\min_d\{\widehat{q}(d) + \rho\|d\|_2^2/2\}$, $\rho > 0$. Write $q_j > 0$ for the positive eigenvalues of Q and $d_j^\star, \xi_j$ for the corresponding coordinates.

Corollary 2 (Geometry-controlled refinement)**.** *With finite second moments, suppose* $\mathbb{E}\xi = 0$, $\mathbb{E}[\xi\xi^\top] = \sigma^2 Q/n$, *and* $\mathbb{E}[d_j^\star \xi_j] = 0$ *for every* $q_j > 0$.[1] *Then*

$$\mathbb{E}[q(V(c_0 + \bar{d})) - q(v^\star)] = \mathbb{E}\mathcal{A}_q(V) + \frac{1}{2}\sum_{j:q_j>0} q_j \left(\frac{\rho}{q_j+\rho}\right)^2 \mathbb{E}[(d_j^\star)^2] + \frac{\sigma^2}{2n} d_{\text{eff}}(\rho),$$
$$d_{\text{eff}}(\rho) := \sum_{j:q_j>0} \left(\frac{q_j}{q_j+\rho}\right)^2 \le \operatorname{rank}(V) \le r. \tag{33}$$

Refinement reduces expected query error in direction j*, relative to retaining initialization, exactly when*

$$(q_j + 2\rho)\,\mathbb{E}[(d_j^\star)^2] > \frac{\sigma^2}{n}. \tag{34}$$

Proof. The regularized normal equation gives $\bar{d} - d^\star = (Q+\rho I)^{-1}(\xi - \rho d^\star)$. Weighted projection gives $q(V(c_0+d)) - q(v^\star) = \mathcal{A}_q(V) + \|d - d^\star\|_Q^2/2$. Substitute the code error, diagonalize Q, and apply the stated moments. Each effective-dimension summand is at most one, and $\operatorname{rank}(Q) = \operatorname{rank}(V)$. Subtracting refined from initialized directional risk gives $\frac{q_j^2}{2(q_j+\rho)^2}[(q_j+2\rho)\mathbb{E}[(d_j^\star)^2] - \sigma^2/n]$, proving the criterion. □

Transfer through the learned maps. At $v_0 = Vc_0$, let g_q and g_s be negative representation gradients of query and support NLL: fresh Adam moments and the zero regularizer gradient give $c_1 - c_0 = \alpha_1 M_1 V^\top g_s$. For L_R-smooth query risk, its decrease is at least $\alpha_1 g_q^\top V M_1 V^\top g_s - L_R\alpha_1^2 \|M_1 V^\top g_s\|_2^2/2$. Thus the positive semidefinite, rank-at-most-r operator $VM_1V^\top$ governs transfer, connecting Observation 3 to learned correction geometry rather than an orthogonal projector.

Shared learning. On source episodes, $\mathcal{R}_{\mathcal{H}}(c_0) = \mathcal{R}_{\mathcal{H}}^\star + \mathcal{A}_{\mathcal{H}} + \mathcal{I}_{\mathcal{H}}$, so the shared S2 population objective targets approximation and initialization error jointly, without a support-fitting inner loop.

Scope. The finite-step NLL bounds use the stated smoothness and influence conditions, not Adam convergence. Independent records are needed only for concentration, not the pathwise comparison or displacement bound. The quadratic corollary additionally uses common curvature and stated moments; it does not predict an optimal code size or gains in every metric.

E Supplementary Materials for Experimental Setup

E.1 Datasets

We use six LaMP tasks (Salemi et al., 2024b) under OPPU's protocol of data processing, target-user partition, and prompt construction (Tan et al., 2024b). Table 5 summarizes their statistics. Following prior OPPU-based evaluations (Liu et al., 2026a; Li et al., 2026), we exclude LaMP-1 because outputs under the OPPU prompt and Llama-2-7B frequently violate the required citation format, hindering reliable evaluation; we omit LaMP-6 because of data privacy concerns. The remaining six datasets still ensure task diversity.

We further partition the source users into fixed 90/10 training/validation sets. These sets and the target-user cohort are mutually disjoint, and the split is unchanged across seeds. Target-user histories are used for personalization, while their held-out queries are reserved for final evaluation.

E.2 Evaluation Metrics

All methods are scored by the same evaluator using the metrics in Table 6.

[1] Without decorrelation, add $-\rho\sum_{j:q_j>0} q_j \mathbb{E}[d_j^\star \xi_j]/(q_j+\rho)^2$ to the risk; the directional criterion then changes.

Table 5: Statistics of the six LaMP tasks. Avg. history is the mean number of history records per target user.

Dataset	Task	Type	Users (source / target)	Source queries	Avg. history	Target queries	# classes
LaMP-2M	Movie tagging	Classification	829 / 100	3,181	55.6	3,302	15
LaMP-2N	News categorization	Classification	274 / 49	3,662	219.9	6,033	15
LaMP-3	Product rating	Prediction	19,899 / 101	22,388	959.8	112	5
LaMP-4	News headline generation	Generation	1,543 / 100	7,275	270.1	6,725	–
LaMP-5	Scholarly title generation	Generation	14,581 / 101	16,075	442.9	107	–
LaMP-7	Tweet paraphrasing	Generation	13,337 / 100	14,826	121.2	109	–

Table 6: Evaluation metrics used across the six LaMP tasks. **Acc** = Accuracy, **F1** = Macro-F1, **R-1** = ROUGE-1, and **R-L** = ROUGE-L. Here, P_c and R_c denote precision and recall for class c; $P_{1,i}$ and $R_{1,i}$ denote unigram precision and recall for the i-th example; and $P_{L,i}$ and $R_{L,i}$ denote the corresponding precision and recall based on the longest common subsequence.

Metric	Datasets	Task type	Formulation
Acc ↑	LaMP-2M, LaMP-2N	Classification	$\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}[\hat{y}_i = y_i]$
F1 ↑	LaMP-2M, LaMP-2N	Classification	$\frac{1}{C}\sum_{c=1}^{C}\frac{2P_cR_c}{P_c+R_c}$
MAE ↓	LaMP-3	Rating prediction	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i-\hat{y}_i\rvert$
RMSE ↓	LaMP-3	Rating prediction	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}$
R-1 ↑	LaMP-4, LaMP-5, LaMP-7	Generation	$\frac{1}{n}\sum_{i=1}^{n}\frac{2P_{1,i}R_{1,i}}{P_{1,i}+R_{1,i}}$
R-L ↑	LaMP-4, LaMP-5, LaMP-7	Generation	$\frac{1}{n}\sum_{i=1}^{n}\frac{2P_{L,i}R_{L,i}}{P_{L,i}+R_{L,i}}$

For generation, we report ROUGE-1 and ROUGE-L F-measures averaged over examples with one reference, using `rouge-score` with Porter stemming and `rougeL`. For LaMP-3, numeric predictions are parsed as floats without rounding or clipping. Unparsable outputs are assigned the farther endpoint in $\{1, 5\}$ from the reference rating, following the official LaMP evaluator.

E.3 Baselines

We use baselines' official implementations. Their personalization algorithms are retained under the common Llama-2-7B backbone and OPPU target-user protocol.

Source-data construction and shared training remain method-specific to ensure fairness. OPPU and PerFit perform task adaptation on the full source pool. P2P trains its hypernetwork on all source users, conditioning adapter generation on user profiles and query-retrieved history. Per-Pcs and MTA retain their released selection procedures, choosing sharers or anchors from up to 1,000 candidates drawn from the full source partition. RAG retrieves the top user-history record using BM25.

Per-Pcs and MTA provide no complete released configuration for LaMP-2N. MTA also excludes LaMP-7 because its adaptation-data construction requires query–answer pairs. These entries are marked as NA.

Table 7: LINEUP hyperparameters. "F7+L7" selects the first and last seven token positions; "L7" selects the last seven.

Hyperparameter	**LaMP-2M**	**LaMP-2N**	**LaMP-3**	**LaMP-4**	**LaMP-5**	**LaMP-7**
Candidate / active factors	4 / 4	4 / 4	4 / 4	16 / 4	16 / 4	16 / 4
Query strength λ	2	2	2	8	8	8
S1 learning rate	2×10^{-4}	1×10^{-4}	2×10^{-4}	5×10^{-5}	2×10^{-4}	2×10^{-4}
S2 layers	14, 15	14–16	14, 15	14–16	15	15, 16
S2 positions	L7	F7+L7	L7	F7+L7	L7	F7+L7
S2 shared learning rate	2.25×10^{-3}	1×10^{-3}	2.25×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
User-code learning rate	2×10^{-3}	2×10^{-3}	4×10^{-3}	8×10^{-3}	1×10^{-3}	1×10^{-3}

E.4 Implementation

We learn LINEUP's shared components on source-training users and select hyperparameters on source-validation users. Selection uses the arithmetic mean of the two task metrics, maximized for classification and generation and minimized for LaMP-3. The selected configurations are fixed before final target-user evaluation.

We report mean $\pm$ standard deviation over three runs. LINEUP and baselines are independently trained with three seeds.

E.5 Hyperparameters

Across tasks, LINEUP uses $K = 32$ rank-4 LoRA factors over the query and value projections of all transformer layers, with 256-dimensional history and query representations. Target-user fitting updates only an eight-dimensional code for 10 Adam steps. Table 7 lists detailed settings.

F Supplementary Experimental Results

F.1 Long-Form Generation on LongLaMP

We extend our evaluation to long-form personalized generation on three LongLaMP tasks (Kumar et al., 2024): LongLaMP-2 (Abstract Generation), LongLaMP-3 (Product Review Generation), and LongLaMP-4 (Topic Writing).

F.1.1 Experimental Setup

We first consolidate the official LongLaMP Temporal release at the user level, yielding one available history and a set of held-out queries for each user. We select users with the longest histories as the target cohort. The remaining users form the source pool and are eligible for LINEUP's training/validation split. User-disjoint splitting and model selection follow Appendix E.1 and Appendix E.4. LongLaMP-1 (Email Completion) is omitted because its underlying Avocado email corpus requires licensed access.

For history-derived supervision, Abstract Generation uses title–abstract pairs, Product Review Generation uses ratings, descriptions, and summaries to predict review text, and Topic Writing uses summary–content pairs. Table 8 summarizes the resulting datasets.

For this supplementary evaluation, we compare Non, Basic, RAG, and OPPU under the same user split and evaluation protocol. These cover the pretrained model, task adaptation, retrieval-based personalization, and full private user adapters. Extending PerFit, Per-Pcs, and MTA would require substantial LongLaMP-specific adaptation. P2P reports LongLaMP results under a different protocol, so aligning it with our user split would require rebuilding user representations and retraining.

Table 8: Statistics of the three LongLaMP tasks. Avg. history is the mean number of history records per target user.

Dataset	Task	Users (source / target)	Source queries	Avg. history	Target queries
LongLaMP-2	Abstract generation	22,620 / 101	31,663	1,291.3	145
LongLaMP-3	Product review generation	15,997 / 100	19,561	560.5	128
LongLaMP-4	Topic writing	16,136 / 100	20,975	260.6	132

Table 9: Performance on LongLaMP-2, LongLaMP-3, and LongLaMP-4. **R-1** = ROUGE-1 and **R-L** = ROUGE-L. **Bold** and underline denote the best and second-best means. Δ reports absolute improvements over the strongest baseline. LINEUP leads on all metrics for long-form generation.

	LongLaMP-2		LongLaMP-3		LongLaMP-4	
Method	**R-1** ↑	**R-L** ↑	**R-1** ↑	**R-L** ↑	**R-1** ↑	**R-L** ↑
Non	$.195 \pm .003$	$.121 \pm .002$	$.143 \pm .010$	$.082 \pm .004$	$.174 \pm .005$	$.090 \pm .002$
Basic	$.367 \pm .010$	$.194 \pm .006$	$.296 \pm .001$	$.112 \pm .001$	$.233 \pm .009$	$.103 \pm .002$
RAG	$.381 \pm .005$	$.211 \pm .002$	$.322 \pm .005$	$.135 \pm .003$	$.242 \pm .006$	$.114 \pm .004$
OPPU	$\underline{.412} \pm .003$	$\underline{.244} \pm .004$	$\underline{.346} \pm .002$	$\underline{.157} \pm .001$	$\underline{.260} \pm .003$	$\underline{.123} \pm .001$
LINEUP	$\mathbf{.435} \pm .006$	$\mathbf{.263} \pm .005$	$\mathbf{.402} \pm .004$	$\mathbf{.205} \pm .001$	$\mathbf{.289} \pm .003$	$\mathbf{.136} \pm .002$
Δ	**+0.023**	**+0.019**	**+0.056**	**+0.048**	**+0.029**	**+0.013**

We therefore retain a focused comparison, with OPPU providing the direct private-adapter reference central to our research question.

F.1.2 Results on LongLaMP

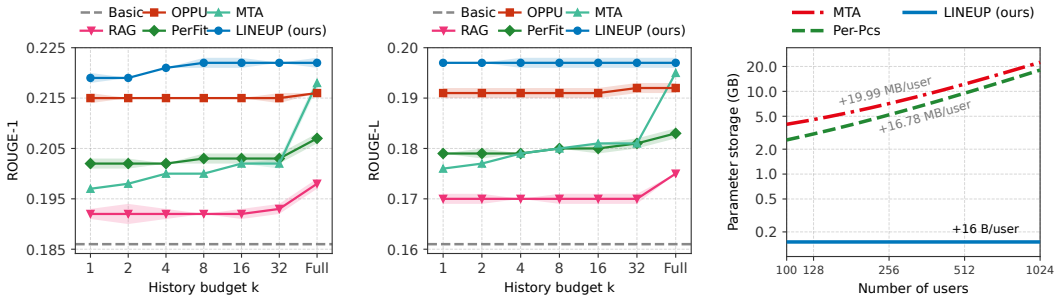
Table 9 shows that **LINEUP achieves the best mean performance on all six metrics across the three tasks**, with OPPU providing the strongest baseline throughout. The largest gains occur on Product Review Generation, where LINEUP improves over OPPU by 0.056 in ROUGE-1 and 0.048 in ROUGE-L.

The advantage over OPPU extends our capacity-allocation findings to long-form generation. Jointly learned factors provide reusable personalization capacity, history–query composition tailors it to each request, and the compact user code supplies residual individual correction. Thus, long-form personalization can also benefit from rich shared capacity without requiring a complete private adapter for every user.

F.2 Additional Cold-Start Results

Protocol. We evaluate LaMP-2N and LaMP-4 using nested prefixes of each user's earliest $k \in \{1, 2, 4, 8, 16, 32, \text{Full}\}$ historical interactions. Target users and evaluation queries remain fixed. Shared parameters are unchanged, and every history-dependent operation uses only the current prefix. LINEUP recomputes history-conditioned recall and code initialization at each budget, then fits a new code using available support examples. The full available prefix is used for history encoding. Results are mean $\pm$ sample standard deviation over three seeds. Basic is history-independent, so its full-history result is repeated across budgets. Tables 12 and 13 report the complete results.

Generation results. Figure 7(a–b) and Table 13 provide complementary results on LaMP-4. LINEUP achieves the best mean ROUGE-1 and ROUGE-L at every history budget. With one history record, it reaches $0.219/0.197$, compared with OPPU's $0.215/0.191$ at the same budget. Several



(a) LaMP-4 ROUGE-1 (b) LaMP-4 ROUGE-L (c) LaMP-3 storage

Figure 7: Supplementary cold-start and adaptation-storage results. (a) One interaction nearly recovers LINEUP's full-history ROUGE-1. (b) LINEUP maintains the highest ROUGE-L across history budgets. (c) LINEUP's storage grows far more slowly than Per-Pcs and MTA.

Table 10: Shared and per-user adaptation parameters and bf16 storage on LaMP-2N and LaMP-3. User columns report persistent state per user.

Method	Shared params	User params	Shared storage	User storage
(a) LaMP-2N				
OPPU	6.29 M	4.19 M	12.58 MB	8.39 MB
PerFit	1.18 M	0.79 M	2.36 MB	1.57 MB
P2P	307.66 M	0	615.33 MB	0
LINEUP	75.61 M	8	151.23 MB	16 B
(b) LaMP-3				
Per-Pcs	454.03 M	8.39 M	908.07 MB	16.78 MB
MTA	999.42 M	9.99 M	1,998.85 MB	19.99 MB
LINEUP	75.58 M	8	151.16 MB	16 B

baselines also change little with additional history; the key finding here is LINEUP's consistently stronger generation performance under restricted history.

Baseline coverage. P2P and Per-Pcs are included in the full-history main comparison but not in this controlled cold-start study, because our reproduced pipelines use user representations precomputed from complete histories. We restrict this study to baselines whose history budgets can be varied within the same reproduced protocol. MTA is omitted on LaMP-2N because no complete released configuration is available.

F.3 Additional Efficiency and Scalability Results

Storage accounting. For the evaluated configurations, we analytically compute storage curves from shared adaptation parameters P_{shared} and persistent parameters per user P_{user} as $S(N) = 2(P_{\text{shared}} + NP_{\text{user}})$ bytes for N users. Storage is normalized to bf16, with $1\,\text{MB} = 10^6$ bytes and $1\,\text{GB} = 10^9$ bytes. Here N is a deployment-population variable, independent of the evaluated target-user cohort; shared parameters and per-user state sizes remain fixed. The per-user term counts parameter state retained for reuse, whether fitted or assembled. We exclude the common frozen Llama-2-7B backbone and auxiliary pretrained encoders. LINEUP's shared count includes its task LoRA and fixed feature projections. P2P's hypernetwork is counted as shared state, while its query-generated LoRAs are transient and excluded from the per-user term. Per-Pcs retains an assembled personalized LoRA, and MTA retains a trained stacked LoRA; reconstructable merged copies are not counted separately. Table 10 summarizes the parameter and storage counts.

Comparisons. The displayed curves begin at 100 users, reflecting the approximately 100 target users in most tasks under our evaluation protocol. For LaMP-2N, LINEUP becomes more storage-efficient than OPPU at 17 users and PerFit at 95 users, while remaining below P2P's 615.33 MB shared state over the displayed range. At LaMP-2N's actual evaluation size of 49 target users, PerFit therefore has lower total adaptation storage than LINEUP. We additionally compare Per-Pcs and MTA using their complete LaMP-3 configurations (Figure 7(c)). At 1,024 users, the projected totals are approximately 151.18 MB for LINEUP, 18.09 GB for Per-Pcs, and 22.47 GB for MTA. Thus, directly sharing personalization capacity avoids the substantial per-user storage that remains when reusable adapter components are assembled into large individual modules.

F.4 Ablation Protocol

We evaluate all variants on LaMP-2M and LaMP-5 over three seeds and report mean $\pm$ standard deviation. The two tasks cover different routing regimes: LaMP-2M uses $M_c = M_a = 4$, whereas LaMP-5 uses $M_c = 16$ and $M_a = 4$. Query calibration therefore reweights recalled factors on LaMP-2M and additionally selects the active subset on LaMP-5.

All variants inherit the corresponding task configuration, optimization settings, data splits, evaluator, and training schedule without re-tuning. The training horizon and reference checkpoint are 5,000 and 4,096 episodes for LaMP-2M, and 500 and 400 for LaMP-5. Full LINEUP and S1 only reuse the corresponding main-experiment runs; the two routing ablations are trained under the same schedule as S1-only and evaluated at the corresponding reference checkpoint.

Random candidates. We replace the history-guided candidate set with a uniformly sampled subset of the same size:

$$\mathcal{C}_u^{\mathrm{rand}} \sim \mathrm{UniformSubset}\big(\{1, \ldots, K\}, M_c\big). \tag{35}$$

The subset is generated deterministically from the experimental seed and user identifier, and remains fixed across that user's queries and batches. The original score $z_{u,x} = s_u + \lambda W_Q \tilde{e}_x$ still determines active-factor selection and mixture weights within this set. Thus, this control removes history-guided retrieval rather than all history information.

No query calibration. We set $\lambda = 0$, so $z_{u,x} = s_u$, throughout training and inference. History-conditioned candidate recall, optimization settings, and all remaining model components are unchanged.

Removing the user-specific correction. S1 only retains shared-factor composition and omits the entire S2 correction mechanism. Full LINEUP adds history-initialized, support-fitted user-code correction to the same S1 model, so this comparison evaluates the contribution of S2 as a whole.

Initialized code without support fitting. *S1 + initialized code* uses the same shared checkpoints and available user history as Full LINEUP for each seed. It retains S1 and the shared correction maps, applying the initialized code $c_u^{(0)} = G\tilde{e}_u$ across the user's queries without target-user optimization. Comparison with S1-only evaluates the shared maps and history-based initialization together; comparison with Full LINEUP isolates the additional contribution of support fitting.

F.5 User-Code Dimension Sensitivity

Protocol. We sweep $r_c \in \{2, 4, 8, 16, 32\}$ on LaMP-2M and LaMP-5, following the data splits and evaluation protocol in Appendix F.4. For each task and seed, the task-adapted backbone and S1 checkpoint are fixed across dimensions. For each new dimension, we retrain the shared correction maps $\{V_\ell\}$ and history initializer G; $r_c = 8$ reuses the existing Full LINEUP runs. All remaining settings are unchanged, including S2 training budgets, checkpoint selection, support construction, learning rates, and regularization. Each target user initializes one code from available history and fits only this code for ten Adam steps, keeping all shared parameters fixed.

Results. Table 11 shows that increasing r_c from 2 to 8 improves mean performance on all four metrics. Moving from 8 to 16 raises both LaMP-2M means by 0.002, while LaMP-5 scores remain unchanged at the reported precision. At $r_c = 32$, LaMP-2M Accuracy reaches its highest mean,

Table 11: Sensitivity of LINEUP to user-code dimension r_c on LaMP-2M and LaMP-5. Underline and **bold** denote the second-best and best means, including ties; † marks the default configuration. Performance improves from $r_c = 2$ to 8, while larger codes yield no consistent gains across metrics.

	LaMP-2M		LaMP-5	
r_c	Acc ↑	F1 ↑	R-1 ↑	R-L ↑
2	.633 ± .002	.530 ± .002	.551 ± .003	.486 ± .001
4	.641 ± .002	.535 ± .002	.563 ± .002	.491 ± .001
8†	.645 ± .003	.539 ± .002	**.564** ± .003	**.494** ± .004
16	.647 ± .004	**.541** ± .001	**.564** ± .002	**.494** ± .002
32	**.653** ± .006	.538 ± .003	.561 ± .002	.490 ± .002

but Macro-F1 and both ROUGE means decrease relative to $r_c = 16$. Larger codes therefore do not yield consistent improvements across metrics. The default $r_c = 8$ provides an effective compact configuration, matching the best reported generation means and remaining close to $r_c = 16$ on classification.

Table 12: Cold-start performance on LaMP-2N across history budgets. **Bold** denotes the best reported mean for each metric and budget.

Method	k = 1	2	4	8	16	32	Full
Acc ↑							
Basic	.783 ± .002	.783 ± .002	.783 ± .002	.783 ± .002	.783 ± .002	.783 ± .002	.783 ± .002
OPPU	.793 ± .002	.793 ± .002	.793 ± .002	.793 ± .002	.793 ± .002	.793 ± .002	.801 ± .001
PerFit	.679 ± .000	.711 ± .001	.710 ± .001	.713 ± .001	.722 ± .001	.724 ± .001	.818 ± .000
RAG	.750 ± .001	.756 ± .001	.768 ± .001	.770 ± .001	.778 ± .001	.788 ± .001	.817 ± .001
LINEUP	**.806** ± .006	**.807** ± .007	**.807** ± .005	**.807** ± .005	**.806** ± .005	**.813** ± .005	**.822** ± .005
F1 ↑							
Basic	.536 ± .002	.536 ± .002	.536 ± .002	.536 ± .002	.536 ± .002	.536 ± .002	.536 ± .002
OPPU	.536 ± .002	.536 ± .002	.536 ± .002	.536 ± .002	.537 ± .002	.537 ± .003	.556 ± .002
PerFit	.457 ± .002	.484 ± .001	.489 ± .002	.510 ± .001	.514 ± .001	.518 ± .002	.589 ± .002
RAG	.507 ± .002	.514 ± .003	.537 ± .004	.554 ± .004	.571 ± .003	**.588** ± .001	.594 ± .001
LINEUP	**.560** ± .005	**.564** ± .005	**.575** ± .004	**.585** ± .004	**.584** ± .003	**.588** ± .003	**.621** ± .003

Table 13: Cold-start performance on LaMP-4 across history budgets. **Bold** denotes the best reported mean for each metric and budget.

Method	k = 1	2	4	8	16	32	Full
R-1 ↑							
Basic	.186 ± .000	.186 ± .000	.186 ± .000	.186 ± .000	.186 ± .000	.186 ± .000	.186 ± .000
OPPU	.215 ± .001	.215 ± .000	.215 ± .000	.215 ± .000	.215 ± .000	.215 ± .001	.216 ± .000
PerFit	.202 ± .001	.202 ± .001	.202 ± .000	.203 ± .001	.203 ± .001	.203 ± .001	.207 ± .001
RAG	.192 ± .001	.192 ± .002	.192 ± .001	.192 ± .000	.192 ± .001	.193 ± .001	.198 ± .001
MTA	.197 ± .000	.198 ± .000	.200 ± .000	.200 ± .000	.202 ± .000	.202 ± .001	.218 ± .001
LINEUP	**.219** ± .001	**.219** ± .000	**.221** ± .000	**.222** ± .001	**.222** ± .001	**.222** ± .000	**.222** ± .001
R-L ↑							
Basic	.161 ± .001	.161 ± .001	.161 ± .001	.161 ± .001	.161 ± .001	.161 ± .001	.161 ± .001
OPPU	.191 ± .001	.191 ± .001	.191 ± .001	.191 ± .001	.191 ± .001	.192 ± .001	.192 ± .001
PerFit	.179 ± .000	.179 ± .001	.179 ± .000	.180 ± .000	.180 ± .001	.181 ± .001	.183 ± .001
RAG	.170 ± .001	.170 ± .001	.170 ± .000	.170 ± .001	.170 ± .001	.170 ± .001	.175 ± .000
MTA	.176 ± .000	.177 ± .000	.179 ± .000	.180 ± .000	.181 ± .000	.181 ± .000	.195 ± .000
LINEUP	**.197** ± .000	**.197** ± .000	**.197** ± .001	**.197** ± .001	**.197** ± .001	**.197** ± .001	**.197** ± .001