

CURP: Codebook-based Continuous User Representation for Personalized Generation with LLMs

Liang Wang^{1*}, Xinyi Mou^{1*}, Xiaoyou Liu¹, Xuanjing Huang³, Zhongyu Wei^{1,2†}

¹School of Data Science, Fudan University

²Shanghai Innovation Institute

³School of Computer Science, Fudan University

liangwang25@m.fudan.edu.cn, {xymou20, zywei}@fudan.edu.cn

Abstract

User modeling characterizes individuals through their preferences and behavioral patterns to enable personalized simulation with Large Language Models (LLMs) in contemporary approaches. However, existing methods, whether personalized prompt-based or parameter-based methods, face challenges in balancing personalization quality against computational and data efficiency. We propose a novel framework CURP, which employs a bidirectional user encoder, a discrete prototype codebook and a dual-aggregator to extract both multi-dimensional user traits and query-aware user characteristic. This design enables plug-and-play personalization with a small number of trainable parameters (about 28M parameters, about 0.36% of the total model size). Through extensive experiments, we show that CURP achieves superior performance and generalization compared to strong baselines on the language model personalization (LaMP) benchmark, while offering better interpretability and robustness.¹

1 Introduction

User simulation aims to capture individual preferences and behaviors, enabling systems both to generate personalized content and interactions and to investigate population-level social dynamics (Mou et al., 2026, 2024; Xi et al., 2025; Tseng et al., 2024). Traditional user modeling typically relies on feature engineering and representation learning to perform coarse-grained attribute classification, limiting the ability to capture more complex user behaviors and preferences (Purificato et al., 2024; He et al., 2023). In contrast, LLMs can generate personalized content that reflects individual preferences, with promising applications in dialogue

¹Code repository: https://github.com/RaidonWong/CURP_code.

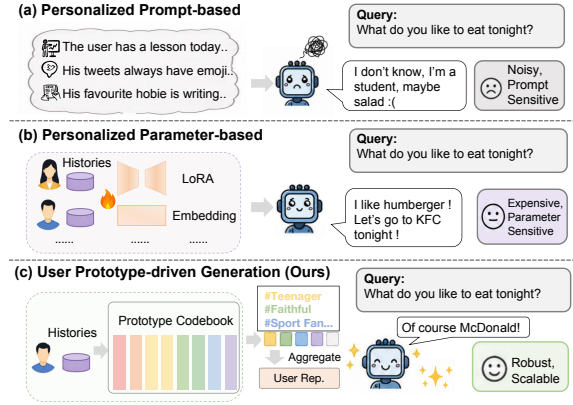


Figure 1: Different paradigms for user response simulation. (a) *Personalized Prompt-based*: prompting LLMs with text-based user behaviors, generalizable but noisy; (b) *Personalized Parameter-based*: Finetuning parameters for each user, accurate but expensive; (c) *Ours*: adopting prototype fusion based user representation via a codebook and an aggregator to guide user content generation, achieve a balance between two paradigms.

systems, recommendation, and healthcare interventions (He et al., 2025; Deng et al., 2024; Li et al., 2024a; Bao et al., 2024).

Current approaches to LLM-based user simulation fall into two paradigms: *Personalized Prompts* and *Personalized Parameters*, as illustrated in Figure 1(a) and (b) (Kumar et al., 2024; Zhang, 2024; Sun et al., 2025). Personalized prompt-based method injects histories and sometimes personas directly into the prompt to condition generation without modifying the model. It is lightweight and general-purpose, but pure descriptions often contain noise and redundancy, leading to imprecise and stereotyped outputs. In contrast, personalized parameter-based method adapts model weights or steers the latent representation using user-specific data, thereby internalizing personalized traits. Although high fidelity is achieved, fine-tuning a separate model per user is computationally prohibitive and impractical at scale. These

limitations expose a fundamental tension between efficiency and personalization fidelity, raising a central question: **How can we achieve faithful user simulation while remaining computationally efficient and scalable?**

Self-Categorization Theory (SCT) (Turner et al., 1987) posits that identity is formed through a context-sensitive blend of shared social prototypes, like *Teenager*, *Faithful*, and *Sport Fan*, rather than as a fixed and isolated self-concept. This theory provides a psychologically grounded and structured framework for user modeling, where identity is represented as meaningful and reusable prototypes. Conceptually motivated by SCT, we propose to model users in a shared latent prototype space, implemented as learnable prototypes (Van Den Oord et al., 2017; Deng et al., 2025; Arik and Pfister, 2020), where each entry captures a salient behavioral or preference pattern. As shown in Figure 1(c), a user might be encoded as a fusion of *#Teenager* + *#Faithful* + *#Sport Fan*. Notably, the shared space enables learning from aggregated signals across users, mitigating data sparsity while preserving individual differences through flexible prototype blending. This latent abstraction also improves robustness to noisy profile descriptions by mapping diverse user expressions into a common semantic space.

In this paper, we introduce **CURP** (Codebook-based Continuous User Representation for Personalized Generation with LLMs), a prototype-based framework for noise-robust and data-efficient personalized generation. At the core of CURP is a learnable **codebook** and a **dual-aggregator**. CURP first encodes and maps user interaction histories into sparse combinations of shared prototypes with the codebook. The dual-aggregator then fuses these selected prototypes into complementary user representations, capturing both stable long-term preferences and dynamic query-specific intent. The resulting personalized representation is subsequently projected and incorporated into the task prompt for personalized generation. Based on this, training proceeds in two stages: the **PCC** stage constructs codebooks from a large-scale behavioral pool, while the **UBA** stage learns the dual-aggregator and a lightweight adapter that maps the user representation into a frozen LLM’s space for personalized generation. To validate our approach, we conduct experiments across six personalized tasks on LaMP benchmark (Salemi et al., 2024). CURP consistently outperforms

strong baselines, with ablation studies revealing key insights into its mechanisms. Our main contributions are summarized as follows:

- (1) We propose a novel framework that encodes each user as fused denoised prototypes and query-aware embeddings, replacing fragile textual descriptions and per-user parametric training.
- (2) We introduce a two-stage training framework that enables the model to understand and utilize dual user representations without additional LLM training, making our approach model-agnostic.
- (3) Extensive experiments across diverse personalization tasks demonstrate the effectiveness, scalability, and generalization ability of CURP, which achieves a practical balance between personalization fidelity and efficiency.

2 Related Works

2.1 User Modeling

Prior to LLMs, user representations were typically constructed from explicit personas (e.g., questionnaires, self-disclosed attributes) or implicit behavioral signals, processed into vectors using TF-IDF (Nanas et al., 2003; Elkahky et al., 2015; Purificato et al., 2024) for traditional machine learning or word embeddings (An et al., 2019; Fazelnia et al., 2022) for deep learning. However, these approaches were largely limited to classification tasks and lacked generative capacity.

Recent advances in LLMs have enabled user simulation and personalized generation conditioned on user representations, either as natural language or learned embeddings (Chen et al., 2024; Li et al., 2024b; Huber et al., 2025), moving beyond a generic one-size-fits-all paradigm toward more structured modeling of user characteristics (Xu et al., 2025; Wang et al., 2023; Woźniak et al., 2024).

2.2 LLM-based Personalization

Personalization methods for LLMs generally fall into two paradigms: *prompt-based* approaches incorporate user profiles or interaction histories directly into the input context (Richardson et al., 2023; Salemi et al., 2024; Zhang, 2024), but often struggle to interpret noisy signals at scale. *Parameter-based* methods adapt model parameters via parameter-efficient fine-tuning (Tan et al., 2024b,a; Zhang et al., 2024) or personalized reward modeling and user-level alignment (Li et al., 2024b; Bose et al., 2025; Li et al., 2026), albeit with

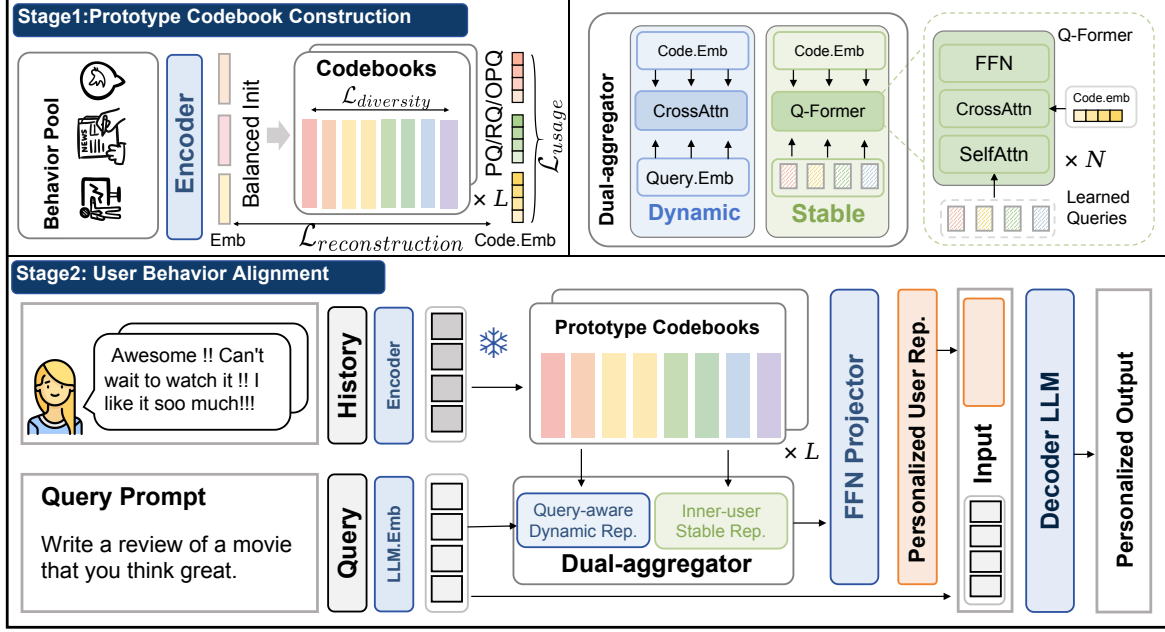


Figure 2: The overall framework of our proposed CURP. Our training procedure can be divided into two stages: Prototype Codebook Construction (PCC) and User Behaviour Aligning (UBA). Rep. is short for Representation.

higher computational costs. Recently, increasing attention has turned to *latent space personalization*, where user features are encoded as soft prompts or adapters (Doddapaneni et al., 2024; Jeong et al., 2024; Hebert et al., 2024; Huber et al., 2025; Zhang et al., 2025b). While some approaches employ discrete codebooks to represent semantic patterns in user utterances (He et al., 2025; Tang et al., 2024), these codebooks focus primarily on dialog policy rather than capturing characteristics. PHF (Wang et al., 2026) further models user behavior hierarchically, both over time and across users, which fundamentally relies on high-quality and stable codebook-based practice representations. These studies highlight that developing interpretable, robust, and scalable user representations is both fundamental to effective personalization and remains a central open challenge.

3 Methodology

In this section, we first define the task and provide a detailed description of the proposed framework and training procedure. Given a set of users $\mathcal{U} = \{u_1, u_2, \dots, u_n\}$, each user u_i is defined as a tuple $u_i = \langle \mathcal{H}_i, \mathcal{Q}_i, \mathcal{R}_i \rangle$. Here, $\mathcal{H}_i$ represents the user’s historical behaviors; $\mathcal{Q}_i$ is the current query for a scenario context; and $\mathcal{R}_i$ is the corresponding ground-truth from the user. Our goal is to leverage LLMs to accurately predict $\mathcal{R}_i$, given $\mathcal{H}_i$ and $\mathcal{Q}_i$.

3.1 Structure of CURP

Figure 2 presents an overview of our proposed framework, which comprises four key components: the *User Encoder* $\mathcal{E}$, the *Prototype Codebook* $\mathcal{C}$, the *Dual-Aggregator* $\mathcal{A}$ and the *Decoder LLM* $\mathcal{D}$.

3.1.1 User Encoder

To encode user historical behaviors, the fundamental source for personalization, we introduce a user encoder $\mathcal{E}$ that distills a user’s histories into a dense tensor representation. Unlike prior works that either couple the encoder with a task-specific adapter (Hebert et al., 2024; Doddapaneni et al., 2024) or require extensive task-specific data (Ning et al., 2025), our encoder is built upon a pretrained encoder-only architecture with bidirectional attention, ensuring strong contextual modeling capacity while maintaining generalizability across diverse downstream tasks. We encode historical behaviors using the encoder to obtain a unified representation of the user’s history. This design addresses the limitations of naively concatenating raw texts, which introduces noise, expands context length, and fails to capture structured user characteristics.

3.1.2 Codebook

The codebook $\mathcal{C}$ is designed to achieve denoised structured behavior prototype for further aggregation. To construct this codebook, we first collect

historical behaviors from a large-scale user pool, encoding them into dense tensors using the user encoder. These dense embeddings are then quantized through Product Quantization (PQ) (Jegou et al., 2011) into L discrete indices from codebook vocabularies: each user event embedding is decomposed into L subspaces, and each subspace is independently quantized to select an index from its own codebook. Through this process, each user event is abstracted as a combination of L prototype essence. This design enables effective user modeling through the composition of learned prototypes, where each codebook entry captures a distinct aspect of user characteristics, eliminating the need for raw textual inputs, and provide informative but denoised representations for further aggregation.

3.1.3 Dual-aggregator

User preferences often exhibit stable long-term characteristics and dynamic context-dependent intent (Mou et al., 2023). Modeling both aspects with a single representation may lead to information entanglement and reduced personalization quality. Therefore, we introduce a dual-aggregator $\mathcal{A}$ to decompose user modeling into two complementary branches: *dynamic* and *stable*. Let $\mathbf{BP}_i = \{\mathbf{bp}_{i,1}, \mathbf{bp}_{i,2}, \dots, \mathbf{bp}_{i,LJ}\}$ denotes all behavior prototypes for user u_i . To capture user interests related to the current query, we employ a Multi-Head Cross-Attention mechanism (Vaswani et al., 2017) that fuses the $\mathbf{BP}_i$ with the encoded task query Q_i , to acquire the query-aware dynamic user representation. Meanwhile, a stable user representation is constructed by compressing historical prototypes of varying lengths into a fixed-length vector, thereby preserving multi-dimensional cross-user characteristics. To achieve this, we utilize a multi-layer Q-Former (Li et al., 2023) equipped with M shared learnable queries. This dual-branch design effectively handles histories of arbitrary length while balancing the preservation of intrinsic user stability with query-specific task performance.

3.1.4 Decoder

To bridge the gap between the aggregated user representation space and the LLM decoder’s embedding space, we employ a Multi-Layer Perceptron (MLP) to project the constructive embeddings into the decoder’s embedding space. With the task query Q_i embedded via the decoder’s input embedding layer, we combine the projected user embeddings with the query embeddings and feed the com-

bined sequence into the decoder to perform personalized conditional generation, predicting the personalized response R_i . Our framework is decoder-agnostic, imposing no architectural constraints on the choice of decoder, which enables flexible application to various decoder architectures. The decoupled design that separates the codebook from the LLM also enables practical applications in cloud-edge collaboration and user privacy protection.

3.2 Training CURP

Based on the architecture described above, we implement a two-stage training pipeline: **Prototype Codebook Construction (PCC)** and **User Behavior Aligning (UBA)**.

3.2.1 Stage 1: Prototype Codebook Construction

For all users $u_i \in \mathcal{U}$, we collect historical behaviors $\mathcal{H}_i$ from dataset $\mathcal{D}$ and encode them using the frozen encoder $\mathcal{E}$ to construct an embedding pool:

$$\mathcal{P} = \{\mathcal{E}(h_{i,j}) \mid h_{i,j} \in \mathcal{H}_i, \forall u_i \in \mathcal{U}\}. \quad (1)$$

We partition each embedding $\mathbf{e} \in \mathbb{R}^d$ into L subspaces: $\mathbf{e} = [\mathbf{e}^{(1)}, \dots, \mathbf{e}^{(L)}]$, where $\mathbf{e}^{(l)} \in \mathbb{R}^{d/L}$. We initialize L codebooks $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_L\}$ by applying balanced k-means algorithm (Deng et al., 2025; Chen et al., 2025) for each subspaces to ensure uniform cluster sizes:

During quantization, each subspace embedding $\mathbf{e}^{(l)}$ is quantized by corresponding codebook:

$$\mathbf{e}_q = [\text{Quant}(\mathbf{e}^{(1)}, \mathcal{C}_1), \dots, \text{Quant}(\mathbf{e}^{(L)}, \mathcal{C}_L)], \quad (2)$$

where $\text{Quant}(\mathbf{e}^{(l)}, \mathcal{C}_l) = \arg \min_{\mathbf{c} \in \mathcal{C}_l} \|\mathbf{e}^{(l)} - \mathbf{c}\|^2$. We use PQ as the default quantizer for a simple and efficient instantiation; the same framework can replace it with stronger quantizers such as Residual Quantization (RQ) or Optimized Product Quantization (OPQ). More details can be found in Appendix A.3. The training loss consists of three components:

$$\mathcal{L}_{\text{PCC}} = \alpha \mathcal{L}_{\text{reconstruction}} + \beta \mathcal{L}_{\text{diversity}} + \gamma \mathcal{L}_{\text{usage}}, \quad (3)$$

where $\mathcal{L}_{\text{reconstruction}}$ measures reconstruction error between original $\mathbf{e}$ and quantized $\mathbf{e}_q$, $\mathcal{L}_{\text{diversity}}$ and $\mathcal{L}_{\text{usage}}$ are used to encourage codebook diversity and usage rate to prevent codebook collapse.

More details for balanced K-Means and loss function can be found in Appendix A.1 and A.2.

3.2.2 Stage 2: User Behavior Alignment

For each user u_i with historical behaviors $\mathcal{H}_i = \{h_{i,1}, h_{i,2}, \dots, h_{i,J}\}$, we independently encode and quantize each historical item $h_{i,j}$ as follows:

$$\mathbf{e}_{i,j} = \mathcal{E}(h_{i,j}), \quad \mathbf{p}_{i,j} = PQ(\mathbf{e}_{i,j}, \mathcal{C}), \quad (4)$$

where $\mathcal{E}(\cdot)$ denotes the encoder, and $PQ(\cdot, \mathcal{C})$ represents Product Quantization using the codebooks $\mathcal{C}$ learned in Stage 1. Here, $\mathbf{p}_{i,j}$ denotes the quantized prototype embedding for $h_{i,j}$.

After that, we aggregate the prototypes into the merged dynamic and stable representations:

$$\mathbf{R}_{dyn} = \text{MHCA}(\text{BP}_i, \mathcal{E}(\mathcal{Q}_i)), \quad (5)$$

where **MHCA** denotes the Multi-Head Cross-Attention module.

$$\mathbf{R}_{stb} = \text{Q-Former}(\text{BP}_i, \mathbf{Q}_{learned}), \quad (6)$$

where $\mathbf{Q}_{learned}$ represents M learnable queries.

Finally, the dual representations are projected into the decoder’s embedding space via MLPs (f) and prepended to the task query embedding. The final input sequence $\mathbf{x}_{in}$ is formulated as:

$$\mathbf{x}_{in} = [f(\mathbf{R}_{dyn}, \mathbf{R}_{stb}); \mathcal{D}_{Emb}(\mathcal{Q}_i)], \quad (7)$$

where $\mathcal{D}_{Emb}$ denotes the embedding layer of the $\mathcal{D}$. The training objective maximizes the likelihood of generating the personalized response Y_i :

$$\mathcal{L}_{UBA} = - \sum_{t=1}^{|Y_i|} \log P(y_t | \mathbf{x}_{in}, y_{<t}; \theta), \quad (8)$$

where θ represents the trainable parameters, y_t is the t -th token of the target response, and $|Y_i|$ is the total length of the response.

3.3 Training details

For the **PCC** stage, we randomly sample 100k user behaviors to balanced initialize the prototype codebooks $\mathcal{C}$. Each codebook consists 1,000 prototype entries for 4 subspaces by product quantization.

For the **UBA** stage, we use the whole histories for each user. We employ Contriever (Izacard et al., 2021) as the base encoder $\mathcal{E}$ and Qwen-2.5-7B-Instruct (Yang et al., 2024) as the decoder $\mathcal{D}$. We apply a 8-head MHCA for dynamic representation and a 2 layer Q-Former with 4 learnable queries for stable representation. A 2-layer MLP is used to project prototype embeddings into the LLM’s embedding space. More details can be found in Appendix D.

4 Experimental Setup

4.1 Datasets

We conduct experiments on six tasks from the LaMP benchmark (Salemi et al., 2024), namely LaMP-1–5 and LaMP-7, following previous studies (Tan et al., 2024b,a). LaMP-6 is excluded because its original Avocado email data are not publicly accessible due to privacy restrictions. The six evaluated tasks comprise three classification/regression tasks and three generation tasks: (1) LaMP-1: Citation Identification; (2) LaMP-2: Movie Tagging; (3) LaMP-3: Product Rating; (4) LaMP-4: News Headline Generation; (5) LaMP-5: Scholarly Title Generation; (6) LaMP-7: Tweet Paraphrasing.

4.2 Evaluation Metrics

We use the default metrics align with LaMP benchmark to evaluate the performance of each tasks: accuracy (Acc) for LaMP-1, accuracy and F1 for LaMP-2, mean absolute error (MAE) and root mean squared error (RMSE) for LaMP-3, and ROUGE-1 (R-1) and ROUGE-L (R-L) for LaMP-4, LaMP-5, and LaMP-7. Besides, apart from the surface overlap ROUGE, we report the BERTScore (Sim) for semantic overlap for a better evaluation.

4.3 Comparison Methods

We compare CURP against a range of baselines, mainly within two broad categories: Personalized Prompt and Personalized Parameters.

Personalized Prompt-based Methods : These methods avoid parameter updates, providing efficiency at the cost of long context. (1) **Zero Shot (ZS):** The LLM generates outputs solely based on the current input, serving as a non-personalized baseline. (2) **In-Context Learning (ICL):** Historical user behaviors (default 8) as reference are appended to the prompt to personalize the LLM. (3) **BM25:** BM25 algorithm to retrieve the most similar histories with the current query via surface overlap. (4) **Dense:** Retrieve the most similar histories with query by dense Contriever embedding. (5) **Chain-of-Thought (CoT)** (Wei et al., 2022): Prompts encourage step-by-step reasoning based on histories. (6) **PAG** (Zhang, 2024): A persona profile is generated by Qwen-2.5-14B from the user’s histories and prepended to the prompt.

Personalized Parameter-based Methods: These methods update parameters by supervision.

Task	Metric	Zero Shot	Personalized prompt					Personalized parameters				Ours
			ICL	BM25	Dense	CoT	PAG	DiffMean	PROPER	SP	PPlug	CURP
Lens & Paras		L	9L	9L	9L	9L	2L	L&4k*	L&8M*	L&4k	L&40M	L&28M
LaMP-1	Acc↑	0.514	<u>0.557</u>	0.579	0.555	0.466	0.534	-	-	0.504	0.502	0.518
LaMP-2	Acc↑	0.345	0.445	0.431	0.448	0.423	0.384	0.345	<u>0.587</u>	0.351	0.565	0.629
	F1↑	0.191	0.291	0.281	0.300	0.080	0.110	0.182	<u>0.504</u>	0.200	0.485	0.547
LaMP-3	MAE↓	0.502	0.368	0.314	0.284	0.381	0.577	0.580	0.284	0.524	0.258	<u>0.276</u>
	RMSE↓	0.839	0.714	0.651	0.604	0.725	0.940	0.923	0.597	0.867	0.567	<u>0.585</u>
LaMP-4	R-1↑	0.138	0.160	0.168	0.177	0.143	0.136	0.136	<u>0.191</u>	0.108	0.190	0.194
	R-L↑	0.119	0.140	0.151	0.159	0.126	0.117	0.118	<u>0.171</u>	0.092	0.168	0.173
	Sim↑	0.845	0.852	0.854	0.855	0.849	0.842	0.844	<u>0.858</u>	0.829	0.857	0.859
LaMP-5	R-1↑	0.411	0.432	0.440	0.435	0.448	0.376	0.396	0.469	0.292	<u>0.500</u>	0.505
	R-L↑	0.338	0.360	0.371	0.364	0.377	0.309	0.324	0.399	0.254	<u>0.444</u>	0.449
	Sim↑	0.876	0.884	0.885	0.884	0.887	0.870	0.874	0.890	0.864	<u>0.897</u>	0.898
LaMP-7	R-1↑	0.457	0.428	0.446	0.447	0.364	0.416	0.463	-	0.451	<u>0.536</u>	0.543
	R-L↑	0.401	0.375	0.390	0.392	0.322	0.363	0.406	-	0.395	<u>0.477</u>	0.483
	Sim↑	0.903	0.894	0.898	0.898	0.881	0.889	0.900	-	0.899	<u>0.909</u>	0.911

Table 1: Performance of all baselines and our model on six tasks. The best results are shown in **bold**, and the second-best are underlined. The reported results represent the average of three independent runs. ↑ means higher is better. In Lens & Paras, for personalized prompt methods we provide the relative average input length; for personalized parameters method we provide the number of personalized parameters as well. - indicates that the method is not adaptive to data formats. * denotes user-specific parameters.

(1) **DiffMean** (Zhang et al., 2025a): DiffMean represents each user as a vector in LLM’s activation space for steered generation. (2) **PROPER** (Zhang et al., 2025b): PROPER further fine-tunes a personalized LoRA module for each user at varying granularities. While this approach maximizes the utilization of personal information, it incurs substantial computational costs, which are known for their high resource demands. (3) **Soft Prompt (SP)** (Lester et al., 2021): Soft Prompt learns a shared trainable embedding prepended to the input sequence while keeping the model frozen. (4) **PPlug** (Liu et al., 2025): A soft prompt tuning based approach that aggregates the user’s entire history into a single token via softmax weights derived from its similarity to the current query, functioning similarly to our dynamic representation.

5 Experiment Result

5.1 Main Results

The main results of inference on diverse scenarios are reported in Table 1, and we can observe that:

(1) **Comparison of different methods**: Overall, CURP achieves the strongest overall performance by jointly modeling stable intrinsic user representations and query-aware dynamic adaptation. Personalized Prompt based methods like ICL, CoT and PAG yield suboptimal results, e.g. perform even worse than non-personalized ZS (R-1 0.457, R-L 0.401) in LaMP-7, primarily due to inflated context lengths and their sensitivity to noise in raw user his-

tories, which also imply overthinking does not help personalization. While training-based methods improve performance, they incur substantial parameter costs for personalization. For example, although PROPER achieves competitive results in LaMP-2 and LaMP-4, it requires 8M parameters per user and the time efficiency is relatively much lower. DiffMean and SP perform way behind other baselines, and perform worse than ZS on LaMP-4 and LaMP-5, showing steering method and one general embedding hard to achieve personalization. In contrast, our approach attains state-of-the-art results on more than 70% of metrics across all tasks, with only about 30M parameters. This demonstrates a superior trade-off: **CURP matches or exceeds the performance of heavier models while maintaining a compact and scalable architecture.**

(2) **Analysis across tasks**: The effectiveness of personalization methods varies with task characteristics, yet our approach delivers robust and leading performance consistently across diverse task types. On classification tasks, methods that better aggregate users’ preferences regarding current query perform better. For instance, RAG-based methods BM25 and Dense show competitive performance even with training-based methods. In contrast, on generation tasks, the difference between methods is smaller than in classification tasks, while training methods PROPER, PPlug and CURP show a large advance than prompting-based methods. **Overall, our CURP model perform competitive performance in both task types.**

Task	LaMP-2		LaMP-3		LaMP-4		LaMP-7	
Metric	Acc $\uparrow$	F1 $\uparrow$	MAE $\downarrow$	RMSE $\downarrow$	R-1 $\uparrow$	R-L $\uparrow$	R-1 $\uparrow$	R-L $\uparrow$
CURP	0.629	0.547	0.276	0.585	0.194	0.173	0.543	0.483
w/o Quantize	0.634	0.548	0.275	0.601	0.198	0.177	0.544	0.483
w/o Dynamic	0.474	0.367	0.290	0.596	0.191	0.170	0.538	0.480
w/o Stable	0.539	0.436	0.287	0.607	0.188	0.168	0.528	0.469
w/o Q-Former	0.590	0.541	0.290	0.620	0.192	0.171	0.536	0.477
Dual MLP	0.604	0.558	0.282	0.605	0.194	0.175	0.545	0.485

Table 2: Overall performance of ablation models.

5.2 Ablation Study

Ablation on the Framework Component We first validate the necessity of each component through ablation studies by comparing performance after removing the corresponding module (Table 2). Removing codebook quantization (*w/o Quantize*) slightly improves performance but compromises robustness due to continuous embeddings (Section 5.3.1). Appendix E further replaces PQ with RQ and OPQ. Stronger quantizers avoid degradation from information loss and can surpass both PQ and the continuous variant by filtering nuisance variation. Eliminating either the dynamic or stable representation (*w/o Dynamic/Stable*) causes clear degradation, confirming their complementary benefits. Replacing the Q-Former with mean pooling for stable representation (*w/o Q-Former*) also yields suboptimal performance, showing that expressive feature fusion better captures user characteristics. Using separate MLPs (*Dual MLP*) for different representations offers no noticeable gains while adding parameters. Therefore, we adopt a shared MLP as the projection module. Overall, each component plays an essential role and collectively contributes to the framework’s effectiveness. Appendix I additionally explores collapsing the dynamic and stable branches into a single embedding, which does not improve downstream performance.

Ablation on Codebook Design Choices We further analyze how different design decisions affect codebook utilization and combination diversity. As illustrated in Figure 3, we compare initialization strategies (e.g. Random), loss designs, vocabulary sizes (e.g. CB-500), and PQ subspace configurations (e.g. PQ1). Random initialization results in extremely low usage and can lead to codebook collapse, whereas balanced initialization with PQ4 and the proposed loss enables effective exploration of the codebook space. Overall, our final design (Balanced+PQ4+Loss) achieves both high entry usage, substantial combination diversity and cost-effective, outperforming alternatives significantly. Appendix F further reports the downstream performance of different subspace counts, codebook

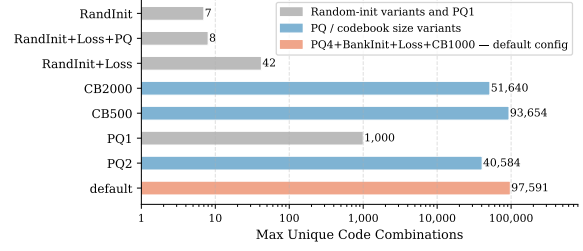


Figure 3: Maximum number of unique PQ code combination tuples observed across all training epochs for each codebook configuration on LaMP-2.

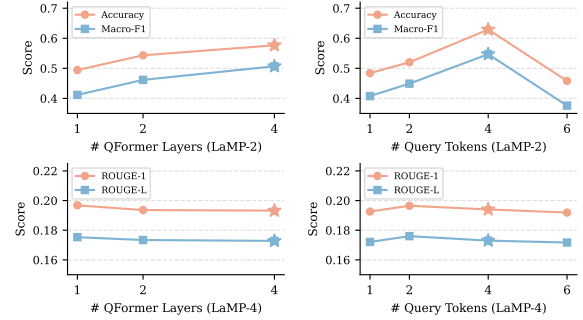


Figure 4: Ablation study on the number of Q-Former layers and learned queries on LaMP-2 and LaMP-4.

sizes, and initialization schemes.

Ablation on Q-Former Layers and Learned Queries Finally, we investigate the impact of the number of Q-Former layers and learned queries on stable representation learning. As shown in Figure 4, for LaMP-2, the performance consistently improves as the number of Q-Former layers increases, whereas for LaMP-4, deeper Q-Former architectures lead to noticeable performance degradation. Regarding the number of learned queries, LaMP-2 achieves the best performance when the query number is set to 4, while the performance drops sharply when increasing it to 6. In contrast, LaMP-4 performs best with 2 learned queries, followed by a gradual decline as the query number increases. Considering the trade-off across different tasks, we adopt a configuration with 2 Q-Former layers and 4 learned queries as the default setting.

5.3 Further Analysis

We further conduct additional analysis on LaMP-2 (classification) and LaMP-4 (generation).

5.3.1 Robustness to Profile Paraphrase

In real-world applications, user profiles often come from diverse sources and may vary in wording or format, despite conveying the same underlying

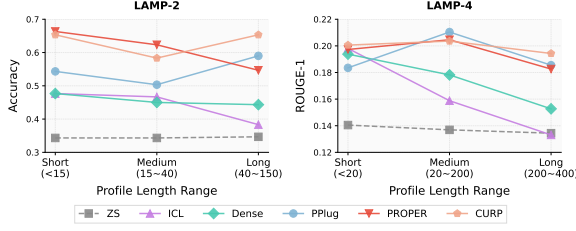


Figure 5: Performance of CURP with users with different history length.

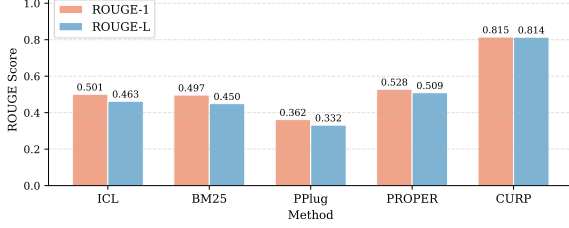


Figure 6: Robustness of personalized generation under profile paraphrase on LaMP-4.

ing preferences. A robust personalization method should therefore rely on underlying semantic content for stable user understanding, rather than being distracted by surface-level lexical variations. To evaluate this, we design a paraphrase robustness test on LaMP-4 using 100 randomly sampled users. Specifically, we rewrite each user’s profile history using Qwen2.5-8B-Instruct while preserving its original meaning, and then run each method on both the original and paraphrased profiles. As shown in Figure 6, CURP achieves the highest consistency, demonstrating strong robustness to profile paraphrasing. This benefits from the quantization mechanism that encourages user representations to capture semantic-level patterns rather than superficial lexical features. In contrast, other methods achieve moderate consistency, indicating partial sensitivity to wording changes. Overall, these results demonstrate **CURP learns a more semantically grounded and robust user representation.**

5.3.2 Interpretability Analysis of Codebook

To demonstrate the interpretability of our codebook-based representation, we analyze PQ indices from the LaMP-4 task. We observe a hierarchically structured representation: (1) **Topic-level consistency:** We first group samples by topic, e.g., Donald Trump, and examine their PQ indices. We observe that all samples within the same topic consistently share identical indices in subspaces indexed 40 and 101. This recurring pattern indi-

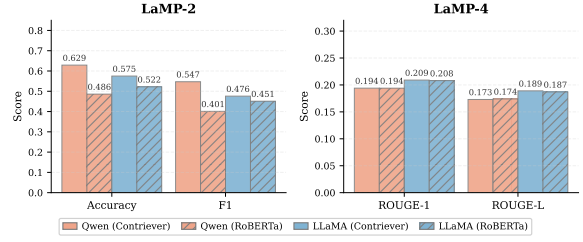


Figure 7: Generalization of CURP across different encoder and LLM backbones on LaMP-2 and LaMP-4.

cates that these subspaces function as a semantic anchor, capturing invariant, topic-level information. (2) **Within-topic variation:** Remaining subspaces capture nuanced semantic distinctions orthogonal to the topic. For instance, subspace 1 distinguishes content sources (e.g., legal proceedings vs. media coverage), while subspace 4 aligns with narrative framing (e.g., court challenges vs. personal impact). This indicates that our codebook effectively decomposes user profiles into stable topical cores and dynamic stylistic variations. Further visualizations and more details are provided in Appendix L.

5.3.3 Generalization to Different Backbones

To evaluate the generalization and modularity of CURP across different backbone models, we deploy the learned codebooks with three distinct $\mathcal{E} + \mathcal{D}$ combinations: *RoBERTa+Qwen*, *Contriever+LLaMA*, and *RoBERTa+LLaMA*. When replacing the decoder, only the UBA stage needs to be re-executed, whereas changing the encoder requires rerunning both stages. As shown in Figure 7, CURP achieves consistently competitive performance across all combinations, indicating that **the learned codebook captures generalizable user characteristics that are not tied to any specific encoder or decoder architecture.**

5.3.4 Scalability Across Different User History Lengths

To evaluate how personalization performance scales with increasing user history, we randomly sample 100 users with 3 different seeds from each of three profile-length groups (short, mid, and long), and guarantee similar difficulty by keeping ZS score stable, as shown in Table 5. ICL and Dense perform weaker as user history length grows, as their context window are limited and can receive more noise. PROPER performs best on short profiles, likely benefiting from its LoRA-based task adaptation under limited user history.

However, its advantage diminishes as profile length increases, likely because it weights all past interactions equally, failing to prioritize more relevant ones, and this may introduce noise and distort the direction. In contrast, PPlug struggles in the short-profile setting, due to its reliance on query-aware fusion, which becomes less effective when historical records are limited and misaligned. However, for CURP, **the codebook-based quantization compresses arbitrarily long user histories into compact and informative discrete representations while focusing on more relative histories in long history, allowing it to benefit consistently from additional history.** As a result, CURP achieves the best overall performance across both tasks.

6 Conclusion

In this work, we propose a novel CURP framework for personalized generation. To address the efficiency-fidelity trade-off in LLM-based personalized generation, we propose to model users as fused prototypes in both dynamic and stable representation. Experimental results demonstrate that our model achieves strong performance across multiple tasks, while further validating the generalization, interpretability, and robustness of our method.

Limitations

In this study, we propose a prototype codebook-based approach for personalized LLMs; however, several limitations remain and warrant future exploration. First, LaMP represents users primarily through behavior histories and does not provide direct persona descriptions; it also lacks realistic cross-domain behavior sequences. Second, discarding the original historical text can lose fine-grained behavioral patterns such as habitual word usage. Although the quantizer comparisons suggest that stronger quantizers preserve task-relevant signals while filtering noise, rare patterns far from learned prototypes may remain unrecoverable. Third, evaluating personalization remains difficult. Following previous work (Salemi et al., 2024; Kumar et al., 2024; Liu et al., 2025; Zhang et al., 2025a), we report ROUGE-1, ROUGE-L, and BERTScore to provide a comprehensive reference-based assessment, but these metrics cannot deeply measure the degree of personalization. We additionally report an LLM-as-a-judge study in Appendix J; however, automated LLM judges can introduce model-specific biases and cannot replace human preference evaluation.

Larger-scale human studies and evaluations on realistic cross-domain histories remain important future work.

Ethical Considerations

We address several ethical considerations in this work. First, all datasets used in our experiments are publicly available, and all models employed are open-source, ensuring that no user privacy information is involved in our research. We ensure that our experiments adhere to ethical guidelines and do not involve any ethical concerns.

Acknowledgments

The research is supported by National Key R&D Program of China (Grant No. 2023YFF1204800) and the AI for Science Program, Shanghai Municipal Commission of Economy and Informatization (Grant No. 2025-GZL-RGZN-BTBX-02028). The project’s computational resources are partially supported by CFFF platform of Fudan University.

We only use AI assistants as tools for paper refinement and code implementation assistance, and we do not engage in any practice of using AI to directly generate paper content.

References

- Mingxiao An, Fangzhao Wu, Chuhan Wu, Kun Zhang, Zheng Liu, and Xing Xie. 2019. [Neural news recommendation with long-and short-term user representations](#). In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 336–345.
- Sercan O Arik and Tomas Pfister. 2020. Protoattend: Attention-based prototypical learning. *Journal of Machine Learning Research*, 21(210):1–35.
- Zhijie Bao, Qingyun Liu, Ying Guo, Zhengqiang Ye, Jun Shen, Shirong Xie, Jiajie Peng, Xuanjing Huang, and Zhongyu Wei. 2024. Piors: Personalized intelligent outpatient reception based on large language model with multi-agents medical scenario simulation. *arXiv preprint arXiv:2411.13902*.
- Avinandan Bose, Zhihan Xiong, Yuejie Chi, Simon Shaolei Du, Lin Xiao, and Maryam Fazel. 2025. Lore: Personalizing llms via low-rank reward modeling. *arXiv preprint arXiv:2504.14439*.
- Ben Chen, Xian Guo, Siyuan Wang, Zihan Liang, Yue Lv, Yufei Ma, Xinlong Xiao, Bowen Xue, Xuxin Zhang, Ying Yang, Huangyu Dai, Xing Xu, Tong Zhao, Mingcan Peng, Xiaoyang Zheng, Chao Wang, Qihang Zhao, Zhixin Zhai, Yang Zhao, Bochao Liu, Jingshan Lv, Xiao Liang, Yuqing Ding, Jing Chen,

- Chenyi Lei, Wenwu Ou, Han Li, and Kun Gai. 2025. Onesearch: A preliminary exploration of the unified end-to-end generative framework for e-commerce search. *arXiv preprint arXiv:2509.03236*.
- Jiangjie Chen, Xintao Wang, Rui Xu, Siyu Yuan, Yikai Zhang, Wei Shi, Jian Xie, Shuang Li, Ruihan Yang, Tinghui Zhu, Aili Chen, Nianqi Li, Lida Chen, Caiyu Hu, Siye Wu, Scott Ren, Ziquan Fu, and Yanghua Xiao. 2024. [From persona to personalization: A survey on role-playing language agents](#). *Transactions on Machine Learning Research*.
- Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. 2025. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *arXiv preprint arXiv:2502.18965*.
- Yang Deng, Wenxuan Zhang, Wai Lam, See-Kiong Ng, and Tat-Seng Chua. 2024. [Plug-and-play policy planner for large language model powered dialogue agents](#). In *The Twelfth International Conference on Learning Representations*.
- Sumanth Doddapaneni, Krishna Sayana, Ambarish Jash, Sukhdeep Sodhi, and Dima Kuzmin. 2024. [User embedding model for personalized language prompting](#). In *Proceedings of the 1st Workshop on Personalization of Generative AI Systems (PERSONALIZE 2024)*, pages 124–131. Association for Computational Linguistics.
- Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. [A multi-view deep learning approach for cross domain user modeling in recommendation systems](#). In *Proceedings of the 24th international conference on world wide web*, pages 278–288.
- Ghazal Fazelnia, Eric Simon, Ian Anderson, Benjamin Carterette, and Mounia Lalmas. 2022. Variational user modeling with slow and fast features. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 271–279.
- Tao He, Lizi Liao, Yixin Cao, Yuanxing Liu, Yiheng Sun, Zerui Chen, Ming Liu, and Bing Qin. 2025. [Simulation-free hierarchical latent policy planning for proactive dialogues](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- Zhicheng He, Weiwen Liu, Wei Guo, Jiarui Qin, Yingxue Zhang, Yaochen Hu, and Ruiming Tang. 2023. [A survey on user behavior modeling in recommender systems](#). In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 6656–6664.
- Liam Hebert, Krishna Sayana, Ambarish Jash, Alexandros Karatzoglou, Sukhdeep Sodhi, Sumanth Doddapaneni, Yanli Cai, and Dima Kuzmin. 2024. Persoma: Personalized soft prompt adapter architecture for personalized language prompting. *arXiv preprint arXiv:2408.00960*.
- Bernd Huber, Ghazal Fazelnia, Andreas Damianou, Sebastian Peleato, Maksym Lefarov, Praveen Chandar, Marco De Nadai, Mounia Lalmas, and Paul N. Bennett. 2025. Embedding-to-prefix: Continual personalization with large language models. In *AI That Keeps Up: NeurIPS 2025 Workshop on Continual and Compatible Foundation Model Updates*.
- Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.
- Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2011. [Product quantization for nearest neighbor search](#). *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128.
- Jihwan Jeong, Yinlam Chow, Guy Tennenholtz, Chih-Wei Hsu, Mohammad Ghavamzadeh, and Craig Boutilier. 2024. Factual and tailored recommendation endorsements using language models and reinforcement learning. In *First Conference on Language Modeling*.
- 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. 2024. Longlamp: A benchmark for personalized long-form text generation. *arXiv preprint arXiv:2407.11016*.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. [The power of scale for parameter-efficient prompt tuning](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059. Association for Computational Linguistics.
- Jia-Nan Li, Jian Guan, Songhao Wu, Wei Wu, and Rui Yan. 2026. [From 1,000,000 users to every user: Scaling up personalized preference for user-level alignment](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30144–30168. Association for Computational Linguistics.
- Junkai Li, Yungwei Lai, Weitao Li, Jingyi Ren, Meng Zhang, Xinhui Kang, Siyu Wang, Peng Li, Ya-Qin Zhang, Weizhi Ma, and Yang Liu. 2024a. Agent hospital: A simulacrum of hospital with evolvable medical agents. *arXiv preprint arXiv:2405.02957*.
- Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR.
- Xinyu Li, Ruiyang Zhou, Zachary C Lipton, and Liu Leqi. 2024b. Personalized language modeling from personalized human feedback. *arXiv preprint arXiv:2402.05133*.

- Jiongnan Liu, Yutao Zhu, Shuting Wang, Xiaochi Wei, Erxue Min, Yu Lu, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. 2025. [LLMs + persona-plugin = personalized LLMs](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.
- Xinyi Mou, Xuanwen Ding, Qi He, Liang Wang, Jingcong Liang, Xinnong Zhang, Libo Sun, Jiayu Lin, Jie Zhou, Xuanjing Huang, and Zhongyu Wei. 2026. [From individual to society: A survey on social simulation driven by large language model-based agents](#). *ACM Computing Surveys*, 58(11):280:1–280:41.
- Xinyi Mou, Zhongyu Wei, and Xuanjing Huang. 2024. [Unveiling the truth and facilitating change: Towards agent-based large-scale social movement simulation](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4789–4809. Association for Computational Linguistics.
- Xinyi Mou, Zhongyu Wei, Qi Zhang, and Xuan-Jing Huang. 2023. [Uppam: A unified pre-training architecture for political actor modeling based on language](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11996–12012. Association for Computational Linguistics.
- Nikolaos Nanas, Victoria Uren, and Anne De Roeck. 2003. [Building and applying a concept hierarchy representation of a user profile](#). In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 198–204. ACM.
- Lin Ning, Luyang Liu, Jiaxing Wu, Neo Wu, Devora Berlowitz, Sushant Prakash, Bradley Green, Shawn O’Banion, and Jun Xie. 2025. [User-llm: Efficient llm contextualization with user embeddings](#). In *Companion Proceedings of the ACM on Web Conference 2025*, pages 1219–1223.
- Erasmus Purificato, Ludovico Boratto, and Ernesto William De Luca. 2024. User modeling and user profiling: A comprehensive survey. *arXiv preprint arXiv:2402.09660*.
- Chris Richardson, Yao Zhang, Kellen Gillespie, Sudipta Kar, Arshdeep Singh, Zeynab Raeesy, Omar Zia Khan, and Abhinav Sethy. 2023. Integrating summarization and retrieval for enhanced personalization via large language models. *arXiv preprint arXiv:2310.20081*.
- Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2024. [LaMP: When large language models meet personalization](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7370–7392. Association for Computational Linguistics.
- Libo Sun, Siyuan Wang, and Zhongyu Wei. 2025. [Identity-driven hierarchical role-playing agents](#). In *Natural Language Processing and Chinese Computing – 14th National CCF Conference, NLPCC 2025, Proceedings, Part II*, volume 16103 of *Lecture Notes in Computer Science*, pages 403–417. Springer.
- Zhaoxuan Tan, Zheyuan Liu, and Meng Jiang. 2024a. [Personalized pieces: Efficient personalized large language models through collaborative efforts](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.
- Zhaoxuan Tan, Qingkai Zeng, Yijun Tian, Zheyuan Liu, Bing Yin, and Meng Jiang. 2024b. [Democratizing large language models via personalized parameter-efficient fine-tuning](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Miami, Florida, USA. Association for Computational Linguistics.
- Yihong Tang, Bo Wang, Dongming Zhao, Xiaojia Jin, Jijun Zhang, Ruifang He, and Yuexian Hou. 2024. [MORPHEUS: Modeling role from personalized dialogue history by exploring and utilizing latent space](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7664–7676. Association for Computational Linguistics.
- Yu-Min Tseng, Yu-Chao Huang, Teng-Yun Hsiao, Wei-Lin Chen, Chao-Wei Huang, Yu Meng, and Yun-Nung Chen. 2024. [Two tales of persona in LLMs: A survey of role-playing and personalization](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16612–16631. Association for Computational Linguistics.
- John C. Turner, Michael A. Hogg, Penelope J. Oakes, Stephen D. Reicher, and Margaret S. Wetherell. 1987. *Rediscovering the Social Group: A Self-Categorization Theory*. Basil Blackwell, Oxford, UK.
- Aaron Van Den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Liang Wang, Xinyi Mou, Xiaoyou Liu, Tiannan Wang, Yuqing Wang, and Zhongyu Wei. 2026. [Beyond isolated behaviors: Hierarchical user modeling for llm personalization](#). *arXiv preprint arXiv:2606.02300*.
- Wenjie Wang, Xinyu Lin, Fuli Feng, Xiangnan He, and Tat-Seng Chua. 2023. Generative recommendation: Towards next-generation recommender paradigm. *arXiv preprint arXiv:2304.03516*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le,

and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.

Stanisław Woźniak, Bartłomiej Koptyra, Arkadiusz Janz, Przemysław Kazienko, and Jan Kocoń. 2024. Personalized large language models. In *2024 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 511–520. IEEE.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, Qi Zhang, and Tao Gui. 2025. [The rise and potential of large language model based agents: A survey](#). *Science China Information Sciences*, 68(2):121101.

Yiyan Xu, Jinghao Zhang, Alireza Salemi, Xinting Hu, Wenjie Wang, Fuli Feng, Hamed Zamani, Xiangnan He, and Tat-Seng Chua. 2025. [Personalized generation in large model era: A survey](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24607–24649. Association for Computational Linguistics.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. [Qwen2.5 technical report](#). *arXiv preprint arXiv:2412.15115*.

Jiarui Zhang. 2024. Guided profile generation improves personalization with llms. *arXiv preprint arXiv:2409.13093*.

Jinghao Zhang, Yuting Liu, Wenjie Wang, Qiang Liu, Shu Wu, Liang Wang, and Tat-Seng Chua. 2025a. [Personalized text generation with contrastive activation steering](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7128–7141. Association for Computational Linguistics.

Kai Zhang, Yejin Kim, and Xiaozhong Liu. 2024. Personalized llm response generation with parameterized memory injection. *arXiv preprint arXiv:2404.03565*.

Linhai Zhang, Jialong Wu, Deyu Zhou, and Yulan He. 2025b. [Proper: A progressive learning framework](#)

[for personalized large language models with group-level adaptation](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16399–16411. Association for Computational Linguistics.

A Training Procedure Details

A.1 Detailed Balanced K-Means Algorithm

To ensure that the codebook entries are utilized uniformly and represent distinct semantic clusters, we employ a *Balanced K-Means* strategy for initialization. Unlike standard K-Means, which may suffer from codebook collapse (where few centroids attract most data points), our approach enforces a cardinality constraint on each cluster.

Formally, let $\mathcal{E}^{(l)} = \{\mathbf{e}_1^{(l)}, \dots, \mathbf{e}_N^{(l)}\}$ be the set of subspace embeddings for the l -th subspace. We aim to partition these embeddings into K clusters $\{\mathcal{P}_k^l\}_{k=1}^K$ by minimizing the within-cluster sum of squares:

$$\min_{\{\mathcal{P}_k^l\}_{k=1}^K, \{\boldsymbol{\mu}_k^l\}_{k=1}^K} \sum_{k=1}^K \sum_{n \in \mathcal{P}_k^l} \|\mathbf{e}_n^{(l)} - \boldsymbol{\mu}_k^l\|^2 \quad (9)$$

subject to the balance constraint:

$$|\mathcal{P}_k^l| = \left\lfloor \frac{N}{K} \right\rfloor \quad \text{or} \quad \left\lceil \frac{N}{K} \right\rceil, \quad \forall k \in \{1, \dots, K\}. \quad (10)$$

Here, $\boldsymbol{\mu}_k^l$ denotes the centroid of the k -th cluster in the l -th codebook. The balance constraint ensures that each codebook entry is responsible for an approximately equal number of training samples, thereby maximizing the information capacity of the discrete codebook and preventing "dead" codes. We solve this constrained optimization problem using a swap-based iterative algorithm, as detailed in Algorithm 1. For initialization, we run 100 iterations as default.

A.2 Detailed Codebook Loss Composition

The training objective for the codebook module consists of three complementary components: reconstruction fidelity, codebook diversity, and usage uniformity. The total loss is defined as:

$$\mathcal{L}_{\text{codebook}} = \alpha \mathcal{L}_{\text{reconstruction}} + \beta \mathcal{L}_{\text{div}} + \gamma \mathcal{L}_{\text{usage}} \quad (11)$$

Reconstruction Loss. To ensure that the quantized vectors faithfully approximate the original continuous embeddings, we minimize the squared

Algorithm 1 Balanced K-Means for Codebook Update

Require: Subspace embeddings $\mathcal{E}^{(l)} = \{\mathbf{e}_n^{(l)}\}_{n=1}^N$, Codebook centroids $\mathcal{C}^l = \{\boldsymbol{\mu}_k^l\}_{k=1}^K$, Max iterations T_{max}

Ensure: Updated centroids $\mathcal{C}^l$, Assignments $\{\mathcal{P}_k^l\}_{k=1}^K$

```
1: Initialize assignments  $\{\mathcal{P}_k^l\}_{k=1}^K$  randomly satisfying the balance constraint  $|\mathcal{P}_k^l| \approx N/K$ 
2: for  $t = 1$  to  $T_{max}$  do
3:   // Step 1: Update Centroids
4:   for  $k = 1$  to  $K$  do
5:      $\boldsymbol{\mu}_k^l \leftarrow \frac{1}{|\mathcal{P}_k^l|} \sum_{n \in \mathcal{P}_k^l} \mathbf{e}_n^{(l)}$ 
6:   end for
7:   // Step 2: Compute Distance Matrix
8:   Compute  $D \in \mathbb{R}^{N \times K}$  where  $D_{nk} = \|\mathbf{e}_n^{(l)} - \boldsymbol{\mu}_k^l\|^2$ 
9:   // Step 3: Balanced Re-assignment (Swap-based)
10:  Initialize priority queue  $Q$  with samples sorted by assignment cost
11:   $changes \leftarrow \text{true}$ 
12:  while  $Q \neq \emptyset$  and  $changes$  do
13:    Select sample  $n$  from  $Q$  assigned to cluster  $a$ 
14:    Find best alternative cluster  $b \neq a$  minimizing distortion while maintaining balance
15:    if swapping  $n$  from  $a$  to  $b$  reduces total distortion then
16:      Move  $n$  from  $\mathcal{P}_a^l$  to  $\mathcal{P}_b^l$ 
17:      Update  $\boldsymbol{\mu}_a^l$  and  $\boldsymbol{\mu}_b^l$  incrementally
18:       $changes \leftarrow \text{true}$ 
19:    else
20:       $changes \leftarrow \text{false}$ 
21:    end if
22:  end while
23:  if convergence criterion met then
24:    break
25:  end if
26: end for
27: return  $\mathcal{C}^l, \{\mathcal{P}_k^l\}_{k=1}^K$ 
```

Euclidean distance between the encoder output $\mathbf{e}_q$ and the selected codebook vector $\mathbf{e}$. Since the $\arg \min$ operation in vector quantization is non-differentiable, we employ the Straight-Through Estimator (STE) to allow gradient flow from the decoder to the encoder:

$$\mathcal{L}_{\text{reconstruction}} = \|\mathbf{e}_q - \text{sg}(\mathbf{e})\|^2 + \|\text{sg}(\mathbf{e}_q) - \mathbf{e}\|^2 \quad (12)$$

where $\text{sg}(\cdot)$ is the stop-gradient operator. The second term acts as a commitment loss, encouraging the encoder outputs to stay close to the codebook embeddings they select.

Diversity Loss. To prevent multiple codebook entries from converging to the same semantic center (redundancy), we introduce a diversity loss that maximizes the minimum pairwise distance between centroids. Directly optimizing the hard minimum

is unstable; thus, we use a smooth surrogate based on the hyperbolic tangent function:

$$\mathcal{L}_{\text{div}} = -\tau \cdot \tanh \left(\frac{1}{\tau} \min_{i \neq j} \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\| \right) \quad (13)$$

where τ is a temperature parameter controlling the smoothness of the approximation. This term encourages the codebook to span a larger volume in the latent space, improving its ability to represent out-of-distribution or diverse user profiles.

Usage Loss. Codebook collapse, where only a small subset of entries is actively used, is a common failure mode in VQ-based models. We propose a composite usage loss $\mathcal{L}_{\text{usage}}$ that penalizes uneven

utilization through three mechanisms:

$$\begin{aligned} \mathcal{L}_{\text{usage}} = & \underbrace{\text{Var}(\{n_k\}_{k=1}^K)}_{\text{Variance Penalty}} \\ & + \underbrace{\left(1 - \frac{|\{k : n_k > 0\}|}{K}\right)}_{\text{Coverage Penalty}} \\ & + \underbrace{\left(1 - \frac{H(\{p_k\}_{k=1}^K)}{\log K}\right)}_{\text{Entropy Penalty}} \end{aligned} \quad (14)$$

Here, n_k is the number of times the k -th codebook entry is selected in the current batch, and $p_k = n_k / \sum_j n_j$ is the empirical probability of usage.

- The *Variance Penalty* minimizes the variance in usage counts, pushing towards equal frequency.
- The *Coverage Penalty* directly maximizes the number of unique active codes (cardinality).
- The *Entropy Penalty* maximizes the entropy of the usage distribution $H(\{p_k\})$, normalized by $\log K$ to range between $[0, 1]$. A uniform distribution yields maximum entropy.

Together, these terms ensure robust and comprehensive utilization of the codebook capacity. The gradient is optimized after a large batch size of 1024 items with AdamW optimizer.

A.3 Alternative Quantizer Formulations

The framework is not restricted to PQ. Residual Quantization (RQ) approximates an embedding through a sequence of codebooks. Let $\mathbf{r}^{(0)} = \mathbf{e}$. At stage m , RQ selects the codeword nearest to the current residual and updates the residual as

$$k_m^* = \arg \min_k \left\| \mathbf{r}^{(m-1)} - \mathbf{c}_k^{(m)} \right\|_2^2, \quad (15)$$

$$\mathbf{r}^{(m)} = \mathbf{r}^{(m-1)} - \mathbf{c}_{k_m^*}^{(m)}. \quad (16)$$

After M stages, the quantized representation is

$$Q_{\text{RQ}}(\mathbf{e}) = \sum_{m=1}^M \mathbf{c}_{k_m^*}^{(m)}. \quad (17)$$

Optimized Product Quantization (OPQ) learns an orthogonal rotation $\mathbf{R}$ before applying PQ so that the rotated dimensions are more suitable for subspace quantization:

$$\begin{aligned} \mathbf{R}^* = & \arg \min_{\mathbf{R}^\top \mathbf{R} = \mathbf{I}, \mathcal{C}} \sum_n \\ & \left\| \mathbf{R} \mathbf{e}_n - Q_{\text{PQ}}(\mathbf{R} \mathbf{e}_n; \mathcal{C}) \right\|_2^2. \end{aligned} \quad (18)$$

The representation returned to the downstream modules is rotated back to the original space:

$$Q_{\text{OPQ}}(\mathbf{e}) = (\mathbf{R}^*)^\top Q_{\text{PQ}}(\mathbf{R}^* \mathbf{e}; \mathcal{C}). \quad (19)$$

Thus, PQ, RQ, and OPQ expose the same quantized-representation interface to the subsequent aggregation and projection modules.

B Dataset Details

The format of question, answer and user histories of the tasks are shown in Table 3. We do data filter to make sure the length distribution. During the UBA stage, task-specific training data is utilized exclusively for the ID component, while other modules remain task-agnostic. Detailed statistics and example input output pair for all six tasks are provided in Table 7.

C Efficiency

We provide an efficiency experiment of the training and inference time on the same setting as shown in Table 4. Our CURP achieve a high efficient, especially compare with DiffMean and PROPER.

D Hyperparameter Settings

All experiments are conducted on 8 NVIDIA 4090 GPUs with a batch size of 2 per GPU. We utilize DeepSpeed ZeRO Stage 2 to enable efficient memory usage and distributed training. The learning rate is set to $1\text{e-}4$ with weight decay of 0.01 and max gradient norm of 1.0. We use gradient accumulation with 4 steps and train for 3 epochs. The MLP projection layer consists of two linear layers: 768-3584 with GELU activation, followed by 3584-3584. We use a 2 layer Q-Former with 4 learnable queries. The head for attention is set to 8. For the codebook training, we set the codebook weight to 1.0, diversity weight to 0.1, and usage weight to 0.3. We employ Contriever as the encoder and Qwen-2.5-7B-Instruct as the decoder. During inference, we set `do_sample` as false to ensure replicability and the reliability of our result. The random seeds are 42, 0, 1 for multiple runs. We use the default val dataset for evaluation on LaMP benchmark. The default optimizer is AdamW for its stable performance. We use `rouge_score` to compute ROUGE.

E Alternative Quantizers

Table 6 replaces the default PQ while keeping the downstream pipeline fixed. RQ and OPQ improve

Table 3: Dataset statistics and input–output formats for different tasks in the LaMP benchmark. “#” denotes the number of instances, “Cls” indicates classification tasks, and placeholders in braces { } are replaced with dataset-specific values.

Task	#Train	#Val	#Len IN	#Cls or #Len OUT	Len HIST	Input Format	Output Format	History Format
LaMP-1	6,542	1,500	51.43	2	84.15	For an author who has written the paper with title “{title}”, which reference is related? Just answer with <1> or <2>. <1>: “{ref1}” <2>: “{ref2}”	<2>	Title: {title} Abstract: {abs}
LaMP-2	5,073	1,410	92.39	15	86.76	Which tag does this movie relate to among the following tags? Just answer with the tag name. tags: {sci-fi, action, ...} description: {movie}	romance	Description: {des} Tag: {tag}
LaMP-3	20,000	2,500	128.18	5	185.40	What is the score of the following review on a scale of 1 to 5? Just answer with 1–5. review: {review}	5	Text: {Review} Score: {score}
LaMP-4	12,500	1,500	29.97	10.07	204.59	Generate a headline for the following article: {article}	How I Got ‘Rich’	title: {title} text: {article}
LaMP-5	14,682	1,500	162.34	9.71	87.88	Generate a title for the following abstract of a paper: {abstract}	Distributed Partial Clustering	title: {title} text: {abstract}
LaMP-7	13,437	1,498	29.72	16.96	15.71	Paraphrase the following tweet without any explanation: {tweet}	gotta make the most of my last full day in ktown	text: {tweet}

Method	LaMP-4		
	Train Time	Infer Time	Memory
Direct	—	3:36	14G
BM25	—	6:16	18G
CoT	—	43:38	18G
PAG	—	12:20	16G
PPlug	7:43	5:02	17G
DiffMean	—	≈640 min [†]	14G
PROPER	≈57 min [†]	≈1056 min [†]	21G
CURP (ours)	7:40	4:28	17G

Table 4: Efficiency comparison between methods.

Table 5: Performance comparison of CURP with and without LoRA fine-tuning across LaMP benchmarks. Arrows indicate direction of improvement (↑ higher is better, ↓ lower is better).

Model	LaMP-2		LaMP-3		LaMP-4		LaMP-7	
	Acc↑	F-1↑	MSE↓	RMSE↓	R-1↑	R-L↑	R-1↑	R-L↑
CURP	0.629	0.547	0.276	0.585	0.194	0.173	0.543	0.483
w/ LoRA	0.633	0.561	0.270	0.574	0.214	0.192	0.538	0.481

over PQ, and the stronger quantizers surpass the continuous representation on most reported tasks. These results show that quantization need not sacrifice task-relevant information: stronger quantizers can instead improve performance by retaining useful structure while filtering nuisance variation. PQ therefore serves as a simple default rather than a restriction of CURP.

Table 6: Quantizer replacement with the downstream pipeline held fixed. The continuous variant omits quantization. Bold denotes the best score for each task.

Task	Continuous	PQ	RQ	OPQ
LaMP-2 Acc↑	0.634	0.629	0.655	0.649
LaMP-3 MAE↓	0.275	0.276	0.271	0.279
LaMP-4 R-1↑	0.198	0.194	0.201	0.203
LaMP-7 R-L↑	0.540	0.543	0.553	0.552

F PQ Configuration Ablation

Table 7 compares subspace counts, codebook sizes, and initialization. PQ-1 denotes full-vector VQ. The default PQ-4 configuration performs best in these controlled downstream comparisons, while the weak result from a randomly initialized codebook confirms that the data-fitted Stage-1 structure transfers useful information to UBA.

Table 7: PQ configuration ablation. PQ-1 denotes full-vector VQ.

Variant	LaMP-2 Acc↑	LaMP-4 R-1↑
PQ-4 (default)	0.629	0.194
PQ-1 (full-vector VQ)	0.500	0.189
PQ-2	0.528	0.187
PQ-8	0.528	0.189
Codebook 1,000 (default)	0.629	0.194
Codebook 500	0.465	0.190
Codebook 2,000	0.522	0.186
Random initialization	0.481	0.181

G Whole-Codebook Interpretability

We also quantify interpretability over the entire LaMP-2 codebook rather than relying only on selected cases. The evaluation covers approximately 26k labeled history items, 14 tags, and 1,348 distinct semantic IDs. Semantic ID and gold label have normalized mutual information of 0.522 and V-measure of 0.522, compared with an NMI of 0.074 after label permutation. Within-ID label homogeneity is 0.951, and histories assigned the same code have pairwise label agreement of 0.957, compared with 0.092 for random pairs (the class-frequency chance level is $\sum_c p_c^2 = 0.089$). These results show that discrete IDs form reproducible, inspectable semantic units; continuous embeddings do not provide directly nameable units for the same analysis. Qualitative prototype factorization is further detailed in Appendix L.

H Robustness under History Perturbations

To isolate robustness due to quantization, we additionally perturb histories by deleting items, injecting irrelevant behaviors from other users, and paraphrasing 20–60% of the profile. Under noise injection and paraphrasing on LaMP-4, CURP’s performance drop is approximately 0%, compared with approximately 2.7% for the continuous variant. The advantage depends on the task and perturbation, and the continuous representation can be equally stable in some settings. Taken together with the clean-data and stronger-quantizer results, the evidence suggests that default PQ discards predominantly nuisance variation rather than critical user signals. Nevertheless, rare fine-grained patterns that lie far from every prototype may be lost and unrecoverable.

I Stable–Dynamic Fusion Ablation

We test whether the stable and dynamic branches can be collapsed into a single embedding. With the same token budget, we mean-pool the stable branch and add it to the dynamic branch as one vector. This fusion does not improve performance: the LaMP-2 score decreases from 0.536 to 0.458, while LaMP-4 changes only from 0.163 to 0.160. Representation-level tests further support keeping the branches separate. When the query is swapped while holding the user fixed, the stable representation is invariant by construction (cosine similarity 1.000 on both tasks), whereas the dynamic representation changes substantially (0.529 ± 0.153 on LaMP-2 and 0.234 on LaMP-4, with 10th percentile 0.127). After deleting half of a LaMP-2 profile, the stable representation also remains more consistent than the dynamic representation (0.982 versus 0.887).

J LLM-as-a-Judge Preference Evaluation

Reference-overlap metrics do not fully measure whether an output fits a particular user. We therefore conduct an LLM-as-a-judge evaluation on approximately 200 examples from each generation task (LaMP-4, LaMP-5, and LaMP-7). Claude Opus 4.8 scores personalization, faithfulness, and overall generation quality on a 1–5 scale and performs forced pairwise comparisons; we report 95% confidence intervals. As shown in Table 8, CURP obtains the highest overall score among the compared compressed-representation methods on

LaMP-4 and LaMP-5 and ties PPlug on LaMP-7. CURP clearly outperforms StyleVector in pairwise comparisons. Against PPlug, CURP leads on LaMP-4, is statistically comparable on LaMP-5, and trades slightly lower personalization for higher faithfulness on LaMP-7. Because automated judges may inherit model-specific preferences, this evaluation complements but does not replace human preference judgments.

K Prompt Templates

We provide detailed prompt templates used for encoding user history across different tasks.

In this section, we detail the prompt templates used for each LaMP benchmark. The prompts are constructed using three components:

- **Static Context:** Encodes long-term user preferences/profile (`stable_str`).
- **Dynamic Context:** Encodes retrieved relevant history (`dynamic_token`).
- **Instruction:** The specific task description and output constraint.

Note that `{input_text}` represents the specific input for the current instance (e.g., the paper abstract, the movie description, or the article body). Ablation studies may remove either the Static or Dynamic context lines.

K.1 LaMP-1: Scholar Profile Matching

Prompt Template

```
The research expertise and dominant
scientific topics that define this
scholar’s work,
distilled from their complete
publication history, are
encoded in: {stable_str}
```

```
The research aspects most
topically aligned with the
target paper are encoded
in: {dynamic_token}
```

```
{input_text}
```

```
Select the reference whose
subject matter best matches
this scholar’s research domain.
Output only [1] or [2].
```

Table 8: LLM-as-a-judge preference evaluation. Scores are on a 1–5 scale. The last column is CURP’s forced-pairwise win rate against the method in the same row; “–” denotes self-comparison.

Task	Method	Personalization	Faithfulness	Quality	Overall [95% CI]	CURP win rate [95% CI]
LaMP-4	CURP	2.65	3.09	3.77	3.04 [2.92, 3.15]	–
	PPlug	2.59	2.64	3.67	2.87 [2.74, 3.01]	58.0% [51.8, 64.1]
	StyleVector	2.17	3.25	2.99	2.64 [2.56, 2.73]	68.3% [62.1, 74.1]
LaMP-5	CURP	4.04	4.11	4.69	4.22 [4.11, 4.32]	–
	PPlug	4.00	4.12	4.72	4.21 [4.10, 4.32]	51.1% [45.8, 56.3]
	StyleVector	1.95	4.14	3.33	2.84 [2.77, 2.91]	92.9% [89.2, 96.1]
LaMP-7	CURP	3.18	4.50	4.55	3.85 [3.76, 3.95]	–
	PPlug	3.37	4.26	4.41	3.85 [3.75, 3.95]	47.0% [41.5, 52.5]
	StyleVector	1.89	4.32	4.75	3.21 [3.13, 3.30]	82.5% [77.5, 87.2]

K.2 LaMP-2: Personalized Text Classification

Prompt Template

A user’s preferences learned from their past writings are encoded in: {stable_str}

The aspects of their history most relevant to the current task are encoded in: {dynamic_token}

{input_text}

Output only the answer.

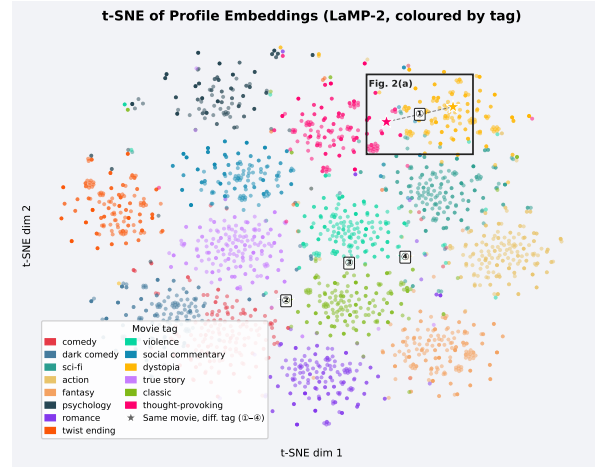


Figure 8: t-SNE projection of 2,800 Contriever embeddings from a controlled LAMP-2 stress test (200 movie descriptions $\times$ 14 candidate tags). We pair every movie with every candidate tag, deliberately creating many semantically implausible movie–tag combinations. Items derived from the same movie remain colocated despite the forced tags, indicating that the representation preserves movie content while suppressing injected tag noise. Stars ①–④ mark examples enlarged in Figure 9.

K.3 LaMP-3: Rating Prediction

Prompt Template

This reviewer’s personal rating calibration--how they translate expressed satisfaction into numerical scores--is encoded in: {stable_str}

The reviewer’s past reviews whose sentiment most closely matches the current review are encoded in: {dynamic_token}

{input_text}

Based on this user’s personal rating behavior, predict the score they assigned.
Output only 1, 2, 3, 4, or 5.

K.4 LaMP-4 / 5 / 7: Text Generation

Prompt Template

This author’s headline writing style and preferences, learned from their past articles, are encoded in: {stable_str}

The aspects of their past articles most relevant to the current article are encoded in: {dynamic_token}

{input_text}

Generate a headline in this author’s personal writing style.
Output only the headline.

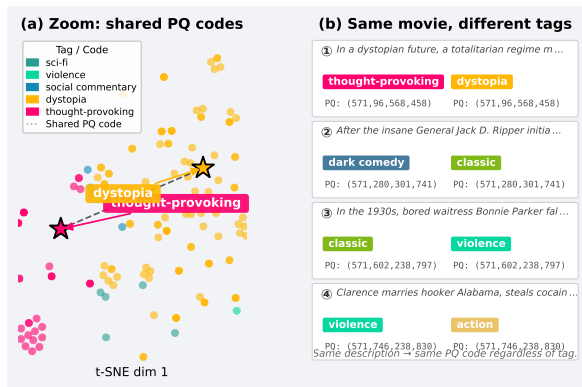


Figure 9: (a) Zoom into the boxed region of Figure 8. Dotted lines connect instances constructed from the same movie description but paired with different forced tags; each connected group receives an identical PQ code tuple. (b) Four examples of the same movie paired with different candidate tags. Because every movie is deliberately combined with all 14 tags, some pairings are inherently unreasonable. Mapping these noisy variants to the same content-consistent code shows that quantization filters the injected tag noise instead of encoding each implausible pairing as a distinct user signal.

L Detailed Codebook Interpretability Analysis

Table 9 provides detailed examples of PQ indices from the LaMP-4, demonstrating how the codebook decouples semantic dimensions. All samples share the same indices [910, 204] in subspaces 2 and 3, serving as a semantic anchor for Donald Trump-related content. Variations in subspaces 1 and 4 reveal fine-grained semantic distinctions in content type, perspective, sentiment, and stance.

Key Observations:

- **Semantic Anchor (Subspaces 2 & 3):** All samples share [40, 101], capturing the core topic semantics (Trump’s Muslim Ban / Travel Ban executive order content). This anchor spans 1,059 profile items with 99.2% topic purity across the entire 3.6M-item training corpus.
- **Content Type (Subspace 1):**
 - 20: Legal/political proceedings (Supreme Court, travel ban litigation, Trump lawyers, judicial review, executive order)
 - 8: Media and civil society coverage (Wall Street Journal, cable news, Muslim voices, personal impact story, civil

rights organizations)

- 122: Expert appellate scrutiny (appeals court judges, anti-Muslim statements, constitutional legality, lawyer cross-examination)

• Perspective/Framing (Subspace 4):

- 29: Court challenge framing (challengers, filibuster, halting, blocking, constitutional challenge)
- 65: Personal impact and press accountability (ripping couples apart, dearth of Muslim voices, mealy-mouthed coverage, human cost)
- 39: Protest/resistance framing (Google workers walkout, large-scale protest, civil action)
- 58: Expert legal proceedings framing (judges hit lawyer hard, appellate questioning, legality hearing)

This decomposition demonstrates that the PQ codebook learns meaningful semantic factorization. Notably, Groups 1 and 3 share the same **S4=29** (court challenge framing) but differ in **S1** (20 vs. 8), separating government-side judicial proceedings from civil-society constitutional challenges—a nuanced distinction captured purely by the content-type subspace. This confirms that different subspaces capture hierarchical and orthogonal semantic dimensions: topic anchor (subspaces 2–3), content type/source (subspace 1), and perspective/framing (subspace 4).

M User Study

To understand what each model actually captures, we sample six representative users across all four tasks and compare predictions across the full baseline family, ZS, ICL, BM25, Dense, PPlug, PROPER, and the CURP ablations *stable-only* and *dynamic-only* against the full model (Table 10).

What retrieval baselines capture through in-context copying. ICL prepends the user’s eight profile items verbatim as few-shot exemplars; BM25 and Contriever retrieve the most similar profile items and inject them as context. This gives retrieval baselines an immediate surface advantage: for the abbreviation-heavy Twitter user (LaMP-7, row 1), ICL, BM25, and Contriever all output “*why didn’t u tell me*” because the user’s actual tweets

(“*I told u already*”, “*why u gotta go2 work*”) appear verbatim in the prompt. Likewise in LaMP-4 (row 5), ICL (“*8 Inappropriate...*”), BM25 (“*8 Cringe-Worthy...*”), and Contriever (“*10 Wedding Comments...*”) all spontaneously adopt the user’s numbered-list format because numbered headlines appear directly in the retrieved context window. The mechanism is shallow: the model copies what it sees. When the query departs from the retrieved exemplars, as in the hallucination case (row 2) where no profile tweet is topically relevant, retrieval baselines produce a generic, coherent paraphrase rather than degenerating, because they never inject the profile into the generation pathway beyond the text prompt.

Where learned representations outperform retrieval: conceptual genre framing. The most discriminating case is LaMP-2, row 3, where an ambiguous film (a near-future vampire plague) can be legitimately tagged either *sci-fi* or *dystopia*. Every retrieval-based model, ICL, BM25, Contriever, and PPlug, predicts *sci-fi*, because they retrieve the user’s sci-fi movies from their profile and surface that label. Yet the user has tagged four films as *dystopia* and only two as *sci-fi*: the genre that dominates their *conceptual* framing is not recoverable by retrieving the single most similar movie. Only PROPER, *stable-only*, and CURP predict *dystopia* correctly, because all three learn something about the user’s overall preference distribution rather than nearest-neighbour matching. This is the clearest evidence that CURP’s gains in LaMP-2 reflect genuine personalisation: it resolves lexically ambiguous films by internalising what genre the user reaches for, not by copying labels from retrieved items.

When dynamic content overrides stable prior. In LaMP-2 row 4, a user whose dominant tag is *violence* ($\times 7$ films) must label a WWII teen-survival story. The stable prior alone is insufficient: *stable-only* predicts *action*, the plausible surface reading, as do Direct, ICL, and BM25. The dynamic branch retrieves the user’s violence-tagged war films and confirms *violence*; CURP and PPlug both follow this content signal correctly. Notably, PROPER fails here (predicting *comedy*), suggesting that PROPER’s nearest-neighbour retrieval selects the wrong profile subset when the target film’s description does not closely match any of the user’s violence films in embedding space.

A failure mode shared by all query-aware

dynamic models. Row 2 (LaMP-7) shows a systematic weakness of profile-embedding injection. When no profile tweet matches the paraphrase query, *dynamic-only* degenerates into a hallucinated list of @handles (“@jessicajay @michelle_1986 @sarahjane1986... ”); PPlug exhibits the same failure (“@jessicajay @michelle-bryant @sarahmccarthy... ”). The retrieval baselines do not hallucinate here because they insert the profile as text: lacking relevant examples, the model simply ignores them. This failure is intrinsic to soft-prompt injection: without a relevant profile–query match, the injected embedding activates @mention patterns from the profile rather than suppressing output. *Stable-only*, PROPER and CURP are the only personalised models to produce a coherent paraphrase on this example.

Style and metrics decouple. Row 5 (LaMP-4) is the clearest illustration. The gold reference headline contains no number (“*Rudest Things Guests Say to the Bride and Groom*”), yet the user’s entire profile follows a numbered format. Every personalised model, ICL (“*8 Inappropriate...*”), BM25 (“*8 Cringe-Worthy...*”), Contriever, PROPER, *Stable*, *Dynamic*, and CURP predicts a numbered headline more consistent with the user’s observable writing habit than the reference itself. All are penalised by ROUGE. This confirms that automatic metric gains systematically undercount style improvements: a model that writes the way the user always writes is penalised whenever one specific reference deviates from that habit.

Overall, by case study, we demonstrate our CURP model can achieve a more personalized performance beyond the automatic metric and keep a balance on self-inner stable representation and query-aware dynamic representation.

N Combine with PEFT

As shown in Table 5, integrating LoRA-based fine-tuning generally enhances the performance of CURP on structured and semantic tasks. Specifically, we observe consistent improvements on LaMP-2 (Classification), LaMP-3 (Rating Prediction), and LaMP-4 (Headline Generation), where LoRA helps the model better align with user-specific preferences and domain-specific patterns, yielding lower error rates (MSE/RMSE) and higher generation quality (ROUGE scores). However, an interesting divergence occurs on LaMP-7 (Tweet Generation). Here, the vanilla CURP outperforms

the LoRA-fine-tuned variant. We attribute this to the unique nature of tweet data, which is characterized by highly diverse, complex, and informal linguistic styles (e.g., slang, abbreviations, and erratic syntax). While LoRA effectively captures stable, long-term user profiles, full parameter freezing with lightweight adaptation may inadvertently smooth out the high-frequency, stochastic variations inherent in casual social media text. In such scenarios, the rigid structure imposed by LoRA fine-tuning might constrain the model’s ability to reproduce the raw, unpolished authenticity required for realistic tweet generation, whereas the retrieval-augmented approach of CURP alone preserves these fine-grained stylistic nuances more effectively.

O Licenses

We use publicly available datasets and models that are already licensed. Our code will be released if accepted with MIT License.

Table 9: PQ Indices Analysis for Trump Muslim Ban / Travel Ban Headlines (LaMP-4). All samples share indices [40, 101] in subspaces S2 and S3, forming a semantic anchor for Trump’s Executive Order on immigration. Variations in S1 and S4 reveal fine-grained distinctions in coverage angle, source type, and perspective.

Sample	Text	Title	PQ Indices			
			S1	S2	S3	S4
Group 1: S1=20 (Legal/Political), S4=29 (Court Proceedings)						
308421-2815421	And all because the Supreme Court could soon determine the legality of Trump’s executive order.	Travel Ban Challengers Lose Bid For Giuliani’s ‘Muslim Ban’ Memo	20	40	101	29
308421-2815431	The president’s campaign promises are proving to be a key issue in the travel ban litigation.	Trump’s ‘Muslim Ban’ Pledge Scrubbed From Website Just As Judges Ask About It	20	40	101	29
308421-2815464	A judge in Maryland agreed that anti-Muslim sentiment was behind the president’s executive orders.	Federal Court In Hawaii Blocks Donald Trump’s New Travel Ban Nationwide	20	40	101	29
308421-2815488	The administration is trying to make the case that judicial review of its executive order imperils national security.	Trump Lawyers To Court Re-viewing Muslim Travel Ban: Stay Out Of It	20	40	101	29
Group 2: S1=8 (Media/Civil Society), S4=65 (Personal Impact & Press Critique)						
302244-745229	They’ve been told to downplay the “Muslim exclusion” part of the Muslim exclusion order.	Wall Street Journal Editor Directs Reporters To Get Mealy-Mouthed Covering Trump	8	40	101	65
304846-1424183	A new report shows a dearth of Muslim voices in discussions about Donald Trump’s executive order.	Cable News Sure Could Talk To More Muslims About The Muslim Ban	8	40	101	65
304846-1424184	They were supposed to get married in Virginia and start a life together. Then Donald Trump signed an executive order.	How The Muslim Ban Is Ripping This Young Couple Apart	8	40	101	65
Group 3: S1=8 (Media/Civil Society), S4=29 (Constitutional Challenges)						
308421-2815426	An appeals court said the president’s executive order “drips with religious intolerance.”	Travel Ban Challengers Demand Trump Hand Over Giuliani’s ‘Muslim Ban’ Memo	8	40	101	29
308421-2815490	It’s perhaps the most sweeping constitutional challenge to the president’s travel ban yet.	Muslim Coalition Takes Donald Trump To Court Over ‘Muslim Exclusion Order’	8	40	101	29
3011697-3528095	A federal judge temporarily stopped elements of the executive order on Saturday.	Read The Court Order Halting Parts Of Trump’s Anti-Muslim Ban	8	40	101	29
Group 4: S1=20 (Legal/Political), S4=39 (Corporate Protest)						
3010214-3138252	Googlers united against the president’s decision to restrict immigration and travel from certain predominantly Muslim countries.	Google Workers Stage Large-Scale Walkout To Protest Trump’s Executive Order On Immigration	20	40	101	39
Group 5: S1=122 (Judicial Expert), S4=58 (Appellate Scrutiny)						
308421-2815432	A key question an appeals court must answer is whether the president’s anti-Muslim statements can be held against him.	Judges Hit Donald Trump’s Lawyer Hard About Legality Of Travel Ban	122	40	101	58

Table 10: Qualitative comparison across models and cases. **Bold** indicates critical differentiating tokens. ✓ = style-consistent or correct; × = inconsistent or incorrect.

Model	R1	R2	R3	R4	R5	R6
(Case Context)	LaMP-7	LaMP-7	LaMP-2	LaMP-2	LaMP-4	LaMP-3
(Trait)	Abbrev.	Sparse	Genre	Content	Format	Rating
Direct	... you ×	<i>coh.</i> ✓	fantasy ×	action ×	<i>Wedding Woes</i> ×	5 ×
ICL	... u ✓	<i>coh.</i> ✓	sci-fi ×	action ×	8 Inapp. ✓	5 ×
BM25	... u ✓	<i>coh.</i> ✓	sci-fi ×	action ×	8 Cringe ✓	5 ×
Cont.	... u ✓	<i>coh.</i> ✓	sci-fi ×	sci-fi ×	10 Wedding ✓	4 ✓
PPlug	... u ✓	@jessica... ×	sci-fi ×	violence ✓	<i>Etiquette</i> ×	4 ✓
PROPER	... you ×	<i>coh.</i> ✓	dystopia ✓	comedy ×	10 Wedding ✓	5 ×
Stable	... you ×	<i>coh.</i> ✓	dystopia ✓	action ×	10 Wedding ✓	4 ✓
Dynamic	... you ×	@jessica... ×	sci-fi ×	violence ✓	10 Wedding ✓	4 ✓
CURP	... u ✓	<i>coh.</i> ✓	dystopia ✓	violence ✓	10 Wedding ✓	4 ✓
Ref.	... u tell me	<i>Good morning...</i>	dystopia	violence	<i>Rudest Things...</i>	4

R1 (Abbrev.): CURP/ICL/BM25 capture “u” (style); Direct/PROPER/Stable/Dyn use “you”. **R2 (Sparse):** Retrieval models remain coherent; Embedding models (PPlug/Dyn/CURP) hallucinate handles. **R3 (Genre):** CURP/PROPER/Stable infer “dystopia” (global prior); others match local “sci-fi” items. **R4 (Content):** CURP/PPlug/Dyn retrieve “violence”; PROPER/Stable miss; others guess “action/comedy”. **R5 (Format):** All personalized models adopt numbered format (style-consistent); Reference is unnumbered. **R6 (Rating):** CURP/Cont./PPlug/Stable/Dyn calibrate to 4; Direct/ICL/BM25/PROPER overshoot to 5.