Rethinking Personalized Generation: Test-Time Alignment via Factorized Ranking Models

Qiyao Ma **Junshan Zhang**[†] **Zhe Zhao**[†]
University of California, Davis
{qiyma,jazh,zao}@ucdavis.edu

Code: https://github.com/Martin-qyma/Rethinking-Personalized-Generation

Abstract

Aligning large language models (LLMs) to diverse user preferences is fundamentally hindered by standard alignment paradigms that optimize for monolithic users. In this work, empirical studies are first used to reveal the existence of a massive, untapped performance headroom for personalized generation through test-time alignment. We demonstrate that personalized generation is uniquely suited for test-time scaling methods like Best-of-N (BoN) because it can be viewed primarily as a candidate matching problem rather than a generator capability bottleneck. While reward models could in principle exploit this headroom, they are poorly calibrated for personalization, and their billion-parameter scale makes scoring large candidate pools prohibitively expensive. To overcome this limitation, we propose a parameter-efficient framework utilizing million-parameter scale multi-layer perceptron (MLP) ranking models. Our personalized ranking model directly reuses the internal embeddings of the base generator with minimal overhead. By scaling train-time data to provide fine-grained personalized preferences, this million-parameter ranking model accurately scores large candidate pools and can seamlessly guide generation to reduce the cost of materializing N candidates. Extensive experiments on nine datasets spanning three personalized generation settings show that our personalized ranking model effectively exploits the discovered headroom, outperforming billion-parameter generalist reward models on every dataset, with under 0.4% of their parameters and four orders of magnitude lower scoring latency.

1 Introduction

As large language models (LLMs) are deployed in increasingly subjective settings, from writing in a user's voice to explaining why an item suits them, the assumption behind standard alignment begins to bind. Reinforcement Learning from Human Feedback (RLHF) and its variants [20, 1] have proven highly effective at instilling general alignment, but they do so under the assumption of a single, monolithic set of human preferences. Human preferences are pluralistic [31, 27], and a monolithic reward averages away exactly the idiosyncratic choices that make a response read as if it were written for a particular person [18, 40, 9, 22].

The prevailing response treats this as a capability deficit of the generator and repairs it at training time, by finetuning per user or learning to adapt the generator from user data [10, 32]. We start from a different question: Does the base model already have the capability of producing well-personalized responses that we are failing to pick out? Figure 1 answers with an oracle experiment. When the evaluation metric itself selects the best of N sampled candidates, alignment with the user's own reference keeps improving as the pool grows; in contrast, when a state-of-the-art reward model selects

[†]Equal Advising.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

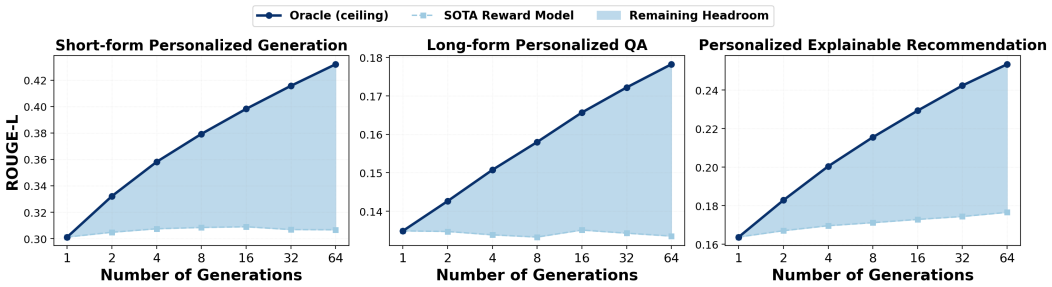


Figure 1: Personalization headroom under Best-of-N sampling. The oracle selects with the evaluation metric itself and upper-bounds any selector; the reward-model line is, at each N, the best of the four generalist reward models of Section 4.1. The shaded region is the gap between them. Results are averaged over the three datasets of each task.

instead, it plateaus almost immediately and captures only a small fraction of the available gain. The oracle shows that well-matched candidates are already in the pool, whereas naively using the reward model cannot find them: **the capability is in the generator, and the bottleneck is selection**.

Exploiting this headroom with a standard reward model is costly in two ways. A reward model is a separate billion-parameter transformer, so scoring a pool means re-encoding the same prompt and every candidate from scratch: selection cost grows with N, with response length, and with the reward model's width, and in our measurements scoring a pool with an 8B reward model costs about as much as generating it. Beyond cost, generalist reward models are trained to rank responses by universal qualities, and Figure 1 shows this is the wrong yardstick for a pool whose candidates are all fluent and all on topic but differ in whose voice they are written in.

We therefore decouple selection from language understanding. The generator has already read the query, the user profile, and each candidate, and its final-layer hidden states carry that understanding; we read those states directly and train a small multi-layer perceptron (MLP) of 1M to 30M parameters to map them to a scalar score. No text is re-encoded and no per-user parameters are learned. Because the ranker operates in a fixed feature space, it cannot learn a representation of its own; whatever it learns about a user's choices must come from the supervision. We build that supervision from the generator itself. For each training prompt we sample a pool of candidates and label every candidate with its ROUGE-L against the reference the user actually wrote. The pool supplies realistic hard negatives, responses that are well formed but not what this user would have written, and pointwise regression on these labels calibrates the ranker for exactly the selection task it faces at test time. The loss itself is not the point: pairwise and listwise objectives perform comparably.

With selection nearly free, the remaining cost of Best-of-N is the pool itself: every candidate must be fully generated before it can be scored. This raises the question of whether the ranker can act before the pool exists. Because it scores candidates from the generator's own hidden states, it can steer decoding directly: when the generator is uncertain about the next token, we nudge its choice toward what the ranker predicts the user would prefer. This recovers part of the selection gain at the cost of a single generation; most of the headroom still requires scoring a pool, so we present guided decoding as a complement to selection rather than a replacement for it.

We evaluate on nine datasets spanning three personalized generation settings: short-form personalized generation [29], long-form personalized question answering [28], and personalized explainable recommendation [17]. Our ranker has a few million parameters; the reward models it is compared against have eight billion. Against four generalist reward models, it is nonetheless the best selector on every dataset and keeps improving with the pool size where they plateau or decline. We then finetune the best of those reward models per dataset on the same candidate pools with the same objective, which sets the ceiling a model of that size can reach on this task. The finetuned model is ahead, as it should be, but the ranker recovers most of its gain with a model more than two hundred times smaller, at orders of magnitude lower latency, and without finetuning a billion-parameter model for every dataset. Without any preference labels, it also matches personalized reward models that fit per-user parameters from labeled comparisons, and its selections transfer to a metric it was never trained on.

In summary, our main contributions are summarized as follows:

- **Empirical Discovery of Personalization Headroom:** An oracle Best-of-N analysis shows that the base generator already produces well-personalized candidates while generalist reward models recover only a small fraction of the gain, reframing personalized alignment as a candidate selection problem.
- **Introduce Personalized Ranking Model:** A million-parameter MLP that reuses the generator's own hidden states in place of a billion-parameter reward model. Trained on candidate pools with realistic negatives, it outperforms generalist reward models over two hundred times its size and extends to single-pass guided decoding.
- **Comprehensive Experimental Validation:** Extensive evaluations across nine datasets and three paradigms demonstrate that our million-parameter MLP recovers most of the gain of explicitly finetuned, billion-parameter SOTA reward models at a fraction of their size and inference cost, and matches personalized reward models without per-user preference labels.

2 Related Work

Generalized Alignment and Reward Modeling. RLHF [20, 1] and DPO [26] align LLMs to a monolithic notion of preference through reward models trained with the Bradley-Terry objective [3]. This averages out pluralistic tastes, and a Bradley-Terry reward is identified only up to a prompt-dependent shift, so its scores are not calibrated across the large candidate pools that test-time selection must compare [41]. We instead frame personalization as candidate matching and train a lightweight ranking model on the generator's own embeddings with a pointwise objective. The objective itself is not a contribution: pairwise and listwise alternatives perform comparably under the same architecture and data (Appendix B.1).

Personalized Generation. Early work injects persona descriptions or retrieved history into the prompt [38, 29] or fine-tunes per user [10, 32], treating personalization as a generator deficit. A second line personalizes the reward model: PAL [9], VPL [22], PReF [30], and LoRe [2] place each user in a low-dimensional preference space that is fit or inferred from labeled pairwise comparisons, while GPO [40] and SynthesizeMe [27] adapt an LLM judge from a user's labeled examples. All of them require per-user preference annotations and, as deployed, a separate encoder pass per candidate. Our ranker needs neither: it derives the user signal from the interaction history already present in the generation context and recycles the generator's hidden states. Appendix B.2 compares against all six on XRec, the setting in which per-user supervision can be constructed.

Test-Time Alignment. Scaling inference compute can rival scaling model size [19, 37]. Best-of-N sampling and guided decoding [35, 11, 36, 8, 12] rely on a reward model to select or steer and inherit its per-candidate encoding cost; even the lightest of them, FUDGE's future discriminators, re-encode the partial sequence with their own network rather than reading the generator's states, and all target a single monolithic reward rather than an individual user. UserAlign [21] and T-POP [24] personalize at inference time but elicit preferences through interactive pairwise queries cast as bandits, whereas our ranker issues no queries and needs no labels beyond the user's existing history. Our ranker brings the selection cost to near zero by scoring the generator's own hidden states, and because it is differentiable in those states it can also steer decoding directly.

3 Methodology

In this section, we formalize the personalized test-time alignment problem and introduce our parameter-efficient, decoupled ranking framework. We begin by defining the core objective of Best-of-N sampling and identifying the severe computational bottlenecks inherent in standard reward models. Subsequently, we detail our lightweight ranking architecture and the preference optimization strategy tailored specifically for learning personal preferences. We conclude with an asymptotic efficiency analysis to theoretically validate the massive scalability of our approach. An overview of our methodology is illustrated in Figure 2.

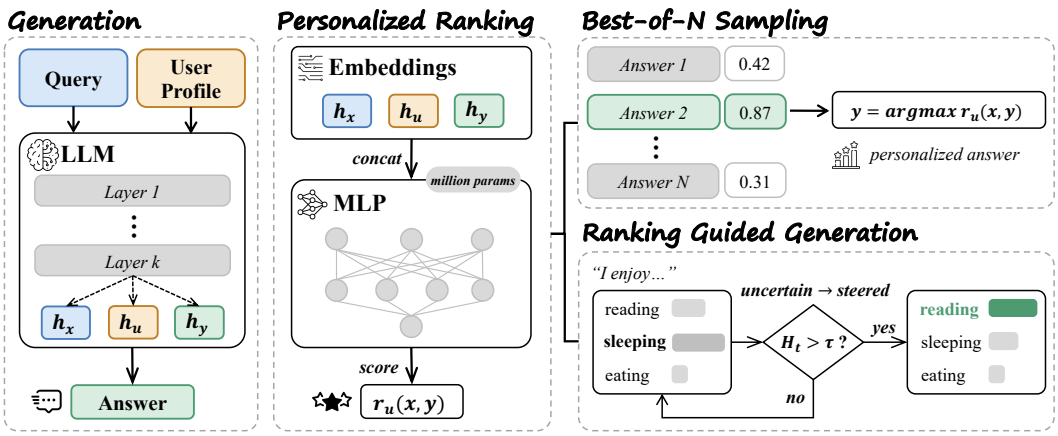


Figure 2: Overview of our decoupled personalized ranking framework. Our personalized ranking model directly recycles the final hidden states of query, user profile, and answer from the LLM generator. This parameter-efficient model is deployed in two inference settings: Best-of-N Sampling and Ranking Guided Generation, where it steers the decoding trajectory when the token distribution entropy (H_t) exceeds a predefined threshold (τ).

3.1 Problem Formulation: Personalized Test-time Alignment

Let U denote a set of users, X the space of input prompts, and Y the space of generated responses. In standard generalized alignment, the goal is to optimize a policy $\pi_\theta(y|x)$ that maximizes a globally shared reward. However, in personalized alignment, the objective shifts to finding an optimal user-specific policy $\pi^*(y|x,u)$ that caters to the idiosyncratic preferences of user $u \in U$.

Instead of actively finetuning the base policy π_θ for every user, test-time alignment employs Best-of-N (BoN) sampling. Given a prompt x, the base generator samples a diverse candidate pool of N responses, denoted as $C = \{y_1, y_2, \ldots, y_N\}$. The objective is to learn a personalized scoring function, $\hat{r}_u(x,y)$, which identifies the optimal candidate y^* that maximizes the user's implicit utility:

$$y^* = \arg\max_{y_i \in C} \hat{r}_u(x, y_i) \tag{1}$$

In standard paradigms, $\hat{r}_u(x,y)$ is parameterized as a massive LLM. Executing Equation 1 requires N independent, full forward passes through this heavyweight encoder, creating a severe computational bottleneck that limits the feasible size of N and, consequently, the ability to exploit the personalization headroom.

3.2 Personalized Ranking Model

Decoupling Understanding from Scoring. To eradicate the redundancy of secondary encoding, we decouple the language understanding component from the preference scoring component. The base LLM generator has already comprehended the prompt x and synthesized the response y_i, so its internal representations already contain dense, high-quality semantic features. Rather than re-encoding the text, our framework directly reads the generator's own final-layer hidden states.

Recycled Embeddings. Three texts take part in personalized generation: the task query x, the user profile u, and the sampled response y_i. The profile is the user's interaction history as provided by the benchmark: retrieved history items for LaMP and LaMP-QA, and a pre-written user summary for XRec. For each of the three texts we take the generator's final-layer hidden state at its last token, denoted $h_x, h_u, h_{y_i} \in \mathbb{R}^d$, where d is the hidden dimension (the last hidden token in Figure 2). Appendix C specifies the profile text used for each benchmark and how the embeddings are extracted in our implementation.

User Embeddings. The user embedding h_u involves no separate user encoder and no per-user learned parameters. It is the generator's own summary of the profile text, so the same frozen model

that writes the response also represents the user. Because h_x and h_u depend only on the prompt, they are computed once per query and shared across all N candidates.

Ranking Heads. We map the three vectors to a scalar utility through a small MLP f_ϕ with L hidden layers:

$$\hat{r}_u(x, y) \;=\; f_\phi\big(h_x, h_u, h_y\big) \;=\; \mathbf{w}^\top g_L \circ g_{L-1} \circ \cdots \circ g_1\big([h_x \,\|\, h_u \,\|\, h_y]\big), \tag{2}$$

where $\|$ denotes concatenation, each block g_ℓ is the activation layer, and $\mathbf{w}$ is the final scalar projection. All configurations are trained from scratch with the same protocol. Because the semantic heavy lifting is offloaded to the LLM generator, the ranking model only needs to learn which candidate in a user-specific pool comes closest to what that user would have written. This is a regression problem in a fixed feature space rather than a representation-learning problem.

3.3 Pointwise Preference Optimization

Objective The ranking model f_ϕ is agnostic to the training objective: any differentiable ranking loss can be applied to the recycled embeddings of Section 3.2. We adopt pointwise regression by default. Let $s^*(x, y_i, u) \in \mathbb{R}$ denote the target utility of candidate y_i for user u. We minimize

$$\mathcal{L}_{\text{MSE}}(\phi) = \mathbb{E}_{(x,y_i,u)\sim\mathcal{D}}\left[\big(f_\phi(h_x, h_u, h_{y_i}) - s^*(x, y_i, u)\big)^2\right] \tag{3}$$

which maps every candidate onto a shared, absolute utility scale.

Target Standardization. In practice we standardize s^* within each candidate pool, to zero mean and unit variance over the N candidates of a prompt. Best-of-N selection is invariant to per-prompt shifts, so this focuses the regression on within-pool discrimination rather than on predicting each prompt's overall difficulty.

Why Pointwise? Two properties of our setting favor pointwise regression. First, our supervision is cardinal rather than ordinal: each candidate carries an absolute target, whereas pairwise or listwise objectives are invariant to monotone transformations of the targets and therefore discard how much better one candidate is than another. Second, Best-of-N selection and the guided decoding of Section 3.4 consume absolute utilities that must be comparable across large pools, whereas pairwise scores are identified only up to prompt-dependent shifts. Pointwise training also scales as $\mathcal{O}(N)$ per prompt rather than $\mathcal{O}(N^2)$ for all-pairs objectives.

Alternative Objectives. Alternatives include the pairwise Bradley-Terry objective standard in reward modeling [3] and RankNet [4], and listwise objectives such as ListNet [7], ListMLE [34], and LambdaRank [5]. Appendix B.1 compares these objectives under identical architecture, data, and protocol. All objectives perform comparably, so pointwise MSE is a well-founded default rather than a requirement, and the gains of our framework stem from embedding recycling and train-time candidate scaling rather than from the choice of loss.

Train-Time Data Scaling. To ensure this minimal architecture achieves high discriminative power, data construction is paramount. Rather than utilizing randomly sampled negative responses, we populate the training distribution $\mathcal{D}$ with candidates directly generated by the base LLM generator π_θ. The target utility $s^*(x, y_i, u)$ is the downstream evaluation metric of y_i against the reference authored by user u (ROUGE-L in our experiments). This strategy yields hard negatives: responses that are structurally sound and factually correct, but stylistically or semantically misaligned with the specific user u, and therefore receive low targets. Training the ranking model to minimize the MSE over these hard negatives bridges the representation gap typically associated with low-parameter models.

3.4 Ranking Guided Generation

Best-of-N sampling pays for N full decodings. We instead use the trained personalized ranking model as a single-pass decoding-time signal: the LLM is run once, and the ranking model only intervenes when the model is genuinely uncertain about the next token.

Confidence Gate. Let $\boldsymbol{\ell}_t \in \mathbb{R}^V$ denote the next-token logits produced by the LLM generator at decoding step t, where V is the vocabulary size. We measure the LLM generator's uncertainty at step t by the next-token entropy

$$H_t \;=\; -\sum_{v \in \mathcal{V}} p_t(v)\, \log p_t(v), \qquad p_t \;=\; \mathrm{softmax}(\boldsymbol{\ell}_t). \tag{4}$$

With threshold τ, we leave confident steps unchanged:

$$\text{if } H_t \le \tau: \quad y_t \;=\; \arg\max_v \boldsymbol{\ell}_t. \tag{5}$$

Steering on Uncertain Tokens. When $H_t > \tau$, we ask the ranking model which direction in hidden space would increase its predicted reward, and project that direction onto the vocabulary through the LLM head. Concretely, let $W \in \mathbb{R}^{V \times d}$ denote the LLM head weight matrix, where V is the vocabulary size and d is the hidden dimension. We then compute:

$$\hat{r}_u(x, y) \;=\; f_\phi\big(h_x, h_u, h_y\big), \qquad g_t \;=\; \nabla_{\mathbf{h}_t} \hat{r}_u(x, y) \;\in\; \mathbb{R}^d, \qquad \mathbf{u}_t \;=\; W\, g_t \;\in\; \mathbb{R}^V, \tag{6}$$

and emit the argmax of the blended logits

$$\boldsymbol{\ell}'_t \;=\; \boldsymbol{\ell}_t \;+\; \alpha_t\, \mathbf{u}_t, \qquad y_t \;=\; \arg\max_v \boldsymbol{\ell}'_t. \tag{7}$$

The step size is entropy-modulated so that the steering is gentle on borderline-uncertain tokens and stronger on near-uniform distributions:

$$\alpha_t \;=\; \alpha_0 \cdot \mathrm{clip}\left(\frac{H_t - \tau}{H_{\max} - \tau},\, 0,\, 1\right). \tag{8}$$

The chosen token is appended to the sequence and decoding proceeds to the next step. Appendix C.3 reports how τ, $H_{\max}$, and α_0 were chosen and how sensitive the results are to them.

3.5 Time Complexity Analysis

The primary advantage of our decoupled architecture lies in its asymptotic test-time efficiency.

Reward Model Selection. Consider a standard Best-of-N scenario where candidate responses have an average sequence length of L. For a standard heavyweight reward model, selecting the best candidate requires passing the prompt and response through the full transformer stack N times. The time complexity for the selection phase scales as $\mathcal{O}(N \cdot L \cdot d_r^2)$, where d_r is the hidden dimension of the reward model. As N grows to exploit the personalization headroom, this linear scaling with respect to sequence length and full model dimensionality becomes prohibitively expensive.

Embedding-Recycling Selection. In contrast, our proposed framework operates with an $\mathcal{O}(1)$ amortized encoding cost. The generation of the N embeddings occurs naturally during the candidate generation phase. The subsequent ranking step merely requires a forward pass through the shallow MLP for each candidate. The time complexity for selection is thus reduced to $\mathcal{O}(N \cdot d^2)$, completely independent of the sequence length L, where d is the hidden dimension of the LLM generator. Because $d^2 \ll L \cdot d_r^2$, the latency overhead introduced by our ranker is virtually negligible. This theoretical efficiency allows our method to scale to large candidate pools in real-time deployments with negligible computational cost; Appendix C.4 reports measured latency and memory.

4 Experiments

4.1 Experimental Settings

Generation and Evaluation. We use *Qwen2.5-7B-Instruct* [25] as the generative backbone. Following the benchmark protocols, we evaluate with ROUGE-L [13] against the gold reference, and use ROUGE-1 as a check where noted. In all nine datasets the reference is authored by the target user, so lexical overlap with it measures alignment with that user's revealed stylistic choices rather than generic quality; semantic metrics such as BERTScore [39] abstract away precisely the surface-level nuances in which personalization manifests. Because the ranker is trained on ROUGE-L targets, we additionally report BLEU [23] in Section 4.3 as a held-out metric, verifying that its gains reflect learned user preference rather than regression to the training metric.

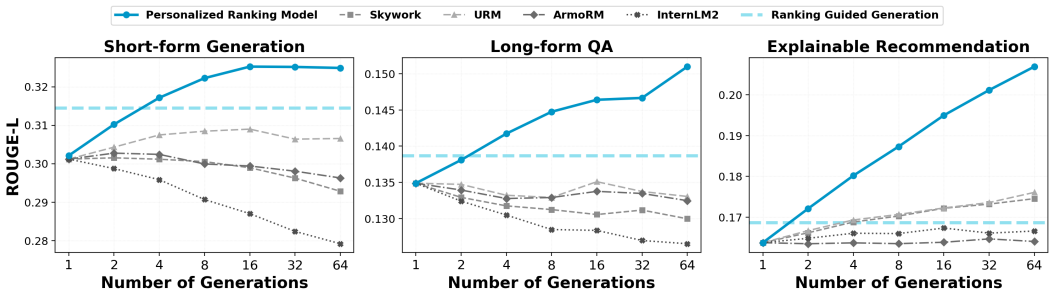


Figure 3: Best-of-N selection with our personalized ranking model and each generalist reward model, averaged over the three datasets of each task. Dashed line: ranking guided generation.

Ranking Baselines. We evaluate our approach using several state-of-the-art discriminative reward models (RMs) as ranking baselines:

- `Skywork-Reward-V2-Llama-3.1-8B` [14]: A data-centric RM trained on the Skywork-Reward Preference 80K dataset, which focuses on high-quality curated pairs and utilizes a robust vanilla Bradley-Terry (BT) loss to maximize reward differences.
- `internlm2-7b-reward` [6]: A model trained on a large-scale corpus of 2.4 million preference samples (human and AI-synthesized), providing comprehensive coverage across diverse areas including dialogue, coding, and mathematics.
- `URM-LLaMa-3.1-8B` [16]: An uncertainty-aware reward model that implements attribute-specific value heads to predict the parameters of a normal distribution, allowing for the estimation of reward reliability.
- `ArmoRM-Llama3-8B-v0.1` [33]: An Absolute-Rating Multi-Objective RM that employs a Mixture-of-Experts (MoE) gating mechanism to scalarize 19 distinct reward dimensions (e.g., helpfulness, correctness, and verbosity) into a single preference score.

Every reward model scores exactly the input our ranker sees: the generation prompt, which carries the query and the user's retrieved profile, together with the candidate response. Personalization signal is therefore available to these models in context, and what we test is whether generically trained preference models can exploit it when ranking large candidate pools. Section 4.3 adds a second, stronger tier in which the best of them is finetuned on our personalized training distribution.

Datasets. Our evaluation spans nine datasets across three distinct personalization categories:

1. **Short-form Personalized Generation (LaMP)** [29]: We focus on three generative tasks requiring models to adapt to user historical profiles: personalized news headline generation (News), scholarly title generation (Scholarly), and tweet paraphrasing (Tweet).
2. **Long-form Personalized QA (LaMP-QA)** [28]: This benchmark evaluates informational seeking tasks across three categories: Arts & Entertainment, Lifestyle & Personal Development, and Society & Culture, using personalized rubrics extracted from user narratives.
3. **Personalized Explainable Recommendation (XRec)** [17]: We utilize three domain-specific datasets: Amazon, Yelp and Google, to generate natural language explanations for user-item interactions based on collaborative signals and user profiles.

The size of the user profiles shipped with these benchmarks varies by more than an order of magnitude, from a median of eight interactions per user on XRec to a median of 143 article–headline pairs on News; following each benchmark's protocol, only a small retrieved subset (three items for LaMP, six for LaMP-QA) or a pre-written summary (XRec) enters the prompt. Appendix C.1 gives self-contained task descriptions, and Table 6 the per-dataset profile statistics.

Implementation Details. The personalized ranking model f_ϕ is an MLP over the concatenated (h_x, h_u, h_y) with GELU activations and dropout 0.1. Width-depth pairs $(H, L) \in \{(96, 3), (256, 3), (1024, 3), (2048, 4)\}$ give the $\{1.0, 2.8, 12.1, 30.4\}$M-parameter sizes used in our

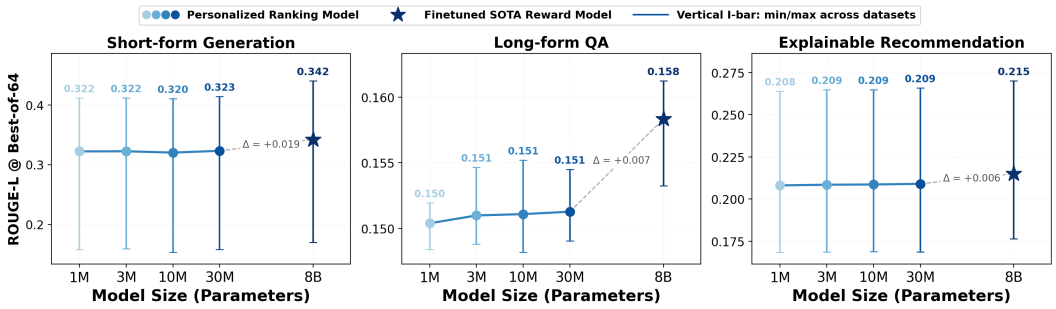


Figure 4: Personalized ranking model size against a finetuned reward model at Best-of-64. Points are means over the three datasets of each task, bars span the per-dataset minimum and maximum.

scaling analysis, which we round to 1M, 3M, 10M, and 30M; the 2.8M model is the default elsewhere. We train with AdamW [15] at learning rate 5×10^{-4} and weight decay 0.01 for 15 epochs at batch size 64, with gradient clipping at 1.0 and a single seed; the same trainer produces every ranker size and every appendix ablation (Appendix C.2). Each prompt contributes 64 on-policy candidates sampled at temperature 1.0, labeled with ROUGE-L against the gold reference. At inference, ranking guided generation uses $\tau = 1$, $H_{\max} = 8$, $\alpha_0 = 5$, and a 3-token warmup; these values were fixed a priori and shared across all nine datasets. Appendix C provides further training details, the sensitivity analysis of the decoding hyperparameters, and measured inference costs. Appendix A expands every aggregated figure below to all nine datasets, and Appendix B collects the additional analyses referenced in this section.

4.2 Main Results

Personalization Headroom. Oracle Best-of-N selection picks the candidate with the evaluation metric itself and therefore upper-bounds any selector. Against it we place, at each N, the best of the four generalist reward models of Section 4.1. Figure 1 shows both averaged per task; Appendix A.1 gives all nine datasets (Figure 5). The oracle rises steadily up to $N = 64$ on every task, with no sign of saturation, so the generator keeps producing better-matched candidates as the pool grows. The best reward model does not follow. It plateaus by $N = 16$ on short-form generation and never beats its $N = 1$ score on long-form QA. At $N = 64$ it captures at most 23% of the headroom, and under 12% on eight of the nine datasets. Because one curve keeps climbing while the other stays flat, the gap between them widens with N, and every additional sample adds headroom that a generalist selector fails to exploit. This has three implications. First, much of the personalization we seek does not need to be trained into the generator. A frozen model conditioned on the user's profile already produces well-matched responses, and what is missing is the ability to recognize them. Second, test-time compute becomes a lever for personalization, but only if the selector's accuracy scales with the pool. That requires a selector calibrated on the fine, user-specific distinctions among candidates that are all fluent and on topic, rather than on the generic quality differences that generalist reward models are trained to detect. Third, exploiting this headroom requires scoring large pools, so the cost of scoring each candidate becomes the limiting factor. This is what motivates a selector that is both personalized and nearly free to run. The capability is already in the generator; the bottleneck is selection.

Best-of-N Selection. Figure 3 compares selectors as N grows from 1 to 64 (per dataset and per reward model in Figure 6, Appendix A.2). The two families of selectors follow opposite trends. The generalist reward models stay flat or decline as the pool grows, improving only slowly on explainable recommendation. For them, a larger pool is not an opportunity but a risk. Because they reward generic qualities rather than fit to the user, each added candidate is one more chance to select a response that looks good in general but is wrong for this user. Our ranker instead improves steadily with N on every task, and its advantage over the reward models widens as the pool grows. The first takeaway is that whether test-time compute helps personalization depends entirely on the selector: the same pools yield steady gains under a calibrated ranker and none under a generalist one. The shape of the curve also varies with the task. Gains saturate early on short-form generation, where candidates of a few words differ little from one another, but keep rising on long-form QA and explainable recommendation, where longer responses leave more room for variation. The second takeaway is

Table 1: Agreement between ROUGE-L and BLEU at the candidate level, computed on the evaluation prompts of each dataset. Within-pool statistics correlate the two metrics over the 64 candidates of a prompt and are averaged over prompts; "same argmax" is the fraction of pools in which both metrics select the same best candidate; the across-prompt column correlates the per-prompt mean scores.

	Short-form Generation			Long-form QA			Explainable Recommendation		
Statistic	News	Scholarly	Tweet	Art	Lifestyle	Society	Amazon	Yelp	Google
Within-pool Spearman	0.548	0.545	0.578	0.385	0.405	0.410	0.594	0.389	0.363
Within-pool Pearson	0.529	0.520	0.614	0.394	0.418	0.434	0.638	0.404	0.383
Same argmax	31%	31%	37%	17%	12%	18%	28%	17%	19%
Across-prompt Pearson	0.778	0.496	0.647	0.564	0.484	0.699	0.743	0.685	0.688

that pool size should be matched to the task, with small pools sufficient for short outputs and larger ones worthwhile for open-ended responses. We attribute the difference to training on the generator's own candidate pools, which teaches the ranker the fine-grained distinctions that selection actually requires.

Ranking Guided Generation. The dashed line marks ranking guided generation, which decodes one response per prompt while the ranker steers uncertain tokens. It reaches the level of Best-of-2 to Best-of-4 of the sampled pool on short-form generation and long-form QA, and of Best-of-1 to Best-of-2 on explainable recommendation, for the cost of a single generation; per dataset it exceeds Best-of-64 only on Tweet and falls below Best-of-1 on News (Figure 6). Steering a single trajectory thus recovers a modest part of the selection gain without materializing candidates, while most of the headroom still requires scoring a pool. We conduct a sensitivity analysis by varying each decoding hyperparameter around its default (Appendix C.3). ROUGE-L changes by at most about 0.005 on every dataset, so the shared defaults need no per-dataset tuning. Only τ has a visible effect, and it controls how often the ranker intervenes rather than the final score.

4.3 More Analysis

Scaling Effect and Parameter Efficiency. We next ask how much ranker capacity is needed and how the ranker compares with a reward model finetuned for the task. Figure 4 varies the ranker from 1M to 30M parameters and adds *Skywork-Reward-V2-Llama-3.1-8B* finetuned on the same training pools with the same pointwise objective (Appendix C.2); points are means over the three datasets of each task with bars spanning the per-dataset range, and Appendix A.3 gives each dataset (Figure 7). Appendix B.2 further compares against personalized reward models that fit user-specific parameters from per-user preference labels.

Ranker size barely matters: from 1M to 30M parameters the Best-of-64 score moves by at most 0.006 ROUGE-L on any dataset, because the semantic work is done by the generator and the ranker only learns a regression in a fixed feature space. The finetuned 8B model is ahead on all nine datasets, by $\Delta = +0.019$ ROUGE-L on short-form generation and below $+0.01$ on long-form QA and explainable recommendation; measured against the headroom, the 30M ranker recovers 45% to 94% of the finetuned model's gain per dataset. This gap is the price of decoupling. The 8B model re-encodes every candidate and must be finetuned per dataset, whereas the ranker scores recycled embeddings at four orders of magnitude lower cost per query (Appendix C.4); for that cost it delivers the bulk of what finetuning a billion-parameter model buys.

Cross-Metric Generalization. Because the ranker is trained on ROUGE-L targets, one may ask whether it merely regresses the training metric. If so, its advantage should vanish under a metric it never saw. Table 4 therefore evaluates the Best-of-64 selection of every ranker under BLEU, keeping all rankers fixed; ours remains trained solely on ROUGE-L labels. Our ranker selects the highest-BLEU candidate on all nine datasets, and the four generalist reward models trail it everywhere, most clearly on explainable recommendation, where our ranker improves on the strongest baseline by 20% to 64% relative.

This transfer is not trivial. Table 1 quantifies how differently ROUGE-L and BLEU order the candidates of a pool. The two metrics select the same best candidate in only 12–37% of pools, and their within-pool rank correlation is 0.36–0.59, so a ranker that had memorized the ROUGE-L

Table 2: BoN ROUGE-L (N=64) stratified by the amount of history each user actually has. The selector is the paper-default ranker, so differences across buckets reflect history available to the pipeline, not a change of model. †Buckets with fewer than 10 users, reported for completeness only.

	Short-form Generation			**Long-form QA**			**Explainable Recommendation**		
History size	News	Scholarly	Tweet	Art	Lifestyle	Society	Amazon	Yelp	Google
<10	0.1498	–	0.4104	–	–	–	0.2640	0.1904	0.1675
10–24	0.1832	–	0.4091	0.1535	0.1470	0.1528	0.2659	0.1958	0.1729
25–49	0.1954	0.3693	0.4197	0.1455†	0.1430	0.1440	–	–	–
50–99	0.1952	0.3900	0.4462	0.1477	0.1524	0.1556	–	–	–
100+	0.1385	0.4103	0.4781†	0.1620	0.1532	0.1407	–	–	–

scoring function would gain little under BLEU. The across-prompt correlation is higher (0.48–0.78), which reflects shared difficulty between prompts rather than agreement on which candidate to pick, and it is the within-pool disagreement that matters for selection. The ranker has therefore learned which candidate comes closest to what the user would have written, a property that transfers across surface metrics. The pattern mirrors the main results: generalist reward models fall short not because the evaluation is n-gram based, but because they lack calibration for user-authored references, a conclusion now corroborated under a metric none of the rankers was tuned toward.

User History Influence. Since the user profile is the only source of personalization signal, we examine how the amount of available history affects the pipeline. Table 6 summarizes the number of historical interactions per user on the evaluation prompts of each dataset, together with the profile text that enters the generation prompt and h_u. History richness spans more than an order of magnitude, from a median of eight recorded interactions on XRec to 143 article–headline pairs on News, and is highly dispersed within datasets as well. The main results therefore aggregate over heterogeneous history regimes. Following each benchmark's protocol, only a retrieved subset (three BM25-retrieved items for LaMP, six for LaMP-QA) or the pre-written summary (XRec) enters the prompt and h_u, never the full history. There is no minimum history and no per-user adaptation step.

Table 2 buckets evaluation users by their number of historical interactions and reports the Best-of-64 ROUGE-L of the default ranker within each bucket. Because the model is unchanged, differences across buckets reflect only the history available to the pipeline. Bucket support mirrors Table 6: XRec users, with at most 22 recorded interactions, fall into the two lowest buckets, Tweet users concentrate below 25 interactions, and Scholarly users concentrate above. Two patterns emerge. First, performance generally rises with available history, most clearly on Tweet (0.410 to 0.478), Scholarly (0.369 to 0.410), and all three XRec datasets. The long-form QA columns are flatter, with no consistent direction, and News is the exception, dropping for users with 100 or more pairs. Second, degradation under sparsity is graceful rather than catastrophic, as the sparsest users still retain most of the benefit of selection. We therefore read the direction of the trend as robust and the ordering of adjacent buckets as within noise.

5 Conclusion

We challenged the assumption that personalized LLM alignment requires expensive train-time intervention. An oracle Best-of-N analysis showed that the base generator already produces well-personalized candidates but generalist reward models fail to select them, which reframes personalized alignment as a candidate selection problem. To solve it at low cost, we introduced a million-parameter MLP ranker that scores candidates directly from the generator's own hidden states. Trained on the generator's own candidate pools labeled against user-authored references, it learns the fine-grained distinctions that personalized selection requires. The ranker outperforms generalist 8B reward models on all nine datasets and keeps improving as the pool grows where they stall. It approaches an 8B reward model finetuned per dataset at four orders of magnitude lower scoring latency, and it is competitive with personalized reward models without any per-user preference labels. It can also steer decoding directly, recovering part of the selection gain from a single generation. Since it attaches to any generator that exposes its hidden states, it offers a practical route to per-user alignment without retraining the generator. Our evaluation rests on lexical overlap with user-authored references, and a human study of style consistency is left for future work.

Acknowledgement

This work is supported in part by NSF Grants OAC-2531126, CNS-2504762, POSE-2449194, and ECCS-2413529, ARO Grant W911NF-2410046, and a Google Research Award.

References

[1] 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.

[2] Avinandan Bose, Zhihan Xiong, Yuejie Chi, Simon Shaolei Du, Lin Xiao, and Maryam Fazel. Lore: Personalizing llms via low-rank reward modeling. *arXiv preprint arXiv:2504.14439*, 2025.

[3] RALPH ALLAN BRADLEY and MILTON E. TERRY. Rank analysis of incomplete block designs: The method of paired comparisons. *Biometrika*, 39(3-4):324–345, 12 1952. ISSN 0006-3444. doi: 10.1093/biomet/39.3-4.324. URL https://doi.org/10.1093/biomet/39.3-4.324.

[4] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 89–96, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102363. URL https://doi.org/10.1145/1102351.1102363.

[5] Chris J.C. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical Report MSR-TR-2010-82, June 2010. URL https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/.

[6] Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, Aijia Guo, Qipeng Guo, Conghui He, Yingfan Hu, Ting Huang, Tao Jiang, Penglong Jiao, Zhenjiang Jin, Zhikai Lei, Jiaxing Li, Jingwen Li, Linyang Li, Shuaibin Li, Wei Li, Yining Li, Hongwei Liu, Jiangning Liu, Jiawei Hong, Kaiwen Liu, Kuikun Liu, Xiaoran Liu, Chengqi Lv, Haijun Lv, Kai Lv, Li Ma, Runyuan Ma, Zerun Ma, Wenchang Ning, Linke Ouyang, Jiantao Qiu, Yuan Qu, Fukai Shang, Yunfan Shao, Demin Song, Zifan Song, Zhihao Sui, Peng Sun, Yu Sun, Huanze Tang, Bin Wang, Guoteng Wang, Jiaqi Wang, Jiayu Wang, Rui Wang, Yudong Wang, Ziyi Wang, Xingjian Wei, Qizhen Weng, Fan Wu, Yingtong Xiong, Chao Xu, Ruiliang Xu, Hang Yan, Yirong Yan, Xiaogui Yang, Haochen Ye, Huaiyuan Ying, Jia Yu, Jing Yu, Yuhang Zang, Chuyu Zhang, Li Zhang, Pan Zhang, Peng Zhang, Ruijie Zhang, Shuo Zhang, Songyang Zhang, Wenjian Zhang, Wenwei Zhang, Xingcheng Zhang, Xinyue Zhang, Hui Zhao, Qian Zhao, Xiaomeng Zhao, Fengzhe Zhou, Zaida Zhou, Jingming Zhuo, Yicheng Zou, Xipeng Qiu, Yu Qiao, and Dahua Lin. Internlm2 technical report, 2024.

[7] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 129–136, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273513. URL https://doi.org/10.1145/1273496.1273513.

[8] Souradip Chakraborty, Soumya Suvra Ghosal, Ming Yin, Dinesh Manocha, Mengdi Wang, Amrit Singh Bedi, and Furong Huang. Transfer q-star: Principled decoding for llm alignment. *Advances in Neural Information Processing Systems*, 37:101725–101761, 2024.

[9] Daiwei Chen, Yi Chen, Aniket Rege, and Ramya Korlakai Vinayak. Pal: Pluralistic alignment framework for learning from heterogeneous preferences. *arXiv preprint arXiv:2406.08469*, 2024.

[10] Chuanyang Jin, Jing Xu, Bo Liu, Leitian Tao, Olga Golovneva, Tianmin Shu, Wenting Zhao, Xian Li, and Jason Weston. The era of real-world human interaction: Rl from user conversations. *arXiv preprint arXiv:2509.25137*, 2025.

[11] Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. Args: Alignment as reward-guided search. *arXiv preprint arXiv:2402.01694*, 2024.

[12] Bolian Li, Yifan Wang, Anamika Lochab, Ananth Grama, and Ruqi Zhang. Cascade reward sampling for efficient decoding-time alignment. *arXiv preprint arXiv:2406.16306*, 2024.

[13] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[14] Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*, 2024.

[15] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[16] Xingzhou Lou, Dong Yan, Wei Shen, Yuzi Yan, Jian Xie, and Junge Zhang. Uncertainty-aware reward model: Teaching reward models to know what is unknown. *arXiv preprint arXiv:2410.00847*, 2024.

[17] Qiyao Ma, Xubin Ren, and Chao Huang. Xrec: Large language models for explainable recommendation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 391–402, 2024.

[18] Qiyao Ma, Dechen Gao, Rui Cai, Boqi Zhao, Hanchu Zhou, Junshan Zhang, and Zhe Zhao. Personalized rewardbench: Evaluating reward models with human aligned personalization. *arXiv preprint arXiv:2604.07343*, 2026.

[19] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332, 2025.

[20] 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.

[21] Victor-Alexandru Pădurean, Parameswaran Kamalaruban, Nachiket Kotalwar, Alkis Gotovos, and Adish Singla. Inference-time personalized alignment with a few user preference queries. *Advances in Neural Information Processing Systems*, 38:85125–85156, 2026.

[22] Sriyash Poddar, Yanming Wan, Hamish Ivison, Abhishek Gupta, and Natasha Jaques. Personalizing reinforcement learning from human feedback with variational preference learning. *Advances in Neural Information Processing Systems*, 37:52516–52544, 2024.

[23] Matt Post. A call for clarity in reporting bleu scores. In *Proceedings of the third conference on machine translation: Research papers*, pages 186–191, 2018.

[24] Zikun Qu, Min Zhang, Mingze Kong, Xiang Li, Zhiwei Shang, Zhiyong Wang, Yikun Ban, Shuang Qiu, Yao Shu, and Zhongxiang Dai. T-pop: Test-time personalization with online preference feedback. *arXiv preprint arXiv:2509.24696*, 2025.

[25] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

[26] 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.

[27] 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)*, pages 8045–8078, 2025.

[28] Alireza Salemi and Hamed Zamani. Lamp-qa: A benchmark for personalized long-form question answering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1139–1159, 2025.

[29] 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)*, pages 7370–7392, 2024.

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

[31] Taylor Sorensen, Jared Moore, Jillian Fisher, Mitchell Gordon, Niloofar Mireshghallah, Christopher Michael Rytting, Andre Ye, Liwei Jiang, Ximing Lu, Nouha Dziri, et al. A roadmap to pluralistic alignment. *arXiv preprint arXiv:2402.05070*, 2024.

[32] Zhaoxuan Tan, Zixuan Zhang, Haoyang Wen, Zheng Li, Rongzhi Zhang, Pei Chen, Fengran Mo, Zheyuan Liu, Qingkai Zeng, Qingyu Yin, et al. Instant personalized large language model adaptation via hypernetwork. *arXiv preprint arXiv:2510.16282*, 2025.

[33] Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10582–10592, 2024.

[34] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1192–1199, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390306. URL https://doi.org/10.1145/1390156.1390306.

[35] Yuancheng Xu, Udari Madhushani Sehwag, Alec Koppel, Sicheng Zhu, Bang An, Furong Huang, and Sumitra Ganesh. Genarm: Reward guided generation with autoregressive reward model for test-time alignment. *arXiv preprint arXiv:2410.08193*, 2024.

[36] Kevin Yang and Dan Klein. Fudge: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535, 2021.

[37] Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, et al. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*, 2025.

[38] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, 2018.

[39] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

[40] Siyan Zhao, John Dang, and Aditya Grover. Group preference optimization: Few-shot alignment of large language models. *arXiv preprint arXiv:2310.11523*, 2023.

[41] Banghua Zhu, Michael Jordan, and Jiantao Jiao. Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. In *International Conference on Machine Learning*, pages 43037–43067. PMLR, 2023.

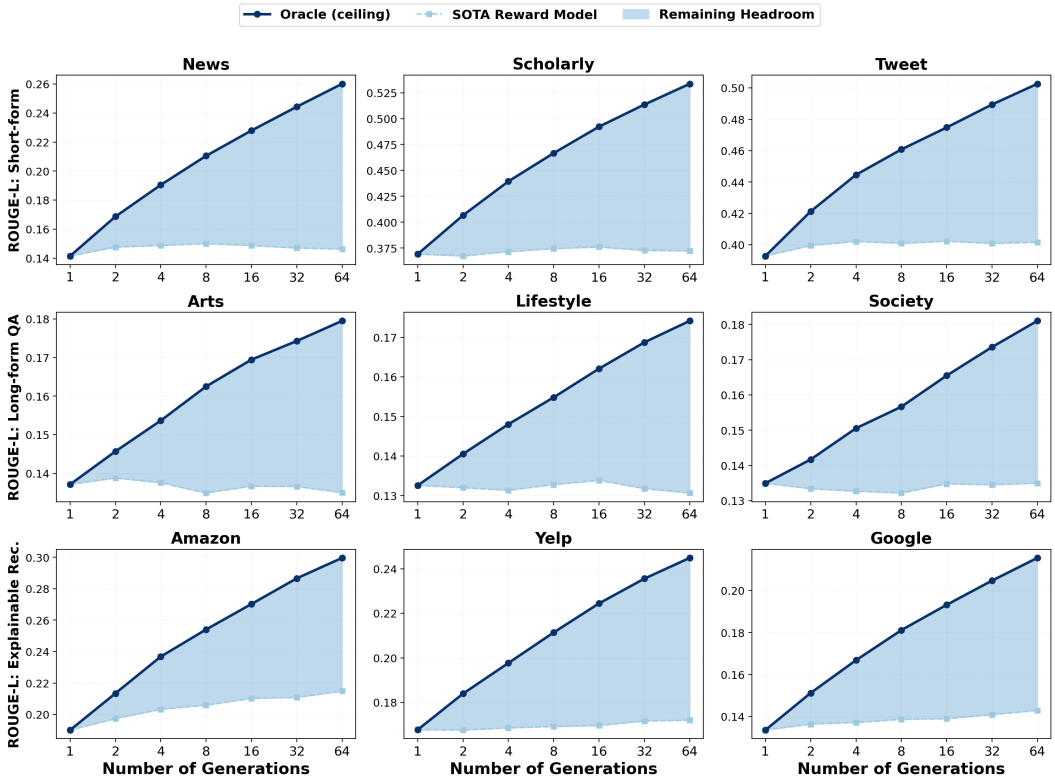


Figure 5: Personalization headroom under Best-of-N sampling ($N \leq 64$, ROUGE-L) on each of the nine datasets. The SOTA reward model line is the per-N maximum over the four generalist reward models, and the shaded region is its gap to the Oracle ceiling.

A Full Per-Dataset Results

This appendix expands the aggregated figures of Section 4 to all nine datasets.

A.1 Personalization Headroom Analysis

In Section 4.2, we presented the aggregated performance headroom across three distinct personalized generation tasks. To provide a more granular view of the test-time scaling behavior, Figure 5 details the full Best-of-N scaling trajectories for all nine individual datasets.

Across all nine datasets the pattern of Section 4.2 holds. The Oracle ceiling (solid dark blue line) rises monotonically as the candidate pool N expands from 1 to 64, whereas the strongest generalized reward model (dashed light blue line; the per-dataset maximum over the four reward models of Section 4.1) plateaus or declines. At $N = 64$ it captures 4%, 2%, and 8% of the oracle headroom on News, Scholarly, and Tweet and 23%, 6%, and 11% on Amazon, Yelp, and Google, and -5%, -5%, and 0% on Art, Lifestyle, and Society, where its Best-of-64 pick is no better than a single sample (ROUGE-L; on ROUGE-1 the shares range from -9% to 22%). The base LLM thus produces well-matched candidates in every domain, and generalist reward models are miscalibrated for picking them out of a large pool.

A.2 Baseline Comparisons Across Datasets

Expanding upon the aggregated results presented in the main text, Figure 6 provides a comprehensive dataset-level breakdown of Best-of-N scaling performance against individual state-of-the-art reward models (`Skywork-Reward-V2`, `URM-LLaMa-3.1`, `ArmoRM-LLaMa3`, and `InternLM2-Reward`).

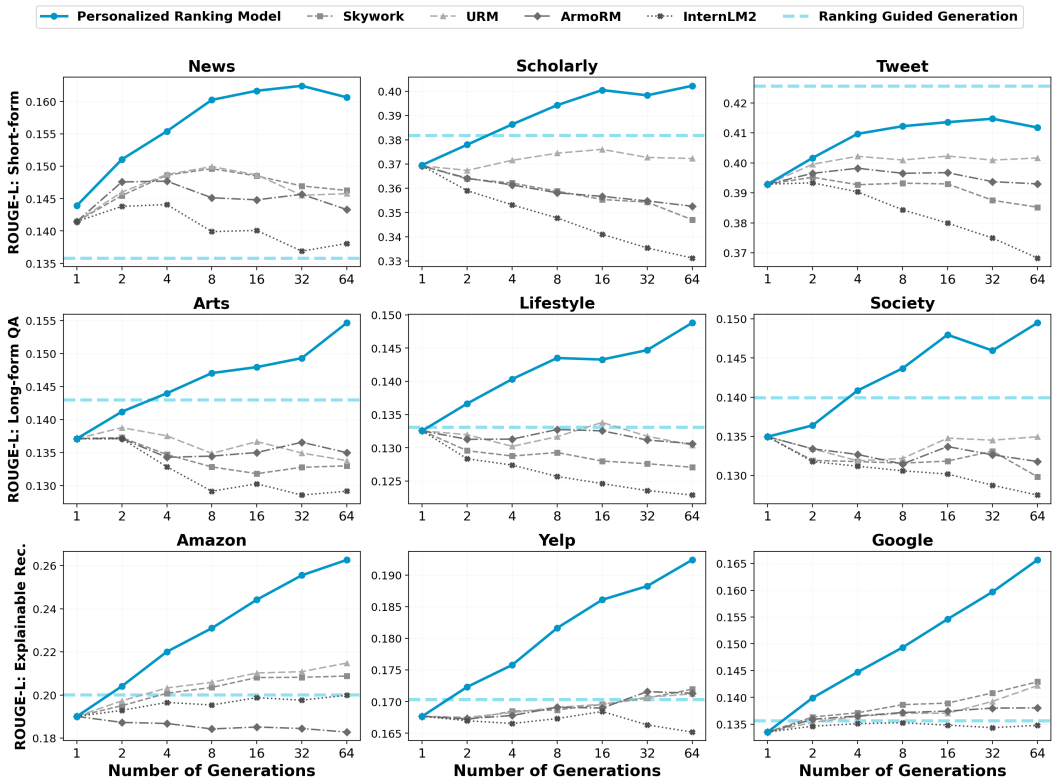


Figure 6: Best-of-N selection (ROUGE-L) on each of the nine datasets with our Personalized Ranking Model and each of the four generalist reward models. The dashed horizontal line is ranking guided generation, one greedily decoded response per prompt (Appendix C.3).

Consistent Degradation of General Reward Models. On the short-form datasets the reward models are flat on News (within 0.005 ROUGE-L of their $N = 1$ score) and decline on Scholarly and Tweet, where `InternLM2-Reward` loses 0.038 and 0.025 between $N = 1$ and $N = 64$ and `Skywork-Reward-V2` loses 0.022 on Scholarly; only `URM` gains, by at most 0.009. On the three long-form QA datasets all four models end at or below their $N = 1$ score. Explainable recommendation is the one task on which generalist models benefit from a larger pool: `Skywork-Reward-V2` and `URM` gain 0.019 and 0.025 on Amazon and up to 0.009 on Yelp and Google, while `ArmoRM` and `InternLM2-Reward` stay within 0.01 of $N = 1$. Even there the best reward model captures no more than 23% of the oracle headroom.

Superiority of the Personalized Ranker and Guided Generation. Our Personalized Ranking Model (solid blue line) is the best selector at $N = 64$ on all nine datasets, ahead of the best reward model by 0.010 (Tweet) to 0.048 (Amazon) ROUGE-L, and it improves with N on every dataset; on News, Scholarly, Tweet, and Society the curve saturates around $N = 16$ to 32 and fluctuates by up to 0.003 thereafter. The dashed horizontal line is ranking guided generation, one greedily decoded response per prompt (Appendix C.3). It lies between Best-of-1 and Best-of-4 of our ranker on seven datasets, above Best-of-64 on Tweet (0.426 against 0.412) and below Best-of-1 on News (0.136 against 0.144), so steering a single trajectory recovers only a small part of the selection gain.

A.3 Scaling Analysis and Parameter Efficiency

Figure 7 shows, for each dataset, the Best-of-64 ROUGE-L of our Personalized Ranking Model at 1M, 3M, 10M, and 30M parameters and of the finetuned 8B reward model of Appendix C.2.

Ranker size. The ranker is insensitive to its size: across 1M to 30M parameters the score moves by at most 0.006 ROUGE-L (News) and by less than 0.002 on Scholarly, Lifestyle, Society, Amazon,

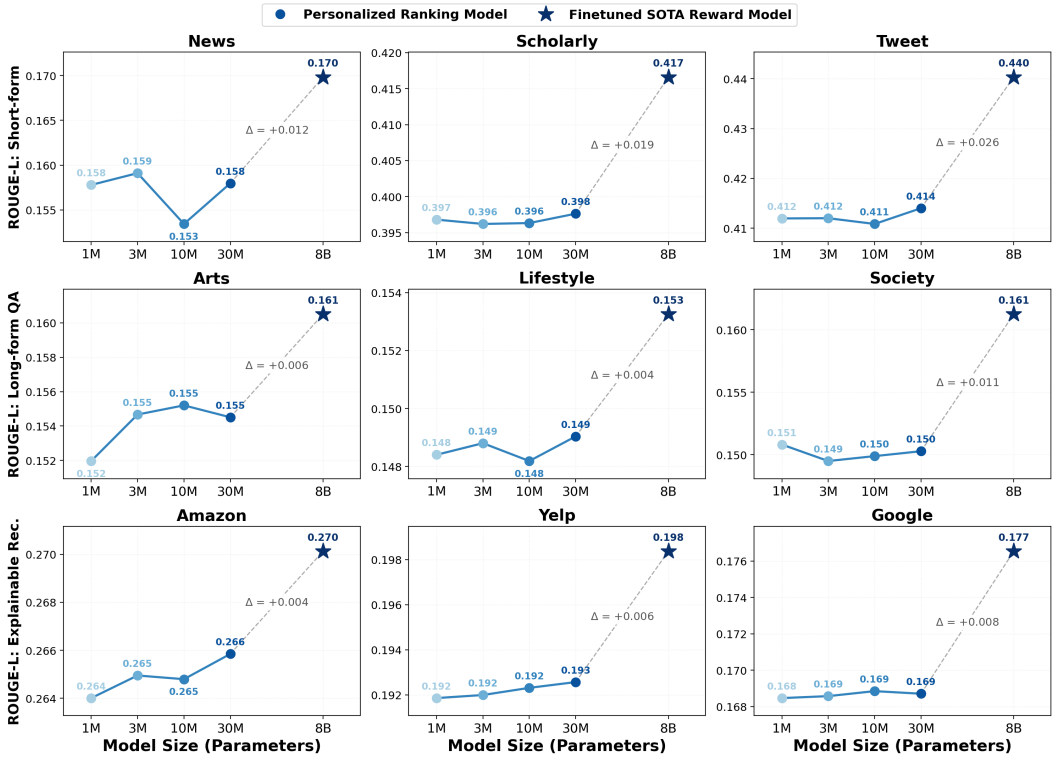


Figure 7: Ranker size against the finetuned 8B reward model at Best-of-64 (ROUGE-L) on each dataset. Δ is the 8B score minus the 30M score.

Yelp, and Google, and the ordering of the four sizes is not consistent across datasets, so we treat the differences as noise and use the 2.8M model by default.

Finetuned 8B reward model. The finetuned model is ahead on every dataset: Δ (8B minus 30M) is $+0.012$, $+0.019$, and $+0.026$ on News, Scholarly, and Tweet, $+0.006$, $+0.004$, and $+0.011$ on Art, Lifestyle, and Society, and $+0.004$, $+0.006$, and $+0.008$ on Amazon, Yelp, and Google. Measured against the headroom between a single sample and the oracle, the 30M ranker captures 12% to 66% and the 8B model 22% to 70%, so the ranker recovers 45% (Tweet) to 94% (Amazon) of the finetuned model's gain at roughly 10^{-4} of its scoring latency and without finetuning an 8B model per dataset (Appendix C.4).

B Additional Analyses

This appendix reports the ablations and comparisons referenced in the main text: the training objective and personalized reward-model baselines.

B.1 Ranking Objectives

Table 3 retrains the identical 2.8M-parameter MLP ranker under the pairwise and listwise objectives discussed in Section 3.3, keeping the architecture, training data, and budget fixed (Appendix C.2). MSE regresses to the ROUGE-L target standardized within each candidate pool; Bradley–Terry is trained on sampled candidate pairs and RankNet on all pairs within a pool; ListNet, ListMLE, and LambdaRank consume the ordering of each pool induced by the targets. No single objective dominates: MSE is best on News, ListMLE on Tweet, Lifestyle, and Society, the pairwise objectives on Scholarly, Art, Yelp, and Google, and ListNet on Amazon. The spread across objectives is small, at most 0.012 ROUGE-L on any dataset and below 0.003 on the three explainable recommendation datasets. The objectives that fall furthest behind on the three short-form datasets and on Art and Lifestyle are ListNet and LambdaRank, which were designed for graded ordinal relevance in retrieval.

Table 3: Ranking-objective ablation across all nine datasets. Identical MLP architecture, training data, and budget; only the training objective changes. Cells report BoN ROUGE-L at N=64. **Bold** marks the best objective per dataset.

	Short-form Generation			Long-form QA			Explainable Recommendation		
Objective	News	Scholarly	Tweet	Art	Lifestyle	Society	Amazon	Yelp	Google
MSE	**0.1591**	0.3962	0.4120	0.1547	0.1488	0.1495	0.2650	0.1920	0.1686
Bradley–Terry	0.1545	0.3980	0.4122	**0.1558**	0.1492	0.1512	0.2664	0.1928	**0.1694**
RankNet	0.1549	**0.3981**	0.4116	0.1557	0.1491	0.1515	0.2663	**0.1931**	0.1691
ListNet	0.1506	0.3936	0.4085	0.1502	0.1482	0.1526	**0.2675**	0.1923	0.1691
ListMLE	0.1551	0.3967	**0.4135**	0.1557	**0.1492**	**0.1527**	0.2670	0.1930	0.1691
LambdaRank	0.1483	0.3949	0.4021	0.1531	0.1465	0.1526	0.2661	0.1919	0.1687

Table 4: Cross-metric generalization: BLEU of the Best-of-N (N=64) selected candidate across all nine datasets. Every ranker is fixed (ours remains trained solely on ROUGE-L labels); only the evaluation metric changes. **Bold** marks the best selector per dataset.

	Short-form Generation			Long-form QA			Explainable Recommendation		
Method	News	Scholarly	Tweet	Art	Lifestyle	Society	Amazon	Yelp	Google
Ours	**0.0387**	**0.0671**	**0.1150**	**0.0259**	**0.0217**	**0.0277**	**0.0822**	**0.0417**	**0.0337**
Skywork	0.0330	0.0528	0.1050	0.0202	0.0166	0.0248	0.0498	0.0347	0.0274
InternLM2	0.0297	0.0503	0.0990	0.0205	0.0148	0.0235	0.0446	0.0332	0.0243
URM	0.0326	0.0593	0.1105	0.0214	0.0168	0.0251	0.0500	0.0347	0.0272
ArmoRM	0.0316	0.0544	0.1083	0.0225	0.0170	0.0247	0.0370	0.0347	0.0256

We draw two conclusions. Pointwise MSE is a well-founded default in our setting, since it uses the magnitude information in the cardinal labels and scales linearly in the pool size, but pairwise objectives are not inherently suboptimal here and perform on par. More importantly, the narrow spread confirms that the gains of the framework stem from embedding recycling and train-time candidate scaling rather than from the loss, which we do not claim as a contribution.

B.2 Personalized Reward Model Baselines

Table 5 compares our ranker with reward models that personalize by fitting or inferring user-specific parameters from labeled preference comparisons of the target user: PAL [9], VPL [22], PReF [30], LoRe [2], GPO [40], and SynthesizeMe [27]. We evaluate on XRec, the only family in which such supervision can be constructed: its users recur across interactions, whereas every LaMP and LaMP-QA prompt belongs to a distinct user with a history but no preference labels, so for those tasks the strongest feasible heavyweight comparator is the reward model finetuned on the personalized training distribution reported in Section 4.3.

Protocol. We follow the official implementations and preserve the mechanism that defines each method, namely how its per-user parameters are obtained. For each evaluation user we hold out up to eight of their training interactions and mine labeled comparison pairs from the on-policy candidate pools of those interactions, with the chosen response being the candidate with higher ROUGE-L against that user's reference. PAL, PReF, and LoRe fit their per-user vectors on these pairs by gradient descent or regularized logistic regression, with their shared prototypes or reward bases trained on the training-user population; VPL encodes the same pairs with its pair encoder to infer its latent user embedding; GPO meta-learns across users and receives the user's labeled context interactions in context at test time; SynthesizeMe induces a persona from the same pairs and selects with a persona-prompted pairwise judge (Llama-3.1-8B-Instruct) run as a single-elimination tournament over the pool. PAL, VPL, PReF, and LoRe operate on the same frozen generator embeddings as our ranker at a matched parameter budget of about 2.8M, with the official default configurations: two prompt-conditioned prototypes with a cosine ideal-point score for PAL, sixteen basis reward features for PReF, a rank-8 basis for LoRe, and the official 128-dimensional, six-layer transformer for GPO; SynthesizeMe runs the released package with a persona search budget of five to ten candidates per user. All methods select from the same Best-of-64 candidate pools and are evaluated on each user's held-out test interaction. Our ranker receives only the content profile and zero preference labels.

Table 5: Comparison with personalized reward-model baselines on XRec (BoN ROUGE-L, N=64). All baselines are instantiated per user from that user's own labeled data, whereas ours uses only the content profile (zero preference labels). Ours is retrained under the seen-user protocol of this comparison, which is why it differs slightly from Table 3.

Dataset	Ours	VPL	PAL	PReF	LoRe	GPO	SynthesizeMe
Amazon	0.2641	0.2628	0.2596	0.2598	0.2503	0.2690	0.1962
Yelp	0.1906	0.1916	0.1902	0.1791	0.1798	0.1957	0.1687
Google	0.1685	0.1697	0.1614	0.1526	0.1544	0.1743	0.1381

Table 6: Historical interactions per user on the evaluation prompts of each dataset (median, mean, 10th and 90th percentiles, maximum), and the profile text that enters the generation prompt and h_u. XRec counts are lower bounds from the interactions present in our splits; the benchmark reports 18–25 interactions per user.

Dataset	**History unit**	**Median**	**Mean**	**p10**	**p90**	**Max**	**Profile text entering the prompt and h_u**
News	article–headline pairs	143	172.4	11	442	634	top-3 BM25-retrieved pairs
Scholarly	abstract–title pairs	80	99.4	53	167	913	top-3 BM25-retrieved pairs
Tweet	past tweets	13	17.6	9	29	226	top-3 BM25-retrieved tweets
Art	past questions	73	159.1	14	370	991	top-6 BM25-retrieved questions
Lifestyle	past questions	36	111.6	12	262	1488	top-6 BM25-retrieved questions
Society	past questions	57	115.8	13	284	1488	top-6 BM25-retrieved questions
Amazon	past reviews	9	8.2	3	11	22	pre-written user summary
Yelp	past reviews	9	7.0	2	10	18	pre-written user summary
Google	past reviews	8	6.9	2	10	15	pre-written user summary

Results. Our ranker matches the strongest reward-factorization baseline, VPL, within noise on all three datasets, while requiring no annotated pairs per user and no per-candidate encoder pass. PAL, PReF, and LoRe are below ours on every dataset. GPO obtains the highest raw scores, by about 0.005 ROUGE-L per dataset, but still depends on per-user labeled examples and on post-hoc scoring of materialized candidates. SynthesizeMe selects close to random: its persona-prompted judge captures 1–2% of the oracle headroom, and a generic judge without the persona is no worse (0.2033, 0.1677, and 0.1378 ROUGE-L on Amazon, Yelp, and Google), mirroring the miscalibration of generalist reward models in Section 4.2. A further diagnostic explains why the user-conditioning machinery of these baselines does not pay off here: giving each user another user's fitted parameters or context set changes the selection by at most 0.0008 ROUGE-L for every trained method, including ours. Because every candidate is generated with the user's profile in the prompt, personalization is already realized at generation time, and what remains at selection is a calibrated judgment of which candidate comes closest to what the user would have written. This is precisely the judgment our lightweight ranker provides without any per-user labels, and it reinforces rather than competes with our central claim that the personalization headroom is a candidate matching problem.

C Implementation Details

C.1 Datasets

We evaluate on nine datasets spanning three personalized generation paradigms. In every case the input pairs a task query with the target user's interaction history, and the reference is text specific to that user, so success requires matching individual style rather than generic quality. Table 6 lists, for each dataset, the profile text that enters the generation prompt and h_u, and the number of evaluation prompts is given below.

Short-form personalized generation (LaMP). Given a document and the user's history of past writings, the model produces a short user-styled text: a news headline for an article (News, 1,889 evaluation prompts), a title for a research-paper abstract (Scholarly, 2,498), or a rephrased tweet in the user's voice (Tweet, 1,495). Following the benchmark's retrieval-augmented protocol, the prompt carries the three profile items retrieved by BM25 for the current input, and the reference is the headline, title, or tweet the user actually wrote.

Long-form personalized QA (LaMP-QA). Given an open-ended information-seeking question and the asker's narrative history, the model generates a long-form answer tailored to the asker's background and needs, across three topical domains: Arts & Entertainment (Art), Lifestyle & Personal Development (Lifestyle), and Society & Culture (Society). The prompt follows the benchmark's retrieval-augmented template with the six past questions retrieved by BM25. LaMP-QA ships rubrics rather than reference answers; we use, for each question, the per-user reference answer released with Personalized RewardBench [18], whose question identifiers coincide with the LaMP-QA test set, and split the questions 90/10 into ranker training and evaluation (77, 99, and 108 evaluation prompts).

Personalized explainable recommendation (XRec). Given a user–item interaction together with the user profile and the item profile, the model generates a natural-language explanation of why the item suits the user, on three platforms (Amazon, Yelp, Google; 3,000 evaluation prompts each), with the user's own review-style explanation as reference. We use the benchmark's explainer prompt with its textual user and item summaries and generate with the plain instruction-tuned LLM, without the collaborative embedding tokens specific to XRec's own model.

Following the original benchmarks, we report ROUGE-L against these references (ROUGE-1 where noted), and BLEU as a held-out metric in Section 4.3.

C.2 Training Details

Candidate pools and labels. *Qwen2.5-7B-Instruct* generates 256 samples per prompt at temperature 1.0 with vLLM; Best-of-N uses the first N candidates in stored order, and $N = 64$ throughout the appendix. Each candidate is labeled with its ROUGE-L against the reference, standardized within its pool for the pointwise objective (Section 3.3).

Embeddings. The three embeddings are final-layer last-token hidden states of the same frozen *Qwen2.5-7B-Instruct* ($d = 3584$), taken over the task query, the serialized profile text of Table 6, and each candidate. In our implementation they are obtained with vLLM's last-token pooling in a pass separate from generation, which is the cost we report as "embedding pass" in Table 8; a serving stack that exposes the generator's hidden states removes this pass entirely, since the states are produced during generation anyway. No embedding is fine-tuned.

Ranker. The default ranker concatenates (h_x, h_u, h_y) and applies three linear layers with hidden width 256, GELU activations, and dropout 0.1, for 2.8M parameters. The main-text protocol is given in Section 4.1. The appendix ablations (Appendices B.1 and C.3) share one trainer: AdamW with learning rate 5×10^{-4} and weight decay 0.01 for 15 epochs at batch size 64, a single seed, and the same candidate pools, so that all rows of a table are directly comparable.

Finetuned reward-model comparator. The 8B comparator of Section 4.3 is *Skywork-Reward-V2-Llama-3.1-8B*, fully finetuned per dataset (bf16, fully sharded data parallel on two NVIDIA B200 GPUs) with the same pointwise objective as our ranker: mean-squared error to the within-pool standardized ROUGE-L, with scores centred within each pool. Each training prompt contributes eight candidates from its pool (the best, the worst, and six sampled at random); we use 1,500 training prompts (all 690 to 966 available prompts for LaMP-QA), 24 prompts per optimizer step, one epoch (two for LaMP-QA), learning rate 2×10^{-5} chosen on a News pilot against 5×10^{-6}, 10% warmup, weight decay 0.01, gradient clipping at 1.0, sequences truncated to 2,048 tokens, and a single seed. The finetuned model then scores the first 64 candidates of every evaluation prompt.

C.3 Ranking Guided Generation Hyperparameters

Selection. The decoding hyperparameters of Section 3.4 were set a priori from interpretable considerations rather than by grid search: τ controls how often the ranker intervenes, $H_{\max}$ and α_0 control the strength of an intervention, and the warmup protects the opening tokens of the response, before the partial-answer state carries content. The same values ($\tau = 1$, $H_{\max} = 8$, $\alpha_0 = 5$, 3-token warmup) are used for all nine datasets.

Sensitivity. Table 7 varies each hyperparameter around its default while holding the others fixed, with greedy decoding and at most 48 new tokens. Performance is stable across the full grid: the

Table 7: Sensitivity of ranking guided generation to its decoding hyperparameters. Each block varies one hyperparameter while holding the others at their defaults ($\tau = 1$, $H_{\max} = 8$, $\alpha_0 = 5$, warmup 3). Cells report ROUGE-L with greedy decoding.

	Short-form Generation			Long-form QA			Explainable Recommendation		
Setting	News	Scholarly	Tweet	Art	Lifestyle	Society	Amazon	Yelp	Google
$\tau = 0.5$	0.1356	0.3823	0.4272	0.1409	0.1348	0.1385	0.2002	0.1701	0.1330
$\tau = 1.0$ (default)	0.1358	0.3819	0.4257	0.1430	0.1331	0.1399	0.2001	0.1703	0.1357
$\tau = 2.0$	0.1359	0.3816	0.4235	0.1423	0.1381	0.1397	0.2030	0.1714	0.1354
$\tau = 4.0$	0.1378	0.3816	0.4229	0.1411	0.1364	0.1388	0.2044	0.1711	0.1354
$H_{\max} = 4$	0.1349	0.3819	0.4276	0.1404	0.1350	0.1361	0.2012	0.1711	0.1322
$H_{\max} = 16$	0.1379	0.3819	0.4257	0.1412	0.1347	0.1401	0.2011	0.1697	0.1357
$\alpha_0 = 1$	0.1379	0.3819	0.4257	0.1406	0.1365	0.1406	0.2015	0.1710	0.1360
$\alpha_0 = 2$	0.1379	0.3819	0.4257	0.1418	0.1350	0.1411	0.2014	0.1700	0.1357
$\alpha_0 = 10$	0.1349	0.3819	0.4281	0.1392	0.1348	0.1363	0.2006	0.1702	0.1333
warmup $= 0$	0.1375	0.3829	0.4257	0.1432	0.1342	0.1398	0.2002	0.1703	0.1357
warmup $= 8$	0.1376	0.3819	0.4240	0.1424	0.1332	0.1397	0.2015	0.1703	0.1359

Table 8: Measured inference cost on one NVIDIA RTX A6000 for one query with $N = 64$ candidates. Reward-model numbers are the mean over Skywork-Reward-V2, InternLM2, URM, and ArmoRM (range in the text); generation and embedding use Qwen2.5-7B-Instruct with vLLM.

Stage	**Per candidate**	**Per query** ($N = 64$)	**Peak GPU memory**
Candidate generation, short-form (Qwen2.5-7B-Instruct)	10.2 ms	0.65 s	≈40 GiB
Embedding pass (only if states are not recycled from generation)	2.08 ms	0.13 s	25–39 GiB
Ranker scoring, ours (2.8M MLP, recycled embeddings)	0.0015 ms	0.098 ms	0.93 GiB
Reward-model scoring (8B, mean of four)	37.1 ms	2.38 s	20.6 GiB

max–min spread over all settings is at most 0.0052 ROUGE-L on every dataset (mean 0.0037). The threshold τ is the one knob with a visible effect on cost: the gate $H_t > \tau$ fires on 14–59% of decoding steps at $\tau = 0.5$, on 4–39% at the default, and on at most 11% at $\tau = 2$, so τ offers practitioners a single interpretable compute–intervention trade-off. $H_{\max}$, α_0, and the warmup length change the outcome by at most 0.005 ROUGE-L. Mean next-token entropy under greedy decoding is 0.19–0.43 nats on short-form generation, 0.46–0.52 nats on explainable recommendation, and 0.83–0.92 nats on long-form QA, which is why the gate fires far more often on the latter.

C.4 Inference Cost

Table 8 reports measured wall-clock and memory costs on a single NVIDIA RTX A6000 (46 GiB) for one query with 64 candidates. Scoring the pool with the 2.8M-parameter ranker takes 0.098 ms at 0.93 GiB peak GPU memory, roughly 655k candidates per second, when the candidate embeddings are recycled from generation. Scoring the same pool with an 8B reward model takes 1.7–3.9 s at 18–22 GiB (mean 2.4 s over Skywork-Reward-V2, InternLM2, URM, and ArmoRM), a latency gap of four orders of magnitude and a memory gap of about 22×. Our current implementation does not yet read the states off the generation pass but recomputes them in a separate pooling pass (Appendix C.2); counting that pass, end-to-end selection costs 2.1 ms per candidate, still 18× cheaper than reward-model scoring, and the recycled figure is the architectural ceiling. The one-off cost of embedding a user's profile is 7 ms (XRec), 96 ms (Scholarly), and 118 ms (LaMP-QA) per user when computed standalone and zero when recycled. For context, generating the pool itself costs 10 ms per candidate (0.65 s per query) on short-form generation and roughly an order of magnitude more on long-form QA (about 11 s per query), so generation rather than selection dominates the test-time budget, and scoring with an 8B reward model costs as much as generating the entire pool it evaluates. Ranking guided generation removes the need to materialize the pool at all: each intervention costs one forward and backward pass through the MLP plus one $V \times d$ matrix–vector product, which is negligible relative to a generator forward pass, and the gate fires on a minority of decoding steps (4–39% at the default threshold, Appendix C.3). Generation and embedding use vLLM, whereas the reward models run with HF transformers at batch size 32 as released, so part of the latency gap reflects serving implementation; the parameter and memory ratios do not.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Abstract and introduction

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: [N/A]

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: after authorlist

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Section 4

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix C.2/C.4

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: reviewed

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Personalized generation is directly related to societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Section 4.1

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release our code repo.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Section 3

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.