PERSJEPA: INTERPRETABLE LANGUAGE MODEL PERSONALIZATION THROUGH SPARSE RESIDUAL PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models can personalize their answers when given a user's profile or past interactions, but repeatedly supplying this information consumes context and computation. Learning personalized parameters can reduce this dependence on prompting, yet offers limited visibility into how user preferences affect the model's responses. Activation steering has attracted growing interest as a way to personalize models through interventions in hidden states without updating their parameters, yet it faces three key challenges: (1) fixed adjustments overlook how user preferences interact with each request, (2) steering confined to the initial decoding step leaves later steps without further intervention, and (3) accurate prediction of hidden representation changes does not reliably translate into personalized responses. To address these challenges, we introduce PERSJEPA, a framework for interpretable personalization of frozen language models through sparse prediction conditioned on user information. Inspired by JEPA, we formulate PREDICTIVE RESIDUAL LEARNING, using paired generic and personalized representations to learn how user context changes the model's hidden states. We design SPARSE RESIDUAL DECODING so that each request activates a small set of learned features whose directions are combined into a personalization residual. User conditioning determines which features contribute and how they adjust the model's representations. We further develop GENERATION-ALIGNED STEERING, applying the decoded residual throughout generation and optimizing the predictor for the likelihood of personalized responses through the frozen model while retaining residual supervision. The language model remains frozen throughout training, and inference requires no profile text in the prompt. Experiments demonstrate substantial improvements in controlled persona generation across multiple language models and gains on real user personalization benchmarks. Analyses of the learned features reveal how their contributions vary across user groups, providing a concrete basis for interpreting the learned personalization interventions. Our implementation is fully open-sourced.[1]

1 INTRODUCTION

A personalized language model should account for both the user's preferences and the request at hand. Different users may expect different examples, explanations, or writing styles in response to the same question, and a given preference may be expressed differently across tasks. Understanding this interaction is central to personalized writing, recommendation, and information assistance (Salemi et al., 2024; Liu et al., 2025).

Including user profiles or retrieved interaction histories in the prompt offers a straightforward way to personalize responses (Salemi et al., 2024; Kumar et al., 2024). However, processing this context for every request adds computational cost and uses space that could otherwise support the task. Learning user representations or personalized parameters can reduce this dependence on prompting (Cao et al., 2026a; Li et al., 2026), yet offers limited visibility into which preferences shape a response.

[1] https://anonymous.4open.science/r/pers-jepa-462F

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

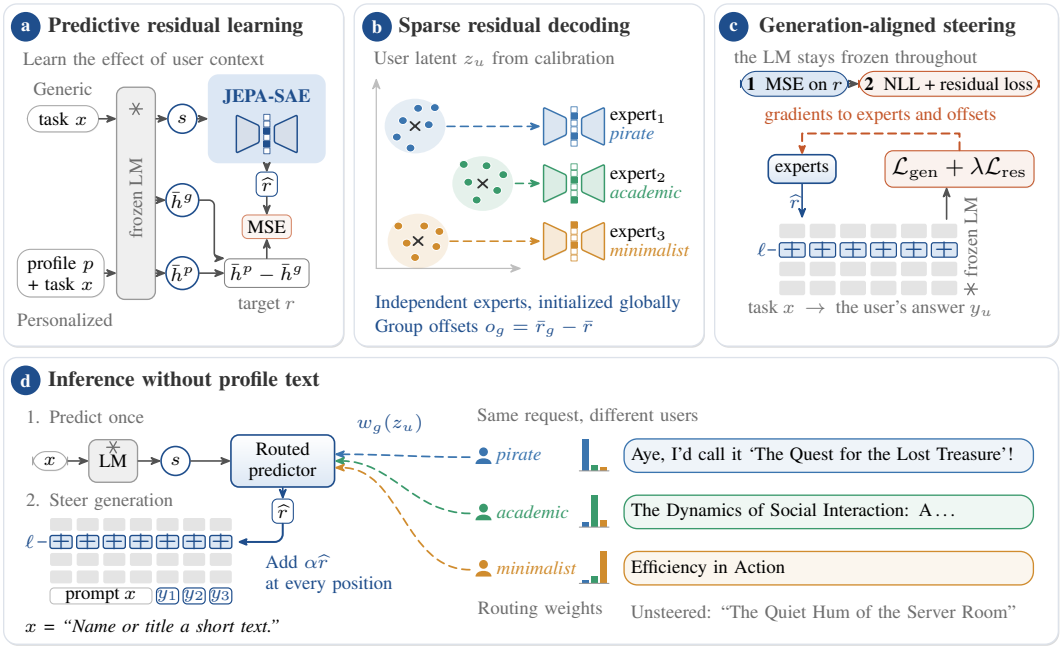

Figure 1: Overview of PERSJEPA. (a) Generic and personalized views define the residual target. The predictor uses a request-only state s; target states $\bar{h}^g$ and $\bar{h}^p$ average matched positions of the same reference answer. Answer conditioning and prediction tokens are omitted from this two-view schematic. (b) Calibration residuals define reusable user representations that route requests to independent sparse experts initialized from a global predictor. (c) Residual pretraining is followed by answer-likelihood fine-tuning with auxiliary residual supervision; only the predictors and offsets are updated. (d) An unsteered source pass obtains s, followed by a separate generation pass that adds the predicted residual at every prompt and decoding position of layer ℓ. The language model remains frozen, and the generation prompt contains no profile text. Colored arrows show alternative user routes into the predictor. Routing weights are schematic; example responses illustrate different persona-conditioned outputs.

Activation steering controls model behavior by intervening in hidden representations while keeping the language model's parameters fixed (Rimsky et al., 2024). Recent work has explored user-specific steering directions (Zhang et al., 2025; Zhao et al., 2025), adjustments tailored to individual requests (Du et al., 2025), and preference-sensitive attention heads (Zhang et al., 2026b).

However, several design choices in existing steering methods can limit personalization. First, fixed user vectors apply the same adjustment across requests (Zhang et al., 2025), motivating request-dependent steering (Du et al., 2025). Second, applying an intervention at too few token positions can weaken personalization (Zhang et al., 2025). Third, reconstructing hidden-state residuals does not directly optimize the likelihood of personalized responses. Prior work on learning interventions from target outputs or preference pairs provides a way to connect representation changes to generation (Wu et al., 2024; Cao et al., 2024).

To address these limitations, we introduce PERSJEPA, a framework for interpretable personalization through conditional sparse residual prediction (Figure 1). Our approach draws on joint-embedding predictive architectures (JEPAs), which learn to predict the representation of one view from another (Assran et al., 2023; Huang et al., 2025). We adapt this idea to paired representations of the same calibration response, encoded with and without user context by the frozen language model.

We formulate PREDICTIVE RESIDUAL LEARNING, using the difference between generic and user-conditioned representations as the predictor's pretraining target. Calibration residuals provide a reusable user representation that conditions the predictor together with the current request. Building on sparse feature decompositions (Huben et al., 2024), we then design SPARSE RESIDUAL DECODING so that each request activates a small set of features whose learned directions contribute to the

predicted residual. User information determines both which features contribute and how they are decoded. To turn these predictions into personalized responses, we develop GENERATION-ALIGNED STEERING. The decoded residual is computed once per request and applied throughout prompt processing and decoding. The predictor is then fine-tuned through the frozen language model to maximize the likelihood of personalized answers, with residual prediction retained as an auxiliary objective. At inference, personalization requires no profile text in the prompt.

We evaluate PERSJEPA on *SynPer*, a controlled synthetic benchmark we introduce, and on classification and text generation tasks from LaMP, introduced by Salemi et al. (2024). Each SynPer example contains a request, a persona description specifying tone, verbosity, and domain framing, and a response written in that persona's style. A counterfactual subset assigns different personas to the same request, allowing us to study persona expression while holding task content fixed. We compare with baselines based on FinTS and SteerX, introduced by Du et al. (2025) and Zhao et al. (2025), respectively. Across Qwen and Llama models, PERSJEPA achieves substantial gains in persona accuracy over these steering baselines on SynPer. It also improves classification accuracy on LaMP-2 and text generation metrics on LaMP-7.

Our analyses of sparse features reveal similar task-related activation patterns across user groups, while their decoding directions are associated with different persona vocabularies. For example, projecting differences in these directions into vocabulary space highlights scholarly terms for an academic persona and legal terms for a lawyer persona.

Our contributions are:

- We formulate personalization as JEPA-inspired prediction between generic and user-conditioned representations, using changes induced by user context as prediction targets.
- We develop PERSJEPA, combining sparse residual decoding with persistent steering and training for personalized responses through a frozen language model.
- We introduce *SynPer* and evaluate on multiple language models and real user tasks, with analyses of how task and persona information appear in the learned features.

2 RELATED WORK

Language model personalization. Language model personalization uses user context to adapt responses to individual preferences (Liu et al., 2025). Profiles and interaction histories provide this context for classification and text generation (Salemi et al., 2024; Kumar et al., 2024). User modeling methods extract preferences from this evidence by comparing users or interpreting edits and interaction trajectories (Qiu et al., 2025; Gao et al., 2024; Aroca-Ouellette et al., 2024). To reuse user information across requests, other approaches learn personalized parameter-efficient modules, compact prefixes, or user-dependent modulation within shared subspaces (Tan et al., 2024; Cao et al., 2026a; Li et al., 2026).

Activation steering and decoding control. Activation steering modifies a frozen language model's behavior through hidden-state directions estimated from contrasting examples (Rimsky et al., 2024; Chen et al., 2025b). Personalized steering extends this approach with user-specific directions, request-dependent adjustments, and signals that isolate preference-related content (Zhang et al., 2025; Du et al., 2025; Zhao et al., 2025). Such interventions can also be learned from target responses or preference comparisons through a frozen model (Wu et al., 2024; Cao et al., 2024). At the output level, decoding methods adjust token probabilities to strengthen contextual faithfulness (Shi et al., 2024; Zhang et al., 2026a) or express user preferences (Chen et al., 2025a; Zhang et al., 2026b).

Predictive and sparse representation learning. Joint-embedding predictive architectures learn to predict representations of one view from another (Assran et al., 2023), with later work incorporating sparsity into visual representations (Hartman & Varshney, 2025). Extensions to language introduce embedding prediction objectives and regularization of hidden-state trajectories (Huang et al., 2025; 2026). Complementary work uses sparse autoencoders to decompose language model activations into sparsely activated features that can be more interpretable than individual neurons (Huben et al.,

2024). This decomposition has also been used to identify behavioral features and steer generation through their activations (Bayat et al., 2025; Wang et al., 2025).

3 METHOD

Problem setup. Let F_ϕ be a pretrained language model whose parameters ϕ remain frozen. Each training example (u_i, x_i, p_i, y_i) contains a user identifier, a request, user context such as a profile or interaction history, and a personalized response. For each user u, a calibration set $\mathcal{C}_u$ provides examples from which to estimate a reusable user representation; evaluation requests are excluded from this set. Given a new request x, PERSJEPA predicts an additive hidden-state residual conditioned on this user representation and uses it to steer generation. The generation prompt contains the request without the profile or interaction history.

3.1 PREDICTIVE RESIDUAL LEARNING

Inspired by prediction between representation views (Assran et al., 2023; Huang et al., 2025), we construct a generic view $x_i^{\mathrm{g}} = \mathrm{Prompt}(x_i)$ and a personalized view $x_i^{\mathrm{p}} = \mathrm{Prompt}(p_i, x_i)$. Both views are encoded by the same frozen language model. Let $H_\ell(\cdot)_t \in \mathbb{R}^d$ denote the output of decoder layer ℓ at position t. To obtain a representation of the request, we append q prediction tokens to the generic prompt and read the last prediction-token state,

$$\mathbf{s}_i = H_\ell(x_i^{\mathrm{g}} \oplus \texttt{[PRED]}^q)_{\mathrm{anchor}}\,, \tag{1}$$

where $\oplus$ denotes concatenation. This pass uses neither the response nor user context.

The prediction target summarizes how user context changes the representation of a response. Under teacher forcing, we encode the same reference answer y_i after each view. Let $\mathbf{h}_{i,t}^{\mathrm{g}}$ and $\mathbf{h}_{i,t}^{\mathrm{p}}$ denote the corresponding layer-ℓ states at answer-token position t. We average their differences over an initial answer span $\mathcal{A}_i$, retaining positions present in both views,

$$\mathbf{r}_i = \frac{1}{|\mathcal{A}_i|} \sum_{t \in \mathcal{A}_i} \left(\mathbf{h}_{i,t}^{\mathrm{p}} - \mathbf{h}_{i,t}^{\mathrm{g}}\right). \tag{2}$$

Using the same answer holds the observed response prefix fixed while varying its conditioning context. The predictor therefore learns to estimate a context-induced change from the request representation, rather than reconstruct the request representation itself. Its residual supervision is $\ell_{\mathrm{res}}(\widehat{\mathbf{r}}, \mathbf{r}) = d^{-1}\|\widehat{\mathbf{r}} - \mathbf{r}\|_2^2$.

Calibration residuals also supply the user representation,

$$\mathbf{z}_u = \frac{1}{|\mathcal{C}_u|} \sum_{i \in \mathcal{C}_u} \mathbf{r}_i. \tag{3}$$

This summary captures the average effect of user context across calibration examples and can be reused for subsequent requests. The reference responses are needed to construct calibration targets, but are not inputs to the predictor at inference.

3.2 SPARSE RESIDUAL DECODING

User conditioning is implemented through G sparse predictors, one for each user group. We use provided group assignments when available and otherwise cluster $\{\mathbf{z}_u\}$ with k-means. Following sparse autoencoder parameterizations (Huben et al., 2024), expert g encodes the request into a sparse feature vector and decodes it into a residual,

$$\mathbf{a}_g(\mathbf{s}) = \mathrm{TopK}_s\big(\mathrm{ReLU}(W_g^{\mathrm{enc}}\mathbf{s} + \mathbf{b}_g^{\mathrm{enc}})\big)\,, \tag{4}$$

$$f_g(\mathbf{s}) = W_g^{\mathrm{dec}}\mathbf{a}_g(\mathbf{s}) + \mathbf{b}_g^{\mathrm{dec}} + \mathbf{o}_g. \tag{5}$$

Here $W_g^{\mathrm{enc}} \in \mathbb{R}^{m \times d}$ and $W_g^{\mathrm{dec}} \in \mathbb{R}^{d \times m}$ are the expert's encoder and decoder, and TopK_s retains at most s nonzero activations. The group offset $\mathbf{o}_g$ supplies a request-independent adjustment alongside the decoder bias. Each expert has its own encoder and decoder, allowing user conditioning to affect both feature activation and the directions in which those features are decoded.

Let $\mathbf{c}_g$ be the centroid of user representations in group g. Routing weights $w_g(u)$ combine the expert predictions,

$$\widehat{\mathbf{r}}(x,u) = \sum_{g=1}^{G} w_g(u)\, f_g(\mathbf{s}(x))\,, \qquad \sum_{g=1}^{G} w_g(u) = 1. \tag{6}$$

The default route selects the nearest centroid, making $w_g(u)$ one-hot. A soft variant applies a temperature-scaled softmax to negative squared centroid distances, normalized by their median for each user. Routing is fixed for a calibrated user, while feature activations vary with the request. Feature j in expert g contributes $w_g(u)a_{g,j}(\mathbf{s})W^{\mathrm{dec}}_{g,:,j}$ to the intervention, giving an explicit decomposition into active feature contributions and request-independent terms.

We first pretrain a global sparse predictor on all calibration pairs, then initialize each expert with a copy of its encoder and decoder. The offsets are initialized as $\mathbf{o}_g = \bar{\mathbf{r}}_g - \bar{\mathbf{r}}$, where the two means are computed over the group's examples and all calibration examples, respectively. Each expert is then pretrained on its assigned examples with its offset held fixed, minimizing $\ell_{\mathrm{res}}(f_g(\mathbf{s}_i), \mathbf{r}_i)$ together with a small activation penalty $\beta\|\mathbf{a}_g(\mathbf{s}_i)\|_1/m$. The global predictor uses the same objective without a group offset. This initialization transfers a common predictive basis to the experts before they specialize.

3.3 Generation-aligned steering

Residual supervision specifies a representation target, but does not directly optimize the generated response. We therefore train the predictor through its effect on the frozen language model. For each request, the source representation is computed in an unsteered pass, and the predictor produces a single residual $\widehat{\mathbf{r}}_i$. A separate forward pass starts from the generic prompt without prediction tokens. At an intermediate decoder layer ℓ, we apply

$$\mathbf{h}_{\ell,t} \leftarrow \mathbf{h}_{\ell,t} + \alpha\widehat{\mathbf{r}}_i \tag{7}$$

at every prompt position and every subsequent decoding step, where α controls the intervention strength. The residual remains fixed throughout the response, and the remaining decoder layers process the adjusted states.

Let $p_{\phi,\alpha\widehat{\mathbf{r}}_i}$ denote the resulting steered model. For a training batch $\mathcal{B}$, we minimize the answer-token negative log-likelihood,

$$\mathcal{L}_{\mathrm{gen}} = -\frac{1}{\sum_{i\in\mathcal{B}} T_i} \sum_{i\in\mathcal{B}} \sum_{t=1}^{T_i} \log p_{\phi,\alpha\widehat{\mathbf{r}}_i}(y_{i,t} \mid x_i^{\mathrm{g}}, y_{i,<t})\,, \tag{8}$$

where T_i counts supervised response tokens, including the end-of-sequence token. The fine-tuning objective retains residual supervision,

$$\mathcal{L} = \mathcal{L}_{\mathrm{gen}} + \frac{\lambda}{|\mathcal{B}|} \sum_{i\in\mathcal{B}} \ell_{\mathrm{res}}(\widehat{\mathbf{r}}_i, \mathbf{r}_i). \tag{9}$$

Gradients propagate through the steered language model to the predictor, updating expert parameters and group offsets while keeping the backbone, user representations, and routing weights fixed. The auxiliary term preserves a connection to the paired-view residual targets as the predictor adapts to personalized generation.

At inference, a calibrated user's routing weights are retrieved, a residual is predicted from the new request, and the frozen model generates under persistent steering. A new user is assigned to the existing experts using residuals from their own calibration examples. Neither the profile text nor a reference response is supplied in the generation prompt.

4 Experiments

4.1 Experimental setup

Datasets and protocol. We introduce *SynPer*, a synthetic benchmark pairing requests spanning ten broad task families with ten personas that differ in tone, formality, verbosity, and domain framing. It contains 10,000 training and 1,000 development examples; all reported comparisons use the

Table 1: **Task performance across three backbones and six datasets** (%). Movie Tag., Head. Gen., Title Gen., and Tweet Para. denote movie tagging (LaMP-2), headline generation (LaMP-4), title generation (LaMP-5), and tweet paraphrasing (LaMP-7), respectively. SynPer reports persona accuracy and task token F1 on held-out examples; Movie Tag. reports classification accuracy; the generation tasks report ROUGE-L against the user's own output (Qwen3.5-9B and Llama: LaMP-4/5/7 and Amazon). Subscripts: change from Base. Bold: best in each block among methods that do not see user text; †sees the profile or history at inference; Scale 1 except CAA, FinTS and SteerX on SynPer (best of 1/2/4). Trained vectors (BiPO-style): frozen backbone, our data and objective. Avg.: mean of the seven columns where all exist.

Method	SynPer		Movie.	Head.	Title.	Tweet.	Amazon	Avg.
	Acc.	F1	Acc.	R-L	R-L	R-L	R-L	
Qwen3.5-4B								
Base (no steering)	33.0	20.0	36.3	11.4	18.4	40.0	10.8	24.3
+ Profile in prompt†	90.3$_{+57.3}$	21.3$_{+1.3}$	45.7$_{+9.4}$	13.8$_{+2.4}$	15.9$_{-2.5}$	40.4$_{+0.4}$	12.3$_{+1.5}$	34.2$_{+10.0}$
+ BM25 in prompt†	83.0$_{+50.0}$	31.4$_{+11.4}$	40.9$_{+4.6}$	14.9$_{+3.5}$	38.6$_{+20.2}$	44.0$_{+4.0}$	14.1$_{+3.3}$	38.1$_{+13.9}$
+ CAA	31.3$_{-1.7}$	19.1$_{-0.9}$	33.7$_{-2.6}$	11.1$_{-0.3}$	18.2$_{-0.2}$	40.6$_{+0.6}$	10.8$_{+0.0}$	23.5$_{-0.7}$
+ FinTS	54.8$_{+21.8}$	20.5$_{+0.5}$	36.7$_{+0.4}$	11.5$_{+0.1}$	18.4$_{+0.0}$	40.4$_{+0.4}$	10.8$_{+0.0}$	27.6$_{+3.3}$
+ SteerX	40.5$_{+7.5}$	22.7$_{+2.7}$	32.5$_{-3.8}$	11.6$_{+0.2}$	19.0$_{+0.7}$	41.1$_{+1.1}$	11.0$_{+0.2}$	25.5$_{+1.2}$
+ Trained vectors (BiPO)	62.6$_{+29.6}$	28.3$_{+8.3}$	47.3$_{+11.0}$	13.9$_{+2.5}$	41.4$_{+23.0}$	43.1$_{+3.1}$	**11.4**$_{+0.6}$	35.4$_{+11.2}$
+ Pers-JEPA (ours)	**93.2**$_{+60.2}$	**36.7**$_{+16.7}$	**66.5**$_{+30.2}$	**14.8**$_{+3.4}$	**41.9**$_{+23.5}$	**43.5**$_{+3.5}$	11.1$_{+0.3}$	**44.0**$_{+19.7}$
Qwen3.5-9B								
Base (no steering)	30.1	19.0	33.2	11.4	19.6	38.9	11.2	23.3
+ Profile in prompt†	89.3$_{+59.2}$	21.1$_{+2.1}$	48.8$_{+15.6}$	12.1$_{+0.7}$	15.9$_{-3.7}$	39.9$_{+0.9}$	11.9$_{+0.7}$	34.1$_{+10.8}$
+ BM25 in prompt†	86.1$_{+56.0}$	32.1$_{+13.1}$	40.0$_{+6.8}$	14.4$_{+2.9}$	37.4$_{+17.9}$	44.7$_{+5.8}$	13.7$_{+2.6}$	38.3$_{+15.0}$
+ CAA	31.4$_{+1.3}$	18.7$_{-0.3}$	32.4$_{-0.9}$	11.7$_{+0.2}$	20.5$_{+1.0}$	39.1$_{+0.2}$	11.0$_{-0.2}$	23.5$_{+0.2}$
+ FinTS	58.8$_{+28.7}$	20.2$_{+1.2}$	32.4$_{-0.9}$	11.7$_{+0.2}$	20.4$_{+0.8}$	38.3$_{-0.7}$	10.9$_{-0.3}$	27.5$_{+4.2}$
+ SteerX	40.2$_{+10.1}$	20.6$_{+1.6}$	24.7$_{-8.5}$	11.5$_{+0.1}$	14.2$_{-5.4}$	40.9$_{+2.0}$	11.2$_{+0.1}$	23.3$_{+0.0}$
+ Trained vectors (BiPO)	49.0$_{+18.9}$	24.6$_{+5.6}$	44.9$_{+11.7}$	11.6$_{+0.2}$	39.4$_{+19.8}$	42.0$_{+3.0}$	11.8$_{+0.6}$	31.9$_{+8.6}$
+ Pers-JEPA (ours)	**93.4**$_{+63.3}$	**36.8**$_{+17.8}$	**58.4**$_{+25.1}$	**15.2**$_{+3.7}$	**42.4**$_{+22.9}$	**44.5**$_{+5.6}$	**12.7**$_{+1.6}$	**43.3**$_{+20.0}$
Llama-3.1-8B-Instruct								
Base (no steering)	32.6	20.4	31.2	8.9	19.4	29.0	11.5	21.9
+ Profile in prompt†	88.8$_{+56.2}$	23.5$_{+3.1}$	46.2$_{+15.0}$	11.1$_{+2.2}$	20.1$_{+0.8}$	31.4$_{+2.4}$	12.7$_{+1.2}$	33.4$_{+11.5}$
+ BM25 in prompt†	84.2$_{+51.6}$	32.9$_{+12.5}$	39.0$_{+7.8}$	14.1$_{+5.2}$	36.4$_{+17.0}$	43.2$_{+14.2}$	14.7$_{+3.2}$	37.8$_{+15.9}$
+ CAA	33.9$_{+1.3}$	20.7$_{+0.3}$	30.9$_{-0.3}$	9.5$_{+0.6}$	19.5$_{+0.1}$	34.1$_{+5.1}$	11.6$_{+0.1}$	22.9$_{+1.0}$
+ FinTS	58.5$_{+25.9}$	15.8$_{-4.6}$	28.6$_{-2.6}$	9.1$_{+0.2}$	19.8$_{+0.4}$	31.0$_{+2.0}$	11.5$_{+0.0}$	24.9$_{+3.0}$
+ SteerX	43.9$_{+11.3}$	21.0$_{+0.6}$	34.1$_{+2.9}$	10.2$_{+1.3}$	14.0$_{5.4}$	33.7$_{+4.6}$	11.7$_{+0.1}$	24.1$_{+2.2}$
+ Trained vectors (BiPO)	69.7$_{+37.1}$	26.1$_{+5.7}$	44.2$_{+13.0}$	13.0$_{+4.1}$	36.2$_{+16.8}$	41.5$_{+12.5}$	12.1$_{+0.6}$	34.7$_{+12.8}$
+ Pers-JEPA (ours)	**92.8**$_{+60.2}$	**34.2**$_{+13.8}$	**57.4**$_{+26.2}$	**14.6**$_{+5.7}$	**38.0**$_{+18.6}$	**42.5**$_{+13.5}$	**13.3**$_{+1.8}$	**41.8**$_{+20.0}$

same fixed subset of 989 development examples. A counterfactual subset contains 111 examples across 13 requests, each paired with multiple personas. We also evaluate movie tagging, headline generation, title generation, and tweet paraphrasing using LaMP-2/4/5/7 (Salemi et al., 2024), together with review generation on Amazon Movies & TV. Historical examples provide calibration pairs and participate in predictor training, while evaluation requests and answers remain held out. The main real-user evaluation therefore concerns new requests from calibrated users; Amazon additionally uses a chronological split. Dataset construction, calibration pools, and evaluation subsets are detailed in Appendix A.

Models and training. We use Qwen3.5-4B as the main backbone and repeat comparisons with Qwen3.5-9B and Llama-3.1-8B-Instruct. The two larger backbones use query subsets for the real-user generation tasks and the full evaluation sets for SynPer and LaMP-2. Each sparse expert has 128 features with at most 16 active per request. SynPer uses persona-group routing; real-user experiments use 20 clusters with soft routing. After residual pretraining, the predictor undergoes two epochs of answer-likelihood training with auxiliary residual weight 0.1. We use injection scale 1, greedy decoding, and a maximum of 64 generated tokens. The backbone remains frozen throughout; full settings appear in Appendix B.

Table 2: **User attribution** (%): accuracy of identifying the user from the output with a classifier trained on the users' own outputs (chance $\leq$0.3%). Task abbreviations follow Table 1; Attr. Acc. denotes user-attribution accuracy. Runs of Table 1; subscripts: change from Base; bold: best steered or trained method without user text. Among the methods shown, only the user's own text in the prompt (†) moves attribution by more than four points.

Method	Qwen3.5-4B				Qwen3.5-9B				Llama-3.1-8B			
	Head. Gen. Attr. Acc.	Title Gen. Attr. Acc.	Tweet Para. Attr. Acc.	Amazon Attr. Acc.	Head. Gen. Attr. Acc.	Title Gen. Attr. Acc.	Tweet Para. Attr. Acc.	Amazon Attr. Acc.	Head. Gen. Attr. Acc.	Title Gen. Attr. Acc.	Tweet Para. Attr. Acc.	Amazon Attr. Acc.
Base (no steering)	0.7	7.6	6.3	1.4	2.6	15.8	10.8	1.6	2.6	21.2	6.0	1.6
Profile in prompt†	1.9$_{+1.2}$	9.8$_{+2.2}$	7.7$_{+1.4}$	12.4$_{+10.9}$	6.2$_{+3.6}$	20.2$_{+4.4}$	10.0$_{-0.8}$	15.4$_{+13.8}$	5.4$_{+2.8}$	23.6$_{+2.4}$	6.8$_{+0.8}$	9.4$_{+7.8}$
BM25 in prompt†	2.1$_{+1.4}$	12.8$_{+5.2}$	9.1$_{+2.8}$	26.8$_{+25.3}$	8.2$_{+5.6}$	22.4$_{+6.6}$	13.2$_{+2.4}$	20.6$_{+19.0}$	6.6$_{+4.0}$	22.4$_{+1.2}$	13.8$_{+7.8}$	21.6$_{+20.0}$
CAA	0.8$_{+0.1}$	7.2$_{-0.4}$	6.1$_{-0.3}$	1.2$_{-0.3}$	2.8$_{+0.2}$	15.8$_{+0.0}$	10.4$_{-0.4}$	1.4$_{-0.2}$	3.4$_{+0.8}$	20.0$_{-1.2}$	6.6$_{+0.6}$	2.2$_{+0.6}$
FinTS	0.8$_{+0.1}$	7.4$_{-0.2}$	6.3$_{+0.0}$	1.5$_{+0.1}$	2.8$_{+0.2}$	16.0$_{+0.2}$	**11.0**$_{+0.2}$	**2.4**$_{+0.8}$	4.4$_{+1.8}$	**20.2**$_{-1.0}$	4.8$_{-1.2}$	1.6$_{+0.0}$
SteerX	0.9$_{+0.2}$	8.1$_{+0.5}$	6.5$_{+0.2}$	**1.8**$_{+0.4}$	2.7$_{+0.1}$	16.2$_{+0.4}$	9.8$_{-1.0}$	1.6$_{+0.0}$	3.6$_{+1.0}$	21.3$_{+0.1}$	5.2$_{-0.8}$	2.0$_{+0.4}$
Trained vectors (BiPO)	0.8$_{+0.1}$	8.3$_{+0.7}$	6.5$_{+0.2}$	1.1$_{-0.3}$	2.3$_{-0.3}$	15.9$_{+0.1}$	10.9$_{+0.1}$	0.8$_{-0.8}$	3.6$_{+1.0}$	21.4$_{+0.2}$	9.0$_{+3.0}$	**2.4**$_{+0.8}$
Pers-JEPA (ours)	**1.0**$_{+0.3}$	**9.1**$_{+1.5}$	**6.9**$_{+0.6}$	1.1$_{-0.3}$	**4.4**$_{+1.8}$	**18.2**$_{+2.4}$	9.6$_{-1.2}$	1.4$_{-0.2}$	**4.4**$_{+1.8}$	17.6$_{-3.6}$	**9.8**$_{+3.8}$	1.6$_{+0.0}$

Baselines. We compare with the unsteered model, profile prompting, and BM25 retrieval of user history. Steering comparisons use adapted implementations of CAA (Rimsky et al., 2024), FinTS (Du et al., 2025), and SteerX (Zhao et al., 2025). We also train one vector per user group using the same calibration data and answer-likelihood objective; these are the trained-vector rows labeled BiPO in the tables. This trained comparator keeps the backbone frozen and uses two epochs of answer-likelihood training. Implementation adaptations are provided in Appendix D.

Metrics. On SynPer, we report persona accuracy using a character-n-gram classifier trained on the training reference answers, together with token F1. For real-user tasks, we report classification accuracy on LaMP-2 and ROUGE-L (Lin, 2004) for generation. We additionally assess user attribution using classifiers trained on calibration answers and compare with a global predictor and permuted routing to distinguish task improvements from the contribution of user conditioning.

4.2 Controlled persona generation

PersJEPA achieves 92.8–93.4% persona accuracy across the three backbones (Table 1). On Qwen3.5-4B, it reaches 93.2%, compared with 54.8% for FinTS and 62.6% for trained group vectors. Token F1 increases from 20.0 for the unsteered model to 36.7, exceeding the steering baselines and both text-prompting references. The gains over steering baselines persist on Qwen3.5-9B and Llama-3.1-8B-Instruct.

Across four Qwen3.5-4B runs covering three seeds and two independent state extractions, persona accuracy ranges from 92.5% to 93.9% (Appendix C).

4.3 Real-user personalization

PersJEPA improves on the unsteered model on all five real-user tasks across the three backbones (Table 1). On Qwen3.5-4B, LaMP-2 accuracy increases from 36.3% to 66.5%, compared with 47.3% for trained group vectors. LaMP-5 ROUGE-L rises from 18.4 to 41.9 on Qwen3.5-4B and from 19.6 to 42.4 on Qwen3.5-9B. These results extend the gains beyond controlled personas to classification and generation with real user histories. History retrieval remains stronger in several settings, including LaMP-7 and Amazon on Qwen3.5-4B.

Routing controls identify where user conditioning contributes beyond shared task adaptation. On LaMP-2, correct routing improves accuracy by 3.2 percentage points over the global predictor (95% confidence interval $[0.3, 6.2]$) and by 12.9 points over permuted routing ($[9.4, 16.3]$). The intervals use 2,000 paired user-level bootstrap resamples (Table 6). Correct routing also improves on permuted routing on LaMP-4, while global and permuted-routing controls recover much of the ROUGE-L improvement on LaMP-5 and LaMP-7.

User attribution provides a complementary view (Table 2). On Llama, PersJEPA raises LaMP-7 attribution from 6.0% to 9.8%, but the changes are smaller or inconsistent across other model–task combinations. History prompting gives stronger attribution gains on several tasks. Task improve-

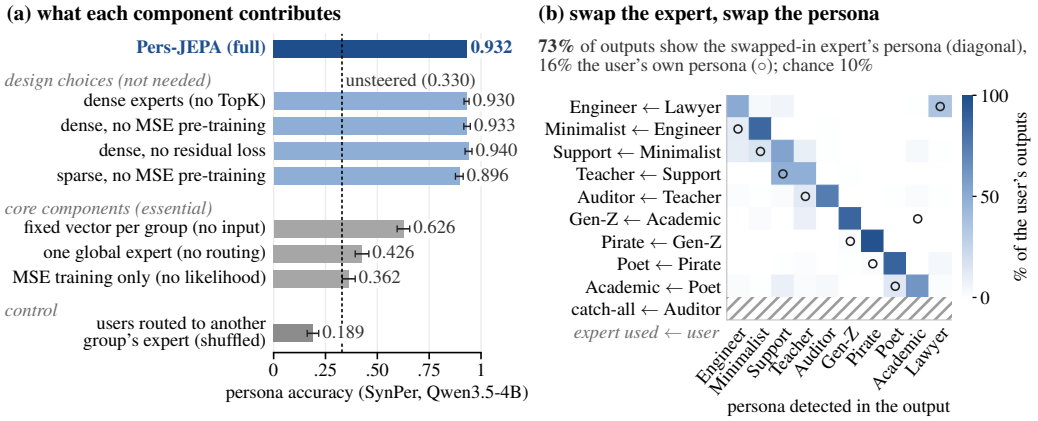

Figure 2: **Routing and the likelihood objective carry the gain; the persona follows the expert.** **(a)** SynPer persona accuracy (Qwen3.5-4B) of variants of the full model (same data and budget); whiskers: paired-bootstrap 95% CI of the difference to the full model. **(b)** Every user is routed to another persona's expert. Rows: expert used ← user; columns: persona detected in the output; circles: the user's own persona; hatched: the persona routed to the catch-all group for unlabeled users (excluded). Across three seeds and two other backbones, 65–73% of outputs take the swapped-in persona (chance 10%). Numbers: Table 3 and Appendix E.

ments therefore do not uniformly imply stronger identification of individual users; we examine the learned interventions directly in Section 5.

5 Analysis

5.1 Design ablations

We examine the roles of the training objective, user conditioning, and sparse prediction on SynPer with Qwen3.5-4B. Figure 2a summarizes the main controls; Table 3 reports the full ablations and paired-bootstrap intervals. Unless stated otherwise, these comparisons use the same 989 evaluation examples and injection scale 1.

Generation-aligned steering. Under persistent injection, the predictor trained only to reconstruct residuals achieves 36.2% persona accuracy, compared with 93.2% after answer-likelihood fine-tuning. Token F1 rises from 19.9 to 36.7, showing that training through the frozen model improves both persona expression and reference overlap. The intervention also determines where this supervision can act. An adjustment applied only to the last prompt position after the final layer directly changes the first-token logits; subsequent effects are mediated by the generated prefix. Persistent steering instead intervenes at every position, allowing answer-likelihood supervision throughout the response. The earlier single-anchor studies are detailed in Appendix G, while Figure 3b compares persistent steering at different layers.

User and request conditioning. A global expert trained with the same likelihood objective reaches 42.6% persona accuracy, while permuting user routes reduces accuracy to 18.9%. Trained group vectors reach 62.6%, indicating that the predictor learns more effective interventions than this constant-vector parameterization. A complementary post-training control replaces each expert's request-dependent predictions with its mean prediction, yielding 87.4% accuracy. The remaining 5.8-point difference shows the additional contribution of adapting the learned intervention to each request. Appendix G provides related representation-space comparisons under homogeneous and mixed-domain calibration.

Sparsity and residual supervision. Sparse and dense experts achieve similar persona accuracy, at 93.2% and 93.0%, respectively, with the same reported token F1. Omitting residual pretraining

lowers the sparse model to 89.6%, whereas dense predictors retain high accuracy even without either residual training loss (Table 3). Residual pretraining therefore benefits the sparse model, while sparsity provides an explicit feature decomposition with performance comparable to dense prediction.

5.2 Interpreting sparse personalization

We examine how feature activations and decoder directions vary across user groups. Common initialization provides corresponding feature indices for comparison, although each expert subsequently learns its own encoder and decoder. In a Qwen3.5-4B replicate achieving 93.9% persona accuracy, 40 of the 128 feature indices activate on more than 1% of an expert's own-persona calibration examples in at least one expert. Across experts and these 40 indices, the ratio of activation rates on the expert's own persona to other personas has median 1.0 and interquartile range 0.90–1.03. Decoder directions at these indices nevertheless differ across experts, with mean pairwise cosine similarity 0.82. Thus, similar activation frequencies coexist with user-dependent feature decoding; they do not imply identical feature meanings across experts.

The decoded interventions also reveal recognizable persona vocabulary. Figure 4 reads each expert's mean predicted residual, centered across experts, through the final normalization and unembedding on Qwen2.5-1.5B. Each expert ranks its own persona's vocabulary first, although this lens is an approximation for intermediate-layer interventions. Generated outputs provide a complementary check on Qwen3.5-4B (Appendix E). Among the twelve words whose frequency increases most under steering, 65% on average belong to the corresponding persona's reference vocabulary, compared with 5% for fixed group-mean residual vectors. For the Pirate persona, these include *treasure*, *yer*, *crew*, and *ahoy*.

Changing the selected expert changes the expressed persona (Figure 2b). Under permuted routing, 73.0% of outputs are classified as the replacement expert's persona and 16.4% as the original persona, excluding the group routed to the catch-all expert. The replacement persona accounts for 64.7–73.0% of outputs across the reported seeds and backbones. This intervention links expert selection to generated behavior, complementing the descriptive feature analysis. Feature statistics, vocabulary definitions, and routing-swap details appear in Appendix E.

5.3 User calibration

Useful expert groups can be learned without persona labels. On SynPer, clustering user residuals and applying soft routing yields 59.0%, 75.2%, and 82.0% persona accuracy with $K = 3$, 5, and 8 groups, respectively (Figure 3c). Held-out calibration likelihood ranks these settings in the same order, selecting $K = 8$ without requiring persona labels.

We also vary the history budget used to estimate routing weights while keeping the trained predictor fixed (Figure 3d). On LaMP-2 with five groups, accuracy increases from 54.3% under uniform routing to 63.3% with one historical example and 65.0% with a budget of five examples per user. One example reaches the accuracy of the global predictor in this experiment; the larger budget adds a further 1.7 percentage points. This experiment measures sensitivity to the history used for routing; as in the main real-user evaluation, users' historical examples also participate in predictor training. Full calibration and hyperparameter sweeps appear in Appendix C.2.

6 Conclusion

We introduced PersJEPA, a framework for interpretable personalization of frozen language models through sparse residual prediction. Inspired by JEPA, it learns from paired generic and personalized representations, decodes request-dependent adjustments through sparse features, and optimizes these interventions for personalized generation. Experiments across multiple language models demonstrate substantial gains over steering baselines in controlled persona generation and improvements on real-user personalization tasks. Our analyses identify generation-aligned training and user conditioning as important drivers of these gains, while sparse prediction provides a feature decomposition with performance comparable to dense predictors. Future work can extend this framework to evolving user preferences and finer-grained causal analysis of individual sparse features.

AI USE STATEMENT

We used generative AI tools to assist with manuscript preparation, code development, and discussions of methodological choices and experimental findings. For manuscript preparation, an LLM-based writing assistant supported drafting and revision, including grammatical corrections, word choice, organization, and clarity of presentation. Coding assistants supported implementation and experimental workflows. We also used LLMs to generate synthetic examples for the SynPer benchmark, as described in Appendix A. The authors made the final decisions concerning the research questions, method design, evaluation protocols, and interpretation of the results. The authors remain responsible for verifying AI-assisted text, code, citations, and numerical claims against the implementation, source materials, and experimental records. We take full responsibility for the final content of this work, including all text, methods, experimental results, claims, and accompanying artifacts.

REFERENCES

Stephane Aroca-Ouellette, Natalie Mackraz, Barry-John Theobald, and Katherine Metcalf. PREDICT: preference reasoning by evaluating decomposed preferences inferred from candidate trajectories. *CoRR*, abs/2410.06273, 2024. doi: 10.48550/ARXIV.2410.06273. URL https://doi.org/10.48550/arXiv.2410.06273.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael G. Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 15619–15629. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01499. URL https://doi.org/10.1109/CVPR52729.2023.01499.

Reza Bayat, Ali Rahimi-Kalahroudi, Mohammad Pezeshki, Sarath Chandar, and Pascal Vincent. Steering large language model activations in sparse spaces. *CoRR*, abs/2503.00177, 2025. doi: 10.48550/ARXIV.2503.00177. URL https://doi.org/10.48550/arXiv.2503.00177.

Heng Cao, Fan Zhang, Jian Yao, Yujie Zheng, Changlin Zhao, Lu Hao, Yuxuan Wei, Wangze Ni, Huaiyu Fu, Yuqian Sun, and Xuyan Mo. Beyond retrieval: Learning compact user representations for scalable LLM personalization. *CoRR*, abs/2606.04547, 2026a. doi: 10.48550/ARXIV.2606.04547. URL https://doi.org/10.48550/arXiv.2606.04547.

Heng Cao, Fan Zhang, Jian Yao, Yujie Zheng, Changlin Zhao, Lu Hao, Yuxuan Wei, Wangze Ni, Huaiyu Fu, Yuqian Sun, and Xuyan Mo. Beyond retrieval: Learning compact user representations for scalable LLM personalization, 2026b. URL https://arxiv.org/abs/2606.04547.

Yuanpu Cao, Tianrong Zhang, Bochuan Cao, Ziyi Yin, Lu Lin, Fenglong Ma, and Jinghui Chen. Personalized steering of large language models: Versatile steering vectors through bi-directional preference optimization. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/58cbe393b4254da8966780a40d023c0b-Abstract-Conference.html.

Ruizhe Chen, Xiaotian Zhang, Meng Luo, Wenhao Chai, and Zuozhu Liu. PAD: personalized alignment of llms at decoding-time. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025a. URL https://openreview.net/forum?id=e7AUJpP8bV.

Runjin Chen, Andy Arditi, Henry Sleight, Owain Evans, and Jack Lindsey. Persona vectors: Monitoring and controlling character traits in language models. *CoRR*, abs/2507.21509, 2025b. doi: 10.48550/ARXIV.2507.21509. URL https://doi.org/10.48550/arXiv.2507.21509.

Kounianhua Du, Jianxing Liu, Kangning Zhang, Wenxiang Jiao, Yuan Lu, Jiarui Jin, Weiwen Liu, Yong Yu, and Weinan Zhang. Fints: Efficient inference-time personalization for llms with fine-grained instance-tailored steering. *CoRR*, abs/2510.27206, 2025. doi: 10.48550/ARXIV.2510.27206. URL https://doi.org/10.48550/arXiv.2510.27206.

Ge Gao, Alexey Taymanov, Eduardo Salinas, Paul Mineiro, and Dipendra Misra. Aligning LLM agents by learning latent preference from user edits. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/f75744612447126da06767daecce1a84-Abstract-Conference.html.

Max Hartman and Lav R. Varshney. Sparsejepa: Sparse representation learning of joint embedding predictive architectures. *CoRR*, abs/2504.16140, 2025. doi: 10.48550/ARXIV.2504.16140. URL https://doi.org/10.48550/arXiv.2504.16140.

Hai Huang, Yann LeCun, and Randall Balestriero. LLM-JEPA: large language models meet joint embedding predictive architectures. *CoRR*, abs/2509.14252, 2025. doi: 10.48550/ARXIV.2509.14252. URL https://doi.org/10.48550/arXiv.2509.14252.

Hai Huang, Yann LeCun, and Randall Balestriero. Semantic tube prediction: Beating LLM data efficiency with JEPA. *CoRR*, abs/2602.22617, 2026. doi: 10.48550/ARXIV.2602.22617. URL https://doi.org/10.48550/arXiv.2602.22617.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

Ishita Kumar, Snigdha Viswanathan, Sushrita Yerra, Alireza Salemi, Ryan A. Rossi, Franck Dernoncourt, Hanieh Deilamsalehy, Xiang Chen, Ruiyi Zhang, Shubham Agarwal, Nedim Lipka, and Hamed Zamani. Longlamp: A benchmark for personalized long-form text generation. *CoRR*, abs/2407.11016, 2024. doi: 10.48550/ARXIV.2407.11016. URL https://doi.org/10.48550/arXiv.2407.11016.

Xinyu Li, Hao Zhou, Jianfeng Zhu, Julina Maharjan, Ruixin Guo, Feodor Dragan, and Ruoming Jin. Plume: Parameter-efficient personalization of large language models via low-rank user modulation in shared subspaces. *arXiv preprint arXiv:2609.04715*, 2026.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christopher Olah. Sparse crosscoders for cross-layer features and model diffing. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/crosscoders/index.html.

Jiahong Liu, Zexuan Qiu, Zhongyang Li, Quanyu Dai, Jieming Zhu, Minda Hu, Menglin Yang, and Irwin King. A survey of personalized large language models: Progress and future directions. *CoRR*, abs/2502.11528, 2025. doi: 10.48550/ARXIV.2502.11528. URL https://doi.org/10.48550/arXiv.2502.11528.

Yilun Qiu, Xiaoyan Zhao, Yang Zhang, Yimeng Bai, Wenjie Wang, Hong Cheng, Fuli Feng, and Tat-Seng Chua. Measuring what makes you unique: Difference-aware user modeling for enhancing LLM personalization. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 21258–21277. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.1095/.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 15504–15522. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.828. URL https://doi.org/10.18653/v1/2024.acl-long.828.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. Lamp: When large language models meet personalization. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 7370–7392. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.399. URL https://doi.org/10.18653/v1/2024.acl-long.399.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. Trusting your evidence: Hallucinate less with context-aware decoding. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Short Papers, NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pp. 783–791. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.NAACL-SHORT.69. URL https://doi.org/10.18653/v1/2024.naacl-short.69.

Zhaoxuan Tan, Qingkai Zeng, Yijun Tian, Zheyuan Liu, Bing Yin, and Meng Jiang. Democratizing large language models via personalized parameter-efficient fine-tuning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 6476–6491. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.372. URL https://doi.org/10.18653/v1/2024.emnlp-main.372.

Mengru Wang, Ziwen Xu, Shengyu Mao, Shumin Deng, Zhaopeng Tu, Huajun Chen, and Ningyu Zhang. Beyond prompt engineering: Robust behavior control in llms via steering target atoms. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 23381–23399. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.1139/.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. Reft: Representation finetuning for language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/75008a0fba53bf13b0bb3b7bff986e0e-Abstract-Conference.html.

Jinghao Zhang, Yuting Liu, Wenjie Wang, Qiang Liu, Shu Wu, Liang Wang, and Tat-Seng Chua. Personalized text generation with contrastive activation steering. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 7128–7141. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.353/.

Weixu Zhang, Fanghua Ye, Qiang Gao, Jian Li, Haolun Wu, Yuxing Tian, Sijing Duan, Nan Du, and Xiaolong Li. Context-fidelity boosting: Enhancing faithful generation through watermark-inspired decoding. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics, ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 42748–42759. Association for Computational Linguistics, 2026a. doi: 10.18653/V1/2026.FINDINGS-ACL.2121. URL https://doi.org/10.18653/v1/2026.findings-acl.2121.

Weixu Zhang, Ye Yuan, Changjiang Han, Yuxing Tian, Zipeng Sun, Linfeng Du, Jikun Kang, Hong Kang, Xue Liu, and Haolun Wu. Preference heads in large language models: A mechanistic framework for interpretable personalization. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 47742–47754. Association for Computational Linguistics, 2026b. doi: 10.18653/V1/2026.ACL-LONG.2205. URL https://doi.org/10.18653/v1/2026.acl-long.2205.

Xiaoyan Zhao, Ming Yan, Yilun Qiu, Haoting Ni, Yang Zhang, Fuli Feng, Hong Cheng, and Tat-Seng Chua. Steerx: Disentangled steering for LLM personalization. *CoRR*, abs/2510.22256, 2025. doi: 10.48550/ARXIV.2510.22256. URL https://doi.org/10.48550/arXiv.2510.22256.

A Dataset Details

A.1 SynPer

SynPer is a synthetic paired-view persona benchmark generated with an LLM. A record has a task prompt (the generic view), a profile naming one of ten personas and its style constraints, a target answer written in that persona's style, and metadata (persona, task type); the profile view places the profile before the same task prompt. The personas and their constraints are the Academic (precise, citation-aware, measured, abstract, avoids slang), the Corporate Lawyer (formal, caveated, risk-focused, exact, contract-like), the Customer Support Lead (empathetic, solution-oriented, calm, structured), the Gen-Z Creator (casual, punchy, meme-aware, enthusiastic, short sentences), the Minimalist Operator (terse, direct, checklist-driven, no filler), the Pirate (nautical voice, playful archaic phrasing, still answers clearly), the Poetic Essayist (lyrical, vivid, metaphorical, reflective), the Skeptical Auditor (critical, evidence-seeking, flags assumptions and edge cases), the Systems Engineer (technical, modular, tradeoff-focused, operational) and the Warm Teacher (encouraging, explanatory, patient, simple analogies).

The training split has $10{,}000$ records (971–$1{,}036$ per persona), almost all from ten task families: explain a technical concept, critique an idea, summarise an article, draft an email, recommend a plan, write a product review, answer a customer question, rewrite a message, name or title a short text, and write a polite rejection. Metadata repair recovered persona and task labels for 75 training rows and no rows were dropped; the SynPer routing table has the ten personas plus a catch-all group of 75 calibration pairs that carry no persona label in the extracted calibration file (the hatched row of Figure 2b). The development split has $1{,}000$ records; every SynPer evaluation in the paper uses the same 989-example subset, at most 100 per persona drawn with a fixed seed (96–100 per persona). The same-task counterfactual split groups identical generic prompts that occur with several personas (13 base tasks, 111 examples, 5–10 personas each). Persona accuracy is the accuracy of a character-n-gram TF–IDF logistic-regression classifier trained on the $10{,}000$ training targets. Many generic prompts are shared by all ten personas, so the unsteered model gives every persona the same answer.

A.2 Real-user datasets

We use the user-based splits of LaMP (Salemi et al., 2024) and the Movies & TV category of Amazon Reviews 2023. Evaluation queries belong to development users and never enter calibration. Each development user's own profile items become its calibration pairs, which define its user latent and, in the likelihood stage, its training answers; $1{,}500$ seeded-random training users are added to the calibration pool for the global SAE and the k-means grouping. Profile text is never shown at inference.

- **LaMP-2** (movie tagging; accuracy and macro-F1): 692 development users with one query each; $17{,}181$ calibration rows, namely up to eight profile items per development user ($4{,}766$ pairs) and the training users' $10{,}915$ profile pairs and $1{,}500$ queries.
- **LaMP-4** (news headlines) and **LaMP-5** (scholarly titles): $1{,}925$ and $2{,}500$ development users with one query each; $27{,}359$ and $33{,}500$ calibration rows ($14{,}392$ and $20{,}000$ profile pairs of the development users, $12{,}967$ and $13{,}500$ rows of the training users).

- **LaMP-7** (tweet paraphrasing): 1,500 development users; 23,847 calibration pairs (11,921 of the development users, 11,926 of the training users). A profile tweet has no source text, so the input of each calibration pair is a neutral paraphrase of the tweet written by the frozen Qwen3.5-4B (greedy, first output line; 153 pairs in which the model returned the tweet unchanged were dropped) and the target is the user's tweet, so the residual carries the user's style.
- **Amazon Movies & TV**: of 800 seeded-random reviewers with at least 50 usable reviews, the 482 that keep 50 after dropping items without a title; 15 examples per user (3–6 earlier reviews as history), split by time into 10 calibration and the 5 latest held-out examples (4,820 calibration and 2,379 held-out reviews, after removing 31 whose target is quoted in the same user's calibration history).

Metrics. ROUGE-L is computed against the user's own held-out output. User attribution is the accuracy of a word- and character-n-gram TF–IDF logistic-regression classifier, trained on the development users' calibration targets, at naming the author of each generated text (1,925-, 2,500-, 1,500- and 482-way on LaMP-4, LaMP-5, LaMP-7 and Amazon).

A.3 Hidden-space and single-anchor studies

The scope and mechanism studies of Sections 5.1–5.1 use Qwen2.5-1.5B on cued LaMP-5 (train3000/dev300 and full-train/dev2500 slices of the original split; titles cleaned by keeping the first title-like segment), Amazon Reviews 2023 All Beauty (train2000/dev300, and a heldout125 split that separates calibration examples from held-out ones) and SynPer. The pooled cross-domain experiment (Appendix G) calibrates one estimator on 5,121 answer-span pairs pooled from the three datasets and evaluates on a held-out 10% of each.

B Implementation Details

B.1 The routed predictor

Backbones and states. The backbone is frozen throughout. Qwen3.5-4B (chat template, thinking disabled) is read and steered at layer 16 of 32, Qwen3.5-9B and Llama-3.1-8B-Instruct at layer 16, and Qwen2.5-1.5B at layer 14 of 28. Three `[PRED]` tokens are appended to the generic prompt and the source state h^g is read at the last one. Calibration residuals are answer-span residuals: with the calibration answer appended to both views under teacher forcing, the profile-view minus the generic-view state, averaged over the first 32 answer positions. They define the user latents z_u, the group constants and the MSE targets.

Groups and routing. On SynPer the groups are the given personas and each user is routed hard to its own group. On real data the groups are k-means clusters of $\{z_u\}$, with $K{=}20$ on every dataset of Tables 1 and 2, and users are routed softly, $w_g(u) = \mathrm{softmax}_g\big(-\|z_u - c_g\|_2^2/(m\tau)\big)$, where m is the median squared distance and $\tau{=}1$. A new user is routed from z_u computed on k of its own calibration examples; $k{=}0$ gives uniform weights.

Experts. Each expert is a TopK JEPA-SAE with 128 latents of which 16 are active. A global SAE is first trained for 80 epochs on the residual MSE (AdamW, learning rate 3×10^{-4}, batch 256, 5% of the pairs held out); each group's expert is a copy of it fine-tuned for 40 epochs on the group's pairs (learning rate 10^{-4}), with a group offset $c_g - \bar{c}$ so that the routed predictor starts exactly at the per-group constant. Groups with fewer than 20 pairs keep the global expert. On Qwen3.5-4B the experts and offsets have 7.27 M trainable parameters.

Likelihood stage. The experts are then trained for two epochs to maximise the likelihood of the user's calibration answers under persistent injection through the frozen model (AdamW, learning rate 10^{-4}, no weight decay, batch 8, gradient clipping at 1, bfloat16, answers truncated to 64 tokens, injection scale 1), with the residual MSE as an auxiliary term of weight 0.1; gradients reach only the experts. On one H100 a Qwen3.5-4B SynPer arm trains in about 17 minutes and generates the 989 development answers in about 23 minutes.

Injection and decoding. The residual is predicted once from h^g and added, scaled by α, at every position of the injection layer during the prompt pass and at every decoding step, so generated

tokens are steered too. α=1, the training scale, unless stated; steering baselines report the best of $\alpha \in \{1, 2, 4\}$. Decoding is greedy with at most 64 new tokens.

B.2 Single-anchor studies and constant baselines

The single-anchor experiments of Sections 5.1–5.1 use Qwen2.5-1.5B with three predictor tokens and inject at the final layer at the last `[PRED]` position; left truncation keeps the end of the prompt and the anchor. Because predictor tokens can themselves damage or improve generation, these experiments always report a predictor-only control. SPAN-MEAN averages the paired-view residual over the training target-output positions (the reference output appended to both views) and injects this fixed vector with no profile text and no reference output at inference. The global PERS-JEPA-SAE has the architecture of one expert and is trained with the hidden-state MSE only. Cleaned LaMP-5 comparisons use 24 new tokens.

C Additional Results and Ablations

C.1 What carries the gain

Table 3 gives the numbers behind Figure 2a. Each arm re-trains the predictor with one change and everything else fixed: the calibration pairs and cached residuals, the 989 development examples, the byte-identical routing table, latent width 128, two likelihood epochs at batch 8 and learning rate 10^{-4}, auxiliary weight 0.1, persistent injection at scale 1, greedy decoding and the frozen backbone. *Dense* sets TopK to the latent width (a ReLU predictor of the same size) in both training stages. *No MSE pre-training* skips the hidden-MSE stage: one fresh encoder shared by all groups, zero decoder weights and a decoder bias at the global mean residual, with the group offsets kept, so every expert still starts at its group constant and users are still routed by the residual-derived latents. *No residual loss* also sets the auxiliary weight to 0. *Fixed vector per group* replaces the experts by K trainable vectors initialised at the group means, $\widehat{\Delta}(x, u) = \sum_g w_g(u)\, v_g$ (28 k parameters instead of 7.27 M), trained with the same objective, so the prediction cannot depend on the prompt.

Routing, the likelihood objective and the learned experts are each needed: without any one of them persona accuracy stays between 0.36 and 0.63. Sparsity, MSE pre-training and the auxiliary loss are not needed by the dense predictor: the differences of its three arms to the full model are not significant; sparsity is therefore kept for interpretability (Appendix E). Without pre-training the sparse predictor loses 0.036, and 0.038 [0.020, 0.057] (to 0.894) when the auxiliary loss is also removed. The fixed vectors lose 0.31, most on the Pirate (0.98 $\rightarrow$ 0.10). This arm changes the parameterisation and the optimisation together, so it shows that the experts are needed, not which of their properties matters. Task-F1 differences follow persona accuracy: -0.001 (dense), -0.017 (sparse, no pre-training), -0.084 (fixed vectors), -0.069 (no routing) and -0.168 (hidden MSE only). Injecting the hidden-MSE routed SAE at the anchor position only (layer 16, scale 3) gives 0.279 / 0.159.

On Qwen2.5-1.5B the same ordering holds. The full predictor reaches 0.878 persona accuracy and 0.329 task F1 after two epochs (0.878 / 0.866 / 0.889 for seeds 42 / 7 / 13), the global expert without routing 0.382 and permuted routing 0.173. After one epoch (full model 0.844), training only the group offsets gives 0.379 and training only the experts with the offsets frozen gives 0.854.

C.2 Hyperparameters

Figure 3 sweeps the injection scale, the injection layer, the number of groups and the cold-start routing budget on Qwen3.5-4B. Scale 1, the training scale, gives the best task F1, scale 2 trades task F1 for persona accuracy, and scale 4 over-injects. Layers 10 and 16 tie and layer 22 is worse. K can be set by held-out calibration likelihood without labels, and one of a new LaMP-2 user's own examples routes it almost as well as twenty. On Qwen2.5-1.5B (one epoch) the auxiliary residual weight trades generation for residual fidelity monotonically (0.869 / 0.844 / 0.785 persona accuracy at weights 0 / 0.1 / 1), and a second epoch helps (0.844 $\rightarrow$ 0.878).

Automatic calibration of the single-anchor baselines. For the single-anchor constant baselines of Sections 5.1–5.1 the scale needs no manual sweep. Two automatic rules, both using only cali-

Table 3: **Routing, the likelihood objective and the learned experts carry the gain; sparsity and residual pre-training do not** (SynPer, Qwen3.5-4B, 989 development examples, scale 1; unsteered: 0.330 / 0.200). Each arm changes one component of the full model; in parentheses, the difference to the full model with a paired-bootstrap 95% interval over the examples. Bold: best per column; the intervals of the three dense arms include 0.

Variant	Persona acc. (Δ [95% CI])	Task F1
Pers-JEPA (full): sparse experts, MSE, then likelihood	0.932	**0.367**
Design choices (not needed)		
dense experts (TopK = width = 128)	0.930 (-0.002 [-0.014, $+0.010$])	**0.367**
dense, no MSE pre-training	0.933 ($+0.001$ [-0.015, $+0.016$])	0.361
dense, no residual loss at all	**0.940** ($+0.008$ [-0.009, $+0.024$])	0.362
sparse, no MSE pre-training	0.896 (-0.036 [-0.056, -0.018])	0.350
Core components (needed)		
fixed learned vector per group (no input)	0.626 (-0.306 [-0.337, -0.276])	0.283
one global expert (no routing)	0.426 (-0.507 [-0.539, -0.473])	0.298
hidden-MSE training only (no likelihood)	0.362 (-0.570 [-0.602, -0.539])	0.199
Input-blind constant and control		
routed group-mean constants (no training)	0.346 (-0.586 [-0.618, -0.555])	0.199
users routed to another group's expert	0.189 (-0.743 [-0.771, -0.715])	0.242

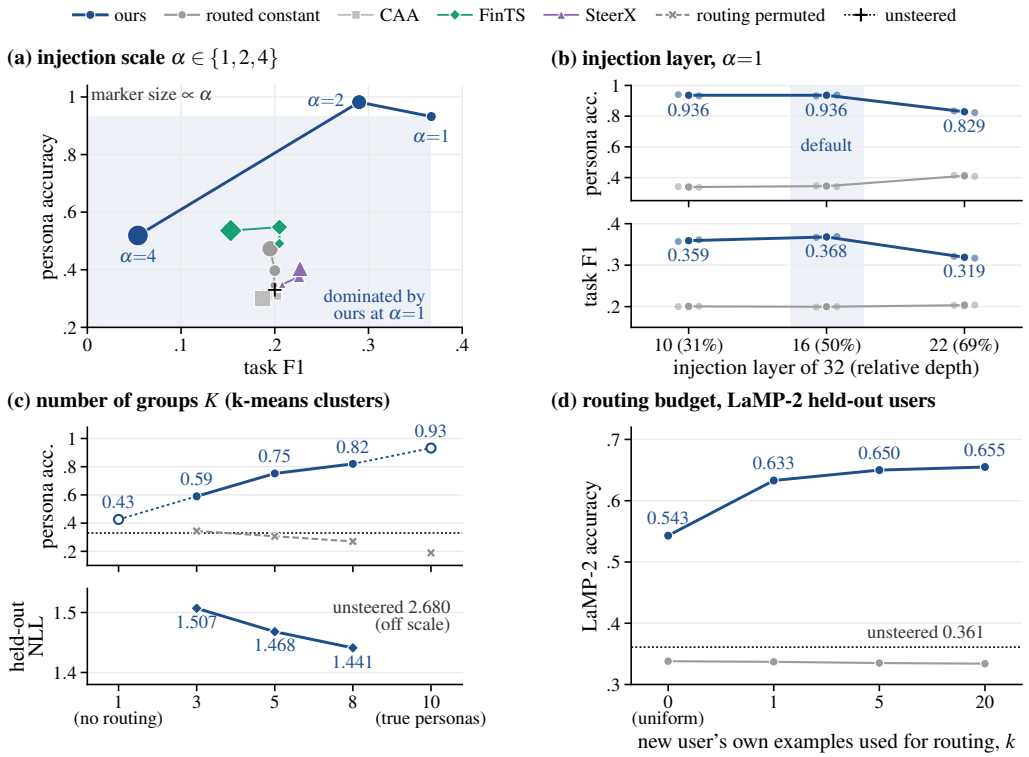

Figure 3: **Every default needs no labels, and no baseline reaches the region our predictor dominates at** α=1 (Qwen3.5-4B; SynPer in (a)–(c), LaMP-2 in (d)). (a) Methods traced over $\alpha \in \{1, 2, 4\}$; α=2 raises persona accuracy (0.93 $\rightarrow$ 0.98) at the cost of F1 (0.37 $\rightarrow$ 0.29) and α=4 over-injects (F1 0.05). (b) Layers 10 and 16 tie; layer 22 loses 0.11 persona accuracy. Dots are replicates of seed 42 on different clusters. (c) Accuracy rises with the number of k-means groups, and held-out calibration NLL ranks K the same way; the open markers (K=1, one global expert; K=10, the true personas with hard routing) come from the main run. (d) A new user routed from one own example reaches 0.633, against 0.655 from twenty.

bration data (no development or test references), were validated against the manual sweeps. Rule (a), the default: choose α by argmax of short (12-token) free generations on $\sim$150 calibration pairs scored against the calibration targets; on Qwen2.5-1.5B this selects 3.5 from a flat 2.25–3.5 plateau matching the dev-optimal range, and on Qwen2.5-7B it independently recovers the manual-sweep optimum. Rule (b): choose α by argmin of calibration-set teacher-forced answer NLL under injection; on Amazon this selects 8.0, which matches or slightly beats the manually tuned 3.0 in generation (0.237 vs. 0.234 F1). By the mechanism of Section 5.1, teacher-forced NLL is flat in α at the final layer on LaMP-5, so rule (b) is a diagnostic rather than the default selector.

C.3 Other backbones

Table 4 repeats the SynPer controls on Qwen3.5-9B and Llama-3.1-8B-Instruct (layer 16, chat template, greedy decoding, the same 989 development examples). The ordering of Qwen3.5-4B holds on both: the routed predictor reaches 0.93, the routed constants and the hidden-MSE routed SAE stay within 0.1 of the unsteered model, likelihood-trained group vectors reach 0.49–0.70, and permuted routing falls below the unsteered model. The global expert trained with the likelihood (no routing) reaches 0.403 on Qwen3.5-9B and 0.426 on Qwen3.5-4B; it was not run on Llama. On Qwen2.5-1.5B the routed predictor reaches 0.878 (Appendix C.1). Table 1 compares the same runs with the steering and prompting baselines.

Table 4: **The routed predictor reaches 0.93 persona accuracy on all three backbones, and every control stays far below it** (SynPer, 989 development examples, persona accuracy and task F1, scale 1 unless stated; one run per backbone, seed 42). Bold: best per column.

	Qwen3.5-4B		Qwen3.5-9B		Llama-3.1-8B-Instr.	
Arm	Persona	F1	Persona	F1	Persona	F1
Unsteered	0.330	0.200	0.301	0.190	0.326	0.204
Routed group-mean constants	0.346	0.199	0.374	0.197	0.424	0.210
Routed SAE, hidden MSE only	0.362	0.199	0.374	0.192	0.421	0.213
Learned group vectors, likelihood	0.626	0.283	0.490	0.246	0.697	0.261
Routing permuted (control)	0.189	0.242	0.178	0.243	0.200	0.229
Pers-JEPA, scale 1	0.932	**0.367**	0.934	**0.368**	0.928	**0.342**
Pers-JEPA, scale 2	**0.982**	0.290	**0.987**	0.307	**0.968**	0.243

D Baseline Re-implementations and Adapter Methods

All baselines run on the frozen backbone with the same user identities (SynPer personas; user ids on real data), the same calibration pairs, greedy decoding and the same 64-token budget. Trained baselines use the same likelihood objective (cross-entropy of the user's answer given the generic prompt), two epochs, batch 8 and AdamW; no method sees the profile text except the two reference rows. **FinTS** (Du et al., 2025): for each calibration example we store the attention- and MLP-sublayer output differences at layer L (last prompt token) between the query with the user's own profile and the query with another user's profile, keyed by the mean layer-L residual of the query; at inference the top-5 samples of the user are retrieved by cosine similarity, weighted by normalized similarity and added after both sublayers at every position; $\alpha = \beta$ follows the scale grid. **SteerX** (Zhao et al., 2025): preference-driven tokens are those whose log-probability rises by more than $\tau = 1$ nat when the user's other history items are in the prompt (leave-one-out); the user style vector at layer L is the mean over anchor positions of the hidden-state difference with versus without history, averaged with the same difference with versus without a style description; the original uses an external LLM to write the description, which we replace by the user's most frequent anchor tokens. The vector is injected persistently at layer L with the scale grid as γ. **TAP-PER** (Cao et al., 2026b): a learned user-state prefix (8 tokens), a query-aware record prefix built from the user's calibration answers (mean-pooled input embeddings scored by a DIN-style attention with an order-gap decay, projected to 8 tokens) and a shared bridge LoRA ($r = 8$ on the attention projections); the original stage-1 task LoRA is omitted because our method has no task adaptation either. **PLUME** (Li et al., 2026): a shared task LoRA ($r = 8$, attention projections) trained on the pooled calibration data, then per user an $r \times r$ subspace mixer per adapted map (initialised to identity), a cross-layer-shared rank-4

pair per module type and rank-1 residuals, trained on the user's own examples with the shared parameters frozen (0.32 M parameters per user on Qwen2.5-1.5B); the original's prompt-side retrieval is disabled. **BM25 history** (Salemi et al., 2024): the five calibration answers of the user that BM25 ranks highest against the request are shown as examples in the prompt (LaMP profile items when the dataset provides them).

Methods that train the backbone. TAP-PER and PLUME train LoRA adapters inside the backbone, so they combine personalization with task adaptation. Table 5 compares them with our frozen-backbone predictor on the three backbones on which both were run. On Qwen3.5-4B and Llama-3.1-8B-Instruct our predictor is within 0.015 persona accuracy of the better adapter, with task F1 0.011 and 0.028 lower. On Qwen2.5-1.5B, where the gap is largest, TAP-PER's bridge LoRA alone, with no user information, already reaches 0.404 / 0.298; TAP-PER adds $+0.54$ persona accuracy to it and our predictor trained with the same LoRA adds $+0.53$, reaching 0.929 / 0.366, so the residual gap is task adaptation rather than personalization.

Table 5: **With the backbone frozen, Pers-JEPA comes within 0.015 persona accuracy of methods that train backbone adapters** on Qwen3.5-4B and Llama-3.1-8B-Instruct (SynPer, 989 development examples; the same users, calibration pairs, likelihood objective and two epochs for every method). (a) Persona accuracy and task F1. (b) On Qwen2.5-1.5B, TAP-PER's bridge LoRA alone (no user information) and our predictor trained with the same LoRA. Bold: best per column.

(a)		Qwen3.5-4B		Llama-3.1-8B-Instr.		Qwen2.5-1.5B	
Method	Trained in the backbone	Persona	F1	Persona	F1	Persona	F1
TAP-PER (user prefix)	bridge LoRA	**0.947**	**0.378**	**0.940**	0.367	**0.947**	**0.371**
PLUME (per-user mixers)	shared LoRA	0.937	0.377	0.929	**0.370**	0.919	0.369
Pers-JEPA (ours)	nothing	0.932	0.367	0.928	0.342	0.878	0.329

(b) Qwen2.5-1.5B	Trained in the backbone	Persona	F1
TAP-PER's bridge LoRA only, no user information (control)	bridge LoRA	0.404	0.298
Pers-JEPA + the same bridge LoRA	bridge LoRA	**0.929**	**0.366**

E Interpretability of the Routed Experts

Shared codes, persona-specific decoders. All experts share the index space of the global SAE they were fine-tuned from, so a feature can be compared across experts. We call a feature *firing* if it is among the active TopK latents on more than 1% of an expert's own calibration examples in at least one expert. On Qwen3.5-4B (the replicate of the main run with 0.939 persona accuracy) 40 of the 128 features fire, and 27 on Qwen2.5-1.5B. Which features fire hardly depends on the persona: over pairs of a persona and a firing feature, the ratio between the feature's activation rate on that persona's calibration examples and on the other personas' has median 1.0 (interquartile range 0.90–1.03), and only 3% of pairs exceed 2 (1% on Qwen2.5-1.5B). What the features decode to does depend on the persona: the mean pairwise cosine between the experts' decoder directions of a firing feature averages 0.82 and falls to 0.63 for the most divergent feature (0.70 and 0.59 on Qwen2.5-1.5B).

The injected direction reads as the persona's vocabulary. Figure 4 reads each expert's injected direction through the logit lens: for every expert of the two-epoch Qwen2.5-1.5B predictor we average its predicted residual over the 989 development anchors, subtract the mean over experts and pass the result through the final norm and the unembedding. Our experts rank their own persona's vocabulary first for 10/10 personas, against 8/10 for the hidden-MSE routed SAE and 6/10 for the routed group-mean constant. On Qwen3.5-4B we read the effect on the output instead. For each persona, we rank words by their increase over the unsteered output using informative-Dirichlet log-odds, excluding stopwords. Of the twelve highest-ranked words, 65% on average belong to the persona's reference vocabulary, against 5% for the routed group-mean constant. The reference vocabulary comprises the top 50 words by log-odds in that persona's training answers relative to the other personas'.

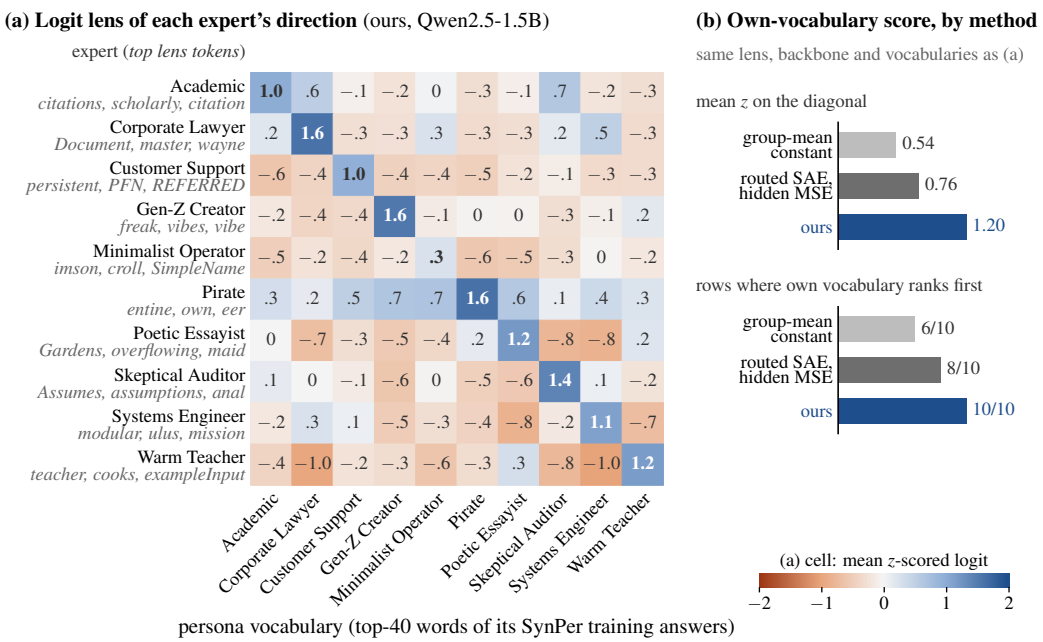

Figure 4: **Each expert's injected direction reads as its persona's vocabulary (Qwen2.5-1.5B, SynPer).** (a) Mean predicted residual of each likelihood-trained expert over the 989 development anchors, minus the mean over experts, read through the final norm and unembedding; a cell is the mean z-scored logit of a persona's top-40 training-answer words. Row labels are the first three ASCII alphabetic lens tokens (otherwise unfiltered). (b) The same score for the routed group-mean constant and the hidden-MSE routed SAE: only our experts rank their own vocabulary first for all ten personas. The mid-layer lens is approximate; the Pirate direction raises every vocabulary.

Swapping the route swaps the persona. Figure 2b uses the permuted-routing control of the main run: a fixed-seed derangement of the routing table over the ten personas and the catch-all group. Leaving out the user routed to the catch-all group, 73.0% of the outputs are classified as the swapped-in expert's persona and 16.4% as the user's own (chance 10%), and the swapped-in persona is the most frequent prediction for 8 of the 9 users (the ninth, the Customer Support Lead routed to the Warm Teacher's expert, is a tie at 49 of 99 outputs). The same analysis gives 68.6% and 64.7% for seeds 7 and 13, which draw different derangements, 72.4% on Qwen3.5-9B and 68.5% on Llama-3.1-8B-Instruct.

Feature cards. For each expert the supplementary material lists, per feature, the activation rate on the persona's own calibration examples versus the other personas', the decoder norm, the top-activating requests and answers, and the logit lens of the decoder direction and of the expert's decoder direction minus the mean over experts.

F Real Users: Paired Bootstrap

Table 6 gives paired bootstrap intervals for the real-user claims of Section 4.3, on the $K{=}20$ runs of Table 1. The resampling unit is the user (2,000 resamples); on LaMP-4, LaMP-5 and LaMP-7 every development user has one query, so this is the per-example paired bootstrap, and on LaMP-2 it respects the correlation between a user's examples. Intervals reflect evaluation-sample uncertainty only. The routed predictor improves every task metric over the unsteered model, and routing beats permuted routing on LaMP-2 at every K ($+0.039$ [$+0.014$, $+0.064$], $+0.123$ [$+0.088$, $+0.160$] and $+0.134$ [$+0.103$, $+0.169$] at $K{=}5$ / 10 / 20 in a second set of runs) and on LaMP-4. LaMP-2 macro-F1 moves with accuracy (0.629 vs. 0.280 unsteered; TAP-PER 0.671, PLUME 0.618). Two results bound the claim. TAP-PER is ahead on LaMP-2 by 0.039 [0.004, 0.071] accuracy. And user

attribution (Table 2) moves significantly only when the user's own text is in the prompt: on LaMP-4 and LaMP-7 no attribution difference between ours and the unsteered, no-routing or permuted arms is significant; on LaMP-5 ours is $+0.015$ [$+0.004$, $+0.026$] above the unsteered model but indistinguishable from the two controls; on Amazon both adapters are slightly but significantly above ours (0.013 and 0.020), all close to chance.

Table 6: **On real users the routed predictor improves every task metric over the unsteered model, and correct routing beats permuted routing on LaMP-2 and LaMP-4** (Qwen3.5-4B, $K{=}20$; ours minus the column's arm, with paired user-level bootstrap 95% intervals; *: the interval excludes 0).

Dataset, metric	Ours − unsteered	Ours − no routing	Ours − permuted
LaMP-2, accuracy	$+0.302^*$ [$+0.259$, $+0.344$]	$+0.032^*$ [$+0.003$, $+0.062$]	$+0.129^*$ [$+0.094$, $+0.163$]
LaMP-4, ROUGE-L	$+0.034^*$ [$+0.027$, $+0.041$]	$+0.004$ [-0.001, $+0.009$]	$+0.015^*$ [$+0.010$, $+0.021$]
LaMP-5, ROUGE-L	$+0.235^*$ [$+0.227$, $+0.243$]	$+0.004^*$ [$+0.001$, $+0.006$]	$+0.002$ [-0.001, $+0.004$]
LaMP-7, ROUGE-L	$+0.035^*$ [$+0.028$, $+0.041$]	$+0.001$ [-0.002, $+0.003$]	$+0.001$ [-0.001, $+0.003$]

G Scope Evidence from the Earlier Estimator Family

Sections 5.1–5.1 summarise hidden-space and single-anchor results of the global Pers-JEPA-SAE and the constant Span-Mean on Qwen2.5-1.5B. This appendix gives the supporting numbers; none of them involves the routed predictor.

Heterogeneous calibration. Table 7 pools the answer-span residuals of LaMP-5, Amazon All Beauty and SynPer and reports held-out gap recovery per domain over four seeds of an extraction independent of the original one. The per-domain mean residuals are nearly orthogonal (pairwise cosines between -0.12 and 0.21), so the pooled constant collapses on Amazon, below no steering in 4/4 seeds, while the pooled Pers-JEPA-SAE, which never sees a domain label, beats the per-domain constant on LaMP-5 and SynPer in 4/4 seeds and is within one standard deviation of it on the small Amazon split.

Table 7: **One pooled conditional estimator matches three separately calibrated constants, while the pooled constant collapses** (held-out hidden gap recovery, mean ± s.d. over 4 seeds; Qwen2.5-1.5B, answer-span residuals, one estimator calibrated on all three domains). Bold: best among the estimators that see no domain label; the last row uses the label.

Estimator (pooled calibration)	LaMP-5	Amazon	SynPer
Constant, pooled (Span-Mean)	0.227 ± 0.010	-0.206 ± 0.025	0.084 ± 0.006
Content clusters, $k{=}8$ (no identity)	0.332 ± 0.015	$\mathbf{0.173} \pm 0.028$	0.262 ± 0.015
Pers-JEPA-SAE (pooled, no domain label)	$\mathbf{0.355} \pm 0.012$	0.170 ± 0.044	$\mathbf{0.387} \pm 0.012$
Per-domain constant (uses the domain label)	0.326 ± 0.015	0.185 ± 0.023	0.244 ± 0.007

Identity-free conditioning on Amazon. On Amazon All Beauty, k-means with $k{=}4$ on the generic source states alone (no user identity) beats the global mean in held-out hidden gap recovery (0.19–0.21 vs. 0.18) with cluster purity with respect to user identity near chance (0.03–0.14). Steering each example with its own cluster's mean residual beats a shuffled-cluster control in generation (0.219 vs. 0.202 token F1), but at the tested scale neither beats the global mean (0.234).

Single-anchor generation. Under single-anchor final-layer injection the estimators do not separate. With pooled checkpoints, clean F1 on LaMP-5 at scale 3 is 0.080 (raw), 0.367 (Span-Mean) and 0.371 (Pers-JEPA-SAE); on Amazon every estimator reaches the same plateau at scales 4–6 (0.166 raw, 0.203 and 0.199). With SynPer-only checkpoints persona accuracy on 300 development examples is 0.310 raw, 0.303 / 0.327 for Span-Mean and 0.287 / 0.273 for Pers-JEPA-SAE at scales 3 / 8.

Crosscoders. Crosscoders for model diffing (Lindsey et al., 2024) trained on the same paired activations (LaMP-5 and SynPer; latent 128 with top-k 16, plus a 1024/32 capacity check) have *zero* view-specific latents at every capacity: the profile shift is a rotation of shared latents' decoder directions (mean paired-decoder cosine 0.45 / 0.26 on anchor pairs for LaMP-5 / SynPer, 0.95 / 0.88 on answer-span pairs). Read from h^g alone, as profile-free use requires, the crosscoder recovers 0.373 / 0.471 of the held-out anchor gap against 0.894 / 0.972 for the asymmetric PERS-JEPA-SAE, which is why the latter reads only the generic view.

Standard activation steering. Under identical injection machinery, calibration data, layer, anchor and scale sweep (LaMP-5 dev300, uncued), the answer-span residual of SPAN-MEAN reaches 0.373 clean F1, stable over scales 2.25–3.0, against 0.276 for difference-of-means at the `[PRED]` anchor (only at scale 1.5), 0.251 for the prompt-token mean difference and 0.214 for difference-of-means at the natural anchor (raw 0.084): estimating the residual over the answer span, where personalization aligns with generation, is worth $+0.10$–0.16 F1.

Persistent injection of persona vectors. Before the routed predictor, persona-conditional constant vectors injected persistently at layer 14 of Qwen2.5-1.5B (SynPer, n=989, scales 0.5–4) raised pooled persona accuracy from 0.319 to 0.374 at scale 4 ($z \approx 2.6$) with task F1 unchanged (0.224 vs. 0.220), while the global vector never significantly exceeded the unsteered model (best 0.338). This was the first generation-level result in which a user-conditioned direction beat the population direction, and it located the bottleneck in the intervention rather than the estimator.