Personalized Text Generation with Contrastive Activation Steering

Jinghao Zhang[1,2], Yuting Liu[3], Wenjie Wang[4*],
Qiang Liu[1,2], Shu Wu[1,2*], Liang Wang[1,2], Tat-Seng Chua[5]
[1]NLPR, Institute of Automation, Chinese Academy of Sciences,
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences,
[3]Northeastern University, China, [4]University of Science and Technology of China,
[5]National University of Singapore
jinghao.zhang@cripac.ia.ac.cn, {qiang.liu,shu.wu,wangliang}@nlpr.ia.ac.cn,
yutingliu@stumail.neu.edu.cn, wenjiewang96@gmail.com, dcscts@nus.edu.sg,

Abstract

Personalized text generation aims to infer users' writing style preferences from their historical texts and generate outputs that faithfully reflect these stylistic characteristics. Existing solutions primarily adopt two paradigms: retrieval-augmented generation (RAG) and parameter-efficient fine-tuning (PEFT). While these approaches have advanced the field, they suffer from two critical limitations: (1) the entanglement of content semantics and stylistic patterns in historical texts impedes accurate modeling of user-specific writing preferences; and (2) scalability challenges arising from both RAG's inference latency by retrieval operations and PEFT's parameter storage requirements for per user model. To overcome these limitations, we propose StyleVector, a training-free framework that disentangles and represents personalized writing style as a vector in LLM's activation space, enabling style-steered generation during inference without requiring costly retrieval or parameter storage. Comprehensive experiments demonstrate that our framework achieves a significant 8% relative improvement in personalized generation while reducing storage requirements by 1700 $\times$ over PEFT method.

1 Introduction

Large language models (LLMs) have demonstrated unprecedented capabilities in text generation and complex reasoning through pre-training on massive corpora. However, these models still function as "one-size-fits-all" systems, optimized for average-case scenarios, and fail to adapt to individual users' unique preferences. The increasing demand for personalized AI assistants highlights the need to customize LLMs to better align with the specific preference of each user (Cai et al., 2024; Au et al., 2025; **?**; Jang et al., 2023; Lin et al., 2024; Lv et al., 2025; Zhang et al., 2024c,a; Zhu et al., 2025; Liu et al., 2025; Xu et al., 2025).

Personalized text generation has emerged as a critical research frontier (Salemi et al., 2024b; Kumar et al., 2024; Alhafni et al., 2024; Chen and Moscholios, 2024). Consider a scenario where given an email subject x and a user u's historical subject-email pairs P_u, the system must infer the user's writing style from P_u to generate stylistically consistent emails. Current approaches predominantly fall into two categories: (1) Retrieval-augmented generation (RAG) methods (Zhang et al., 2023; Salemi and Zamani, 2024a,b), which enhance input prompts by retrieving personalized information from P_u, and (2) parameter-efficient fine-tuning (PEFT) methods (Salemi and Zamani, 2024a; Tan et al., 2024a; Zhuang et al., 2024), which train per-user adapter modules using P_u. Despite their merits, these methods suffer from critical limitations: (a) The inherent entanglement of *user-agnostic content semantics* and *user-specific stylistic patterns* in historical data impedes accurate style inference. (b) The substantial inference latency of RAG's retrieval mechanisms and storage requirements of PEFT's per-user parameters renders these solutions impractical for real-world deployment at scale.

Recent advances in activation engineering (Zou et al., 2023; Liu et al., 2023; Rimsky et al., 2024) reveal that LLMs encode features and concepts as linear directions in hidden activation space. These directional vectors can effectively steer model behavior through simple linear interventions during inference. Building on these insights, we reveal that *user-specific writing styles* can similarly be represented as directional vectors in activation space. This leads to an elegant solution for personalized generation: (1) By contrasting the hidden activations between *user-authentic responses* (containing both content and style) and *model-generated generic responses* (content-preserving but style-agnostic), we can derive "style vector" that contains personal stylistic signatures. (2) The derived

*Corresponding authors

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 7128–7141
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

style vector could be used to steer model generation towards user-specific writing styles through simple linear interventions during inference, without parameter updates or extensive retrieval.

To this end, we present StyleVector, an efficient, training-free framework that only requires storing one vector for each user to achieve high-quality personalized text generation. As shown in Figure 1, our methodology comprises three key steps: (1) generating style-agnostic responses for historical inputs using a base LLM, (2) deriving style vectors by contrasting hidden activations between authentic user responses and generated neutral responses, and (3) steering generation during inference through linear activation interventions with the obtained style vectors.

Comprehensive evaluations on LaMP (Salemi et al., 2024b) and LongLaMP (Kumar et al., 2024) benchmarks for short- and long-form personalization respectively demonstrate our method's effectiveness. Experimental results show that StyleVector achieves 8% relative improvement in personalization quality while reducing storage requirements by 1700× over PEFT-based methods.

Our contributions are summarized as follows:

- We reveal that user-specific writing styles can be represented as linear directions in activation space through contrastive analysis between authentic user responses and style-agnostic model outputs.
- We propose a training-free personalized generation framework through simple linear activation interventions, requiring only $2|P_u|$ forward passes (zero back-propagation) per user and compresses personalized information into a single vector.
- Experiments on both short- and long-form personalization benchmarks show the effectiveness of our method, while significantly reducing storage and inference latency compared to retrieval-based and adapter-based approaches.

2 Preliminaries

2.1 Problem Formulation

Personalized text generation aims to infer the user's writing style preferences based on the text created from their history and generate outputs that align with those preferences. Formally, for each user u: given an input prompt x specifying task requirements (e.g., an email subject), the language model M generates output $\hat{y} = M(x, P_u)$ conditioned on both x and the user's historical data $P_u = \{(x_i, y_i)\}_{i=1}^{|P_u|}$, where each pair (x_i, y_i) represents previous interactions (e.g., subject-email pairs). The ground truth output y represents the user-customized response that reflects u's unique writing style (e.g., personalized email drafts).

2.2 Base Solutions

Retrieval-Augmented Generation (RAG) RAG-based approaches achieve personalization through context-aware retrieval. Given input x, the system retrieves k most relevant historical responses from P_u using retriever R, then generates personalized responses by combining retrieved documents $R(x, P_u, k)$ with the input prompt:

$$\hat{y} = M(x, R(x, P_u, k)) \tag{1}$$

Parameter-Efficient Fine-Tuning (PEFT) PEFT methods customize LLMs by training lightweight adapters (e.g., LoRA (Hu et al., 2021)) on user-specific data while keeping base model parameters frozen (Tan et al., 2024b). For each user u, a distinct adapter θ_u is trained via:

$$\theta_u^* = \arg\min_{\theta} \sum_{(x_i, y_i) \in P_u} \mathcal{L}(M(x_i; \theta), y_i) \tag{2}$$

where $\mathcal{L}(\cdot)$ denotes the sequence-to-sequence cross-entropy loss. During inference:

$$\hat{y} = M(x; \theta_u) \tag{3}$$

2.3 Limitations of Base Solutions

Existing approaches face the following two fundamental constraints.

Entangled Style-Content Representation Both RAG and PEFT methods process historical entries p_i as monolithic units. However, each historical entry contains both the *user-agnostic semantics* corresponding to the input x_i and the *user-specific writing style* (Fisher et al., 2024). This entanglement impedes accurate style modeling, particularly for RAG methods that retrieve documents based on semantic matching, and the semantic-dominated retrieved contexts lead to style dilution (see Section 4.5 for examples).

Scalability Bottlenecks As summarized in Table 1, existing methods suffer from three critical scalability constraints: training time, inference latency and storage requirement. Due to space constraints, we have placed the complexity analysis of

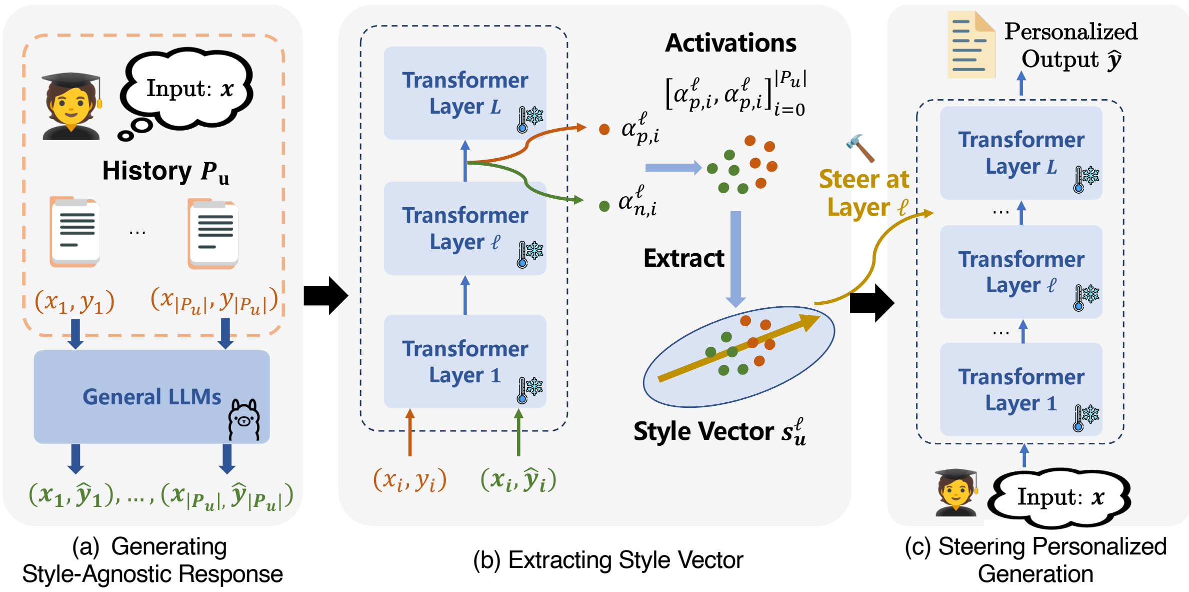

Figure 1: The overall framework of StyleVector.

Metric	RAG	PEFT	StyleVector
Training Time/User	$O(\|P_u\|)$*	$O(\|P_u\|)$	$O(\|P_u\|)$*
Latency/Query	$O(\|P_u\|)$	O(Load+Merge)	$O(1)$
Storage/User	$O(\|P_u\|D)$	$O(rDL)$	$O(D)$

* Training-free. Denotes pre-processing cost.

Table 1: System Efficiency Comparison.

the baseline in the Appendix D. These compounded costs render existing methods challenging for real-world deployment at scale (Salemi and Zamani, 2024a). We also provide empirical cost comparisons in Section 4.2.

3 Method

Our StyleVector framework aims to identify a user-specific style vector through contrastive activation analysis, then steer LLM generation via targeted activation intervention. As shown in Figure 1, the process comprises three stages: (1) Style-agnostic response generation, (2) Style vector extraction through contrastive activation analysis, and (3) Activation steering during inference.

3.1 Generating Style-Agnostic Response

Given a user u with historical interactions $P_u = \{(x_i, y_i)\}_{i=1}^{|P_u|}$, where x_i denotes an input and y_i the user-authored response, we first generate style-agnostic responses $\{\hat{y}_i\}_{i=1}^{|P_u|}$ by instructing any general LLM M_g with the input x_i:

$$\hat{y}_i = M_g(x_i). \tag{4}$$

Please note that the general LLM M_g is designed to generate responses that are independent of the user's style and only related to the input semantics. It does not necessarily need to be the same as a personalized large model M; it can be any model, whether open-source or closed-source. We conduct experiments in Appendix C.2 to show the robustness on M_g of our method.

In this way, y_i denotes the user-authentic content, containing both content semantics and stylistic patterns. Model-generated generic $\hat{y}_i$ only preserves content semantics related to x_i but stripped of personal style. By contrasting y_i and $\hat{y}_i$, we could disentangle user-specific style from user-agnostic semantics.

3.2 Extracting Style Vector

We extract style vectors through contrastive analysis of hidden activations. Let $h_\ell(r) \in \mathbb{R}^d$ denote the hidden states of the last token at layer ℓ when processing text r. The positive and negative activations of history piece i can be represented as:

$$a^\ell_{p,i} = h_\ell(x_i \oplus y_i), \quad a^\ell_{n,i} = h_\ell(x_i \oplus \hat{y}_i), \tag{5}$$

where $\oplus$ denotes concatenation the strings of input and output. Then we can obtain the user style vector by considering all history pieces:

$$s^\ell_u = f([a^\ell_{p,i}, a^\ell_{n,i}]^{|P_u|}_{i=0}), \tag{6}$$

where $f(\cdot)$ is an extracting function that takes all the positive and negative activations and returns a single style vector. The essence of the function f is to find a direction in the activation space that points from style-agnostic samples to user-authentic samples. There could be many possible functions, and here we discuss three strategies:

1) Mean Difference. The most straightforward approach computes the mean difference between positive and negative activations:

$$s_u^\ell = \frac{1}{|P_u|} \sum_{i=1}^{|P_u|} (a_{p,i}^\ell - a_{n,i}^\ell). \tag{7}$$

s_u^ℓ represents the average direction in the activation space that distinguishes user-specific style patterns from style-agnostic ones.

2) Logistic Regression. We can also employ logistic regression to find a direction that best separates positive and negative examples. Let $X = [a_{p,1}^\ell; ...; a_{p,|P_u|}^\ell; a_{n,1}^\ell; ...; a_{n,|P_u|}^\ell]$ be the matrix of all activations, and $y = [1, ..., 1, -1, ..., -1]$ be the corresponding labels. The style vector is obtained by:

$$w = \arg\min_w \sum_i \log(1 + e^{-y_i X_i w}), \tag{8}$$

where w denotes the normal vector to the decision boundary. When moving in the direction of w, the model's predicted probability of being a positive sample will monotonically increase. We use the normalized w as the style vector:

$$s_u^\ell = \frac{w}{\|w\|_2}, \tag{9}$$

3) Principal Component Analysis. The Principal Component Analysis (PCA) approach finds the steering vector s_u^ℓ by identifying the direction of maximum variance in the differences between positive and negative activations. Let $\Delta_i = a_{p,i}^\ell - a_{n,i}^\ell$ be the difference between the i-th pair of positive and negative activations. PCA computes the first principal component of the set $\{\Delta_i\} \cup \{-\Delta_i\}$, which can be formulated as:

$$s_u^\ell = \arg\max_{v:\|v\|=1} \sum_{i=1}^{|P_u|} (\Delta_i^T v)^2. \tag{10}$$

This formulation ensures that: 1. The resulting vector s_u^ℓ has unit norm 2. It maximizes the projected variance of the activation differences 3. The inclusion of $-\Delta_i$ enforces symmetry around the origin, making the solution invariant to the choice of which sample is positive or negative

The solution to this optimization problem is given by the first eigenvector of the matrix $\sum_{i=1}^{|P_u|}(\Delta_i \Delta_i^T + (-\Delta_i)(-\Delta_i^T))$, which can be efficiently computed using Singular Value Decomposition (SVD).

3.3 Steering Personalized Generation

After obtaining the style vector, we can steer the model's generation by intervene the hidden states at inference time. In this work, we only consider intervene one layer ℓ, which could be selected via validation set. Let $h_\ell(x)$ denote the hidden states at layer ℓ when processing input x. We use the most straightforward approach directly adds the scaled style vector to the hidden states of the token position t:

$$h'_\ell(x)_t = h_\ell(x)_t + \alpha s_u^\ell \tag{11}$$

where α is a scaling factor controlling the strength of steering. Following (Rimsky et al., 2024), we intervene every token position of the generated text after the end of the initial prompt $t \geq |x|$. We also try different positions experimentally in Section 4.2.

Efficiency Analysis For pre-processing, our method requires only $2|P_u|$ forward passes of LLMs to obtain activations and the style vector extracting is negligible when compared with the cost of LLMs. For storage, the final style vector s_u^ℓ only requires D-dimensional vector storage. For additional inference latency, activation steering only introduces D element-wise addition overhead. The complexity analysis is summarized in Table 1.

4 Experiments

4.1 Experimental Setup

Benchmarks and Evaluation We adopt LaMP benchmark (Salemi et al., 2024b) and LongLaMP benchmark (Kumar et al., 2024), which are designed for evaluating short-form and long-form personalized text generation, respectively. We exclude email generation tasks for both datasets since it involves private data that we cannot access. We choose the user split for both benchmarks and the dataset statistics are presented in Table 4. Following previous works (Tan et al., 2024a; Salemi and Zamani, 2024a), we use ROUGE-L and METEOR as evaluation metrics.

Benchmark	Metric	Non-personalized	RAG-based		PEFT-based		Ours	Improv.
		LLaMA2	BM25	Contriever	SFT	DPO		
LongLaMP: Abstract Generation	ROUGE-L	<u>0.2056</u>	0.2020	0.2035	0.2038	0.2020	**0.2060**	0.2%
	METEOR	<u>0.2950</u>	0.2911	0.2922	0.2929	0.2933	**0.2973**	0.8%
LongLaMP: Topic Writing	ROUGE-L	0.1299	0.1235	0.1256	<u>0.1303</u>	0.1277	**0.1361**	4.7%
	METEOR	0.1874	0.1782	0.1853	<u>0.1914</u>	0.1901	**0.1949**	4.0%
LongLaMP: Review Generation	ROUGE-L	0.1380	0.1388	<u>0.1391</u>	0.1364	0.1320	**0.1448**	5.0%
	METEOR	0.1614	0.1655	<u>0.1663</u>	0.1574	0.1446	**0.1804**	11.8%
LaMP: News Headline Generation	ROUGE-L	0.0398	0.0403	0.0403	<u>0.0407</u>	0.0401	**0.0411**	3.2%
	METEOR	0.0790	0.0792	0.0807	<u>0.0800</u>	0.7910	**0.0809**	2.5%
LaMP: Scholarly Title Generation	ROUGE-L	0.1086	0.0909	0.0919	<u>0.1100</u>	0.1047	**0.1366**	25.8%
	METEOR	0.2337	0.2066	0.2086	<u>0.2348</u>	0.1930	**0.2575**	10.2%
LaMP: Tweet Paraphrasing	ROUGE-L	0.2506	0.2554	<u>0.2571</u>	0.2341	0.2204	**0.2827**	12.8%
	METEOR	0.2588	0.2603	<u>0.2634</u>	0.2503	0.2389	**0.3042**	17.5%

Table 2: The performance results on LongLaMP and LaMP personalized text generation benchmarks. The best score is in **bold** and the second best is <u>underlined</u>.

Baselines We compare our proposed StyleVector with RAG-based personalization methods and PEFT-based personalization methods.

For RAG-based personalization, we employ two widely-used retrievers BM25 (Robertson et al., 2009) and Contriever (Lei et al., 2023).

For PEFT-based personalization, we fine-tune user-specific LoRA adapter (Hu et al., 2021) for each user using their profile $P_u = (x_i, y_i)_{i=0}^{|P_u|}$, using SFT loss in Equation 2. Additionally, since we obtained style-agnostic responses, we also employ DPO loss (Rafailov et al., 2024) to guide the model to generate user-authentic responses rather than style-agnostic responses.

Implementation Details We implement our proposed StyleVector and all baselines with Llama-2-7B-chat (Touvron et al., 2023). For the RAG approach, we set the number of retrieved documents $k = 2$; for the PEFT approach, we set the rank of LoRA to 8. For StyleVector, unless otherwise specified, we will use gpt-3.5-turbo to generate style-neutral responses and employ the simplest mean difference extracting function. We conduct experiments on the validation set to select the appropriate number of intervention layer ℓ and intervention strength α for each task. For more details, please refer to Appendix B.

4.2 Main Results

By comparing our method with the baseline in terms of generation performance and efficiency, we demonstrate that our approach can achieve strong generation performance while maintaining high efficiency.

Task	Averaged Cost ↓	SFT	RAG	Ours
Abstract Generation	TT/User (s)	131.98	**0.64**	27.23
	IL/Query (s)	22.59	18.90	**15.59**
	IL/5-Query (s)	94.75	96.97	**79.23**
	SS/User (MB)	17.00	0.35	**0.01**
Review Generation	TT/User (s)	62.45	**0.44**	11.65
	IL/Query (s)	18.88	**8.23**	11.75
	IL/5-Query (s)	77.33	**52.52**	59.69
	SS/User (MB)	17.00	0.10	**0.01**
News Headline Generation	TT/User (s)	123.28	**1.22**	22.16
	IL/Query (s)	25.52	12.47	**10.32**
	IL/5-Query (s)	105.00	78.08	**57.80**
	SS/User (MB)	17.00	0.83	**0.01**
Scholarly Title Generation	TT/User (s)	112.31	**0.51**	22.53
	IL/Query (s)	25.43	**9.52**	10.49
	IL/5-Query (s)	104.33	**50.68**	54.30
	SS/User (MB)	17.00	0.26	**0.01**

Table 3: Comparison of Training Time (TT, for train-free RAG and StyleVector, represents pre-processing time), Inference Latency (IL) and Storage Space (SS) requirements across different methods. The lowest cost in in **bold**.

Generation Performance Comparison Table 2 shows the generation performance comparison and We can observe that:

- StyleVector demonstrates superior performance across both short-term and long-term personalized text generation tasks. Notably, StyleVector achieves averaged 11% and 8% relative improvements on ROUGE-L and METEOR compared with RAG-based methods and PEFT-based methods, respectively.
- Both RAG-based and PEFT-based methods

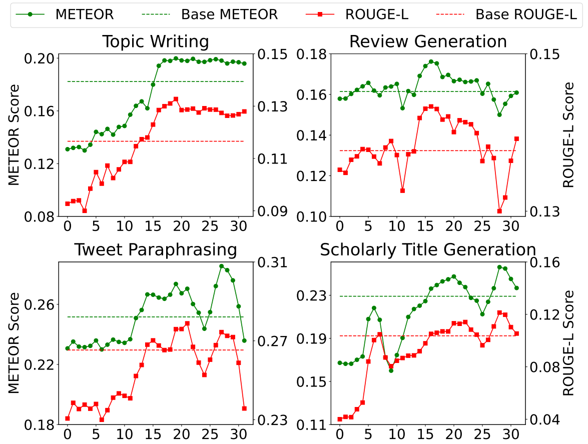

Figure 2: Performance comparison across different intervention layers l.

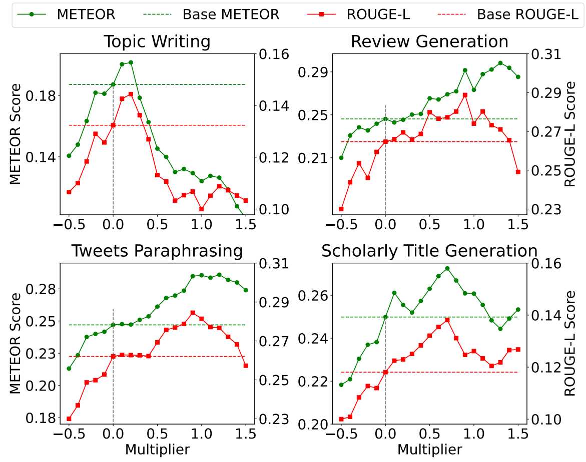

Figure 3: Performance comparison across different intervention strengths α.

show unstable performance and cannot consistently improve base model across all tasks. RAG-based methods are more effective in tasks with less user history (review generation and tweet paraphrasing are the two tasks with the least user history), while PEFT performs better in scenarios with more historical data as it provides more training texts.

Efficiency Comparison Table 3 shows the scalability comparison, where we implement Contriver as the retriever of RAG.

- In terms of training time, our method is training-free and requires only 1/5 of the preprocessing time compared to SFT. However, since RAG uses smaller retrievers (e.g., the Contriever model we use is no larger than 0.1B), RAG's preprocessing time is the shortest.
- In terms of inference latency, RAG is faster on tasks with less user history, but it becomes significantly slower on tasks with more user history. SFT takes too long to load and merge LoRA, making it unsuitable for scenarios that require frequent updates. Our method is independent of user history and does not require prolonged loading, making it a more versatile approach.
- In terms of storage space, our method only requires storing a single vector per user, making it unquestionably the most space-efficient, which occupies about $1/1700$ of the space required by SFT.

4.3 Steering Analysis

Analysis of layers and multipliers The intervention layer ℓ and the intervention strength α are two important hyperparameters of our method. In this section, we analyze the impact of different values of ℓ and α on generation performance. The results are shown in Figure 2 and Figure 3, from which we can observe that:

- **The activations controlling the model's writing style are typically reflected in the middle to later layers**. As shown in Figure 2, although there may be subtle differences across tasks, in general, the most effective intervention occurs when modifying the middle to later layers of the model (around layer 15 and beyond). Linear probing results in Section 4.4 also lead to the similar conclusion.
- **Positive intervention can guide the model to generate in the user's style, while negative intervention can push it away from that style.** As shown in Figure 3, when $\alpha < 0$, the negative intervention causes the model's generated content to drift away from the user's style, resulting in a score lower than that of the non-personalized model. However, if α is too large, it can cause abnormal activation values, thereby disrupting the generation process.

4.4 Style Vector Analysis

Probing Study To investigate how writing style features are encoded in the model's hidden states, we conduct a linear probing analysis across different layers of the base LLM. For each user $u \in \mathcal{U}$, we construct a binary classification task where the positive samples are the user's authentic historical

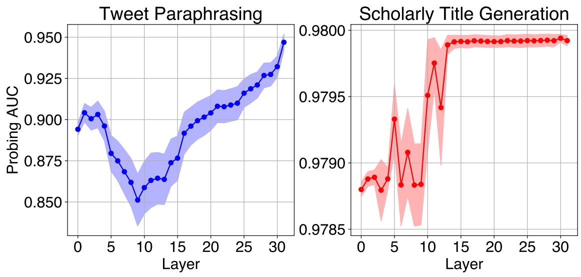

Figure 4: Probing results on LaMP benchmark.

texts $y_i \in \mathcal{P}_u$, and the negative samples are our framework's style-agnostic responses $\hat{y}_i$ generated for the same input contexts. We extract hidden states at layer l for all samples and train a logistic regression classifier to distinguish between authentic and generated texts. Figure 4 shows the averaged probing results across all users, which reveals two key findings:

- **High Layer-wise Separability.** All layers achieve strong classification performance (AUC > 0.85), suggesting that user-specific stylistic patterns are robustly encoded throughout the network. This confirms our hypothesis that style information persists in the model's internal representations, even when not explicitly supervised.
- **The activations controlling the model's writing style are typically reflected in the middle to later layers**. The AUC increases with the depth of the layers, which aligns with our empirical findings in Section 4.3, where style steering interventions in these layers yielded optimal generation quality. The progressive feature refinement suggests that stylistic attributes are gradually distilled through the forward pass, reaching maximal linear separability in higher layers.

4.5 Case Study

To demonstrate the effectiveness of our method, we analyze a representative case from user_310 in LaMP: News Headline Generation benchmark in Figure 5, demonstrating three key insights about our style vector approach:

- **Style vector encodes user preferences**. The highlighted tokens are the top 5 tokens that most closely match the style vector among all historical tokens. We can observe that the top-5 tokens (":", "ips", "for", "What", "Need") in historical headlines reveal consistent stylistic patterns of using subtitles and combinations such as "tips for" or "what need".
- **Style vector can steer personalized generation**. Our method generates "Keeping Your Teen Safe Online: Tips and Strategies for Parents", which naturally incorporates 3 key style tokens (":", "ips", "for") while maintaining content fidelity. However, the generation by baselines can not match user style preferences.
- **It's necessary to decouple style from semantic**. We list style ranking and semantic ranking of each historical headline, where style ranking represents the ranking results based on the similarity between the historical headline embeddings with the style vector, and semantic ranking represents the ranking results obtained by Contriver (Lei et al., 2023). We can observe that headlines with higher style rankings exhibit stronger alignment with user-preferred stylistic patterns. However, there exists significant divergence between style ranking and semantic ranking. For RAG-based methods, the semantic-dominated retrieved headlines fail to provide useful patterns about stylistic preferences.
- **Style Transfer**.We tried rewriting the user's historical texts in a certain style (by instructing GPT) to recalculate the style vector, in order to observe whether we can steer the model to generate in the desired style. We targeted two styles: "exclamatory tone, ending with an exclamation mark" and "removal of colons and subheadings." The results show that our method can achieve style transfer while maintaining semantic fidelity, further demonstrating that the style vector can indeed encode the user's writing style.

5 Related Work

5.1 Personalized Text Generation

The rapid evolution of LLMs has fundamentally transformed content generation paradigms, shifting from generic outputs to sophisticated personalized text generation. Current methodologies in personalized generation predominantly fall into two technical categories: Retrieval-Augmented Generation (RAG) approaches leverage users' historical content (P_u) through dynamic retrieval mechanisms. While foundational work (Zhang et al., 2023; Salemi and Zamani, 2024a; Salemi et al., 2024a; Richardson et al., 2023; Deng et al., 2025) established basic retrieval frameworks, re-

Input	Generate a headline for the following article: "Here are a few tips to keep your teen safe when using the Internet and other web-based technologies. If you think it's an awkward conversation; you can hand them this blog to read."		
Ground Truth Output	Social Media Gone Awry: Tips for Teens to Stay Safe		
User History	**User-created Headlines**	**Style Ranking**	**Semantic Ranking**
	The Anxiety of Hiring a Nanny: Tips for the Screening Process	1	16
	Leading Causes of Injury Death Among Children: What Parents Need to Know	2	14
	Summer Camp Safety: Essential Questions Parents Should Ask	3	12
	Protecting your child after a disclosure of sexual abuse: What parents need to know.	4	13
	Parent Alert: Tips for Keeping Your Children Safe this Summer	5	2
	Internet Predators: Parents, Monitor Your Children!	8	1
	Talking to Your Child About the School Shooting in Newtown, CT	12	4
	Stop Bullying: Teach Your Child Empathy and Limit Their Intake of Violence	15	3
	If You See Something, Please Do Something to Prevent Child Abuse	17	11
LLaMA	Set boundaries and rules for internet use		
+SFT	Use parental controls to limit access to inappropriate content		
+RAG	Talking to Your Teen About Online Safety: It's Time to Get Real		
+Ours	Keeping Your Teen Safe Online: Tips and Strategies for Parents		
Style Transfer + '!'	Tips for Keeping Your Teen Safe Online: A Must-Read for Parents!		
Style Transfer - ':'	5 Tips to Keep Your Teen Safe Online		

Figure 5: Case study of user_310 in News Headline Generation task. The highlighted tokens are the top 5 tokens that most closely match the style vector among all historical tokens. The underline words are the words that match the ground truth. 'Style Ranking' represents the ranking results based on the similarity between the historical headline embeddings with the style vector. 'Semantic Ranking' represents the ranking results obtained by Contriver (Lei et al., 2023).

cent innovations have enhanced these paradigms. Richardson et al. (2023); Zhang (2024); Tan et al. (2025) developed profile-augmented prompting strategies, while Salemi and Zamani (2024b) introduced feedback-driven retrieval model optimization, demonstrating improved personalization accuracy. Parameter-Efficient Fine-Tuning (PEFT) methods adopt an alternative paradigm by adapting per-user parameters through lightweight adapter modules. Comparative studies (Salemi and Zamani, 2024a) reveal that PEFT approaches, particularly those employing user-specific adapter tuning (Tan et al., 2024a; Zhuang et al., 2024; Liu et al., 2024; Ding et al., 2025), achieve competitive personalization while maintaining computational efficiency.

5.2 Activation Engineering

Emerging research in activation engineering has uncovered that LLMs encode semantic concepts as linear subspaces within hidden activation representations (Zou et al., 2023; Liu et al., 2023; Rimsky et al., 2024). This geometric interpretation enables targeted behavioral steering through linear interventions during inference. Turner et al. (2023) pioneered activation addition using contrastive-derived steering vectors for sentiment and topic control, while Rimsky et al. (2024) enhanced steering precision through mass-mean activation differentials. Zhang et al. (2024b) identified truth-correlated heads via linear probing, achieving enhanced veracity through targeted modulation. Complementing this, Chen et al. (2024) developed multi-directional orthogonal steering to amplify truthfulness in model responses.

6 Conclusion

In this work, we demonstrate that user's writing style can be represented as a vector in LLM's activation-space. Based on this insight, we introduces a simple yet effective frame, StyleVector, that achieves personalized text generation through inference time intervention, without parameter updates or retrievals. Experiments on both short- and long-form personalization benchmarks show our method can achieve strong generation performance while maintaining high efficiency.

Limitations

While our framework demonstrates significant advantages in efficiency and effectiveness, several limitations warrant discussion to guide future research:

Our training-free style vector derivation, though efficient, may not achieve optimal disentanglement of style from content. The current contrastive approach relies on the model's inherent ability to separate these features through simple activation arithmetic. Future work could explore hybrid approaches that combine our parametric-free method with lightweight optimization techniques to refine the style vectors while maintaining storage efficiency.

The single-vector user representation, while storage-efficient, potentially conflates multiple stylistic dimensions (e.g., lexical preferences, syntactic structures, and discourse patterns). A more granular approach could represent users through sparse combinations (Cunningham et al., 2023; Lieberum et al., 2024) of concept-specific vectors, enabling precise control over individual style components.

Our evaluation focuses on established benchmarks (LaMP and LongLaMP) that assume domain homogeneity within each user's historical data. However, real-world personalization scenarios often involve *cross-domain style consistency* – users may employ distinct stylistic registers across different tasks (e.g., formal emails vs. casual social media posts). Current benchmarks lack the capability to assess whether learned style vectors can: (1) preserve task-appropriate stylistic variations within users, or (2) prevent negative interference between conflicting domain-specific patterns. Future work should develop cross-domain personalization benchmarks that incorporate mixed-task histories.

Ethics Statement

The experimental datasets are publicly available from some previous works, downloaded via official APIs. The information regarding users in all datasets has been anonymized, ensuring there are no privacy concerns related to the users. We do not disclose any non-open-source data, and we ensure that our actions comply with ethical standards. We use publicly available pre-trained models, i.e., LLaMA-2, Contriver, and APIs, i.e., GPT-3.5-turbo, DeepSeek-Chat. All the checkpoints and datasets are carefully processed by their authors to ensure that there are no ethical problems.

Acknowledgements

This work is supported by National Natural Science Foundation of China (62372454, 62206291, 62141608, 62236010).

References

Bashar Alhafni, Vivek Kulkarni, Dhruv Kumar, and Vipul Raheja. 2024. Personalized text generation with fine-grained linguistic control. *ArXiv*.

Steven Au, Cameron J Dimacali, Ojasmitha Pedirappagari, Namyong Park, Franck Dernoncourt, Yu Wang, Nikos Kanakaris, Hanieh Deilamsalehy, Ryan A Rossi, and Nesreen K Ahmed. 2025. Personalized graph-based retrieval for large language models. *arXiv preprint arXiv:2501.02157*.

Hongru Cai, Yongqi Li, Wenjie Wang, Fengbin Zhu, Xiaoyu Shen, Wenjie Li, and Tat-Seng Chua. 2024. Large language models empowered personalized web agents. *arXiv preprint arXiv:2410.17236*.

Zhongzhi Chen, Xingwu Sun, Xianfeng Jiao, Fengzong Lian, Zhanhui Kang, Di Wang, and Chengzhong Xu. 2024. Truth forest: Toward multi-scale truthfulness in large language models through intervention without tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20967–20974.

Ziyang Chen and Stylios Moscholios. 2024. Using prompts to guide large language models in imitating a real person's language style. *ArXiv*.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. 2023. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*.

Boyi Deng, Wenjie Wang, Fengbin Zhu, Qifan Wang, and Fuli Feng. 2025. Cram: Credibility-aware attention modification in llms for combating misinformation in rag. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23760–23768.

Yucheng Ding, Yangwenjian Tan, Xiangyu Liu, Chaoyue Niu, Fandong Meng, Jie Zhou, Ning Liu, Fan Wu, and Guihai Chen. 2025. Personalized language model learning on text data without user identifiers. *ArXiv*.

Jillian R. Fisher, Skyler Hallinan, Ximing Lu, Mitchell Gordon, Zaid Harchaoui, and Yejin Choi. 2024. Styleremix: Interpretable authorship obfuscation via distillation and perturbation of style elements. *ArXiv*, abs/2408.15666.

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. *arXiv preprint arXiv:2106.09685*.

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 preprint arXiv:2310.11564*.

Ishita Kumar, Snigdha Viswanathan, Sushrita Yerra, Alireza Salemi, Ryan A Rossi, Franck Dernoncourt, Hanieh Deilamsalehy, Xiang Chen, Ruiyi Zhang, Shubham Agarwal, et al. 2024. Longlamp: A benchmark for personalized long-form text generation. *arXiv preprint arXiv:2407.11016*.

Yibin Lei, Liang Ding, Yu Cao, Changtong Zan, Andrew Yates, and Dacheng Tao. 2023. Unsupervised dense retrieval with relevance-aware contrastive pre-training. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10932–10940.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. 2024. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *arXiv preprint arXiv:2408.05147*.

Zihao Lin, Zichao Wang, Yuanting Pan, Varun Manjunatha, Ryan Rossi, Angela Lau, Lifu Huang, and Tong Sun. 2024. Persona-sq: A personalized suggested question generation framework for real-world documents. *arXiv preprint arXiv:2412.12445*.

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*.

Sheng Liu, Haotian Ye, Lei Xing, and James Zou. 2023. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*.

Yuting Liu, Jinghao Zhang, Yizhou Dang, Yuliang Liang, Qiang Liu, Guibing Guo, Jianzhe Zhao, and Xingwei Wang. 2025. Cora: Collaborative information perception by large language model's weights for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 12246–12254.

Zheqi Lv, Tianyu Zhan, Wenjie Wang, Xinyu Lin, Shengyu Zhang, Wenqiao Zhang, Jiwei Li, Kun Kuang, and Fei Wu. 2025. Collaboration of large language models and small recommendation models for device-cloud recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, page 962–973. Association for Computing Machinery.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 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*.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. 2024. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand. Association for Computational Linguistics.

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

Alireza Salemi, Surya Kallumadi, and Hamed Zamani. 2024a. Optimization methods for personalizing large language models through retrieval augmentation. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2024b. 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, Bangkok, Thailand. Association for Computational Linguistics.

Alireza Salemi and Hamed Zamani. 2024a. Comparing retrieval-augmentation and parameter-efficient fine-tuning for privacy-preserving personalization of large language models. *arXiv preprint arXiv:2409.09510*.

Alireza Salemi and Hamed Zamani. 2024b. Learning to rank for multiple retrieval-augmented models through iterative utility maximization. *arXiv preprint arXiv:2410.09942*.

Zhaoxuan Tan, Zheyuan Liu, and Meng Jiang. 2024a. Personalized pieces: Efficient personalized large language models through collaborative efforts. *arXiv preprint arXiv:2406.10471*.

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*.

Zhaoxuan Tan, Zinan Zeng, Qingkai Zeng, Zhenyu Wu, Zheyuan Liu, Fengran Mo, and Meng Jiang. 2025. Can large language models understand preferences in personalized recommendation? *ArXiv*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. 2023. Activation addition: Steering language models without optimization. *arXiv e-prints*, pages arXiv–2308.

T Wolf. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

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. *arXiv preprint arXiv:2503.02614*.

Jiarui Zhang. 2024. Guided profile generation improves personalization with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*.

Jinghao Zhang, Yuting Liu, Qiang Liu, Shu Wu, Guibing Guo, and Liang Wang. 2024a. Stealthy attack on large language model based recommendation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5839–5857.

Kai Zhang, Fubang Zhao, Yangyang Kang, and Xiaozhong Liu. 2023. Memory-augmented llm personalization with short-and long-term memory coordination. *arXiv preprint arXiv:2309.11696*.

Shaolei Zhang, Tian Yu, and Yang Feng. 2024b. Truthx: Alleviating hallucinations by editing large language models in truthful space. *arXiv preprint arXiv:2402.17811*.

Zhehao Zhang, Ryan A Rossi, Branislav Kveton, Yijia Shao, Diyi Yang, Hamed Zamani, Franck Dernoncourt, Joe Barrow, Tong Yu, Sungchul Kim, et al. 2024c. Personalization of large language models: A survey. *arXiv preprint arXiv:2411.00027*.

Jianfeng Zhu, Ruoming Jin, and Karin G. Coifman. 2025. Investigating large language models in inferring personality traits from user conversations. *ArXiv*.

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 preprint arXiv:2406.02888*.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. 2023. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.

Algorithm 1 Personalized Generation with Style Steering

Require: User interaction history $P_u = \{(x_i, y_i)\}_{i=1}^{|P_u|}$, general LLM M_g, intervention layer ℓ, scaling factor α, input query x

Ensure: Personalized generation model M

```
Stage 1: Generate Style-Agnostic Responses
for each data pair (x_i, y_i) ∈ P_u do
    Generate style-agnostic response ŷ_i ← M_g(x_i)
end for
Stage 2: Extract Style Vector
for each data pair (x_i, y_i) ∈ P_u do
    Compute positive activation a^ℓ_{p,i} ← h_ℓ(x_i ⊕ y_i)
    Compute negative activation a^ℓ_{n,i} ← h_ℓ(x_i ⊕ ŷ_i)
end for
Extract style vector s^ℓ_u ← f({a^ℓ_{p,i}, a^ℓ_{n,i}}_{i=1}^{|P_u|})
Stage 3: Activation Steering
for each generation position t ≥ |x| do
    Retrieve original activation h_ℓ(x)_t
    Inject style vector h'_ℓ(x)_t ← h_ℓ(x)_t + α s^ℓ_u
end for
```

A Algorithm

The complete procedure is formalized in Algorithm 1.

B Experiment Details

DPO Baseline The DPO algorithm (Rafailov et al., 2024) reframes preference learning by directly optimizing a policy to align with human preferences without explicit reward modeling. Since we obtained style-agnostic responses, we also employ DPO loss (Rafailov et al., 2024) to guide the model to generate user-authentic responses y_i rather than style-agnostic responses $\hat{y}_i$.

$$\theta_u^* = \arg\min_\theta \sum_{(x_i, y_i, \hat{y}_i) \in P_u} -\log \sigma\Bigg(\beta \log \frac{M_\theta(y_i \mid x_i)}{M_{\text{ref}}(y_i \mid x_i)} - \beta \log \frac{M_\theta(\hat{y}_i \mid x_i)}{M_{\text{ref}}(\hat{y}_i \mid x_i)}\Bigg) \tag{12}$$

where M_θ is the policy with adapter θ_u, M_{ref} is the reference policy (base model M with frozen parameters), σ denotes the sigmoid function, and β controls deviation from the reference policy. This approach enables parameter-efficient preference alignment through lightweight adapters while maintaining the base model's capabilities.

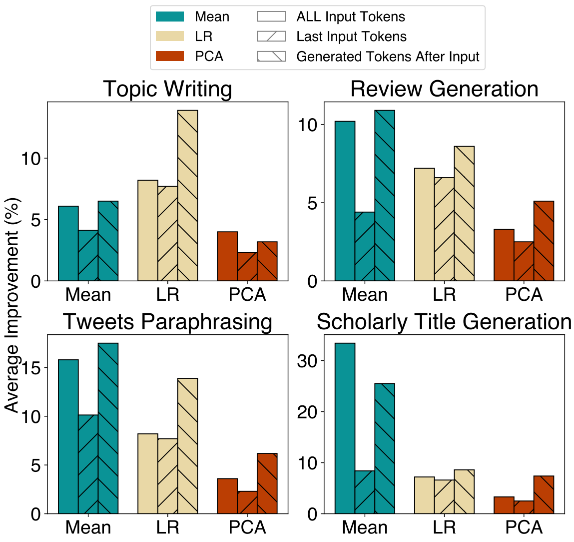

Figure 6: Performance comparison across different extracting functions and intervention positions t.

Implementation Details All experiments were performed on a cluster of 8 NVIDIA RTX 3090 GPUs, with implementations built upon the PyTorch framework (Paszke et al., 2019), HuggingFace Transformers (Wolf, 2019) library. To save computational resources, we apply 8-bit quantization and greedy decoding for all methods.

C Additional Experimental Analysis

C.1 Analysis of Extracting Function and Intervention Position

We compare three different extracting functions in Equation 5 and different intervention token positions t in Equation 11. We use three different intervention positions: intervening on all input tokens, intervening only on the last input token, and intervening on each newly generated token. The results are shown in Figure 6. As we can see, using any extracting function and intervention position results in significant improvements in personalized text generation. Although it is very simple and does not introduce excessive complexity, the performance of the Mean Difference function is still highly superior. Moreover, the more tokens are intervened, the more pronounced the performance improvement.

C.2 Analysis of General Model Selection

We compare the different choices of the general LLM M_g in Equation 4 which is designed to gener-

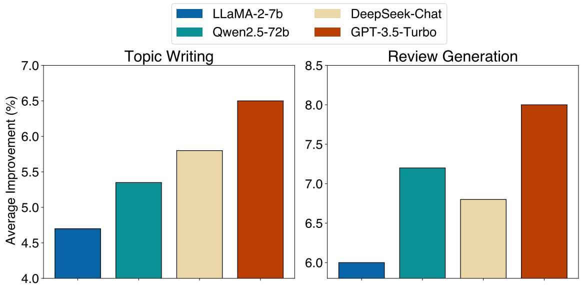

Figure 7: Performance comparison with different generic models M_g.

ate style-agnostic responses. The results are shown in Figure 7, from which we can observe that the proposed StyleVector is robust over different general models. The general model does not have to be the same as the model being intervened (LLaMA-2-7b); in fact, text generated by a more powerful model tends to have a higher relevance to the input x, greater diversity, and is more conducive to the extraction of style vectors.

C.3 Clustering

Figure 8 illustrates the distribution of clustered style vectors for all users in two tasks of the LaMP benchmark. As can be seen, the dimensionality-reduced user style vectors can be grouped into several clusters, indicating that different users may share similar writing styles.

Additionally, in Figure 9, we provide examples of some clusters and highlight the significant writing style patterns of these clusters. For example, in the case of cluster 1, the users within it share two writing style patterns: one prefers starting with numbers, and likes adding parentheses at the end to supplement the content. For cluster 2, all users share one pattern: they tend to use the dash '–' to connect elements in the title.

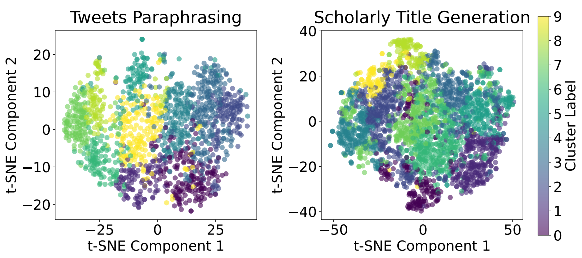

Figure 8: Clustering results of style vectors of all users.

	# User	Input Length	Output Length	# History
Abstract Generation	4560	33.82	144.28	120.30
Topic Writing	2453	28.36	263.03	50.39
Review Generation	1822	119.39	304.54	34.39
News Headline Generation	2376	29.97	9.78	287.16
Scholarly Title Generation	2500	152.81	9.26	89.61
Tweet Paraphrasing	1496	29.76	16.93	17.74

Table 4: Datasets statistics. We report the number of users in test set, the averaged length of input x and output y, and the averaged number of histories $|P_u|$.

D Scalability Bottlenecks of Baselines

As summarized in Table 1, existing methods suffer from three critical scalability constraints:

- **Training Time.** PETF demands user-specific adapter optimization with complexity $O(|P_u|)$, incurring significant costs for large user bases due to the heavy back-propagations. RAG is training-free, eliminating gradient-based training overhead but requiring $O(|P_u|)$ vectorization pre-processing.
- **Inference Latency.** In addition to the normal decoding latency of the language model, RAG suffers from $O(|P_u|)$ retrieval latency, which makes it inefficient for users with long histories. For PEFT, the process of loading these adapters can introduce overhead, particularly in scenarios requiring frequent updates or real-time interactions.
- **Storage Overhead.** RAG stores all historical interactions ($O(|P_u|)$ per user), scaling poorly for long-term usage. PETF maintains $O(rDL)$ storage for each user (typically 0.1%-1% of base model parameters), where r is the rank of LoRA, L is the number of layers and D is the model hidden dimension.

E Datasets and Task Definition

This paper utilizes the LaMP benchmark and LongLaMP benchmark for evaluation. We only select generation tasks in the two benchmarks and the statistics are shown in Table 4. We show the input-output pair formats where the text in {BRACES} can be replaced with content specific to different users and queries:

LongLaMP: Abstract Generation This task focuses on generating personalized abstracts for technical documents or articles based on the provided title and keywords.

	User ID	User-created Headlines
Cluster 1	127	The 10 Least Affordable Major Metro Areas (PHOTOS)
		5 Things Your Real Estate Agent Won't Tell You (VIDEO)
	628	13 Classic Photos Of Phil Jackson Back When He Was The Knicks' Hipster Iconoclast
		13 Ways Johnny Manziel's Pro Day Was The Most Johnny Football Thing Ever (GIFs/PHOTOS)
	1325	On 'Cats' 30th Anniversary, A Brief History (SLIDESHOW)
		'La Boheme' At Philadelphia Opera Uses High-Tech Van Goghs And Renoirs (PHOTOS)
	1652	9 Reasons You Should Get A Hair Gloss Treatment (Instead Of A Normal Dye Job)
		13 Lessons We Can Learn From Veteran Actresses' Style (PHOTOS)
Cluster 2	288	The Sponsors Of Obamacare Repeal Are Trying To Fool America -- And Fellow Republicans
		Clinton Lays Out Agenda For Making Child Care Better -- And More Affordable'
	1900	Chris Christie Video Shows That GOP Empathy Is Real -- And Limited
		Trump Has 2 Events This Weekend -- And Both Benefit His Businesses
	1401	Obama Hits The Trail For Hillary Clinton -- And To Cement His Legacy For Generations
		Trump Scrambles For A Win -- Any Win, Really -- As He Nears 100 Days

Figure 9: Case study of clustering writing patterns in News Headline Generation task. The highlighted tokens are the shared writing styles in cluster.

INPUT: Generate an abstract for the title "{title}" using the following items: "{keywords}"
OUTPUT: {abstract}

LongLaMP: Review Generation This task involves generating personalized product reviews that align with the user's preferences, based on the product description and the score assigned to the product by the user.

INPUT: Generate the review text written by a reviewer who has given an overall rating of {rating} for a product with description "{description}". The summary of the review text is "{summary}".
OUTPUT: {review}

LongLaMP: Topic Writing This task focuses on generating a personalized long-form Reddit post on a given topic from its summary written by user.

INPUT: Generate the content for a Reddit post "{summary}".
OUTPUT: {post}

LaMP: News Headline Generation This task focuses on generating a personalized news headline for a given user-created article.

INPUT: Generate a headline for the following article "{article}".
OUTPUT: {headline}

LaMP: Scholarly Title Generation This task requires language models to generate titles for an input abstract of an paper.

INPUT: Generate a title for the following abstract of a paper "{abstract}".
OUTPUT: {title}

LaMP: Tweet Paraphrasing This task requires language models to generate a tweet in the style of a user given an input tweet.

INPUT: Paraphrase the following tweet without any explanation before or after it "{original tweet}".
OUTPUT: {tweet}