TEST-TIME PERSONALIZATION VIA DPO WITH LATENT REWARD FEATURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Standard preference alignment of language models (LMs) relies on a single reward model (RM), overlooking the diversity of human preferences. While recent work on personalized RMs address this by learning reward features—representing shared latent preference dimensions—together with user-specific weights, updating the alignment objective to these personalized weights may still require repeatedly performing costly preference optimization for each user. In this paper, we propose `TPD` (Test-time Personalized Decoding), a framework for on-the-fly personalization of the LM that avoids model retraining for each user. `TPD` first learns a set of shared feature-level policies through a DPO-like objective, bypassing the explicit learning of reward features. At test-time, `TPD` then balances these policy logits using weights adapted to the user's preferences, following a multi-objective decoding approach. Importantly, we show that `TPD` naturally extends to a broad class of alignment objectives based on f-divergences. Experiments on synthetic and real-world datasets demonstrate effective steering of generation toward maximizing individual user satisfaction.

1 INTRODUCTION

Alignment of language models (LMs) with human preferences is a key step to steer their behavior toward safe and desirable outcomes (Stiennon et al., 2020; Ouyang et al., 2022). Mainstream aproaches address this in two stages. First, a utility model of human preferences, called a reward model, is learned from preference data (e.g., pairwise comparisons of LM responses to the same prompt) collected from researchers or crowd workers. In the second stage, often referred to as preference optimization, the LM is fine-tuned to increase the likelihood of generating high-reward responses, typically through reinforcement learning (RLHF) (Stiennon et al., 2020; Grattafiori et al., 2024) or, without explicitly learning a reward model, direct preference optimization (DPO) (Rafailov et al., 2023). This population-level alignment has proven effective at steering models to satisfy well-defined and broadly accepted objectives such as helpfulness, harmlessness, and honesty (HHH) (Ouyang et al., 2022). However, once answers satisfy HHH requirements, what constitutes a good answer to an open-ended question remains inherently subjective (Kirk et al., 2024; Zhang et al., 2026), suggesting the need for *personalized* alignment to maximize individual user satisfaction (Guan et al., 2025; Xie et al., 2025a). At the societal level, personalized LM alignment may also contribute to preserving cultural diversity and plurality of opinions (Sorensen et al., 2024).

Personalized alignment comes with several important challenges. As user bases tend to be very large, per-user model updates are typically prohibitive. Moreover, preference data are often scarce per user and evolve over time, motivating dynamic, continually learning strategies as opposed to static model updates. *Test-time* personalization has recently emerged as a way to address these challenges. Multi-objective alignment methods (Shi et al., 2024; Lin et al., 2025; Li et al., 2026b), which combine customizable weighted sums of reward models (RMs), offer a natural way to adapt a model on the fly at test time. However, these approaches focus on exploring trade-offs among predefined objectives, typically the HHH objectives, for which separate single-objective RMs can readily be obtained. As a consequence, these methods do not address the discovery of the preference dimensions underlying the preferences of a target user population, which are often *unknown*, as they may be difficult for users themselves to articulate, and may vary across populations and contexts (Movva et al., 2026; Li et al., 2026a). Recent work on personalized reward modeling suggests that these preference dimensions can be captured as learnable *reward features* (Barreto et al., 2025;

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.

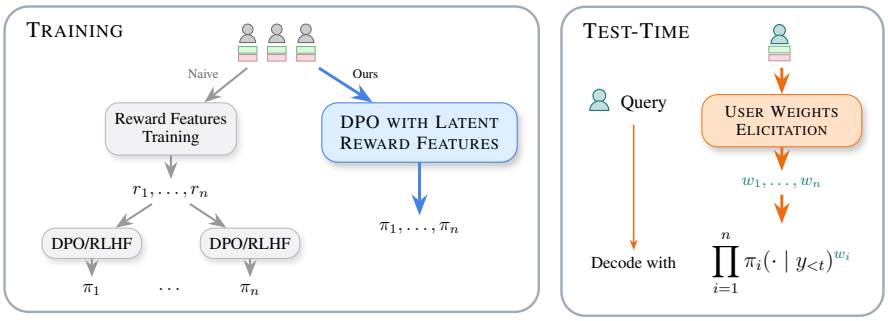


Figure 1: **Overview of TPD .** *Training-time:* Instead of sequentially learning reward features and fine-tuning policies (Naive), our formulation jointly learns feature-level policies $\pi_1, \ldots, \pi_n$ from historical preference data. *Test-time:* Given a query and minimal feedback, we elicit user weights $w_1, \ldots, w_n$ for multi-objective decoding without updating model parameters.

Bose et al., 2025; Chen et al., 2025a), providing a shared representation in which each user can be characterized by their relative valuation of these different dimensions. This enables preference elicitation to be amortized across users: rather than learning a reward model from scratch for each user, one can first learn a shared latent preference space from historical data, and then infer a new user's position in this space from only a few preference signals (Park et al., 2015)

Building on this idea, we introduce a framework for test-time personalization that first learns latent reward features from historical data, then uses minimal feedback to infer user weights and steer generation through multi-objective decoding. This enables personalization from limited user feedback while avoiding per-user model updates at test time. Making this framework practical presents a primary computational challenge: a naive implementation would require a heavy sequential procedure, in which the reward basis is first learned and the corresponding policies are then obtained through independent preference fine-tuning using DPO or RLHF. A central contribution of our paper is to address this bottleneck by generalizing DPO to directly learn a basis of feature-level policies through a joint optimization problem, whose resulting logits are combined at test time using personalized weights. The proposed approach is illustrated in Figure 1, and our main contributions are summarized below:

- We propose a unified framework for test-time personalization that decouples offline learning of a shared latent preference space from online user-specific weight elicitation and multi-objective steering.
- We show that a generalized formulation of DPO can directly learn a basis of feature-level policies, bypassing the need for explicit reward model training while enabling the personalized combination of logits at test time. We further show that the formulation naturally accommodates a broad class of alignment objectives based on f-divergences (Csiszár, 1967).
- We evaluate our method across synthetic and real-world benchmarks (PersonalLLM (Zollo et al., 2025), Roleplay (Singh et al., 2026), PRISM (Kirk et al., 2024), and Community Alignment (Zhang et al., 2026)) on language models including Qwen3-0.6B-Instruct and Llama-3.2-3B-Instruct, demonstrating superior personalization performance over baselines. Our code will be released upon acceptance.

2 Related Work

LM Personalization. At training time, standard personalization approaches involve fine-tuning the LM with dedicated per-user parameters (Li et al., 2024; Zhuang et al., 2024), or using training examples paired with user-specific context (Wang et al., 2024b; Singh et al., 2026). The in-context learning abilities of the model can also be leveraged at test time by augmenting the user prompt with past preference examples or user profile summaries (Gao & Das, 2024; Salemi et al., 2024a; Ryan et al., 2025). However, such approaches are limited by the model's context length and incur additional computational overhead. An alternative line of work, which we follow in this paper, instead leverages reward signals representing user preferences to guide the generation process.

Table 1: Reward-based test-time personalized alignment strategies.

Method	*Prompt free*	*General alignment objective*	*Unstructured preferences*
AMULET (Zhang et al., 2025)	✗	✗	✓
PAD (Chen et al., 2025b), OPAD (Zhu et al., 2025)	✗	✗	✗
LoRe (Bose et al., 2025), RFM (Barreto et al., 2025)	✓	✗	✓
PReF (Shenfeld et al., 2025)	✓	✗	✓
MATO (Luo et al., 2026)	✓	✗	✗
MODPO (Zhou et al., 2024), RLPHF (Jang et al., 2023)	✓	✗	✗
MOD (Shi et al., 2024)	✓	✓	✗
GenARM, PARM (Xu et al., 2025; Lin et al., 2025)	✓	✗	✗
`TPD` (Ours)	✓	✓	✓

Reward-based Test-time Personalization. Reward-guided decoding methods personalize LM generation by adjusting token probabilities using user-specific token-level RMs. Among these, several works achieve personalization via prompting by conditioning the RMs on preference instructions (Chen et al., 2025b; Zhu et al., 2025; Zhang et al., 2025), requiring users to explicitly articulate their preferences in natural language. Other approaches instead elicit user preferences by learning user-specific sequence-level RMs (as combinations of reward features) from pairwise comparisons of answers (Bose et al., 2025; Barreto et al., 2025; Chen et al., 2025a), but do not directly address the alignment objective. This is further addressed in Shenfeld et al. (2025) by combining these sequence-level RMs with best-of-N, or by training associated token-level RMs. In this work, we similarly adopt reward-feature-based sequence-level RMs. However, we achieve personalized alignment by leveraging multi-objective decoding (MOD) (Shi et al., 2024) to sample directly from the optimal policy of a general, f-divergence-based alignment objective—enabling flexible trade-offs between generation quality and diversity across different divergence choices (Wang et al., 2024a).

Multi-objective (MO) Test-time Alignment. MO methods enable test-time alignment by guiding decoding with weighted combinations of reward signals (Rame et al., 2023; Shi et al., 2024; Xu et al., 2025; Lin et al., 2025), allowing models to adapt to preferences without updating their parameters. However, they rely on predefined objective dimensions and hand-crafted weights. Recent works exploit these methods for test-time personalization (Jang et al., 2023; Zhou et al., 2024; Luo et al., 2026), but similarly assume that preferences are structured around predefined objectives. Building on MOD (Shi et al., 2024), we instead learn the preference dimensions and user weights directly from unstructured preference. The positioning w.r.t existing works is summarized in Table 1.

3 Background and Notations

We begin with the standard formalization of the alignment problem for autoregressive LMs at training-time. Let $\mathcal{X}$ denote the input space (e.g., user prompts), and $\mathcal{Y}$ the output space (e.g., responses generated by the model). An LM is defined by a policy π which, for each prompt $x \in \mathcal{X}$, induces a probability distribution $\pi(\cdot \mid x)$ over $\mathcal{Y}$. Before the alignment, the policy is initialized with a reference policy π_{ref}, obtained by instruction fine-tuning a pretrained LM (Ouyang et al., 2022).

Preference Learning. A first step in preference alignment is to collect a preference dataset $D = \{(x, y_w, y_l)\}_{j=1}^{N}$, where, for any prompt x, answer y_w is preferred to y_l, which we denote by $y_w \succ y_l$. A parametrized reward model $r_\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ representing these preferences can then be learned by maximizing the likelihood of the observed preference data. Under the Bradley-Terry (BT) model (Bradley & Terry, 1952), the probability that $y_w \succ y_l$ given the prompt x, is modeled as a logistic function of the reward difference:

$$P(y_w \succ y_l \mid x) = \sigma\left(r_\phi(x, y_w) - r_\phi(x, y_l)\right)$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ denotes the logistic sigmoid function. Consequently, maximum log-likelihood estimation reduces to solving the following optimization problem:

$$\max_{\phi} \sum_{(x,y_w,y_l)\in D} \log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l)) \tag{1}$$

Preference Optimization. Then, the objective is to find a policy π that maximizes the expected reward of its generated responses while limiting its deviation from the reference policy, i.e.:

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{X}} \left[\mathbb{E}_{y\sim\pi(.|x)} \left[r(x, y)\right] - \beta D \left(\pi(.|x)\|\pi_{\text{ref}}(.|x)\right)\right] \tag{2}$$

where the regularization term $D(\pi(.|x)\|\pi_{\text{ref}}(.|x))$ measures the discrepancy between the fine-tuned and reference policies conditioned on prompt x, and $\beta \in \mathbb{R}_+$ controls the strength of regularization. A standard choice for D is the reverse KL divergence (Ouyang et al., 2022). However, alternative discrepancy measures, including the forward KL divergence, Jensen-Shannon (JS) divergence, and α-divergence—all of which belonging to the family of f-divergences (Csiszár, 1967)—have been empirically shown to yield different trade-offs between alignment performance and generation diversity (Go et al., 2023; Wang et al., 2024a). To allow for the exploration of such trade-offs, we consider in what follows a generic f-divergence D_f, parametrized by a convex function $f : \mathbb{R}_+ \to \mathbb{R}$ satisfying $f(1) = 0$, and defined by $D_f(p\|q) := \mathbb{E}_{y\sim q}\left[f\left(\frac{p(y)}{q(y)}\right)\right]$ for any distributions p and q. In particular, the reverse KL divergence corresponds to $f(t) = t \log t$.

While Problem 2 can be solved using reinforcement learning (typically with Proximal Policy Optimization (PPO) (Schulman et al., 2017)), the optimization stage can be bypassed. Provided that ∇f is invertible and $0 \notin \text{dom}(\nabla f)$, Problem 2 admits the closed-form solution (Wang et al., 2024a):

$$\pi(y|x) = \pi_{\text{ref}}(y|x)\nabla f^{-1}\left(\frac{1}{\beta}r(x, y) - Z(x)\right) \Leftrightarrow r(x, y) = \beta\nabla f\left(\frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)}\right) + \beta Z(x) \tag{3}$$

where $Z(x)$ is a normalization factor that ensures $\pi(\cdot|x)$ defines a valid probability distribution, and π_{ref} is assumed to have full support. Although computing $Z(x)$ is intractable as it involves summing over the response space $\mathcal{Y}$, the one-to-one correspondence between reward functions and optimal policies makes it possible to eliminate the reward model altogether. Specifically, substituting Equation 3 into Equation 1 yields the generalized Direct Preference Optimization (DPO) objective (Wang et al., 2024a), which is now parameterized in terms of the policy parameter θ:

$$\max_{\theta} \sum_{(x,y_w,y_l)\in D} \log \sigma\left(\beta\nabla f\left(\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)}\right) - \beta\nabla f\left(\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right) \tag{4}$$

This formulation allows the policy to be fine-tuned directly on human preferences by maximizing the likelihood of the observed preference data. Note that the original DPO objective (Rafailov et al., 2023) is recovered by taking $\nabla f(t) = \log(t) + 1$, corresponding to the reverse KL divergence. Beyond reverse KL, other choices satisfying $0 \notin \text{dom}(\nabla f)$ include JS-divergence and the broader α-divergence family, which encompasses both forward and reverse KL (see Appendix A).

4 TPD : Test-time Personalization with Latent Reward Features

To personalize the preference alignment above, we replace the single reward with a user-specific reward model r^u, defined as a weighted linear combination of n reward features $r_1, \ldots, r_n$ representing shared preference dimensions across users:

$$r^u(x, y) = \sum_{i=1}^{n} w_i^u r_i(x, y) \tag{5}$$

where $w^u = (w_1^u, \ldots, w_n^u)$ denotes the weight vector associated with user u. Note that we additionally constrain w^u to lie in the n-dimensional simplex $\triangle_n = \{w \in \mathbb{R}_+^n \mid \sum_{i=1}^n w_i = 1\}$, making these reward features interpretable as *criteria*, in that the overall reward is monotonically increasing in each component r_i, following a "higher is better" principle.

A key property of the formulation in Equation 5 is that the exact solution to the alignment problem (Equation 2) for the user-specific reward r^u can be expressed as a combination of n policies, each corresponding to the respective solution of the alignment problem w.r.t. reward feature r_i, weighted by the user's preference vector w^u (Shi et al., 2024). In multi-objective settings, where the preference space is structured around pre-defined HHH-like objectives, this property can be used directly for test-time alignment. Prior to inference, the corresponding policies are obtained by performing DPO or RLHF on single-dimension preference data, such as comparisons of responses based on helpfulness or harmlessness. At test time, the LM can then be adapted on the fly to different preferences by varying only the weights w^u.

In what follows, we show how to exploit this mechanism for test-time personalization from unstructured preference data in the form of *holistic* pairwise comparisons $y_w \succ y_l$, where, because the reward features are *latent*, we must first learn them. Specifically, we directly learn the policies associated with the reward features through a DPO formulation, avoiding a costly two-step procedure that first estimates the features and then fine-tunes the corresponding policies (see Figure 1). We then show how the user-specific weights can be inferred from a handful of preference examples at test time and used to decode answers through personalized combinations of the policy basis.

4.1 Direct Preference Optimization with Latent Reward Features

Following previous work on learning personalized rewards via linear combinations of reward features (Barreto et al., 2025; Bose et al., 2025), we formulate the learning of these latent features as a collaborative task across a pool of *seen* users U_{seen}. More precisely, we seek to maximize the log-likelihood of the seen user preference data $D_u, u \in U_{\text{seen}}$:

$$\max_{\phi} \sum_{u \in U_{\text{seen}}} \sum_{(x, y_w, y_l) \in D_u} \log \sigma\big(r^u_\phi(x, y_w) - r^u_\phi(x, y_l)\big) \tag{6}$$

where in contrast to the monolithic preference learning task (see Equation 1), Equation 6 is now a joint optimization problem over the parameters of all users including both the personal weights and the shared reward feature parameters, i.e., $\phi = \{W, \phi_1, \ldots, \phi_n\}$ with $W = \{w^u, u \in U_{\text{seen}}\}$.

Starting from this formulation, we next derive its DPO counterpart by considering the policy basis $\pi_1, \ldots, \pi_n$, where each policy is defined as the solution to the alignment problem with respect to the corresponding reward feature r_i. From Equation 3, policies and RMs are related as follows:

$$r_i(x, y) = \beta \nabla f \left(\frac{\pi_i(y|x)}{\pi_{\text{ref}}(x, y)} \right) + \beta Z_i(x) \tag{7}$$

where $Z_i(x)$ are normalization factors ensuring policies $\pi_i(\cdot \mid x)$ are properly normalized. Then, remarking that normalization factors cancel out in the objective of Problem 6 since $r_{\phi^u}(x, y_w) - r_{\phi^u}(x, y_l) = \sum_{i=1}^n w_i^u (r_{\phi_i}(x, y_w) - r_{\phi_i}(x, y_l))$, we can use Equation 7 to substitute each reward feature r_{ϕ_i} in Problem 6, in the spirit of DPO. This yields the following equivalent optimization problem re-parameterized by policy parameters $\theta = (\theta_1, \ldots, \theta_n)$[1]:

$$\max_{\theta, W \in \triangle_n} \sum_{u \in U_{\text{seen}}} \sum_{(x, y_w, y_l) \in D_u} \log \sigma \left(\sum_{i=1}^n w_i^u \Big(\beta \nabla f \left(\frac{\pi_{\theta_i}(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} \right) - \beta \nabla f \left(\frac{\pi_{\theta_i}(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \right) \Big) \right) \tag{8}$$

In practice, the policies can be parameterized by n LoRA adapters (Hu et al., 2021) plugged into a shared frozen backbone, ensuring parameter-efficient training as long as the size of the reward feature basis n remains manageable. The training procedure is detailed in Algorithm 1. We use separate learning rates η_θ and η_w for the policy parameters and preference weights, respectively. During optimization, the preference weights w are reparameterized in terms of unconstrained logits, with the corresponding simplex-constrained weights obtained through a softmax transformation. Overall, this formulation bypasses the need to explicitly learn the latent reward features, while allowing the aligned policies to be directly combined with weights that can be adapted to any *unseen* user preferences at test-time as we show in the next subsection.

[1] A similar KL-divergence instantiation of this formulation is given in the appendix of Bose et al. (2025) for personalized reward learning; here, we derive the corresponding generalized f-divergence objective to establish its connection with multi-objective decoding.

Algorithm 1 DPO with Latent Reward Features

Require: Preference dataset $\mathcal{D}_{\text{seen}}$, batch size b, policy basis size n, DPO parameters (β, f), policy learning rate η_θ, user-weight learning rate η_w.

1: Initialize adapters $\boldsymbol{\theta}_0 = \{\theta_{i,0}\}_{i=1}^n$ and, for each user $u \in \mathcal{U}$, a logit vector $z_0^u \in \mathbb{R}^n$.
2: **for** $t = 1 \dots T$ **do**
3: Sample a batch $\mathcal{B} = \{(x_j, y_j^w, y_j^l, u_j)\}_{j=1}^b$ from $\mathcal{D}_{\text{seen}}$.
4: Compute per-user weights $w^u \leftarrow \text{softmax}(z_{t-1}^u)$ for each $u \in \mathcal{B}$.
5: Compute loss $\mathcal{L}(\boldsymbol{\theta}_{t-1}, \{w^u\}_{u\in\mathcal{B}}, \mathcal{B})$ as the negative objective in Eq. 8 averaged over $\mathcal{B}$.
6: $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_\theta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{t-1}, \{w^u\}_{u\in\mathcal{B}}, \mathcal{B})$.
7: $z_t^u \leftarrow z_{t-1}^u - \eta_w \nabla_{z^u} \mathcal{L}(\boldsymbol{\theta}_{t-1}, \{w^u\}_{u\in\mathcal{B}}, \mathcal{B})$ for each $u \in \mathcal{B}$.
8: **end for**
9: **return** Final adapters $\boldsymbol{\theta} = \{\theta_i\}_{i=1}^n$ and per-user weights $\{w^u = \text{softmax}(z^u)\}_{u\in\mathcal{U}}$.

4.2 Test-time Personalized Decoding

Adaptation to unseen users. At test time, user-specific weights can be adapted to any unseen user's value system by maximizing the log-likelihood of the user's preference data D_u – potentially containing only a handful of preference examples – with the shared reward features kept frozen:

$$\max_{w^u \in \triangle_n} \sum_{(x,y_w,y_l)\in D_u} \log \sigma \left(\sum_{i=1}^n w_i^u \psi_i(x, y_w, y_l) \right) + \frac{\lambda}{2} \|w^u\|_2^2 \tag{9}$$

where $\psi_i(x, y_w, y_l) = r_{\phi_i}(x, y_w) - r_{\phi_i}(x, y_l) = \beta \nabla f\left(\frac{\pi_i(y_w|x)}{\pi_{\text{ref}}(y_w|x)}\right) - \beta \nabla f\left(\frac{\pi_i(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$ is constant and $\lambda \geq 0$ controls the strength of the ℓ_2 regularization, which helps prevent overfitting to the limited preference data. This optimization problem is convex—akin to a logistic regression with positive examples—with only n parameters, making the adaptation to new preferences computationally and sample-efficient. In the cold-start setting, where no prior user preferences are available, we hypothesize that a uniform combination of the policy basis reflects the average preferences learned during training, thus serving as an effective baseline. Optimization details are provided in Appendix C.2.

Decoding with f-divergences. After adapting the user's weights, the user-aligned policy π^u can be derived by combining Equations 3, 5, and 7, yielding the following closed-form expression:

$$\pi^u(y \mid x) = \pi_{\text{ref}}(y \mid x) \nabla f^{-1}\left(\sum_{i=1}^n w_u^i \left(\nabla f\left(\frac{\pi_i(y \mid x)}{\pi_{\text{ref}}(y \mid x)} \right) \right) + \tilde{Z}(x) \right) \tag{10}$$

where $\tilde{Z}(x) = \sum_{i=1}^n w_u^i Z_i(x) - Z_u(x)$, and $Z_u(x)$ is the normalization factor associated with $\pi^u(.|x)$. In the standard case of the reverse KL divergence, exploiting $\sum_{i=1}^n w_i^u = 1$, the user-aligned policy simplifies to $\pi^u(y|x) = \prod_{i=1}^n \pi_i(y|x)^{w_i^u} \exp \tilde{Z}(x)$. This naturally leads to the following personalized greedy token-level decoding rule:

$$y_t \in \text{argmax} \prod_{i=1}^n \pi_i(\cdot \mid y_{<t})^{w_u^i} \tag{11}$$

In contrast to the reverse KL case, $\tilde{Z}(x)$ no longer acts as a multiplicative factor for JS-divergence and α-divergences (see Appendix A for explicit expressions), thus preventing direct computation of the policy mode. Nevertheless, we adopt the MOD approximation (Shi et al., 2024), which amounts to setting $\tilde{Z}(x) = 0$, before considering the associated token-level decoding rule. The overall test-time personalization procedure is summarized in Algorithm 2.

5 Numerical Experiments

This section evaluates `TPD` using two instruction-tuned base models of different sizes, Qwen3-0.6B (Yang et al., 2025) and Llama-3.2-3B-Instruct (Grattafiori et al., 2024), across a diverse set of synthetic and real-world preference datasets (links to datasets are provided in Appendix B):

Algorithm 2 TPD : Test-Time Personalized Decoding

Training time : Run Algorithm 1 on $\mathcal{D}_{seen}$ to obtain the policy basis $\pi_1, \ldots, \pi_n$.
Test time
1: **Input:** User u, preference data D_u, prompt x, policy basis $\pi_1, \ldots, \pi_n$
2: Compute w^u by solving Problem 9.
3: Decode response y using $y_t \in \operatorname{argmax} \pi_{\text{ref}}(. \mid y_{<t}, x) \nabla f^{-1}\left(\sum_{i=1}^{n} w_i^u \nabla f\left(\frac{\pi_i(.|y_{<t},x)}{\pi_{\text{ref}}(.|y_{<t},x)}\right)\right)$
4: **return** y

- **PersonalLLM** (Zollo et al., 2025): 10,402 open-ended prompts drawn from Anthropic Helpful-base (Bai et al., 2022), HelpSteer (Wang et al., 2024c), and RewardBench (Lambert et al., 2025), paired with responses from 8 LLMs. Users are modeled as linear combinations of RewardBench models, with preferences induced by their aggregated scores.
- **Roleplay (RP)** (Singh et al., 2026): Over 420K preference examples on answers to open-ended questions, generated by LLMs conditioned on 700 synthetic personas defined by demographic and personality attributes.
- **PRISM** (Kirk et al., 2024): A human preference dataset comprising 8,011 multi-turn conversations involving 21 LLMs and 1,500 real human participants from 75 countries. The dataset includes sociodemographic profiles, stated preferences, and turn-level feedback.
- **Community Alignment (CA)** (Zhang et al., 2026): We use the English subset of **CA** (Zhang et al., 2026), a multilingual dataset of human preferences across 21 LLM responses. After filtering, this yields 119,339 pairwise comparisons across 545 users.

Baselines. We evaluate against **DPO** (Rafailov et al., 2023), trained on pooled preference data across seen users without personalization. We also compare with **P-DPO** (Li et al., 2024), which learns user-specific (or cluster-specific) embeddings during training and integrates them into the LM via soft prompting. We further consider test-time personalization approaches applicable to our setting, namely methods that can adapt to unseen users from preference feedback without requiring natural-language instructions. We first consider retrieval- and profile-augmented in-context learning methods (Zollo et al., 2025; Salemi et al., 2024b), that we refer to as **RAG-ICL** and **PAG-ICL**. For **RAG-ICL**, we retrieve $k = 3$ preference examples whose embeddings are closest to the test prompt. **PAG-ICL** additionally uses the base LM to summarize these examples into a user profile. Following (Shenfeld et al., 2025), we use **LoRe** (Bose et al., 2025) to learn personalized RMs as weighted combinations of shared reward features, combined with Best-of-N generation ($N = 128$, temperature $= 1$), which we refer to as **LoRe+BoN**. Unless otherwise specified, all other baselines use temperature sampling with the base model's recommended parameters. Furthermore, all DPO-based methods, including TPD , use LoRA adapters (Hu et al., 2021) with rank $r = 8$ and scaling factor $\alpha = 16$. For TPD , each basis policy uses a separate LoRA adapter. All training runs are conducted on a single NVIDIA A6000 GPU. See Appendix C for baseline details.

5.1 Experiments with Controlled Preference Diversity

To evaluate performance under controlled preference diversity, we construct a diverse user base within PersonalLLM (Zollo et al., 2025) by simulating users as weighted combinations of three reward models selected for their low mutual agreement (see Appendix B). Then, we control diversity by sampling user weights from a Dirichlet distribution over the simplex with varying concentration parameter $\gamma \in [0, 1]$. In particular, we generate pools of 500 seen and 500 unseen users for $\gamma \in \{0.01, 0.1\}$, with 50 training examples for each seen user and N adaptation examples for each unseen user, where $N \in \{2, 8, 32\}$. For evaluation, we randomly sample one prompt from each user's test set. We evaluate personalization using the *win rate* against the base model, defined as the fraction of user for which the user-specific reward model assigns a higher score to the personalized output than to the base model output. We use Qwen3-0.6B as the base model and report the results in Table 2 for $N = 8$ adaptation examples on unseen users.

Results. Overall, TPD outperforms all methods across both user-diversity regimes. For seen users, TPD improves the win rate by approximately 10 percentage points over the base model for $\gamma = 0.01$,

Table 2: **Win rate (%) against the base model.** Simplex plots show user preference concentration under two diversity regimes ($\alpha \in \{0.01, 0.1\}$). Values report mean $\pm$ standard error across users.

	High diversity ($\alpha = 0.01$)			Moderate diversity ($\alpha = 0.1$)		
Method	Seen users	Unseen users	Overall	Seen users	Unseen users	Overall
Generic						
DPO	64.6±2.1	62.6±2.2	63.6±1.5	63.2±2.2	70.6±2.0	66.9±1.5
DPO (greedy)	63.2±2.2	67.0±2.1	65.1±1.5	64.8±2.1	70.2±2.0	67.5±1.5
Personalized						
P-DPO	54.2±2.2	51.6±2.2	52.9±1.6	53.4±2.2	56.0±2.2	54.7±1.6
RAG-ICL	51.4±2.2	50.0±2.2	50.7±1.6	46.8±2.2	47.0±2.2	46.9±1.6
PAG-ICL	51.8±2.2	49.8±2.2	50.8±1.6	47.0±2.2	48.0±2.2	47.5±1.6
LoRe + BoN	67.8±2.1	61.0±2.2	64.4±1.5	62.4±2.2	57.8±2.2	60.1±1.5
TPD	**74.6**±1.9	**71.2**±2.0	**72.9**±1.4	**71.4**±2.0	**74.4**±2.0	**72.9**±1.4

indicating that learning a shared policy basis with user-specific weights provides a substantial benefit in settings with diverse user preferences, while P-DPO does not yield a significant improvement over the base model. As expected, this advantage decreases in the more uniform regime ($\gamma = 0.1$). For unseen users, TPD achieves a win rate above 70% against the base model with only $N = 8$ adaptation examples, demonstrating effective personalization from limited feedback. As shown in Figure 2, increasing the adaptation budget to $N = 32$ further improves user satisfaction, while the other test-time personalization methods fall behind generic DPO. In particular, the in-context methods fail to improve over the base model.

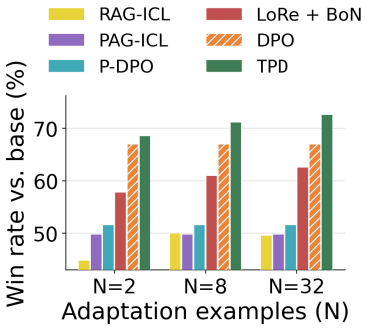


Figure 2: Win rate (%) on unseen users w.r.t. N ($\gamma = 0.01$).

5.2 Experiments with Synthetic Personas and Real-World Users

We now evaluate TPD on synthetic personas from **RP** and real-world users from **PRISM** and **CA** with Llama-3.2-3B-Instruct. For **RP**, seen and unseen users follow the predefined train/test split, while for the other two datasets, users are randomly split 80%/20%. Each user's preference data is split into 90% for training and 10% for evaluation. Evaluating personalization is challenging because it requires assessing whether a model's output aligns with a specific user's preferences based on available user information. A standard approach is to employ an LLM-as-a-judge to perform pairwise comparisons based on user descriptions (Xie et al., 2025b). Following this paradigm, we conduct an AlpacaEval-style (Dubois et al., 2024) evaluation using Qwen3.5-35B-A3B (Qwen Team, 2026) as our judge, which predicts user preferences with an accuracy of $86.1\% \pm 0.2\%$ ($59.6\% \pm 0.3\%$ without ties) on average across the three datasets. To mitigate position bias, each response pair is evaluated in both presentation orderings; a *win* is assigned if and only if the personalized response is strictly preferred across both permutations (see Appendix D).

In Table 3, we report the overall win rate, defined as $win/(win + loss)$, averaged across seen and unseen users (see Appendix E for per-pool breakdowns). Figure 3 also presents preference accuracy on held-out test pairs for reward-based baselines (**DPO**, **LoRe**, TPD). Both **DPO** and TPD are evaluated across all three divergence choices. We also include a *capacity-matched* **DPO** baseline, which scales the default LoRA rank r and scaling factor α by the number of policies n to ensure an equivalent parameter count with TPD . We adopt a default policy basis size of $n = 3$ to balance expressivity against computational and memory overhead, reducing this to $n = 2$ on **PRISM** and **CA** to account for smaller dataset scale.

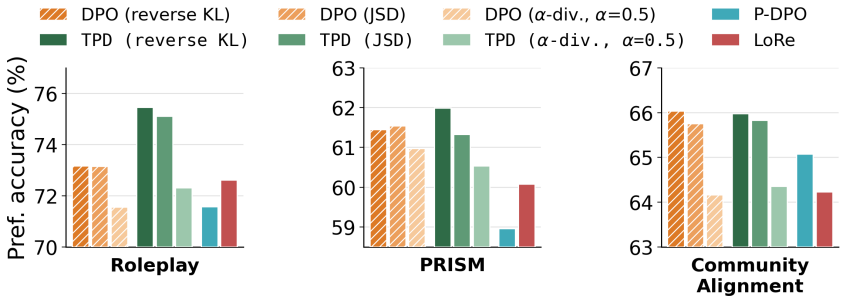


Figure 3: **Preference accuracy (%) on held-out (chosen, rejected) pairs.** Bars report the *Overall* accuracy, i.e. the mean over seen and unseen users weighted by the number of users.

Table 3: **Win rate (%) against the base model, win / (win + lose).** Values report mean ± standard error over users. *Overall* denotes the aggregated win rate across both seen and unseen users.

Method	Roleplay	PRISM	Community Alignment	Average
Generic				
DPO (reverse KL)	55.3±3.5	45.1±1.9	54.6±3.0	51.7±1.7
DPO (*capacity-matched*, reverse KL)	45.6±3.5	41.0±1.8	**60.9**±3.0	49.2±1.7
DPO (JSD)	51.6±3.6	44.7±1.8	58.6±3.1	51.6±1.7
DPO (α-div., α=0.5)	49.9±4.2	47.5±2.0	51.4±3.3	49.6±1.9
Personalized				
P-DPO	56.7±3.3	43.8±1.8	46.8±3.0	49.1±1.6
RAG-ICL	22.4±1.9	25.9±1.4	25.8±2.3	24.7±1.1
PAG-ICL	41.1±2.4	33.1±1.5	32.8±2.5	35.7±1.3
LoRe + BoN	53.8±3.1	50.5±1.8	52.6±3.0	52.3±1.6
TPD (reverse KL)	**57.9**±3.4	47.2±1.8	53.8±3.0	**53.0**±1.6
TPD (JSD)	54.8±3.4	45.8±1.8	53.1±3.1	51.2±1.6
TPD (α-div., α=0.5)	55.0±3.5	**50.6**±1.8	51.9±3.1	52.5±1.7

Results. TPD (rev. KL) achieves the highest average win rate (53.0%), slightly above LoRe+BoN (52.3%), as well as the highest average preference accuracy (67.8%). A closer look at the different datasets reveals different benefits from personalization. On **RP** and **PRISM**, TPD (rev. KL) and TPD (α-div) improve upon the best generic **DPO** variant by 2.6 and 3.1 percentage points in win rate, respectively, placing slightly ahead of other personalization baselines. In contrast, while TPD (rev.KL) remains the top-performing personalized method on **CA**, its generic counterpart outperforms it by over 5 percentage points. This variation in personalization benefits across the three datasets is also reflected in preference accuracy: TPD (rev. KL) achieves a 2.3% accuracy improvement on **RP**, whereas its generic counterpart already approximates user preferences well on **CA**, suggesting moderate preference diversity within the user base. ICL baselines perform poorly across datasets, even though adding explicit user profiles (PAG) offers a notable improvement over RAG, nearly doubling performance from 22.4% to 41.1% on **RP**.

6 Conclusion

We presented TPD , a framework for test-time language model personalization that bypasses explicit reward learning by directly learning a basis of policy features through a generalized f-divergence DPO formulation. By decoupling offline preference space learning from online user-weight elicitation, TPD enables personalized decoding for unseen users without per-user model updates. Extensive evaluations across synthetic and real-world benchmarks demonstrate that TPD effectively steers generations towards higher user satisfaction than other personalization baselines.

Reproducibility Statement

To ensure the reproducibility of our empirical results and methodology, we provide complete implementation details, and hyperparameter configurations throughout the paper and appendix. Algorithm specifications for joint training and test-time adaptation are detailed in Algorithm 1 and Section C.2. Experimental setups, base model Hugging Face identifiers, and baseline implementations are provided in Appendix C. All training and evaluation runs use a fixed random seed (`seed=42`) for model initialization and data splitting across all methods to ensure consistent experimental conditions and reproducibility.

AI use statement

In this work, we used generative AI tools for implementing methods. We have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, assisting with translation, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, or interpreting results.

Additionally, we used generative AI tools for drafting parts of a research paper and editing a research paper to improve readability. All other recommended disclosure tasks are not applicable to this work.

We have thoroughly reviewed and validated all AI-assisted work. Specifically, LLM-assisted implementation code was verified and rigorously tested for correctness by the authors, and all AI-edited text was manually reviewed to ensure accuracy, factual precision, and academic rigor. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Andre Barreto, Vincent Dumoulin, Yiran Mao, Mark Rowland, Nicolas Perez-Nieves, Bobak Shahriari, Yann Dauphin, Doina Precup, and Hugo Larochelle. Capturing individual human preferences with reward features. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Avinandan Bose, Zhihan Xiong, Yuejie Chi, Simon Shaolei Du, Lin Xiao, and Maryam Fazel. Lore: Personalizing llms via low-rank reward modeling. In *Second Conference on Language Modeling*, 2025.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. ISSN 00063444, 14643510. URL `http://www.jstor.org/stable/2334029`.

Daiwei Chen, Yi Chen, Aniket Rege, Zhi Wang, and Ramya Vinayak. Pal: Sample-efficient personalized reward modeling for pluralistic alignment. In *International Conference on Learning Representations*, volume 2025, pp. 15755–15799, 2025a.

Ruizhe Chen, Xiaotian Zhang, Meng Luo, Wenhao Chai, and Zuozhu Liu. Pad: Personalized alignment of llms at decoding-time. In *International Conference on Learning Representations*, volume 2025, pp. 9014–9039, 2025b.

Imre Csiszár. On information-type measure of difference of probability distributions and indirect observations. *Studia Sci. Math. Hungar.*, 2:299–318, 1967.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

Xiang Gao and Kamalika Das. Customizing language model responses with contrastive in-context learning. In *Proceedings of the aaai conference on artificial intelligence*, volume 38, pp. 18039–18046, 2024.

Dongyoung Go, Tomasz Korbak, Germán Kruszewski, Jos Rozen, Nahyeon Ryu, and Marc Dymetman. Aligning language models with preferences through f-divergence minimization. *arXiv preprint arXiv:2302.08215*, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jian Guan, Junfei Wu, Jia-Nan Li, Chuanqi Cheng, and Wei Wu. A Survey on Personalized Alignment—The Missing Piece for Large Language Models in Real-World Applications. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5313–5333, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.277. URL https://aclanthology.org/2025.findings-acl.277/.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

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

Hannah R Kirk, Alexander Whitefield, Paul Röttger, Andrew Bean, Katerina Margatina, Juan Ciro, Rafael Mosquera, Max Bartolo, Adina Williams, He He, et al. The prism alignment dataset: What participatory, representative and individualised human feedback reveals about the subjective and multicultural alignment of large language models. *Advances in Neural Information Processing Systems*, 37:105236–105344, 2024.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1755–1797, 2025.

Stella Li, Avinandan Bose, Faeze Brahman, Simon Du, Pang Wei Koh, Maryam Fazel, and Yulia Tsvetkov. Prefdisco: Benchmarking proactive personalized reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 80844–80908, 2026a.

Xinyu Li, Ruiyang Zhou, Zachary C. Lipton, and Liu Leqi. Personalized Language Modeling from Personalized Human Feedback, December 2024. URL http://arxiv.org/abs/2402.05133. arXiv:2402.05133 [cs].

Zhuo Li, Guodong Du, Weiyang Guo, Yigeng Zhou, Xiucheng Li, Wenya Wang, Fangming Liu, Yequan Wang, Deheng Ye, Min Zhang, et al. Multi-objective large language model alignment with hierarchical experts. In *International Conference on Learning Representations*, volume 2026, pp. 128920–128949, 2026b.

Baijiong Lin, Weisen Jiang, Yuancheng Xu, Hao Chen, and Ying-Cong Chen. Parm: Multi-objective test-time alignment via preference-aware autoregressive reward model. In *International Conference on Machine Learning*, pp. 37874–37888. PMLR, 2025.

Linhao Luo, Thuy-Trang Vu, Van-Anh Nguyen, Junae Kim, Gholamreza Haffari, and Dinh Phung. Mato: Multi-objective personalized alignment with test-time optimization for large language models. *arXiv preprint arXiv:2605.25342*, 2026.

Rajiv Movva, Smitha Milli, Sewon Min, and Emma Pierson. What's in my human feedback? learning interpretable descriptions of preference data. In *International Conference on Learning Representations*, volume 2026, pp. 59420–59455, 2026.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Dohyung Park, Joe Neeman, Jin Zhang, Sujay Sanghavi, and Inderjit Dhillon. Preference completion: Large-scale collaborative ranking from pairwise comparisons. In *International Conference on Machine Learning*, pp. 1907–1916. PMLR, 2015.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Alexandre Rame, Guillaume Couairon, Corentin Dancette, Jean-Baptiste Gaya, Mustafa Shukor, Laure Soulier, and Matthieu Cord. Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. *Advances in Neural Information Processing Systems*, 36:71095–71134, 2023.

Michael J Ryan, Omar Shaikh, Aditri Bhagirath, Daniel Frees, William Barr Held, and Diyi Yang. Synthesizeme! inducing persona-guided prompts for personalized reward models in llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8045–8078, 2025.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. Lamp: When large language models meet personalization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7370–7392, 2024a.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. LaMP: When Large Language Models Meet Personalization. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7370–7392, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.399. URL https://aclanthology.org/2024.acl-long.399/.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Idan Shenfeld, Felix Faltings, Pulkit Agrawal, and Aldo Pacchiano. Language model personalization via reward factorization. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=E7Tu5yjqXw.

Ruizhe Shi, Yifang Chen, Yushi Hu, Alisa Liu, Hannaneh Hajishirzi, Noah A Smith, and Simon S Du. Decoding-time language model alignment with multiple objectives. *Advances in Neural Information Processing Systems*, 37:48875–48920, 2024.

Anikait Singh, Sheryl Hsu, Kyle Hsu, Eric Mitchell, Stefano Ermon, Tatsunori Hashimoto, Archit Sharma, and Chelsea Finn. FSPO: Few-shot optimization of synthetic preferences effectively personalizes to real users. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=SzEc5fSBXv.

Taylor Sorensen, Jared Moore, Jillian Fisher, Mitchell L Gordon, Niloofar Mireshghallah, Christopher Michael Rytting, Andre Ye, Liwei Jiang, Ximing Lu, Nouha Dziri, et al. Position: A roadmap to pluralistic alignment. In *International Conference on Machine Learning*, pp. 46280–46302. PMLR, 2024.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

Leandro Von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning, 2020.

Chaoqi Wang, Yibo Jiang, Chenghao Yang, Han Liu, and Yuxin Chen. Beyond reverse kl: Generalizing direct preference optimization with diverse divergence constraints. In *International Conference on Learning Representations*, volume 2024, pp. 10450–10480, 2024a.

Danqing Wang, Kevin Yang, Hanlin Zhu, Xiaomeng Yang, Andrew Cohen, Lei Li, and Yuandong Tian. Learning personalized alignment for evaluating open-ended text generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 13274–13292, 2024b.

Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Scowcroft, Neel Kant, Aidan Swope, et al. Helpsteer: Multi-attribute helpfulness dataset for steerlm. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3371–3384, 2024c.

Zhouhang Xie, Junda Wu, Yiran Shen, Raghav Jain, Yu Xia, Xintong Li, Aaron Chang, Ryan A Rossi, Tong Yu, Sachin Kumar, et al. A survey on personalized and pluralistic preference alignment in large language models. In *Second Conference on Language Modeling*, 2025a.

Zhouhang Xie, Junda Wu, Yiran Shen, Yu Xia, Xintong Li, Aaron Chang, Ryan Rossi, Sachin Kumar, Bodhisattwa Prasad Majumder, Jingbo Shang, Prithviraj Ammanabrolu, and Julian McAuley. A Survey on Personalized and Pluralistic Preference Alignment in Large Language Models, April 2025b. URL http://arxiv.org/abs/2504.07070. arXiv:2504.07070 [cs].

Yuancheng Xu, Udari Sehwag, Alec Koppel, Sicheng Zhu, Bang An, Furong Huang, and Sumitra Ganesh. Genarm: Reward guided generation with autoregressive reward model for test-time alignment. In *International Conference on Learning Representations*, volume 2025, pp. 73970–73990, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Lily H Zhang, Smitha Milli, Karen Long Jusko, Jonathan Smith, Brandon Amos, Wassim Bouaziz, Manon Revel, Jack Kussman, Yasha Sheynin, Lisa Titus, Bhaktipriya Radharapu, Jane Yu, Vidya Sarma, Kristopher Rose, and Maximilian Nickel. Cultivating pluralism in algorithmic monoculture: The community alignment dataset. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=4NtoAVqfhA.

Zhaowei Zhang, Fengshuo Bai, Qizhi Chen, Chengdong Ma, Mingzhi Wang, Haoran Sun, Zilong Zheng, and Yaodong Yang. Amulet: Realignment during test time for personalized preference adaptation of llms. In *International Conference on Learning Representations*, volume 2025, pp. 74039–74069, 2025.

Zhanhui Zhou, Jie Liu, Jing Shao, Xiangyu Yue, Chao Yang, Wanli Ouyang, and Yu Qiao. Beyond one-preference-fits-all alignment: Multi-objective direct preference optimization. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10586–10613, 2024.

Mingye Zhu, Yi Liu, Lei Zhang, Junbo Guo, and Zhendong Mao. On-the-fly preference alignment via principle-guided decoding. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yuchen Zhuang, Haotian Sun, Yue Yu, Rushi Qiang, Qifan Wang, Chao Zhang, and Bo Dai. Hydra: Model factorization framework for black-box llm personalization. *Advances in Neural Information Processing Systems*, 37:100783–100815, 2024.

Thomas Zollo, Andrew Siah, Naimeng Ye, Li Li, and Hongseok Namkoong. Personalllm: Tailoring llms to individual preferences. In *International Conference on Learning Representations*, volume 2025, pp. 66949–66971, 2025.

A f-DIVERGENCES AND ASSOCIATED DECODING RULES

Table 4 details standard f-divergences along with their respective derivatives $\nabla f(t)$ and domain characteristics. Table 5 further makes explicit Equation 10 for the possible choices satisfying $0 \notin \mathrm{dom}(\nabla f)$. Finally, Table 6 provides the simplified expressions obtained after integrating the approximation $\tilde{Z}(x) = 0$ proposed by MOD (Shi et al., 2024).

Table 4: Overview of common f-divergence functions and their gradients.

f-divergence	$f(t)$	$\nabla f(t)$	$0 \notin \mathrm{dom}(\nabla f)$
α-divergence ($\alpha \in (0,1)$)	$(t^{1-\alpha} - (1-\alpha)t - \alpha)/(\alpha(\alpha-1))$	$(1-t^{-\alpha})/\alpha$	✓
Reverse KL ($\alpha = 0$)	$t \log t$	$\log t + 1$	✓
Forward KL ($\alpha = 1$)	$-\log t$	$-1/t$	✓
JS-divergence	$t \log t - (t+1)\log((t+1)/2)$	$\log(2t/(1+t))$	✓
Total Variation	$\frac{1}{2}\lvert t-1\rvert$	$\frac{1}{2}\,\mathrm{sgn}(t-1)$	✗
Chi-squared	$(t-1)^2$	$2(t-1)$	✗

Table 5: User-aligned policy under different f-divergence objectives.

f-divergence	Optimal Policy $\pi^u(y \mid x)$
α-divergence ($\alpha \in (0,1)$)	$\pi_{\mathrm{ref}}(y \mid x)\left(\alpha\tilde{Z}(x) + \sum_{i=1}^n w_i^u \left(\frac{\pi_{\mathrm{ref}}(y\mid x)}{\pi_i(y\mid x)}\right)^{\alpha}\right)^{-\frac{1}{\alpha}}$
Reverse KL ($\alpha = 0$)	$\left(\prod_{i=1}^n \pi_i(y \mid x)^{w_i^u}\right)\exp(-\tilde{Z}(x))$
Forward KL ($\alpha = 1$)	$\pi_{\mathrm{ref}}(y \mid x)\left(\tilde{Z}(x) + \sum_{i=1}^n \frac{w_i^u \pi_{\mathrm{ref}}(y\mid x)}{\pi_i(y\mid x)}\right)^{-1}$
JS-divergence	$\pi_{\mathrm{ref}}(y \mid x)\left(-1 + \exp(\tilde{Z}(x))\prod_{i=1}^n \left(\frac{\pi_{\mathrm{ref}}(y\mid x)}{\pi_i(y\mid x)} + 1\right)^{w_i^u}\right)^{-1}$

Table 6: Simplified user-aligned policy under the MOD (Shi et al., 2024) approximation.

f-divergence	Simplified Policy $\pi^u(y \mid x)$ ($\tilde{Z}(x) = 0$)
α-divergence ($\alpha \in (0,1)$)	$\left(\sum_{i=1}^n \frac{w_i^u}{\pi_i(y\mid x)^{\alpha}}\right)^{-\frac{1}{\alpha}}$
Reverse KL ($\alpha = 0$)	$\prod_{i=1}^n \pi_i(y \mid x)^{w_i^u}$
Forward KL ($\alpha = 1$)	$\left(\sum_{i=1}^n \frac{w_i^u}{\pi_i(y\mid x)}\right)^{-1}$
JS-divergence	$\pi_{\mathrm{ref}}(y \mid x)\left(-1 + \prod_{i=1}^n \left(\frac{\pi_{\mathrm{ref}}(y\mid x)}{\pi_i(y\mid x)} + 1\right)^{w_i^u}\right)^{-1}$

B PREFERENCE DATASETS

The Hugging Face repositories for the datasets used in our experiments are listed below:

- **PersonalLLM**: `namkoong-lab/PersonalLLM`
- **Roleplay**: `sher222/persona-iterative-responses`
- **PRISM**: `HannahRoseKirk/prism-alignment`
- **Community Alignment**: `facebook/community-alignment-dataset`

PersonalLLM RewardBench Models. To emulate a highly diverse user base, we select a subset of three reward models from the 10 PersonalLLM RewardBench models by evaluating their pairwise agreement rates. For each pair of reward models, the pairwise agreement rate is defined as the bidirectional average fraction of prompts where one model ranks the other's top output above its bottom output—among the 8 available candidate outputs. In Figure 4, we report the mean pairwise agreement rate for all $\binom{10}{3} = 120$ candidate triplets. We specifically choose the triplet with the lowest mutual agreement score (48%) to construct our preference simplex. The selected models are:

- `weqweasdas/RM-Gemma-7B`
- `PKU-Alignment/beaver-7b-v1.0-cost`
- `OpenAssistant/oasst-rm-2.1-pythia-1.4b-epoch-2.5`

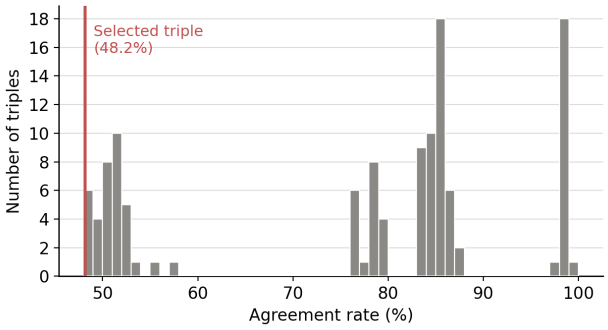


Figure 4: **Pairwise agreement distribution across candidate model triplets.** Distribution of mean agreement rates for all $\binom{10}{3} = 120$ possible RewardBench triplets. Highlighted is the selected triplet exhibiting the lowest mutual agreement (48%), which maximizes preference diversity across the simplex.

C TPD AND BASELINES IMPLEMENTATION DETAILS

All methods depend strictly on the base LM, requiring no external reward models or retrieval mechanisms.

C.1 DPO

DPO is a generic alignment method trained on the pooled preference data of all seen users, performing no adaptation to unseen users at test time. We implement DPO using the Hugging Face `TRL` (Transformer Reinforcement Learning) library (Von Werra et al., 2020), setting $\beta = 0.1$ as in the original DPO and TRL papers. We fine-tune the base model using a LoRA adapter (Hu et al., 2021) with standard parameters: rank $r = 8$, scaling $\alpha = 16$, and dropout 0.05. Complete training hyperparameters are listed in Table 7. `TPD` and all DPO variants inherit these training parameters. The variants include varying the f-divergence in the DPO loss formulation (reverse KL, JSD, or α-divergence with $\alpha = 0.5$) using TRL's f-divergence configuration. We also consider a *capacity-matched* baseline that multiplies the default LoRA rank r and scaling factor α by the number n of policies in `TPD` , ensuring both approaches utilize the exact same number of trainable parameters.

C.2 `TPD`

Training. Our training pipeline extends TRL's DPO trainer to accommodate our multi-adapter policy mixture and perform joint optimization over the adapter weights $\boldsymbol{\theta}$ and per-user logit vectors z^u (as detailed in Algorithm 1). We adopt a default policy basis size of $n = 3$ to balance expressivity against computational and memory overhead, reducing this to $n = 2$ on PRISM and Community Alignment to account for smaller dataset scale (see Table 8).

RAG-ICL System Instruction

```
Below are some examples of past conversations with liked and disliked
responses per prompt.
User: {prompt_1}
Liked Response: {response_1}
Disliked Response: {rejected_1}
User: {prompt_2}
Liked Response: {response_2}
Disliked Response: {rejected_2}
...
User: {prompt_k}
Liked Response: {response_k}
Disliked Response: {rejected_k}
Use the contexts above to generate a good response for the user
prompt below. Your response should be similar to the winning
responses and dissimilar from the losing responses.
```

Figure 5: RAG-ICL system prompt: the top-k retrieved examples of the user, each with its liked and disliked response, followed by the instruction.

Adaptation problem optimization. At test time, the adapters remain frozen, allowing the features $\psi_i(x, y_w, y_l)$ to be computed only once per adapter for each adaptation example of the new user. Problem 9 then reduces to a lightweight n-dimensional optimization over w^u, which we solve via full-batch gradient descent (using Adam) with learning rate $\eta_w^{\text{ada}} = 0.05$ on unconstrained logits z^u, setting $w^u = \text{softmax}(z^u)$ to enforce the simplex constraint by construction. Optimization is initialized at $z^u = \mathbf{0}$, corresponding to uniform weights $w^u = (\frac{1}{n}, \ldots, \frac{1}{n})$. Since this optimization operates strictly on cached features, we select the ridge regularization hyperparameter λ from the grid $\{0, 0.03, 0.1, 0.3, 1, 3\}$ via cross-validation at no additional computational cost—evaluating preference accuracy on held-out sets of preference examples—allowing the optimization to adapt across different user dataset sizes N.

C.3 P-DPO

P-DPO (Li et al., 2024) personalizes DPO by prepending learned user-specific embedding tokens to the prompt and training them jointly with a LoRA adapter on preference data from seen users, while unseen users at test time share a single generic embedding, without any adaptation to their data. Each user embedding is a learned convex combination of C shared cluster embeddings, each made of T tokens. We use the official Personalized-RLHF codebase with its cluster-based user model. Following the original paper's setup, we use $T = 10$ user tokens and $K = 10$ clusters (except on PersonalLLM where we use 3), setting $\beta = 0.5$ and $\alpha = 0.5$; the rest of the optimization recipe is shared with DPO.

C.4 ICL-BASED METHODS

We implement RAG-ICL in the spirit of PersonalLLM (Zollo et al., 2025). For seen users, we retrieve the $k = 3$ training preference pairs most similar to the test prompt from the user's own data, using cosine similarity over prompt embeddings extracted from the base LM's last-token hidden states. For unseen users, we first identify similar users by representing each user as the average difference between their chosen and rejected response embeddings. From this representation, we retrieve the 20 nearest seen users via cosine similarity. The k in-context shots are then retrieved based on prompt similarity from these users' pooled training pairs. Our instruction template follows PersonalLLM (Figure 5), placing the shots in the system message. Each shot presents the prompt alongside its winning (liked) and losing (disliked) responses, with all prompts and responses truncated to 512 tokens. Finally, PAG-ICL uses the same retrieved examples, but first prompts the model to generate a user profile from them (see Figure 6), conditioning generation on that profile alone.

PAG-ICL Pass 1/2: User Description Generation

```
User: {prompt_1}
Liked Response: {chosen_1}
Disliked Response: {rejected_1}

User: {prompt_2}
Liked Response: {chosen_2}
Disliked Response: {rejected_2}

...

# Task
Generate a user description based on the examples above.
```

PAG-ICL Pass 2/2: System Instruction

```
## User Description
{generated_description}
Use the user description above to generate a good response for the
user prompt below.
```

Figure 6: PAG-ICL prompts: the retrieved examples are first summarized into a user description (pass 1), which then serves as the system prompt for the final response (pass 2).

C.5 LoRe

LoRe (Bose et al., 2025) learns a low-rank basis of B reward features from seen user preferences and adapts the user-specific weights for unseen users at test time. We utilize the official LoRe codebase but adapt its embedding pipeline to ensure a fair comparison: while the original implementation relies on an external pretrained reward model to extract response embeddings, we extract them directly from our shared base LM. For a given (prompt, response) pair formatted with the model's standard chat template, the embedding is defined as the last-layer hidden state of the final token. Similar to our adaptation setup, these embeddings are pre-computed and cached prior to fitting. This enables highly efficient hyperparameter optimization via cross-validation to maximize preference accuracy on a held-out data split. Specifically, we tune the rank of the reward basis $B \in \{1, 2, 3, 4, 5, 10, 15, 20, 25, 50\}$, as well as the learning rates for the reward feature matrix, $\eta_A \in \{0.1, 0.5, 1.0\}$, and for the user-specific weights, $\eta_w \in \{0.001, 0.01, 0.1, 0.5, 1.0\}$. At test time, the learned personalized reward is combined with best-of-N sampling, with $N = 128$ candidates. This pipeline is similar to the approach proposed in (Shenfeld et al., 2025).

C.6 TPD and baselines Hyperparameters Summary

The training and adaptation hyperparameters of all methods are summarized in Table 7.

Generation parameters All generations are limited to 1024 new tokens. We use temperature sampling with the default generation parameters of each base model: Llama-3.2-3B-Instruct ($T = 0.6$, top-$p = 0.9$) and Qwen2.5-0.5B-Instruct ($T = 0.6$, top-$p = 0.95$, top-$k = 20$). The exceptions are TPD , which uses greedy decoding, and LoRe + BoN, which uses $T = 1.0$ to diversify the 128 candidates. To ensure that performance gains are not confounded by decoding strategy differences, we also evaluate a greedy decoding variant for the main DPO baseline.

D Details on the evaluation with synthetic personas and real users

This appendix provides extended details regarding our evaluation methodology. First, we describe the data filtering and train/test splitting procedures used to construct the seen and unseen user pools

Table 7: **Hyperparameters.** Values used for each method in all the experiments of the paper; braces list the values tried for swept hyperparameters.

Hyperparameter	Value
DPO parameters (shared by DPO, P-DPO, `TPD`)	
LoRA (r, α, dropout, targets)	8, 16, 0.05, q_proj/v_proj
Batch size	2
Gradient accumulation steps	8
Model dtype	bfloat16
Learning rate η_θ	5e-5
Optimizer	AdamW
Weight decay	0
Warmup steps	150
LR scheduler	cosine
Epochs	1
DPO β	0.1
P-DPO (Li et al., 2024)	
Number of user tokens T	10
Number of clusters K	$\{3, 10\}$
DPO β	0.5
P-DPO α	0.5
`TPD`	
Policy basis size n	$\{2, 3\}$
Embedding learning rate η_w	0.03
Adaptation optimizer	Adam
Adaptation iterations	100
Adaptation regularization λ	$\{0, 0.03, 0.1, 0.3, 1, 3\}$
Adaptation embedding learning rate η_w^{ada}	0.05
ICL parameters (shared by RAG-ICL, PAG-ICL)	
Retrieved shots k	3
Retriever max input tokens	512
Max tokens per in-context prompt / response	512
PAG-ICL	
Max new tokens (user-description pass)	512
LoRe (Bose et al., 2025)	
Number of basis reward functions B	$\{1, 2, 3, 4, 5, 8, 16\}$
Optimizer	Adam
Learning rate η_A (reward feature matrix)	{1e-3, 1e-2, 1e-1}
Iterations for A	2000
Learning rate η_w (user weights)	{1e-4, 1e-3, 1e-2}
Iterations for few-shot w	2000
BoN	
Best-of-N candidates	128

across all three datasets (Section D.1). Next, we detail our LLM-as-a-judge protocol (Section D.2), including the exact evaluation prompts, the specific user profile fields used to contextualize the judge, and a validation of the judge's agreement with ground-truth human preferences.

D.1 Preference Data

Data filtering and pre-processing. For **Community Alignment**, we keep the English-language portion and the annotators who wrote at least one prompt-level feedback, which leaves 545 annotators. Each best-of-four annotation is expanded into up to three pairs (the preferred response against

Table 8: **Dataset statistics.** Number of seen and unseen users, training pairs of the seen users and adaptation pairs of the unseen users, on average per user and in total.

	Roleplay	PRISM	Community Alignment
Users			
Seen	420	1,026	436
Unseen	280	261	109
Training pairs (seen users)			
Per user	776.3	25.5	225.2
Total	326,051	26,204	98,189
Adaptation pairs (unseen users)			
Per user	128.0	25.2	127.0
Total	35,840	6,578	13,842

each of the other three), which leads to 126,241 preference pairs. For **PRISM**, following (Bose et al., 2025), we select users with more than five dialogs containing at least one preferred and one rejected response, which leaves 1,287 users; each preferred response is paired with each rejected one shown in the same turn, which gives 39,337 preference pairs. For **Roleplay**, a single-turn dataset, we apply no filtering and keep all 700 personas and their 420,564 preference pairs. Preference examples are consistently truncated to at most 512 tokens for the prompt and 1024 tokens for the prompt and response together across all datasets.

Data and user splits. For **Community Alignment** and **PRISM**, we randomly split users 80/20 into seen and unseen users. For each user, we hold out 10% of their own sessions (dialogs for PRISM) as test data and use the remaining sessions as training data for seen users and as an adaptation pool for unseen users, capped at 128 pairs per user. Splitting at the level of whole sessions ensures that no turn from a test conversation appears in the prompt of a training example. For **Roleplay**, the original train and test personas already form disjoint seen and unseen populations; we hold out 10% of the questions for each seen persona as test data, and for each unseen persona, we use at most 128 pairs for adaptation and 50 for testing. Preference accuracies are computed over the full test sets of held-out pairwise comparisons, whereas win rates are computed using one prompt randomly drawn from each user's test set. The resulting numbers of users and examples are summarized in Table 8.

D.2 LLM-AS-A-JUDGE EVALUATION

Judge: System Instruction

```
You are a highly efficient assistant, who evaluates and selects the best large language
model (LLMs) based on how well their responses match the given user's preferences.
This process will be used to create a leaderboard reflecting the most personalized and
human-preferred answers.
```

Judge: User Instruction

```
I require a leaderboard for various large language models.  I'll provide you with
prompts given to these models, their corresponding outputs, and a description of the
user who asked.  Your task is to assess these responses, and select the model that
produces the best output for that specific user.
## Instruction
{
   "instruction":  """{instruction}""",
}
## Model Outputs
Here are the unordered outputs from the models.  Each output is associated with a
specific model, identified by a unique model identifier.
{
   {
      "model_identifier":  "X",
      "output":  """{output_X}"""
   },
   {
      "model_identifier":  "Y",
      "output":  """{output_Y}"""
   }
}
## User Description
{
   "user_description":  """{user_description}"""
}
## Task
Evaluate the models based on how well each output matches the user description above.
The user description may include feedback on previous LLM responses (e.g.  "Response
A/B/C/D") -- these do NOT correspond to the current "X"/"Y" labels.  Use the following
rule to decide:
1.  If response "X" is significantly better than response "Y" for this user, output "X".
2.  If response "Y" is significantly better than response "X" for this user, output "Y".
3.  If both responses are nearly similar, equivalent in quality, or differ only in minor
formatting/length, output "tie".
A response's length should not influence your judgment.  First write your reasoning,
then give the model identifier of the best model.  Respond with exactly the following
JSON format and nothing else:
{"reasoning":  "<your brief explanation>", "answer":  "<X, Y, or tie>"}
## Response
```

Figure 7: Prompt of the LLM judge, an AlpacaEval-style prompt augmented with user descriptions. Each pair is judged twice with the two responses swapped (X/Y), and counted as a tie unless both orders agree.

User descriptions. To condition the LM judge on each user's profile, we construct user descriptions using metadata fields available in the **Community Alignment, PRISM, Roleplay** datasets, as detailed in Table 9.

Judge configuration. We use Qwen3.5-35B-A3B (`https://huggingface.co/Qwen/Qwen3.5-35B-A3B`) as the judge, served with vLLM on two NVIDIA A6000 GPUs with a maximum context length of 8,704 tokens. We use deterministic greedy decoding (temperature 0, top-$p = 1$) with a budget of 1,024 generated tokens. The output is constrained to a JSON schema with a short reasoning field followed by the verdict (X, Y, or tie), and the judge runs with the model's thinking mode enabled. The full prompt, based on AlpacaEval-style templates (Dubois et al., 2024), is given in Figure 7.

Table 9: **Fields used for the user description given to the judge.** Field names correspond to the columns of the original datasets. The italic text gives example values taken from one user of each dataset, shortened with [...].

Dataset	Demographics	Stated preferences	Feedback

PRISM	`age`, `gender`, `employment_status`, `education`, `marital_status`, `english_proficiency`
	Age: 25-34 years old; Gender: Male; [...]
		`self_description`
		Respect everyone, even if they don't respect, treat the people like you want to be treated
		`system_string`
		Don't be a menace for anyone [...]
		`order_stated_prefs`
		(top 3)
		values, creativity, helpfulness
			`open_feedback`
			The idea is good but they need more information about music [...]

Community Alignment	`annotator_*` *Age: 55+; Gender: male; [...]*	—	`*_feedback` *Response A provided a story line and a visual hint [...]*
Roleplay	`persona_description` *A 45-year-old woman from Switzerland, a financial advisor who empowers women to take control of their finances and plan for their futures [...]*		

Table 10: **Agreement between the LLM judge and human preference labels.** Pairwise accuracy is reported with ties (100% − Disagree) and without ties (exact match). Values report mean ± standard error across users.

			Agreement Rate (%)	
Dataset	Users	Pairs	**w/ Ties**	**w/o Ties**
Roleplay	700	14,000	$88.4_{\pm 0.3}$	$60.4_{\pm 0.5}$
PRISM	1,287	37,901	$78.6_{\pm 0.3}$	$47.8_{\pm 0.3}$
Community Alignment	545	119,339	$91.2_{\pm 0.3}$	$70.7_{\pm 0.6}$
Average			$86.1_{\pm 0.2}$	$59.6_{\pm 0.3}$

Two-sided judging procedure. To mitigate position bias, each response pair is evaluated in both presentation orderings. A *win* (or *loss*) is assigned if and only if the personalized (or baseline) response is preferred in both orderings; inconsistent judgments are treated as ties. The win rate is then computed as $\frac{win}{win+loss}$.

Agreement between the LLM judge and human preferences. To evaluate the reliability of our LLM-as-a-judge setup in simulating user preferences, we assess its accuracy in predicting the pairwise preferences available in the three datasets (Table 10). More precisely, for each preference example (y_w, y_l), the judge (prompted with the corresponding user description) agrees with the human label with ties if it predicts $y_w \succeq y_l$ (y_w preferred or tied with y_l). Without ties, agreement strictly requires predicting $y_w \succ y_l$. Across 171k+ comparisons, the judge achieves an average agreement of $86.1\% \pm 0.2\%$ with ties and $59.6\% \pm 0.3\%$ without ties.

E Additional Results

Tables 11, 12, and 13, together with Tables 14, 15, and 16, respectively, provide the disaggregated results by seen and unseen users for Table 3 and Figure 3, in terms of win rate and preference accu-

racy. The sample sizes, including the numbers of seen and unseen users and training and adaptation pairs, are reported in Table 8.

Table 11: **Win rate (%) against the base model, win / (win + lose).** Values report mean ± standard error over users. *Overall* denotes the aggregated win rate across both seen and unseen users.

	Roleplay		
Method	Seen users	Unseen users	Overall
Generic			
DPO (reverse KL)	**60.7**±4.4	47.4±5.7	55.3±3.5
DPO (capacity-matched, reverse KL)	45.0±4.3	46.4±6.0	45.6±3.5
DPO (JSD)	51.8±4.8	51.2±5.6	51.6±3.6
DPO (α-div., α=0.5)	44.7±5.4	57.7±6.9	49.9±4.2
Personalized			
P-DPO	47.3±4.4	**70.7**±5.0	56.7±3.3
RAG-ICL	23.2±2.4	21.4±3.0	22.4±1.9
PAG-ICL	41.8±3.1	40.0±3.8	41.1±2.4
LoRe + BoN	45.8±4.2	65.7±4.7	53.8±3.1
TPD (reverse KL)	56.4±4.3	60.2±5.4	**57.9**±3.4
TPD (JSD)	49.2±4.4	63.1±5.3	54.8±3.4
TPD (α-div., α=0.5)	52.8±4.4	58.4±5.6	55.0±3.5

Table 12: **Win rate (%) against the base model, win / (win + lose).** Values report mean ± standard error over users. *Overall* denotes the aggregated win rate across both seen and unseen users.

	PRISM		
Method	Seen users	Unseen users	Overall
Generic			
DPO (reverse KL)	45.8±2.1	42.3±4.2	45.1±1.9
DPO (capacity-matched, reverse KL)	41.8±2.0	37.7±4.0	41.0±1.8
DPO (JSD)	44.6±2.1	44.8±4.0	44.7±1.8
DPO (α-div., α=0.5)	48.1±2.2	45.0±4.3	47.5±2.0
Personalized			
P-DPO	43.9±2.0	43.5±4.0	43.8±1.8
RAG-ICL	25.6±1.6	26.8±3.1	25.9±1.4
PAG-ICL	33.3±1.7	32.4±3.4	33.1±1.5
LoRe + BoN	**52.2**±2.0	43.9±4.1	50.5±1.8
TPD (reverse KL)	47.1±2.0	47.6±3.9	47.2±1.8
TPD (JSD)	45.1±2.0	48.4±3.9	45.8±1.8
TPD (α-div., α=0.5)	49.5±2.0	**54.9**±4.0	**50.6**±1.8

Table 13: **Win rate (%) against the base model, win / (win + lose).** Values report mean ± standard error over users. *Overall* denotes the aggregated win rate across both seen and unseen users.

	Community Alignment		
Method	Seen users	Unseen users	Overall
Generic			
DPO (reverse KL)	52.9±3.5	61.4±6.4	54.6±3.0
DPO (capacity-matched, reverse KL)	**59.0**±3.4	**68.5**±6.3	**60.9**±3.0
DPO (JSD)	58.9±3.5	57.4±6.3	58.6±3.1
DPO (α-div., α=0.5)	50.3±3.7	56.0±7.0	51.4±3.3
Personalized			
P-DPO	47.1±3.3	45.8±6.5	46.8±3.0
RAG-ICL	24.9±2.6	29.6±5.4	25.8±2.3
PAG-ICL	32.1±2.8	35.6±5.6	32.8±2.5
LoRe + BoN	55.6±3.3	40.8±7.0	52.6±3.0
TPD (reverse KL)	54.0±3.3	52.9±7.0	53.8±3.0
TPD (JSD)	52.5±3.5	55.6±6.3	53.1±3.1
TPD (α-div., α=0.5)	53.4±3.5	45.8±7.2	51.9±3.1

Table 14: **Preference accuracy (%) on held-out (chosen, rejected) pairs.** Values report mean ± standard error over users. *Overall* denotes the aggregated accuracy across both seen and unseen users. ICL-based baselines (RAG-ICL, PAG-ICL) are not reported, as they do not directly provide a reward model to score answers.

	Roleplay		
Method	Seen users	Unseen users	Overall
Generic			
DPO (reverse KL)	73.6±2.2	72.5±2.7	73.2±1.7
DPO (capacity-matched, reverse KL)	74.4±2.1	73.0±2.7	73.9±1.7
DPO (JSD)	73.6±2.2	72.5±2.7	73.1±1.7
DPO (α-div., α=0.5)	72.8±2.2	69.8±2.7	71.6±1.7
Personalized			
P-DPO	72.7±2.2	69.9±2.7	71.6±1.7
LoRe	74.5±2.1	69.8±2.7	72.6±1.7
TPD (reverse KL)	**75.7**±2.1	**75.0**±2.6	**75.5**±1.6
TPD (JSD)	75.5±2.1	74.5±2.6	75.1±1.6
TPD (α-div., α=0.5)	73.5±2.2	70.5±2.7	72.3±1.7

Table 15: **Preference accuracy (%) on held-out (chosen, rejected) pairs.** Values report mean ± standard error over users. *Overall* denotes the aggregated accuracy across both seen and unseen users. ICL-based baselines (RAG-ICL, PAG-ICL) are not reported, as they do not directly provide a reward model to score answers.

	PRISM		
Method	Seen users	Unseen users	Overall
Generic			
DPO (reverse KL)	61.9±1.5	59.9±3.0	61.5±1.4
DPO (capacity-matched, reverse KL)	61.9±1.5	59.8±3.0	61.5±1.4
DPO (JSD)	61.8±1.5	60.4±3.0	61.5±1.4
DPO (α-div., α=0.5)	61.3±1.5	59.7±3.0	61.0±1.4
Personalized			
P-DPO	59.4±1.5	57.4±3.1	59.0±1.4
LoRe	60.1±1.5	60.0±3.0	60.1±1.4
TPD (reverse KL)	**62.0**±1.5	**62.0**±3.0	**62.0**±1.4
TPD (JSD)	61.2±1.5	**62.0**±3.0	61.3±1.4
TPD (α-div., α=0.5)	60.5±1.5	60.6±3.0	60.5±1.4

Table 16: **Preference accuracy (%) on held-out (chosen, rejected) pairs.** Values report mean ± standard error over users. *Overall* denotes the aggregated accuracy across both seen and unseen users. ICL-based baselines (RAG-ICL, PAG-ICL) are not reported, as they do not directly provide a reward model to score answers.

	Community Alignment		
Method	Seen users	Unseen users	Overall
Generic			
DPO (reverse KL)	65.3±2.3	69.1±4.4	66.0±2.0
DPO (capacity-matched, reverse KL)	65.2±2.3	**69.7**±4.4	**66.1**±2.0
DPO (JSD)	65.1±2.3	68.4±4.5	65.8±2.0
DPO (α-div., α=0.5)	63.6±2.3	66.5±4.5	64.2±2.1
Personalized			
P-DPO	64.1±2.3	69.0±4.4	65.1±2.0
LoRe	64.1±2.3	64.7±4.6	64.2±2.1
TPD (reverse KL)	**65.6**±2.3	67.5±4.5	66.0±2.0
TPD (JSD)	65.2±2.3	68.2±4.5	65.8±2.0
TPD (α-div., α=0.5)	63.7±2.3	66.9±4.5	64.4±2.1