Is One Vector Enough for Personalized Generation? CARD: A Compact Adaptation with Reward-Guided Decoding Framework

Anonymous ACL submission

Abstract

Adapting large language models to individual users remains challenging due to the tension between fine-grained personalization and scalable deployment. We present CARD, a hierarchical framework that achieves effective personalization through progressive refinement. CARD first clusters users according to shared stylistic patterns and learns group-specific LoRA adapters, enabling robust generalization and strong low-resource performance. To capture individual differences within each cluster, we propose an implicit preference learning mechanism that contrasts user-authored text with cluster-level generations, allowing the model to infer user-specific style preferences without manual annotation. At inference time, CARD injects personalization exclusively at decoding via lightweight user preference vectors and low-rank logit corrections, while keeping the base model frozen. Experiments on the LaMP and LongLaMP benchmarks show that CARD achieves superior generation quality compared to baselines, while significantly improving efficiency and scalability for practical personalized text generation.

1 Introduction

Large language models (LLMs) have substantially advanced natural language generation (NLG) (Salemi et al., 2024b). In many real-world deployments, however, models must produce text that satisfies explicit constraints, motivating *controllable text generation* (CTG) (Liang et al., 2024). Among CTG settings, *personalization* aims to tailor outputs to an individual user's preferences and writing style, which is critical for applications such as dialogue systems, content recommendation, and advertising (Liu et al., 2024).

The code is available at https://anonymous.4open.science/r/CARD-86BC/

Existing personalized text generation methods are commonly grouped into two paradigms: Retrieval-Augmented Generation (RAG) and Parameter-Efficient Fine-Tuning (PEFT). RAG-based methods (Richardson et al., 2023; Kumar et al., 2024; Salemi et al., 2024a; Izacard et al., 2021a) retrieve user history and prepend it to the prompt, whereas PEFT-based methods (Tan et al., 2024a) adapt the model with lightweight modules (e.g., LoRA (Hu et al., 2021)) to learn user-conditioned parameters. Both paradigms face notable limitations (Balaguer et al., 2024). RAG is sensitive to prompt design and retrieval quality, and often yields shallow personalization because the generator remains frozen. PEFT can capture deeper user-level behavior, but scales poorly: maintaining per-user parameters becomes expensive (Tan et al., 2024b) as the user base grows, and onboarding new users typically requires additional optimization.

From a supervision perspective, PEFT requires converting user preferences into preference pairs for objectives like direct preference optimization (Rafailov et al., 2023; Shenfeld et al., 2025). However, explicit annotations are prohibitively expensive, and heuristic constructions (e.g., contrasting user text with random negatives) often entangle topical content with stylistic traits. Consequently, PEFT-based personalization faces a systemic scarcity of high-quality preference data, leading to unreliable signals and brittle performance under sparse user histories.

Fundamentally, PEFT-based personalization faces a rigid granularity trade-off. Recent work explores decomposing personalization into progressive group-level adaptations to improve efficiency (Zhang et al., 2025b). However, achieving fine-grained individual fidelity without incurring prohibitive per-user parameter costs or suffering from sparse user histories remains elusive. This raises a key question:

Can we leverage group-level priors for efficiency

while pushing individual preferences entirely to lightweight decoding-time control?

To address these challenges, we introduce **CARD**, a framework grounded in the insight that personalization signals are inherently hierarchical: broad preferences are shared as group-level priors, while fine-grained nuances manifest as stable individual differences. Based on this structure, CARD first (i) achieves hierarchical scalable adaptation by clustering users to learn shared adapters that capture common group preferences, thereby amortizing adaptation costs and establishing robust priors for low-resource users.

Building on these group-level priors and addressing the challenge of constructing high-quality user preference pairs, CARD (ii) introduces an implicit preference learning strategy, which explicitly reduces semantic confounding and yields stable supervision for learning individual stylistic deviations.

Finally, CARD (iii) executes personalization via lightweight decoding-time steering. At inference time, both the backbone and cluster parameters remain frozen, and generation is modulated via reward-guided logit editing. This enables rapid user switching with minimal per-user storage, radically improving deployment scalability while maintaining strong personalization fidelity.

Our contributions can be summarized as follows:

- We propose CARD, a hierarchical personalization framework that decouples personalization into shared preferences and ultra-lightweight individual vectors. This drastically reduces per-user storage overhead and enables massively scalable deployment without maintaining heavy per-user parameters.
- We introduce an implicit preference learning mechanism that derives stable supervision signals by contrasting user texts against cluster baselines. This effectively mitigates data sparsity, enabling the model to achieve robust personalization even with minimal user history.
- We internalize user personalization data into lightweight parameters. By guiding text generation via logit corrections on a frozen LLM, CARD achieves personalization without exposing raw data in the context window, inherently safeguarding privacy and eliminating long context latency.

2 CARD Model Design

2.1 Task Formulation

Personalized text generation aims to produce outputs that align with individual users' styles and preferences based on their historical contexts and interactions.

Formally, given a user u and a raw input query x^{raw}, we construct a task-specific prompt by injecting the user's historical profile:

$$\tilde{x} = \phi_{\text{task}}(x^{\text{raw}}, \mathcal{H}_u), \qquad \mathcal{H}_u = \{p_i\}_{i=1}^{|\mathcal{H}_u|}, \quad (1)$$

where $\mathcal{H}_u$ denotes the user's historical profile records (task-dependent fields such as posts or writing examples), and $\tilde{x}$ is the transformed input after task-specific prompt construction.

The goal is to generate personalized output $\hat{y}$ that captures the user u's authentic writing style for the given task, conditioned on both the query $\tilde{x}$ and the user's stylistic characteristics. To address this challenge efficiently while maintaining low resource start robustness, we propose a two-stage framework that combines cluster-level adaptation with user-level personalization.

2.2 Overall Framework

We first define a cluster conditioned language model that captures group-level stylistic patterns:

$$p^{(c(u))}(y \mid \tilde{x}) = M^{(c(u))}(\tilde{x};\, W_b, \Theta^{\text{LoRA}}_{c(u)}), \quad (2)$$

where W_b are the frozen backbone parameters and $\Theta^{\text{LoRA}}_{c(u)}$ denotes the LoRA adapter corresponding to cluster assignment $c(u)$. The ground truth y reflects the user's authentic writing style for the given task. Each cluster learns shared PEFT parameters that generalize across similar users, making the system low resource start friendly.

Given the cluster-level distribution, we perform user-specific customization at decoding time:

$$\hat{y} = \text{Decode}\Big(p^{(c(u))}(\cdot \mid \tilde{x});\, \Psi, \lambda_u\Big), \quad (3)$$

where Ψ denotes cluster-level shared personalization parameters, and λ_u is a compact user preference vector trained to modulate the decoding process, without updating either the backbone parameters W_b or the cluster-specific LoRA parameters $\Theta^{\text{LoRA}}_{c(u)}$. At inference time, we feed the formatted prompt $\tilde{x}$ into the cluster model $M^{(c(u))}$ as a group-level prior, and then inject λ_u via logit steering to generate $\hat{y}$. Our framework is illustrated in Figure 1.

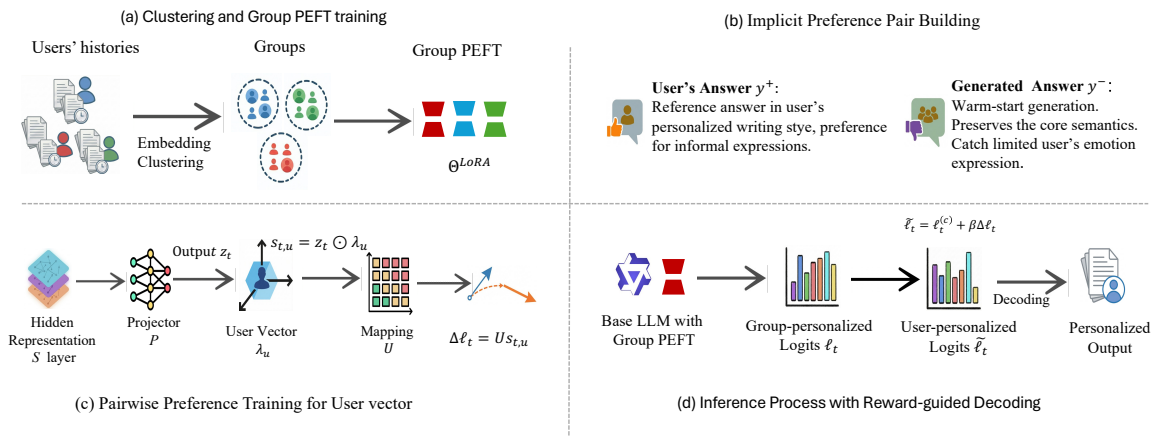

Figure 1: (a) CARD clusters user histories and trains group-specific PEFT modules as warm-start priors. (b) It then constructs input-aligned implicit preference pairs using the user's answer and the corresponding group-generated answer. (c) Preference data pairs are used to learn a dense user vector for user-level personalization on the frozen LLM. (d) During inference, CARD starts from group-conditioned logits as a shared prior and then refines them with a user-specific modification for personalized decoding.

2.3 Group-level Adaptation: Clustering and PEFT

Each user u is represented by an embedding e_u computed from the user's historical profile using a frozen encoder. We apply clustering to partition users into K clusters based on embedding similarity:

$$\mathcal{C} = \{C_1, C_2, \ldots, C_K\},$$
$$C_k = \{u \in \mathcal{U} : \|e_u - \mu_k\| \leq \|e_u - \mu_j\|, \ \forall j \neq k\}. \tag{4}$$

where $\mu_k \in \mathbb{R}^D$ denotes the centroid of cluster C_k, and $c(u)$ denotes the cluster assignment for user u.

To improve computational efficiency, we employ Low-Rank Adaptation (LoRA) (Hu et al., 2021). This cluster-level adaptation serves not as a fine-grained personalization endpoint, but as a stable, amortized prior that prevents catastrophic failure for low-resource users.

So for each cluster $c \in \{1, \ldots, K\}$, we train a distinct LoRA adapter Θ_c^{LoRA} by supervised fine-tuning on aggregated instances:

$$\mathcal{D}_c = \bigcup_{u \in C_c} \left\{ \left(\phi_{\text{task}}(x_{u,n}^{\text{raw}}, \mathcal{H}_u), \ y_{u,n} \right) \right\}_{n=1}^{N_u}. \tag{5}$$

The cluster-specific LoRA parameters are optimized via supervised fine-tuning with the cross-entropy loss:

$$\mathcal{L}_c = \sum_{(\tilde{x}, y) \in \mathcal{D}_c} \sum_{t=1}^{|y|} -\log p_{\theta_c}(y_t \mid \tilde{x}, y_{<t}). \tag{6}$$

where $\theta_c = \{W_b, B_c, A_c\}$ denotes the model parameters with cluster-specific LoRA weights, and $y_{<t}$ represents the tokens preceding position t. During inference, each user u is assigned to their corresponding cluster $c(u)$, and we exclusively use the cluster-specific LoRA $\Theta_{c(u)}^{\text{LoRA}}$.

2.4 Preference Pair Construction

To train the subsequent personalization components while keeping the cluster-LoRA and backbone LLM frozen, we construct preference pairs that emphasize intra-cluster stylistic differences. For each user interaction (x_n^{raw}, y_n) from user u_n, we create:

$$\begin{aligned} \tilde{x}_n &= \phi_{\text{task}}(x_n^{\text{raw}}, \mathcal{H}_{u_n}), \\ \mathcal{D} &= \{(\tilde{x}_n, y_n^+, y_n^-, u_n)\}_{n=1}^{N}. \end{aligned} \tag{7}$$

where $y_n^+ = y_n$ is the ground truth user response (preferred) and $y_n^- = M^{(c(u_n))}(\tilde{x}_n)$ is the response generated by the user's cluster-LoRA on the same prompt (dispreferred). This creates hard negatives that share the same semantic content but differ in stylistic execution. The cluster-LoRA response serves as a strong baseline representing the group-level style. By sharing identical semantic context but differing in stylistic execution, this input-aligned negative effectively isolates pure stylistic deviations from topical confounding. To efficiently generate the cluster-LoRA baseline, we run inference with the vLLM engine (Kwon et al., 2023).

2.5 User-level Personalization

Given the group-level priors and the constructed preference pairs, we introduce a shared personal-

ization head $\Psi = (P, U)$ to map the internal representations to explicit stylistic controls.

Preference Space Mapping and User Modulation At each decoding step t, the matrix P projects the aggregated hidden states h_t into a compact, J-dimensional stylistic subspace: $z_t = P(h_t) \in \mathbb{R}^J$. To distinguish individual traits, we learn a lightweight preference vector $\lambda_u \in \mathbb{R}^J$ for each user. This vector acts as a dynamic scaling mechanism, modulating z_t in a channel-wise manner:

$$s_{t,u} = z_t \odot \lambda_u, \tag{8}$$

where $\odot$ denotes element-wise multiplication. By dynamically amplifying or attenuating specific latent dimensions, λ_u strictly captures fine-grained individual preferences.

Reward-Guided Logit Modification To inject personalization into the generation process, we introduce a compact vocabulary mapping matrix $U \in \mathbb{R}^{|\mathcal{V}| \times J}$ that projects the user-modulated signal $s_{t,u}$ to the vocabulary space. This yields a low-rank adjustment to the cluster-LoRA baseline logits $\ell_t^{(c)} \in \mathbb{R}^{|\mathcal{V}|}$:

$$\Delta\ell_t = U s_{t,u}, \qquad \tilde{\ell}_t = \ell_t^{(c)} + \beta \Delta\ell_t, \tag{9}$$

where $\beta > 0$ is a hyperparameter controlling personalization strength.

To further improve efficiency, we apply this correction only to the Top-k candidate tokens based on the cluster-LoRA logits, reducing computational complexity from $\mathcal{O}(|\mathcal{V}| \cdot J)$ to $\mathcal{O}(k \cdot J)$ where $k \ll |\mathcal{V}|$. Let $I_{t,k}$ denote the Top-k index set at step t:

$$\tilde{\ell}_{t,v} = \begin{cases} \ell_{t,v}^{(c)} + \beta(U_{v,:} \cdot s_{t,u}), & \text{if } v \in I_{t,k} \\ \ell_{t,v}^{(c)}, & \text{otherwise.} \end{cases} \tag{10}$$

The final token distribution is obtained via softmax normalization:

$$\tilde{p}_t(v \mid \tilde{x}, u) = \frac{\exp(\tilde{\ell}_{t,v})}{\sum_{v' \in \mathcal{V}} \exp(\tilde{\ell}_{t,v'})}. \tag{11}$$

Importantly, this approach can be interpreted as reward-guided decoding, where the user preference vector λ_u defines a reward signal that re-ranks candidate tokens according to user-specific stylistic preferences, without modifying the underlying LLM or cluster-LoRA parameters.

Learning Objective With the cluster-LoRA Θ_c^{LoRA} and backbone LLM W_b frozen, we optimize the personalization parameters $\Psi = (P, U)$ and user vectors $\Lambda = \{\lambda_u\}_{u \in \mathcal{U}}$ using a Bradley–Terry (Bradley and Terry, 1952; Rafailov et al., 2023) pairwise loss on the constructed dataset $\mathcal{D}$:

$$\mathcal{L} = -\sum_{(\tilde{x}, y^+, y^-, u) \in \mathcal{D}} \log \sigma\Big(\log p_{\text{pers}}(y^+ \mid \tilde{x}, u) - \log p_{\text{pers}}(y^- \mid \tilde{x}, u) \Big). \tag{12}$$

where $p_{\text{pers}}(y \mid \tilde{x}, u) = \prod_{t=1}^{|y|} \tilde{p}_t(y_t \mid \tilde{x}, y_{<t}, u)$ is the personalized generation probability under user u.

2.6 New User Adaptation.

For a new user u, we compute the profile embedding e_u and assign the user to a cluster $c(u)$. Keeping the backbone and the corresponding LoRA fixed, we estimate the user preference vector λ_u from the user's historical data, which is then used for decoding-time personalization.

3 Experiment Settings

3.1 Benchmarks and Evaluations

We adopt the LaMP benchmark (Salemi et al., 2024b) and the LongLaMP benchmark (Kumar et al., 2024), which are designed to evaluate short-form and long-form personalized text generation, respectively. For each benchmark, we use the user-split setting, we evaluate model performance using the same metrics ROUGE-1(R-1) and ROUGE-L(R-L), more details are illustrated in Appendix C. Beyond reporting standard automatic metrics, we further assess performance using GPT-5.2 (OpenAI, 2025) as an LLM judge and conduct human evaluation, as detailed in Appendix B.

3.2 Baselines

We compare CARD against representative personalization baselines spanning different paradigms: (i) Context-based retrieval augmentation methods, including RAG (Salemi et al., 2024a) (evaluated with both BM25 (Robertson and Zaragoza, 2009) and the dense retriever Contriever (Izacard et al., 2021a)) and PAG (Richardson et al., 2023); (ii) Decoding-alignment baseline PAD (Chen et al., 2025); (iii) PEFT-based baselines, including OPPU (Tan et al., 2024a) and the hierarchical framework PROPER (Zhang et al., 2025b); and (iv)

Table 1: The comparison results of CARD against baselines on LaMP and LongLaMP benchmarks. The best results are in **bold**, and the second best results are underlined.

Task	Metric	Non-pers.	RAG		PAG	PAD	PPLUG	OPPU	PROPER	CARD
			BM25	Contriever						
LaMP4: News Headline	R-1	0.146	0.166	0.178	0.164	0.158	0.157	0.152	0.165	**0.218**
	R-L	0.128	0.148	0.160	0.146	0.139	0.138	0.128	0.144	**0.195**
LaMP5: Scholarly Title	R-1	0.425	0.456	0.448	0.415	0.442	**0.464**	0.426	0.449	0.459
	R-L	0.342	0.372	0.365	0.352	0.360	**0.386**	0.342	0.362	0.387
LaMP7: Tweet Paraphrasing	R-1	0.497	0.500	0.506	0.507	0.502	0.511	0.498	0.515	**0.521**
	R-L	0.439	0.431	0.436	0.435	0.437	0.433	0.422	0.439	**0.448**
LongLaMP1: Abstract Gen.	R-1	0.331	0.372	0.382	0.381	0.355	0.391	0.382	0.386	**0.411**
	R-L	0.184	0.203	0.210	0.201	0.194	0.214	0.202	0.204	**0.216**
LongLaMP2: Topic Writing	R-1	0.247	0.244	0.250	**0.255**	0.248	0.243	0.245	0.246	0.252
	R-L	0.119	0.118	0.121	0.125	0.121	0.122	0.112	0.115	**0.127**
LongLaMP3: Product Review	R-1	0.292	0.382	0.398	0.322	0.308	0.396	0.295	0.384	**0.405**
	R-L	0.130	0.152	0.155	0.141	0.136	0.149	0.132	0.141	**0.156**

Soft prompt generation baseline PPLUG (Liu et al., 2024).

3.3 Implementation Details

We implement CARD and all base models using Qwen/Qwen3-8B (Yang et al., 2025). For the RAG and PAG baselines, we rank user histories using either the sparse BM25 scoring function (Robertson and Walker, 1994) or the dense Contriever (Izacard et al., 2021b), retrieving the top-k=4 items. Crucially, to ensure a fair comparison, all retrieval-augmented baselines are restricted to the exact same historical context limits. Additional hyperparameter settings, training details, and evaluations across different model scales (0.6B to 32B) are provided in Appendix D.8. Our code is available at `https://anonymous.4open.science/r/CARD-86BC/`.

4 Results and Analysis

We present comprehensive experiments aiming to address the following Research Questions (RQs):
RQ1: How does CARD perform compared to existing personalization baselines under multiple evaluation settings?
RQ2: How do group LoRA and user vectors respectively contribute to personalization?
RQ3: How effective is CARD in handling low resource users with limited historical data?
RQ4: Can CARD provide scalable personalization with low per-user storage overhead and efficient inference?

4.1 Performance Results

To answer **RQ1**, we compare the performance of CARD with other baseline models (PEFT-based and soft prompt-based models) in the regular setting and the results are shown in Table 1. LLM judgments and human judgments results in LaMP are illustrated in Figure 2. **CARD achieves the best or near-best performance across multiple tasks on both LaMP and LongLaMP**, with advantages spanning tasks of varying text lengths and generation difficulty. Across 6 tasks and 2 metrics, CARD ranks 1st in 10/12 settings. The remaining two settings are near-best: LaMP5 R-1 (0.459 vs. 0.464) and LongLaMP2 R-1 (0.252 vs. 0.255). Demonstrating stronger cross-task generalization and a more robust personalization mechanism. LaMP and LongLaMP task relative improvements of CARD over the non-personalized baseline are summarized in Appendix A (Table 7).

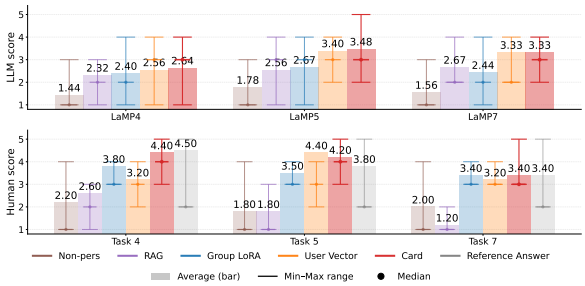

Figure 2: LLM judgments and Human judgments results among methods.

Figure 2 shows that CARD consistently matches or outperforms strong personalization baselines in both LLM-based and human evaluations. In LLM scores, CARD improves over the non-personalized baseline by 76.4%, 95.5%, and 113.5% on LaMP-4, LaMP-5, and LaMP-7 and the gains are also substantial in human evaluation. Notably, CARD even exceeds the reference answer by 10.5% on Task 5, suggesting that human judgments

Table 2: Ablation study (*w/o*) on LaMP.

Task	Metric	w/o LoRA	w/o Vec	CARD
LaMP4	R-1	0.207	0.148	**0.218**
	R-L	0.179	0.127	**0.195**
LaMP5	R-1	0.449	0.428	**0.459**
	R-L	0.376	0.345	**0.387**
LaMP7	R-1	0.507	0.498	**0.521**
	R-L	0.442	0.439	**0.448**

of personalization are inherently subjective and may prefer user-aligned style over strict agreement with a single gold response.

We further find that LLM and human evaluations are broadly aligned in ranking personalized methods above non-personalized baselines, but they are not perfectly matched. In particular, Group LoRA improves LLM scores over Non-pers by 50.0% on LaMP-5, while the corresponding human judgments gain are even larger at 94.4%. This suggests that group-level adaptation captures preference-relevant stylistic signals that are only partially reflected by automatic metrics. A comprehensive statistical analysis of the agreement and correlation between the automated LLM judgments and human annotations is given in Appendix B.

4.2 Ablation Study

To answer **RQ2**, we conduct an ablation study and results in Table 2. We further illustrate their roles with a representative case study shown in Figure 3.

Both group-level adaptation and user-specific deviation are important, with the user vector being the stronger driver in ROUGE-based evaluation. Table 2 shows that removing either component consistently degrades performance across all tasks, confirming that CARD benefits from both a shared group prior and individual-level preference modeling. Removing the user vector causes the largest drop: for example, R-1 decreases from 0.218 to 0.148 on LaMP-4, indicating that fine-grained user-specific deviation is the primary driver of lexical-overlap gains.

Importantly, the contribution of Group LoRA is much more evident in preference-oriented evaluation than in ROUGE alone. Although its ROUGE gains are relatively modest, Group LoRA improves LLM-based scores over the non-personalized baseline by 66.7% on LaMP-4, 50.0% on LaMP-5, and 56.4% on LaMP-7. In human evaluation, the gains are even larger: 72.7% on LaMP-4, 94.4% on LaMP-5, and 70.0% on LaMP-7. Results suggest that group-level adaptation captures meaningful stylistic and preference-related signals that are under-reflected by lexical-overlap metrics. We also have a deep analysis of clustering robustness in Appendix F. **The ablation results support CARD's hierarchical design: Group LoRA provides a strong coarse prior, while the user vector further refines it into fine-grained personalization.**

4.3 User Vector Analysis

To deeper understand user vector, we conduct experiments on dimension showing in Table 3, strength in Figure 5 and representation depth in Table 4.

Moderate personalization strength yields the highest generation quality. As the strength increases from low to moderate values, user-specific signals are effectively amplified, leading to improved personalization. However, further increasing the strength causes the user vector to dominate generation, overwhelming semantic content and resulting in sharp performance drops across tasks.

User vectors with moderate dimensionality and representation depth achieve the best personalization performance. Increasing dimensionality from small sizes improves the expressive capacity of the user vector, enabling it to capture richer user preferences. Performance peaks at an intermediate dimensionality, after which larger vectors introduce noise or overfitting, particularly under limited data.

Aggregating user representations from an intermediate hidden states depth performs the best. Using too few layers limits the representational richness of the user vector, as it relies on a single highly compressed abstraction. In contrast, aggregating too many layers introduces heterogeneous signals with varying levels of abstraction, which can dilute user-specific information and add noise.

4.4 Case Study

As illustrated in the case study showing in Figure 3, CARD's output faithfully retains the central themes while aligning closely with the user's habitual expressive style through informal wording, heightened emotional cues, and light emojis. **Compared with using group-level LoRA alone, CARD achieves a better balance between readability, semantic stability, and stylistic personalization**, resulting in outputs that are most similar to the reference and demonstrating the effective

Task: Tweet Paraphrasing (LaMP-7)

User History

i Thought Fathers day had already been :/ I made richard hammond sign his book for my dad
@ijustine - doesnt work for me either
Morning tweets, I went bed at 4:30 and finally just got up, looking forward to tonight though
i dont want to go out today i wish i got hit by a bus :/
whilst tidying my room, i found 3 easter eggs which i havent even opened
@emmaMay88 i need a job too im broke all the time!
just woke up, i dont want to get out of bed yet
Just about to meet Richard hammond <3 I effin luvvv the Hammond
Farmer cutting the field, now I can lay in it all day
Steve Jobs had a liver transplant i hope he is very well! We miss you Steve come back soon!!
i love having a lie in and not having to rush around

Inputs and Reference Answer

Query: I long for my iPhone and eagerly anticipate Friday.
Instruction: Paraphrase the following tweet without any explanation before or after it.
Reference answer: I miss my iPhone I wish it was friday so much!!

Generated Outputs

Non-personalized: i want my iphone and Looking forward to Friday
RAG: I miss my iPhone and can't wait for Friday
LoRA only: i miss my iphone and i keep waiting for friday!! Happy fridaaays!!
User vector: i miss my iphone and I luvvv friday!!
CARd: i miss my iPhone :/ I wish friday to come already!!

Figure 3: A case study for LaMP-7. The gray background highlights the longest contiguous span shared with the reference/user's answer, while the yellow background highlights content overlapping with the user history.

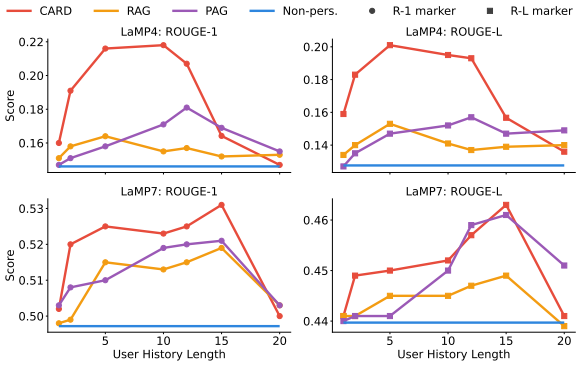

Figure 4: CARD performance comparison for low resource setting

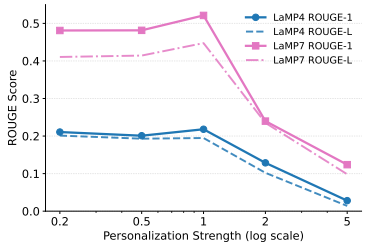

Figure 5: User vector personalization strength.

Table 3: Results on LaMP tasks across different user vector dimensions J.

Task	Metric	32	64	128	256
LaMP-4	R-1	0.149	0.195	**0.218**	0.183
	R-L	0.131	0.189	**0.195**	0.177
LaMP-5	R-1	0.402	0.446	**0.459**	0.451
	R-L	0.325	0.375	**0.387**	0.381
LaMP-7	R-1	0.447	0.511	**0.521**	0.473
	R-L	0.379	0.439	**0.448**	0.419

Table 4: Results on LaMP tasks under different depth S of hidden layers.

Task	Metric	Non-pers.	1	4	8
LaMP-4	R-1	0.146	0.125	**0.218**	0.193
	R-L	0.128	0.110	**0.195**	0.174
LaMP-5	R-1	0.425	0.388	**0.459**	0.442
	R-L	0.342	0.312	**0.387**	0.365
LaMP-7	R-1	0.497	0.406	**0.521**	0.486
	R-L	0.440	0.323	**0.448**	0.429

fusion of shared group semantics with fine-grained individual preferences.

4.5 Low-Resource Users Analysis

To answer **RQ3**, we evaluate the effectiveness of CARD under low-resource settings, we mask user histories by retaining only the first L histories for each testing user, corresponds to User History Length in the figure 4. We observe that: **CARD remains effective for low resource users.** With very limited history($L = 5$), CARD achieves an R-1 of ∼0.216 on LaMP-4, visibly outperforming the non-personalized baseline ∼0.146.
CARD is more sample-efficient in the low-resource situations. With only a few histories, CARD quickly approaches its peak performance (e.g., LaMP4 peaks around L≈10 with ROUGE-1 ∼0.219), while other methods improve more gradually, indicating CARD can extract preference signals more efficiently from limited user history.
Long-history gains are constrained by history quality. Performances drop on LaMP-7 when L is large and for CARD this likely reflects noisy histories that weaken the learned user vector.

4.6 Robustness Across Model Scales and Families

To evaluate the scalability and backbone-agnostic properties of CARD, we extend our experiments across the Qwen model family, spanning 0.6B to 32B parameters. In Table 5, absolute generation quality naturally improves with larger base models.

Table 5: Performance of CARD across varying Qwen model scales.

Task	Metric	Model Scale (Qwen3 Backbone)			
		0.6B	1.7B	8B	32B
LaMP-4:	R-1	0.165	0.188	0.218	**0.220**
News Headline	R-L	0.146	0.168	0.195	**0.212**
LaMP-5:	R-1	0.435	0.447	0.459	**0.463**
Scholarly Title	R-L	0.365	0.373	0.387	**0.391**
LaMP-7:	R-1	0.514	0.518	0.521	**0.528**
Tweet Paraph.	R-L	0.439	0.443	0.448	**0.455**

Table 6: Efficiency Comparison among methods.

Metric	RAG	PEFT	CARD
Training Time/User	$O(\|P_u\|)^1$	$O(\|P_u\|)$	$O(\|\mathcal{D}_u\|)$
Latency/Query	$O(\|P_u\|)$	$O(\text{Load} + \text{Merge})$	$O(kj)$
Storage/User	$O(\|P_u\| \cdot d_e)$	$O(rHL)$	$O(D)$

CARD maintains robust personalization efficacy across all capacities.

Furthermore, we verify CARD's generalizability across different model architectures by providing consistent results on the Llama family backbone in Appendix E.

4.7 Efficiency Analysis

To answer **RQ4**, we compare the complexity of CARD against existing baselines in Table 6. For a new user, CARD incurs only a lightweight, training-free preprocessing cost to encode $|P_u|$ profile items for group assignment. User-specific personalization is then achieved by optimizing a compact J-dimensional preference vector λ_u (with J=128), while freezing the backbone and LoRA modules. CARD is highly efficient in both computation and storage. It can be stored directly on the user device and used for on-device personalization during inference, reducing memory overhead while offering stronger privacy protection for user-specific preference information.

5 Related Work

5.1 Personalized LLMs

LLM personalization involves conditioning a frozen backbone or updating parameters, with LaMP (Salemi et al., 2024b) serving as standard benchmarks. Conditioning approaches include retrieval and profile summarization (e.g., PEARL (Mysore et al., 2024), ROPG-RL (Salemi et al., 2024a)), alongside user representation injection and memory editing methods like PPLUG (Liu et al., 2024), MEMPROMPT (Madaan et al., 2022), TEACHME (Dalvi Mishra et al., 2022), and RECAP (Liu et al., 2023). Beyond explicit conditioning, works such as TEACHLLMS (Li et al., 2023), PUGC (Tan et al., 2025), and DPL (Qiu et al., 2025) exploit implicit supervision from user content.

5.2 PEFT for Personalization

PEFT encodes user information via lightweight adapters on a shared backbone, ranging from per-user adapters to compositional assembly from shared adapter pieces (Tan et al., 2024a,b). While methods like Prefix-Tuning (Li and Liang, 2021) and P-Tuning (Liu et al., 2022) optimize continuous prompts, recent work explores group-level or hierarchical designs that amortize learning across users, as well as factorization-based views that enable black-box alignment (Zhang et al., 2025b; Zhuang et al., 2024; Zhang et al., 2024b,a; Zhu et al., 2024; Kong et al., 2024).

5.3 Decoding-time Alignment with User Preferences

Decoding-time alignment adjusts the outputs of frozen language models at inference time without parameter updates (Kim et al., 2025; Zhang et al., 2025a; Shenfeld et al., 2025; Deng and Raffel, 2023). Related work employs preference-vectors-based steering to induce desired behaviors, frequently in personalization scenarios(Jang et al., 2023; Cao et al., 2024; Zhang et al., 2025c; Tang et al., 2024; Bu et al., 2025).Some studies further formalize personalization as inference time alignment driven by lightweight user interaction signals (Chen et al., 2025; Gao et al., 2024; Kirk et al., 2024).

6 Conclusion

We presented **CARD**, a coarse-to-fine personalization text generation framework. Combining group-level adapters with lightweight user-specific modulation at the logit level, CARD achieves fine-grained personalization without per-user model fine-tuning or long-context history retrieval. Experiments on LaMP and LongLaMP demonstrate that CARD consistently improves personalization quality while maintaining strong generalization and efficiency.

[1]Training-free. Denotes pre-processing cost. D=128. H denotes the hidden size of the LLM and L denotes the number of Transformer layers. For CARD, k denotes the number of selected components and j the projection dimension.

Limitations

While CARD is effective and efficient for personalized text generation, it has several limitations. First, its offline group modeling relies on unsupervised K-means clustering, which may not fully capture complex user relationships or latent personalization structure. Second, CARD represents each user with a single preference vector during decoding, which may limit expressiveness for diverse or evolving preferences, and the learned dimensions are not directly interpretable. Third, although the overall framework is lightweight, its multi-stage pipeline still requires coordination across several components. Finally, noisy or weakly relevant histories may degrade the learned user vector and reduce personalization quality. Incorporating history filtering, relevance weighting, or noise-robust profile selection could alleviate this issue, but we leave such extensions to future work.

Ethical Considerations

The LaMP and LongLaMP benchmarks used in this work are publicly available and anonymized, and therefore do not raise direct privacy concerns. All datasets were obtained from prior work through official APIs, and no proprietary or non–open-source data is involved. Personalized language generation may introduce risks related to user privacy, as it often relies on historical user data. Our approach mitigates these risks by decoupling personalization signals from raw user text. Users can locally construct personal representations without uploading their historical data, while service providers only release a lightweight personalization module. Compared to retrieval-based or user-specific fine-tuning methods, this design substantially reduces the risk of data leakage. All experiments were conducted using publicly available models and APIs in compliance with standard research ethics.

References

Lisa P. Argyle, Ethan C. Busby, Nancy Fulda, Joshua R. Gubler, Christopher Rytting, and David Wingate. 2023. Out of one, many: Using language models to simulate human samples. *Political Analysis*, 31(3):337–351.

Angels Balaguer, Vinamra Benara, Renato Luiz de Freitas Cunha, Roberto de M. Estevão Filho, Todd Hendry, Daniel Holstein, Jennifer Marsman, Nick Mecklenburg, Sara Malvar, Leonardo O. Nunes, Rafael Padilha, Morris Sharp, Bruno Silva, Swati Sharma, Vijay Aski, and Ranveer Chandra. 2024. RAG vs. fine-tuning: Pipelines, tradeoffs, and a case study on agriculture. *arXiv:2401.08406*.

Ralph Allan Bradley and Milton E. Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Hyungjune Bu, Chanjoo Jung, Minjae Kang, and Jaehyung Kim. 2025. Personalized llm decoding via contrasting personal preference. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33946–33966.

Yuanpu Cao, Tianrong Zhang, Bochuan Cao, Ziyi Yin, Lu Lin, Fenglong Ma, and Jinghui Chen. 2024. Personalized steering of large language models: Versatile steering vectors through bi-directional preference optimization. *arXiv:2406.00045*.

Ruizhe Chen, Xiaotian Zhang, Meng Luo, Wenhao Chai, and Zuozhu Liu. 2025. Pad: Personalized alignment of llms at decoding-time.

Bhavana Dalvi Mishra, Oyvind Tafjord, and Peter Clark. 2022. Towards teachable reasoning systems: using a dynamic memory of user feedback for continual system improvement. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Haikang Deng and Colin Raffel. 2023. Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11781–11791.

Ge Gao, Alexey Taymanov, Eduardo Salinas, Paul Mineiro, and Dipendra Misra. 2024. Aligning llm agents by learning latent preference from user edits. *arXiv:2404.15269*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. *Preprint*, arXiv:1902.00751.

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

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021a. Unsupervised dense information retrieval with contrastive learning. *arXiv:2112.09118*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021b. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Joel Jang, Seungone Kim, Bill Yuchen Lin, Yizhong Wang, Jack Hessel, Luke Zettlemoyer, Hannaneh Hajishirzi, Yejin Choi, and Prithviraj Ammanabrolu. 2023. Personalized soups: Personalized large language model alignment via post-hoc parameter merging. *arXiv:2310.11564*.

Minbeom Kim, Kang il Lee, Seongho Joo, Hwaran Lee, Thibaut Thonet, and Kyomin Jung. 2025. Drift: Decoding-time personalized alignments with implicit user preferences.

Hannah Rose Kirk, Alexander Whitefield, Paul Röttger, Andrew Bean, Katerina Margatina, Juan Ciro, Rafael Mosquera, Max Bartolo, Adina Williams, He He, Bertie Vidgen, and Scott A. Hale. 2024. The prism alignment dataset: What participatory, representative and individualised human feedback reveals about the subjective and multicultural alignment of large language models. *arXiv preprint arXiv:2404.16019*.

Xiaoyu Kong, Jiancan Wu, An Zhang, Leheng Sheng, Hui Lin, Xiang Wang, and Xiangnan He. 2024. Customizing language models with instance-wise lora for sequential recommendation. In *NeurIPS*.

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:2407.11016*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention.

Cheng Li, Mingyang Zhang, Qiaozhu Mei, Yaqing Wang, Spurthi Amba Hombaiah, Yi Liang, and Michael Bendersky. 2023. Teach llms to personalize – an approach inspired by writing education. *arXiv:2308.07968*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation.

Xun Liang, Hanyu Wang, Yezhaohui Wang, Shichao Song, Jiawei Yang, Simin Niu, Jie Hu, Dan Liu, Shunyu Yao, Feiyu Xiong, and Zhiyu Li. 2024. Controllable text generation for large language models: A survey. *arXiv:2408.12599*.

Jiongnan Liu, Yutao Zhu, Shuting Wang, Xiaochi Wei, Erxue Min, Yu Lu, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. 2024. Llms + persona-plug = personalized llms. *arXiv:2409.11901*.

Shuai Liu, Hyundong J. Cho, Marjorie Freedman, Xuezhe Ma, and Jonathan May. 2023. Recap: Retrieval-enhanced context-aware prefix encoder for personalized dialogue response generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve gpt-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Sheshera Mysore, Zhuoran Lu, Mengting Wan, Longqi Yang, Bahareh Sarrafzadeh, Steve Menezes, Tina Baghaee, Emmanuel Barajas Gonzalez, Jennifer Neville, and Tara Safavi. 2024. Pearl: Personalizing large language model writing assistants with generation-calibrated retrievers. In *CustomNLP4U*.

OpenAI. 2025. Update to gpt-5 system card: Gpt-5.2. Technical report, OpenAI.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. *Preprint*, arXiv:2304.03442.

Joon Sung Park, Lindsay Popowski, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2022. Social simulacra: Creating populated prototypes for social computing systems. *Preprint*, arXiv:2208.04024.

Yilun Qiu, Xiaoyan Zhao, Yang Zhang, Yimeng Bai, Wenjie Wang, Hong Cheng, Fuli Feng, and Tat-Seng Chua. 2025. Measuring what makes you unique: Difference-aware user modeling for enhancing llm personalization. In *Findings of ACL*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36.

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:2310.20081*.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Stephen E. Robertson and Stephen Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241, London, UK. Springer-Verlag.

Alireza Salemi, Surya Kallumadi, and Hamed Zamani. 2024a. Optimization methods for personalizing large language models through retrieval augmentation. In *SIGIR*.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2024b. Lamp: When large language models meet personalization. In *ACL*.

Murray Shanahan, Kyle McDonell, and Laria Reynolds. 2023. Role-play with large language models. *Preprint*, arXiv:2305.16367.

Idan Shenfeld, Felix Faltings, Pulkit Agrawal, and Aldo Pacchiano. 2025. Language model personalization via reward factorization. *arXiv:2503.06358*.

Zhaoxuan Tan and Meng Jiang. 2023. User modeling in the era of large language models: Current research and future directions. *arXiv:2312.11518*.

Zhaoxuan Tan, Zheng Li, Tianyi Liu, Haodong Wang, Hyokun Yun, Ming Zeng, Pei Chen, Zhihan Zhang, Yifan Gao, Ruijie Wang, Priyanka Nigam, Bing Yin, and Meng Jiang. 2025. Aligning large language models with implicit preferences from user-generated content. In *ACL*.

Zhaoxuan Tan, Qingkai Zeng, Yijun Tian, Zheyuan Liu, Bing Yin, and Meng Jiang. 2024a. Democratizing large language models via personalized parameter-efficient fine-tuning. In *EMNLP*.

Zhaoxuan Tan, Qingkai Zeng, Yijun Tian, Zheyuan Liu, Bing Yin, and Meng Jiang. 2024b. Personalized pieces: Efficient personalized large language models through modular and collaborative peft. In *EMNLP*.

Xiangru Tang, Xingyao Zhang, Yanjun Shao, Jie Wu, Yilun Zhao, Arman Cohan, Ming Gong, Dongmei Zhang, and Mark Gerstein. 2024. Step-back profiling: Distilling user history for personalized scientific writing. *arXiv:2406.14275*.

The Vicuna Team. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. Blog post.

Andrew Trotman, Antti Puurula, and Blake Burgess. 2014. Improvements to bm25 and language models examined. In *Proceedings of the Open Source Information Retrieval Workshop*, pages 116–123.

Zekun Moore Wang, Zhongyuan Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Jian Yang, Man Zhang, Zhaoxiang Zhang, Wanli Ouyang, Ke Xu, Stephen W. Huang, Jie Fu, and Junran Peng. 2024. Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. *Preprint*, arXiv:2310.00746.

Stanisław Woźniak, Bartłomiej Koptyra, Arkadiusz Janz, Przemysław Kazienko, and Jan Kocoń. 2024. Personalized large language models. *arXiv:2402.09269*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint*, arXiv:2505.09388.

Jinghao Zhang, Yuting Liu, Wenjie Wang, Qiang Liu, Shu Wu, Liang Wang, and Tat-Seng Chua. 2025a. Personalized text generation with contrastive activation steering. *arXiv:2503.05213*.

Kai Zhang, Yejin Kim, and Xiaozhong Liu. 2024a. Personalized llm response generation with parameterized memory injection. *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, Vienna, Austria. Association for Computational Linguistics.

You Zhang, Jin Wang, Liang-Chih Yu, Dan Xu, and Xuejie Zhang. 2024b. Personalized lora for human-centered text understanding. In *AAAI*.

Zhaowei Zhang, Fengshuo Bai, Qizhi Chen, Chengdong Ma, Mingzhi Wang, Haoran Sun, Zilong Zheng, and Yaodong Yang. 2025c. Amulet: Realignment during test time for personalized preference adaptation of llms. *arXiv:2502.19148*.

Zhehao Zhang, Ryan A. Rossi, Branislav Kveton, Yijia Shao, Diyi Yang, Hamed Zamani, Franck Dernoncourt, Joe Barrow, Tong Yu, Sungchul Kim, Ruiyi Zhang, Jiuxiang Gu, Tyler Derr, Hongjie Chen, Junda Wu, Xiang Chen, Zichao Wang, Subrata Mitra, Nedim Lipka, and 2 others. 2024c. Personalization of large language models: A survey. *arXiv preprint*, arXiv:2411.00027.

Jiachen Zhu, Jianghao Lin, Xinyi Dai, Bo Chen, Rong Shan, Jieming Zhu, Ruiming Tang, Yong Yu, and Weinan Zhang. 2024. Lifelong personalized low-rank adaptation of large language models for recommendation. *arXiv:2408.03533*.

Yuchen Zhuang, Haotian Sun, Yue Yu, Rushi Qiang, Qifan Wang, Chao Zhang, and Bo Dai. 2024. Hydra: Model factorization framework for black-box llm personalization. *arXiv:2406.02888*.

Appendix Contents

A Relative Improvements of CARD

Table 7: Relative improvement (%) of CARD over the non-personalized baseline.

Task	R-1	R-L
LaMP4: Headlines	+49.3%	+53.5%
LaMP5: Scholarly	+8.0%	+13.16%
LaMP7: Tweets	+4.8%	+2.1%
Long1: Abstract	+24.2%	+71.7%
Long2: Topic	+2.0%	+6.7%
Long3: Review	+38.7%	+20.0%

B LLM-as-judge prompts and human evaluation rubrics

Persona Score evaluation prompt

Task Description: You are given: (1) an instruction (which may include an input), (2) a response to evaluate, (3) a reference answer that should receive a score of 5, (4) a user profile containing the user's preferences and background, and (5) a score rubric describing the evaluation criteria.
Your task is to:
1. Write detailed feedback that assesses how well the response is personalized to this specific user, strictly following the given score rubric. Do **not** comment on general quality unrelated to personalization. 2. Carefully consider how the response aligns with the user's preferences, interests, and background information in the user profile. 3. After writing the feedback, output a single integer score between 1 and 5 that best matches the rubric. You **must** choose an integer. 4. The output format must be: "(write a feedback for criteria) [RESULT] (an integer number between 1 and 5)" 5. Do not add any extra opening, closing, or explanations.
The instruction to evaluate: instruction
Response to evaluate: response
Reference Answer (Score 5): reference answer
User Profile: user profile
Score Rubric: rubric
Feedback:

Score Rubric

criteria: "Evaluate how well the response is personalized to the specific user."
score1 description: "Generic or impersonal. Ignores the provided user profile and personality cues. Style does not match the user; may feel robotic, off-topic, or contradict stated preferences. No meaningful use of user details; largely boilerplate."
score2 description: "Minimal personalization. Mentions a profile detail superficially but remains mostly generic. Weak style match; limited relevance to the user's interests or situation. Includes filler or distracting disclaimers. Significant deviation from the reference's intent or emphasis."
score3 description: "Basic personalization. References a few relevant details and partially adapts to them. Generally on topic but misses important user nuances (interests, constraints, or personality cues). Moderate similarity to the reference; may be verbose or somewhat generic."
score4 description: "Good personalization. Integrates multiple user details accurately; content is relevant and helpful. Tone mostly matches the user's personality and preferred style. Clear and engaging, with only minor misses compared to the reference."
score5 description: "Excellent personalization. Seamlessly weaves in key profile details; highly relevant and tailored guidance or conversation. Tone closely matches the user's personality—empathetic, engaging, and concise. Avoids boilerplate and unnecessary disclaimers. Very closely aligned with the user's likely preference as indicated by the reference."

C Dataset Statistics and Task Details

Detailed statistics for the six tasks are provided in Table 8. The formats of input, output, and user histories of the tasks are shown in Table 9. In all experiments, we use the validation(Val) dataset as testing dataset since the official testing dataset is not public.

D Baseline Details

This section documents how we reproduced the baselines reported in Table 1. Across all baselines,

Table 8: Data statistics of experimented tasks in the LaMP benchmark.

Task	Type	Train	Val	In Len.	Out Len.	Hist.	#Cls
LaMP-1	Binary	6,542	1,500	51.4 ± 5.7	–	84.1 ± 47.5	2
LaMP-2	Category	5,073	1,410	92.4 ± 21.9	–	86.8 ± 189.5	15
LaMP-3	Ordinal	20,000	2,500	128.2 ± 146.2	–	185.4 ± 129.3	5
LaMP-4	Gen	12,500	1,500	30.0 ± 12.1	10.1 ± 3.1	204.6 ± 250.7	–
LaMP-5	Gen	14,682	1,500	162.3 ± 65.6	9.7 ± 3.2	87.9 ± 53.6	–
LaMP-7	Gen	13,437	1,498	29.7 ± 7.0	17.0 ± 5.7	15.7 ± 14.8	–

Table 9: Format of input, output, and user history.

Task	Input	Output	User History
LaMP-4	Gen headline: {*article*}	How I Got 'Rich'	title: {*title*} text: {*article*}
LaMP-5	Gen title for abstract: {*abstract*}	Distributed Partial Clustering	title: {*title*} text: {*abstract*}
LaMP-7	Paraphrase tweet: {*tweet*}	gotta make the most of my last day	text: {*tweet*}

we keep the same backbone LLM, task instruction templates, and decoding setup as CARD (Appendix 10), and only vary how user information is incorporated. Recent surveys highlight that comparing these pipelines systematically reveals distinct trade-offs between context-window utilization and adaptation cost (Balaguer et al., 2024; Zhang et al., 2024c; Tan and Jiang, 2023; Woźniak et al., 2024; Argyle et al., 2023). Furthermore, theoretical frameworks suggest that effective personalization relies on the model's inherent capacity for role-play and mimicry (Shanahan et al., 2023), which we aim to steer via different context augmentation strategies.

Common task instructions. For LaMP tasks, we use the original task instructions/templates (same as Table 9 in our appendix). For LongLaMP tasks, we use the official task instructions provided with the benchmark release. No extra few-shot demonstrations are added beyond user history augmentation (when applicable).

Zero-shot inference acceleration. For prompt-only baselines (Non-pers., RAG, and PAG), we run batched inference with vLLM for throughput, while using identical decoding parameters (greedy decoding, same max new tokens, and repetition penalty). This does not change model behavior; it only improves serving efficiency.

Shared retrieval settings (RAG/PAG/OPPU-hybrid). Following the LaMP retrieve-then-prompt protocol, we retrieve from the *same user*'s history/profile and augment the prompt with the retrieved items. We use BM25 as the sparse retriever and set k=4. The query function is the current task input (i.e., $\phi_q(x) = x$), as in LaMP (Salemi et al., 2024b), and we treat each history entry as a BM25 "document." Retrieving past behaviors to ground current generation is a foundational technique in social simulation (Park et al., 2022). BM25 is reported to be a robust term-matching retriever and performed competitively on LaMP tasks (Salemi et al., 2024a). see also analyses of BM25-style scoring and improvements (Trotman et al., 2014). When a user has fewer than k entries, we use all available items.

Shared truncation / context budgeting. We keep the task instruction and the current input intact. If the concatenation of profile summary (PAG), retrieved history (RAG/PAG), and the task input exceeds the context limit, we truncate in the following order: (1) trim each history item to a fixed per-item budget, (2) trim the profile summary, (3) finally, if still necessary, reduce the number of retrieved items (keeping the top-scored ones first).

Profile generation model (PAG and OPPU-hybrid). For any baseline requiring a textual user profile s_u (PAG and OPPU-hybrid), we generate s_u once per user offline using GPT-5.2 with deterministic decoding (temperature $= 0$), and cache it for inference. The summary prompt instructs the model to capture writing style, recurring topics, and formatting patterns; similar summary generation has been shown to improve retrieval-augmented personalization. This approach aligns with methodologies in role-playing agents, where extracting stylistic nuances from observational data is key to constructing faithful user simulacra (Wang et al., 2024; Park et al., 2023). Compared with summarization models like Vicuna or ChatGPT used in prior work (Team, 2023), GPT-5.2 offers a larger context window.

LongLaMP task-specific considerations. The LongLaMP benchmark introduces three long-form personalization tasks beyond LaMP: personalized abstract generation (LongLaMP1), personalized topic writing (LongLaMP2), and personalized review writing (LongLaMP3). Each task requires adapting the retrieval query ϕ_q to use salient non-templated parts and adjusting profile summarization to handle long histories (Kumar et al., 2024).

- **LongLaMP1 (Personalized Abstract Gen-**

eration). The expected output y is a scientific abstract conditioned on the paper's title and selected keywords. The user profile P_u consists of the author's previous papers, and we generate this profile using the Citation Network Dataset. When constructing the retrieval query, we set $\phi_q(x)$ to be the concatenation of the paper title and keywords. Because abstracts are longer than typical LaMP outputs, we cap each retrieved paper at a fixed token budget and include up to four documents in the prompt.

- **LongLaMP2 (Personalized Topic Writing).** This task generates the content of a Reddit post y from a post summary x and the author's prior posts. The user profile P_u is a set of (summary, content) pairs from the same author, taken from the Reddit TL;DR dataset. We set $\phi_q(x)$ to the post summary; retrieval uses BM25 to fetch up to four of the author's previous posts. Given the variability of Reddit writing (creative writing, sarcasm, domain-specific jargon), we rely on profile summaries to encode writing style and on retrieval to provide topic-specific context.

- **LongLaMP3 (Personalized Review Writing).** The output is a comprehensive product review; the input x comprises the product description, the user's product rating, and a summary of the user's experience. The user profile contains the author's other lengthy reviews (text, summary, rating, product description). We set $\phi_q(x)$ to the concatenation of the product description and rating, and we include retrieved past reviews (up to four) to provide exemplars of tone and preference. Since reviews are long and domain-specific, our profile summary distills consistent sentiment and product features across the user's prior reviews.

The remainder of this section details each baseline and includes the prompt templates used for reproduction.

Baseline prompt skeleton

[USER PROFILE] (empty for RAG; filled for PAG / OPPU-hybrid)
{profile_summary s_u}

[RETRIEVED HISTORY] (empty for Non-pers.; BM25 top-k for RAG/PAG/OPPU-hybrid)
(1) {history_item_1}
(2) {history_item_2}
...
(k) {history_item_k}

[TASK INSTRUCTION + INPUT]
{task_instruction_template_from_Table 9}
{current_input x}

[MODEL OUTPUT]
(model generates the answer text only)

History item serialization used for BM25 indexing and prompt insertion

We treat each user history entry as one "document" for BM25, and use the same serialization when inserting into prompts.
LaMP4 (Headline) / LongLaMP writing tasks:
title: {title}
text: {article or long text}
LaMP5 (Scholarly) / LongLaMP1 (Abstract):
title: {title}
abstract: {abstract}
LaMP7 (Tweet Paraphrasing):
text: {tweet}
If a history entry contains an (input, output) pair:
We keep both fields in the serialized form (e.g., {input_field} + {output_field}), so the model can pick up patterns and style.

Profile generation prompt for PAG

SYSTEM:
You are a careful assistant that summarizes a user's historical behaviors into a compact profile for personalization. Only use information supported by the provided history. Do not invent facts.

USER:
Given the user's historical behaviors below, write a concise user profile that will help a language model generate outputs aligned with this user. The profile should capture: (i) writing style/tone, (ii) recurring topics or preferences, and (iii) any consistent formatting patterns. Keep it compact and directly useful for

downstream prompting. Output only the profile text, no extra headings.

USER HISTORY (chronological):
(1) {history_entry_1}
(2) {history_entry_2}
...
(n) {history_entry_n}

OUTPUT:
{profile_summary s_u}

D.1 Non-personalized (Non-pers.)

The non-personalized baseline removes all user-specific signals. The prompt contains only the task instruction and the raw task input x. This baseline is equivalent to a generic prompt without retrieval; even random retrieval can improve results (Salemi et al., 2024b), so Non-pers. serves as a conservative lower bound.

Non-personalized inference prompt

[TASK INSTRUCTION + INPUT]
{task_instruction_template}
{current_input x}

[MODEL OUTPUT]
(model generates the answer text only)

D.2 RAG (Retrieval-Augmented Generation)

Retriever. We implement retrieval-augmented prompting using BM25 over the current user's history. BM25 is considered a robust term-matching retrieval model and outperformed other baselines like random selection and recency on many LaMP tasks (Salemi et al., 2024b). Dense retrieval methods (e.g., Contriever) sometimes yield marginally higher accuracy but incur more latency (Izacard et al., 2021a); we adopt BM25 for efficiency. While generation-calibrated retrievers (Mysore et al., 2024) offer advanced personalization capabilities, we adhere to the standard BM25 setup for consistent benchmarking.

Query and retrieval. We use the current task input text as the retrieval query ($\phi_q(x) = x$), as described in LaMP. For each example, we retrieve the top-k=4 history entries by BM25 score. If a user has fewer than k entries, we include all available items. Increasing k beyond 4 can slightly improve performance but is constrained by the context length of our backbone LLM.

Prompt construction. We leave the **[USER PROFILE]** section empty and insert the retrieved entries into the **[RETRIEVED HISTORY]** section using the serialization described above. For generation tasks where a history item is an (input, output) pair, we serialize it as (`history_input` → `history_output`) so the LLM can imitate formatting and style. The remainder of the prompt comprises the task instruction and the current input.

LongLaMP adaptation. For LongLaMP1/2/3, we adjust the query and retrieval source according to each task:

- *Abstract generation (LongLaMP1).* Use the title + keywords as the query; retrieve top-4 previous papers (Kumar et al., 2024).
- *Topic writing (LongLaMP2).* Use the post summary as the query; retrieve top-4 prior posts.
- *Review writing (LongLaMP3).* Use the product description and rating as the query; retrieve top-4 past reviews.

RAG inference prompt (BM25)

[RETRIEVED HISTORY]
(1) {history_item_1}
(2) {history_item_2}
(3) {history_item_3}
(4) {history_item_4}

[TASK INSTRUCTION + INPUT]
{task_instruction_template}
{current_input x}

[MODEL OUTPUT]
(model generates the answer text only)

D.3 PAG (Profile-Augmented Generation)

Offline profile summary. PAG extends RAG by including a concise user profile s_u generated offline. Following, we generate s_u via an instruction-tuned LLM (GPT-5.2, temperature 0), which summarises salient information from the user history. Summaries are generated once per user and cached for inference, reducing runtime costs.

Inference-time prompt. At inference time, we prepend s_u in the **[USER PROFILE]** section and include BM25 top-k=4 retrieved history in **[RETRIEVED HISTORY]**. This matches

the profile-augmented prompt $\phi_p(x_u, D_u, s_u)$ described in Richardson et al. (2023), where $D_u = R(\phi_q(x_u), H_u, k)$ and $s_u = \mathrm{LLM}(H_u)$.

LongLaMP adaptation. For LongLaMP tasks, we generate user profiles summarizing the author's prior papers, posts, or reviews, respectively, and we use task-specific queries for retrieval:

- *Abstract generation.* Summarize past papers and use title+keywords to retrieve relevant papers.
- *Topic writing.* Summarize past posts and use the post summary to retrieve relevant posts.
- *Review writing.* Summarize past reviews and use the product description and rating to retrieve relevant reviews.

Context budgeting. To respect the backbone context limit, we truncate the profile and/or retrieved items when needed (see the "Shared truncation" paragraph).

PAG inference prompt (profile + BM25)

[USER PROFILE]
{profile_summary s_u}

[RETRIEVED HISTORY]
(1) {history_item_1}
(2) {history_item_2}
(3) {history_item_3}
(4) {history_item_4}

[TASK INSTRUCTION + INPUT]
{task_instruction_template}
{current_input x}

[MODEL OUTPUT]
(model generates the answer text only)

D.4 PPLUG (Persona-Plug User Embedding)

Persona-Plug (PPlug) introduces a plug-and-play user embedder that produces a single personal embedding $P_u(x)$ from all user histories, guiding a frozen LLM without explicit retrieval or textual profile (Liu et al., 2024).

User behavior encoder and aggregation. Each historical behavior $h_i \in H_u$ is encoded into a dense vector; the current input x is encoded into a query vector. An input-aware attention mechanism computes weights

$$w_i = \frac{\exp(x_u^\top h_i)}{\sum_j \exp(x_u^\top h_j)},$$

and the personal embedding is

$$P_u(x) = \sum_i w_i \cdot \mathrm{Proj}(h_i),$$

where $\mathrm{Proj}(\cdot)$ is a learned projection mapping user embeddings to the LLM representation space.

Embedding attachment. After computing $P_u(x)$, we attach it as a continuous prefix in the embedding sequence sent to the backbone LLM:

$$X_u = [\mathrm{Emb}(\text{instruction});\ P_u(x);\ \mathrm{Emb}(x);\ \mathrm{Emb}(y_{<t})],$$

where the instruction embedding is trainable. Only the instruction embedding, input encoder, and projection network (a 2-layer MLP) are trained; the backbone LLM remains frozen.

Training. We train the plug-in user embedder with the next-token prediction loss on the training set:

$$\mathcal{L} = -\sum_u \sum_i \log p_{\mathrm{LLM}}(y_{u,i} \mid X_u).$$

This approach allows efficient personalization since the backbone parameters are not updated, and the user embedder is shared across users.

LongLaMP adaptation. For LongLaMP tasks, we encode each long-form document (paper, post, review) in the user history as a behavior vector. The query vector is derived from the input (title + keywords, post summary, or product description + rating). This ensures that the attention weights w_i reflect the relevance of each past long document to the current long-text generation task.

PPLUG input construction

The prompt presented to the LLM in text form remains identical to Non-pers (task instruction + input). PPlug modifies the *embedding-level* input to the backbone:
Input embedding sequence to the frozen LLM:
$X_u = [\mathrm{Emb}(\text{instruction});\ P_u(x);\ \mathrm{Emb}(x);\ \mathrm{Emb}(y_{<t})]$
where $P_u(x)$ is the aggregated personal embedding computed from all histories conditioned on the current input.

D.5 OPPU (One PEFT Per User, LoRA)

OPPU is a parametric personalization baseline that equips each user u with a personalized low-rank adapter (LoRA) module (Tan et al., 2024a). Unlike PPlug, OPPU updates task-specific parameters for each user while keeping the base LLM frozen.

Stage 1: task adaptation. We first adapt the backbone LLM to each task using LoRA (Hu et al., 2021). LoRA updates only ∼0.5% of the parameters; after training, the LoRA parameters are merged into the base model, producing a task-adapted base checkpoint.

Stage 2: per-user LoRA. For each user u, we train personal LoRA parameters $\Delta\Theta_u^{(B)}$, $\Delta\Theta_u^{(R)}$, and $\Delta\Theta_u^{(P)}$ that augment the base model under three settings: vanilla, retrieval-augmented, and profile-augmented. These personal PEFT modules are small (rank r=8 in our reproduction), and the base model parameters remain frozen during this stage. This differs from methods that optimize continuous prompts (Li and Liang, 2021) or insert heavy adapter layers (Houlsby et al., 2019), by focusing on modular weight updates.

Training objectives. Given user history H_u and query x_u, the per-user objectives follow Eq. (5) in Tan et al. (2024a):

$$\mathcal{L}_u^{(B)} = \mathrm{CE}[\Theta_u^{(B)}(\phi_t(x_u)),\ y_u],$$
$$\mathcal{L}_u^{(R)} = \mathrm{CE}[\Theta_u^{(R)}(\phi_r(x_u, D_{<t}(x_u))),\ y_u],$$
$$\mathcal{L}_u^{(P)} = \mathrm{CE}[\Theta_u^{(P)}(\phi_p(x_u, D_{<t}(x_u), s_u)),\ y_u],$$

where CE is cross-entropy loss, $D_{<t}(x_u)$ denotes top-k retrieved items from the user history using BM25, and s_u is the profile summary. For tasks where the user history does not align with the supervised format (e.g., tweet paraphrasing), we replace y_u by the right-shifted history and perform unsupervised next-token training.

Hybrid prompting at inference. At inference time, we load the user's LoRA module and augment the prompt with the profile summary s_u and BM25 top-k=4 retrieved histories. This yields a hybrid parametric–nonparametric prompt that combines user-specific parameters with retrieval and summary signals. Following OPPU, we use BM25 for all retrieval operations. For LongLaMP tasks, we use the same task-specific queries and profiles described above.

D.6 PROPER

For PROPER, we follow its progressive 3-stage adaptation setup. For CARD, we use K-means clustering and configure the cluster-LoRA with rank r=16.

D.7 Implementation Details for PAD

We employ **PAD** (Personalized Alignment at Decoding-time) (Chen et al., 2025) as our primary inference-time steering baseline. PAD operates as a **policy-training-free** framework: it modulates the output distribution of a frozen base language model (π_{LM}) via a separately trained *personalized reward model* (PersRM), thereby decoupling preference injection from the base model's context window.

Reward Model Architecture. The PersRM evaluates the compatibility between a user's preference p_u and a candidate token a at step t. We parameterize the reward function as a bilinear form $R(p_u, s_t, a) = w_{p_u}^\top \phi(s_t, a)$. Here, $w_{p_u} \in \mathbb{R}^d$ denotes the user preference embedding, and $\phi(s_t, a) \in \mathbb{R}^d$ represents the state-action feature vector (where $d = 4096$). The PersRM backbone $\hat{\pi}_\theta^*$ is initialized from a reference model $\hat{\pi}_{\mathrm{ref}}$. We note that $\hat{\pi}_{\mathrm{ref}}$ serves as the KL-divergence constraint for the reward model and is conceptually distinct from the inference base model π_{LM}, although they may share similar architectures.

Preference Representation. To ensure a fair comparison with prompt-based baselines, we instantiate the user preference input p_u using the identical profile summaries s_u generated for PAG. We encapsulate the profile into a structured prompt

PAD Preference Prompt (Input to PersRM)

[USER PREFERENCE p_u]
`[Guidelines] Generate outputs aligned with this user's preferences.`
`[Principles] {profile_summary` s_u`}`

Training Protocol. We adhere to the two-stage optimization protocol proposed by Chen et al. (2025): (i) pre-training preference-agnostic features, followed by (ii) freezing the backbone to exclusively optimize the preference mapping $p_u \mapsto w_{p_u}$. Training is performed on triples (p_u, x, y^+, y^-), where the positive sample y^+ is the ground-truth user response, and the negative sample y^- is generated by π_{LM} using **greedy decoding** conditioned on a non-personalized prompt. The model is optimized using the pairwise ranking loss defined in Eq. (9) of Chen et al. (2025).

Inference-Time Steering. During decoding, π_{LM} (Qwen3) remains frozen. At each timestep t, we steer the generation by combining the base

model's likelihood with the reward signal:

$$\text{score}(a) = \log \pi_{\text{LM}}(a \mid s_t) + \beta \cdot \left(w_{p_u}^{\top} \phi(s_t, a) \right)$$

Note that in PAD's implementation, the feature term ϕ is derived from the log-ratio of probabilities between the optimized PersRM and the reference model. We set the candidate pool size $k = 10$ and tune the penalty coefficient β on the validation set.

Since PAD necessitates concurrent forward passes through three models (π_{LM}, $\hat{\pi}_{\theta}^{*}$, and $\hat{\pi}_{\text{ref}}$), efficient implementation is critical. We utilize a vectorized decoding strategy: we compute full-vocabulary logits for all models in a single batch step, but restrict the computationally intensive reward aggregation (dot product and scaling) exclusively to the top-k indices identified by π_{LM}. This significantly reduces inference latency compared to naive implementations.

D.8 Implementation Details and Hyperparameters

Table 10: Implementation details and hyperparameters used in CARD. Note that λ_u denotes the new-user preference learning stage.

Hyperparameter	Value
Backbone Model	
Base Model	Qwen/Qwen3-8B
Thinking Mode	Disabled (`enable_thinking=False`)
Cluster-LoRA Training	
Target Modules	`q,k,v,o,gate,up,down`
LoRA Rank r / α	16 / 16
LoRA Dropout	0.05
Batch Size	8 (2/device $\times$ 4 accum)
Learning Rate	2×10^{-4} (Cosine decay)
Epochs / Warmup	10 / 100 steps
Precision	bf16 (train) + tf32
Max Seq Length	4096 (Label masking active)
New-User λ_u Training	
Trainable Params	User embeddings only
Objective	Pairwise Logistic (Bradley-Terry)
Learning Rate	10^{-2} (AdamW)
Steering Strength β	1.0
Batch Size / Epochs	4 / 3
Inference & Assignment	
Decoding Strategy	Greedy (Temp=0)
Repetition Penalty	1.1
Cluster Embedder	BAAI/bge-m3
Assignment Rule	Nearest Centroid (L_2)

E Generalization Across Model Families: LLaMA-3.1 Results

To demonstrate that the CARD framework is truly backbone-agnostic and generalizes across different model architectures and pre-training distributions, we conduct a full suite of experiments using the **LLaMA-3.1-8B** model. To ensure a strict and fair comparison, we maintain the exact same evaluation metrics (ROUGE-1 and ROUGE-L) and hyperparameter settings as used in our primary Qwen-8B experiments.

As shown in Table 11, CARD achieves state-of-the-art performance across the vast majority of tasks and metrics, confirming the universal effectiveness of our decoding-time steering mechanism. The relative performance trends observed with the Qwen backbone hold consistently: Group PEFT provides a robust baseline, while dense (Contriever) and sparse (BM25) retrievers excel in semantics-heavy and lexical-heavy tasks, respectively.

Importantly, we observe that CARD does not absolutely dominate every single metric, which provides valuable insights into the trade-offs of decoding-time personalization. For instance, on **LaMP5 (Scholarly Title)**, RAG-BM25 marginally outperforms CARD on ROUGE-1. This is highly expected: academic titles rely heavily on exact matching of rare scientific terminologies, a scenario where sparse retrieval excels by directly placing exact keywords into the prompt context for the LLM to copy. Similarly, on **LongLaMP2 (Topic Writing)**, Group PEFT slightly edges out CARD on the ROUGE-L metric. Since Reddit posts are highly divergent and lengthy, the global group-level prior occasionally maintains long-range structural coherence slightly better than the fine-grained, token-level modulation introduced by the user vector.

Despite these minor and predictable trade-offs, CARD delivers the most balanced and superior overall personalization capability without relying on heavy per-user parameter storage.

F Clustering analysis

We provide a deeper look at two clustering-related design choices in CARD: (i) the granularity controlled by the cluster number K, and (ii) the choice of clustering algorithm used to obtain user groups.

Table 11: Performance comparison on LaMP and LongLaMP benchmarks using the **LLaMA-3.1-8B** backbone. The best results are in **bold**, and the second best results are underlined.

Task	Metric	Non-pers.	RAG (k=4)		PEFT		Group PEFT	CARD
			BM25	Contriever	SFT	OPPU		
LaMP4: News Headline	R-1	0.155	0.170	0.185	0.165	0.160	0.175	**0.215**
	R-L	0.135	0.148	0.160	0.142	0.138	0.152	**0.192**
LaMP5: Scholarly Title	R-1	0.420	0.448	**0.468**	0.435	0.428	0.445	0.465
	R-L	0.335	0.370	0.360	0.350	0.342	0.358	**0.385**
LaMP7: Tweet Paraphrase	R-1	0.510	0.515	0.518	0.512	0.505	0.528	**0.535**
	R-L	0.445	0.440	0.442	0.440	0.435	0.452	**0.460**
LongLaMP1: Abstract Gen.	R-1	0.335	0.375	0.385	0.365	0.370	0.380	**0.415**
	R-L	0.180	0.205	0.212	0.198	0.200	0.208	**0.215**
LongLaMP2: Topic Writing	R-1	0.245	0.248	0.252	0.246	0.244	0.255	**0.262**
	R-L	0.115	0.118	0.122	0.116	0.114	**0.137**	0.132
LongLaMP3: Product Review	R-1	0.295	0.385	0.375	0.315	0.300	0.378	**0.405**
	R-L	0.135	0.155	0.150	0.140	0.136	0.152	**0.165**

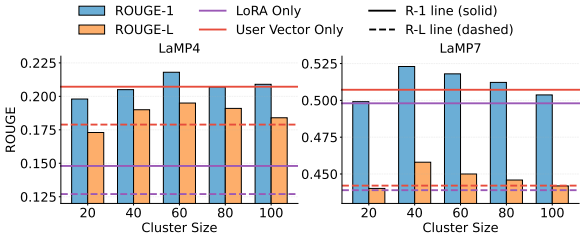

Figure 6: Performance via different K group clusters.

Table 12: Effect of the clustering algorithm on LaMP. *Rand.*: random grouping; *GMM*: Gaussian Mixture clustering; CARD uses K-means by default. *w/o LoRA* and *w/o Vec* are reproduced from Table 2 for reference.

Task	Metric	w/o LoRA	w/o Vec	Rand.	GMM	CARD
LaMP4	R-1	0.207	0.148	0.199	0.211	**0.218**
	R-L	0.179	0.127	0.169	0.183	**0.195**
LaMP5	R-1	0.449	0.428	0.445	**0.464**	0.459
	R-L	0.376	0.345	0.361	**0.396**	0.387
LaMP7	R-1	0.507	0.498	0.512	**0.536**	0.521
	R-L	0.442	0.439	0.439	0.441	**0.448**

F.1 Effect of Cluster Number K

We test the personalization performance with different cluster sizes in Figure 6.

Figure 6 shows that CARD remains relatively stable across a practical range of cluster sizes, while the best performance is typically achieved at moderate granularity. On LaMP4, performance peaks around K=40–60, reaching the highest ROUGE-1 and ROUGE-L scores, whereas on LaMP7 the best result is obtained at K=40, followed by a gradual decline as K increases further. This suggests that overly coarse grouping may fail to provide sufficiently informative shared priors, while overly fine grouping may reduce the amount of data available per cluster and weaken the robustness of group-level adaptation. Importantly, across all tested K, full CARD consistently outperforms both LoRA-only and User-Vector-only variants, indicating that its hierarchical design is beneficial beyond any single cluster configuration.

CARD shows good robustness for clustering size. While group-based PEFT approaches are generally sensitive to clustering design and granularity, CARD maintains stable performance across a wide range of cluster sizes, suggesting CARD is less dependent on precise clustering and that its group-level LoRA component functions as a robust shared prior for personalization.

F.2 Effect of the Clustering Algorithm

To verify that CARD's gains do not hinge on a particular clustering algorithm, we further compare K-means against two alternatives under identical training and inference settings: (i) *Random Grouping*, which assigns each user to one of K groups uniformly at random, and (ii) *GMM Clustering*, which replaces K-means with a Gaussian Mixture Model fitted on the same user embeddings. Results are reported in Table 12, alongside the two ablation variants from the main paper for reference.

Meaningful grouping matters. Replacing K-means with random grouping consistently hurts performance across all three tasks (e.g., LaMP-4 R-1 drops from 0.218 to 0.199), confirming that the group prior must reflect genuine user similarity. With random groups, the group LoRA is forced to fit users with conflicting stylistic patterns and

produces a noisy prior; the user vector must then spend part of its limited capacity correcting this misleading prior rather than refining a meaningful shared style.

CARD is not tightly coupled to K-means. Replacing K-means with GMM yields comparable or even slightly better numbers on some metrics (e.g., LaMP-5 R-1: 0.464 vs. 0.459; LaMP-7 R-1: 0.536 vs. 0.521), indicating that the gains come from the hierarchical group→user design itself rather than from K-means specifically. We adopt K-means as the default for its efficiency, scalability, and reproducibility on large user populations.

The group-level LoRA is not intended to fully recover the personalized model on its own, it provides a coarse warm-start prior used to construct preference pairs, which the user vector then refines at decoding time.

G Alignment between LLM and Human Judgments

To rigorously validate the reliability of our LLM-as-a-judge evaluation, we conduct a comprehensive statistical analysis of the agreement and correlation between the automated LLM judgments and human annotations on the LaMP benchmark. Following standard practices for evaluating subjective text generation on a 1-5 Likert scale, we report five distinct metrics: Pearson correlation (r), Spearman's rank correlation (ρ), Kendall's rank correlation (τ), standard Cohen's Kappa (κ), and Quadratic Weighted Kappa (QWK).

As shown in Table 13, the LLM scores exhibit strong positive correlations with human judgments. The high Spearman's ρ and Kendall's τ indicate that the LLM reliably preserves the relative ranking of personalization quality among different baselines.

Notably, the agreement metrics exhibit expected task-dependent variations. Objective and structure-heavy tasks (e.g., LaMP5: Scholarly Title) demonstrate the highest alignment, as both LLM and humans consistently capture factual fidelity. Conversely, highly stylistic tasks (e.g., LaMP7: Tweet Paraphrasing) show slightly lower, yet still robust, agreement due to the inherent divergence in human aesthetic preferences for informal texts.

Importantly, while the standard Cohen's κ yields relatively lower values (around 0.38)—an expected phenomenon since it heavily penalizes even minor 1-point deviations on a 5-point scale—the Quadratic Weighted Kappa (QWK) demonstrates strong agreement (average 0.618). QWK applies penalties proportional to the squared difference between scores, appropriately capturing the ordinal nature of our rating scale. These comprehensive metrics confirm that our automated evaluation protocol serves as a trustworthy proxy for human evaluation.

Table 13: Correlation and agreement metrics between LLM judgments and human evaluations. The inclusion of Quadratic Weighted Kappa (QWK) accounts for the ordinal nature of the 1-5 Likert scale, appropriately capturing fine-grained stylistic alignment.

Metric	LaMP4	LaMP5	LaMP7	Average
Pearson (r)	0.680	0.725	0.655	**0.687**
Spearman (ρ)	0.665	0.704	0.642	**0.670**
Kendall (τ)	0.524	0.568	0.495	**0.529**
Standard κ	0.382	0.415	0.354	**0.384**
Weighted QWK	0.615	0.652	0.588	**0.618**