

Capturing Individual Human Preferences with Reward Features

André Barreto¹ Vincent Dumoulin¹ Yiran Mao¹ Nicolas Perez-Nieves¹ Bobak Shahriari¹ Yann Dauphin¹
Doina Precup¹ Hugo Larochelle¹

Abstract

Reinforcement learning from human feedback usually models preferences using a reward model that does not distinguish between people. We argue that this is unlikely to be a good design choice in contexts with high potential for disagreement, like in the training of large language models. We propose a method to specialise a reward model to a person or group of people. Our approach builds on the observation that individual preferences can be captured as a linear combination of a set of general reward features. We show how to learn such features and subsequently use them to quickly adapt the reward model to a specific individual, even if their preferences are not reflected in the training data. We present experiments with large language models comparing the proposed architecture with a non-adaptive reward model and also adaptive counterparts, including models that do in-context personalisation. Depending on how much disagreement there is in the training data, our model either significantly outperforms the baselines or matches their performance with a simpler architecture and more stable training.

1. Introduction

Reinforcement learning from human feedback (RLHF) may be useful when we do not have an explicit reward function but can distinguish between good and bad behaviour (Christiano et al., 2017). This is the case, for example, in the training of large language models (LLMs), in which RLHF has been applied with great success (Radford et al., 2018; Ramachandran et al., 2016; Wei et al., 2022).

The most common way to collect human feedback is to ask humans to rank examples of an agent’s behaviour. Usually, all the feedback is integrated to derive a single reward function that reflects as well as possible the preferences of the population of interest. Although this is a sensible strategy

when preferences are homogeneous across the population, it may be less effective when there is considerable disagreement. This is likely the case in the training of LLMs.

As an illustration, suppose that, given a pair of alternative responses to a subjective question, 51% of the target audience prefer the first option while the remaining 49% prefer the second. If we do not distinguish between users, we are left with two options: either we pick the preferred answer and leave 49% of the users unhappy 100% of the time, or we sample the answers proportionally to how often they are preferred and leave 100% of the users unhappy approximately half of the time. Both solutions are clearly unsatisfactory.

We need reward models that can be specialised to users. Crucially, we want the model to be able to adapt to people outside of the group who provided feedback for training. This means that we must capture somehow the subjective criteria that underlie human preferences—a non-trivial endeavour, as humans often cannot fully articulate the reasons why they prefer one behaviour over the other.

One alternative is to specialise the reward model using the same type of data used in RLHF, since in this case the preference criteria are indirectly elicited rather than explicitly spelled out. This raises another challenge: unlike in the standard RLHF problem, in which humans may be willing to provide abundant feedback, here we assume that users will only answer very few questions regarding their preferences. So, the adaptation of the reward model must take place fast.

We propose a simple, principled method to adapt a reward model to a new user using little data. Our approach leverages the fact that individual preferences can be captured as a linear combination of a set of general *reward features*. These features are hidden (in some cases from the person themselves), and must be inferred from data. We show how this can be accomplished through small modifications to the reward model’s architecture and training. During the regular RLHF process, we use data coming from different individuals to learn common features that capture the preferences of the group. When the reward model is being specialised to an unknown user, the features are frozen, and only the coefficients of the linear combination must be learned. This results in a simple classification problem that can be reliably solved with a few training examples provided by the user.

¹Google DeepMind. Correspondence to: André Barreto <andrebarreto@google.com>.

2. Background

We adopt the terminology of LLMs, although the proposed approach is not specific to this application of RLHF. Let \mathcal{T} be a finite set of symbols called *tokens*, let $\mathcal{X} \equiv \cup_{i=1}^{\ell_x} \mathcal{T}^i$ be the *context space*, where ℓ_x is the maximum context length, and let $\mathcal{Y} \equiv \cup_{i=1}^{\ell_y} \mathcal{T}^i$ be the *response space*. Let $\Pi \equiv \{\pi \mid \pi : \mathcal{X} \rightarrow \Delta(\mathcal{Y})\}$ be the *policy space*, where $\Delta(\mathcal{Y})$ is the set of discrete probability distributions over \mathcal{Y} . We can think of a policy $\pi \in \Pi$ as the distribution over responses y conditioned on context x that is encoded in an LLM.

Given a context $x \in \mathcal{X}$ and two responses $y, y' \in \mathcal{Y}$, we can ask human raters for their preferred option; we use $y \succ y' \mid x$ to indicate that y is preferred over y' given x (Rafailov et al., 2024; Azar et al., 2024). We want to model humans' preferences as accurately as possible.

We will use data generated as follows. First, a human rater is sampled, $h \sim \mathcal{D}_{\mathcal{H}}$, where $\mathcal{D}_{\mathcal{H}} \in \Delta(\mathcal{H})$ is a distribution over the set of users \mathcal{H} we are interested in. Second, a context is sampled, $x \sim \mathcal{D}_{\mathcal{X}}^h$, where $\mathcal{D}_{\mathcal{X}}^h \in \Delta(\mathcal{X})$ is a distribution over \mathcal{X} that may depend on h . Third, two responses are sampled, $y, y' \sim \mu(\cdot \mid x)$, where $\mu \in \Pi$ is an appropriately defined exploration policy (possibly implemented by an LLM). Finally, y and y' are shown to h , who ranks them by sampling $z \sim \mathcal{Z}(x, y, y', h)$, where $\mathcal{Z} \in \Delta(\{0, 1\})$ is a Bernoulli distribution whose mean is $p(y \succ y' \mid x, h)$, the probability of response y being preferred over y' given the context x . This procedure results in the dataset $D \equiv \{(x_i, y_i, y'_i, z_i)\}_{i=1}^n$. We assume the tuples in D are independent of each other.

The preferences are assumed to be generated by an underlying reward function $r : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ which we do not have access to; our goal is to use the data in D to compute an approximation $r_{\theta} \approx r$, where $\theta \in \mathbb{R}^k$ is a vector of parameters. Model r_{θ} can then play the role of the reward function in the standard RL loop, which eventually yields a policy $\pi \in \Pi$ that reflects the preferences of the population \mathcal{H} (Christiano et al., 2017; Sutton and Barto, 2018).

2.1. The Bradley-Terry model

A common approach to learn r_{θ} is to use the Bradley-Terry model (Bradley and Terry, 1952). One way to instantiate this model is to assume that (Azar et al., 2024)

$$p(y \succ y' \mid x) = \sigma(r(x, y) - r(x, y')), \quad (1)$$

where σ is the sigmoid function. Since $\sigma(-x) = 1 - \sigma(x)$, using (1) we can write, for a given (x, y, y', z) :

$$p(z \mid x, y, y') = \sigma(r(x, y) - r(x, y'))^z (\sigma(r(x, y') - r(x, y))^{1-z}). \quad (2)$$

Recalling that $r_{\theta} \approx r$, using (2) we can write the likelihood of θ with respect to z as $\mathcal{L}(\theta) = p(z \mid x, y, y'; \theta) = \sigma(\delta_{xyy'z}^{\theta})$, where $\delta_{xyy'z}^{\theta} = r_{\theta}(x, y) - r_{\theta}(x, y')$ if $z = 1$

and $\delta_{xyy'z}^{\theta} = r_{\theta}(x, y') - r_{\theta}(x, y)$ otherwise. Since the samples in the dataset D are independent and identically distributed (i.i.d.), the likelihood function of θ given the data can be written as $\mathcal{L}(\theta \mid D) = \prod_i p(z_i \mid x_i, y_i, y'_i; \theta) = \prod_i \sigma(\delta_{x_i y_i y'_i z_i}^{\theta})$. The problem of learning r_{θ} can thus be formalised as the following maximum likelihood estimation:

$$\max_{\theta} \log \mathcal{L}(\theta \mid D) = \max_{\theta} \sum_{i \in D} \log \sigma(\delta_{x_i y_i y'_i z_i}^{\theta}). \quad (3)$$

2.2. The issue

One of the key assumptions underlying (3) is that

$$p(y \succ y' \mid x) = \mathbb{E}_{H \sim \mathcal{D}_{\mathcal{H}}} [p(y \succ y' \mid x, H)]. \quad (4)$$

That is, we are modelling the probability of response y being preferred over response y' given context x as the average opinion among users \mathcal{H} computed according to $\mathcal{D}_{\mathcal{H}}$.

Clearly, (4) will only be a good assumption if the distribution $\mathcal{D}_{\mathcal{H}}$ is representative of the target users, which may or may not be the case in practice. Here we call attention to another issue: when we use (3) to derive a reward function, the accompanying assumption (4) implies that we are neglecting individual differences in the preferences of the population \mathcal{H} . Formally, we are ignoring the variance of the random variable $\bar{z}(H) \equiv p(y \succ y' \mid x, H)$.

Let us focus on the simple problem of choosing which of two possible responses y and y' should follow a context x . An intuitive way to understand the deleterious effect of making this decision based on (4) is to think of it as a game. At each round, the player picks either y or y' . Then, a user $h \sim \mathcal{D}_{\mathcal{H}}$ is sampled and a reward is drawn from $\mathcal{Z}(x, y, y', h)$ if y was selected, and from $\mathcal{Z}(x, y', y, h)$ otherwise. Clearly, the best one can do in this game is to always pick y if $\mathbb{E}\bar{z}(H) \geq 0.5$, and always pick y' otherwise. This results in a expected win rate of $\max(\mathbb{E}\bar{z}(H), 1 - \mathbb{E}\bar{z}(H))$. We are proposing to change the game in a way that we only choose between y and y' after the user h has been revealed. In this case the optimal strategy is to pick y if $\bar{z}(h) \geq 0.5$, and y' otherwise. This increases the expected win rate to $\mathbb{E} \max(\bar{z}(H), 1 - \bar{z}(H)) \geq \max(\mathbb{E}\bar{z}(H), 1 - \mathbb{E}\bar{z}(H))$.

3. Reward-model personalisation

In this section we introduce the main concepts underpinning the problem of learning a reward model that is specialised to a person or group of people. We will focus on the case with one reward function per person, but this can be easily extended to a group.

As the game example above suggests, our strategy will be to model $p(y \succ y' \mid x, h)$ instead of $p(y \succ y' \mid x)$. We will use data similar to that used in conventional RLHF, with a small extra requirement: tuples in the dataset have to

be labelled with (non-identifying) IDs of the human raters. That is, we have $D^+ \equiv \{(x_i, y_i, y'_i, z_i, h_i)\}_{i=1}^n$, where h_i is effectively an index indicating which rater defined z_i based on x_i, y_i and y'_i . This is not a strong requirement: we are just surfacing a piece of information used to generate D .

As usual, the objective in the reward personalisation problem is not to model the dataset D^+ itself, but rather to use the data to generalise over the sets of interest: \mathcal{H} , \mathcal{X} , and \mathcal{Y} . Let $\hat{\mathcal{H}} \subseteq \mathcal{H}$ be the set of human raters \hat{h} who provided feedback for the generation of D^+ . We distinguish between two types of generalisation. In *intra-user generalisation* one uses D^+ to model $p(y \succ y' | x, \hat{h})$ over $\mathcal{X} \times \mathcal{Y} \times \hat{\mathcal{H}}$ only. That is, although we are able to predict preferences over contexts x and responses y beyond the training data, we are unable to extrapolate to users not in the set of raters $\hat{\mathcal{H}}$. In *inter-user generalisation* one uses the same data to model $p(y \succ y' | x, h)$ over the entire set $\mathcal{X} \times \mathcal{Y} \times \mathcal{H}$. That is, we are able to predict the preferences of *any user* in \mathcal{H} over the entire sets \mathcal{X} and \mathcal{Y} . We are interested in inter-user generalisation.

While the current practice of not distinguishing between users may appear to achieve inter-user generalisation, we argue that it collapses the two types of generalisation by (likely erroneously) assuming a single user whose preferences coincide with the average preference in \mathcal{H} (cf. 4).

To achieve true generalisation across all users in \mathcal{H} , we require a distinct reward function $r_h(x, y)$ per user $h \in \mathcal{H}$. A naive way to accomplish this is to employ a separate parametric function r_{θ_h} per user, but it is unclear how to obtain inter-user generalisation with this model. Instead, we suggest to use two disjoint sets of parameters: a common vector of parameters $\theta \in \mathbb{R}^c$ that is shared among all users $h \in \mathcal{H}$, including those $h \notin \hat{\mathcal{H}}$, and parameters $\theta_h \in \mathbb{R}^d$ that are specific to user h . This is related to the concept of “adapters” (Rebuffi et al., 2018; Houlsby et al., 2019; Poth et al., 2023).

This partitioning of the parameters induces a division of the training procedure itself. The data in D^+ can be used to learn θ and $\theta_{\hat{h}}$ for a rater $\hat{h} \in \hat{\mathcal{H}}$. However, to learn the parameters θ_h associated with a user not in D^+ , $h \in \mathcal{H} - \hat{\mathcal{H}}$, one needs additional data containing feedback provided by that particular user, $D_h \equiv \{(x_i, y_i, y'_i, z_i)\}_{i=1}^m$, with $z_i \sim \mathcal{Z}(x_i, y_i, y'_i, h)$. We will refer to the use of D^+ to learn θ and $\theta_{\hat{h}}$, with $\hat{h} \in \hat{\mathcal{H}}$, as *training*; learning θ_h for $h \in \mathcal{H}$ using D_h will be referred to as *adaptation*.

We assume we have abundant data for training in D^+ , but adaptation must take place with as few additional training examples as possible in D_h (that is, $n \gg m$). Because of this, and also because we want the adaptation of the model to a specific user $h \in \mathcal{H} - \hat{\mathcal{H}}$ to be quick, we generally want to have $c \gg d$ —in words, we want the number of shared

parameters to be much larger than the number of parameters that are specific to a given individual. Another, slightly more practical reason to have a large number of shared parameters is that this allows for a distributed learning architecture in which the learning of θ can make use of a centralised, powerful computational infrastructure, while each θ_h can be learned “locally” using less resources.

4. Reward-feature models

In this section we discuss an architecture for r_{θ, θ_h} that has particularly nice properties. We define *reward features* as a function $\phi(x, y) : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$. We then assume that the reward for individual h is given by

$$r_h(x, y) = \langle \phi(x, y), \mathbf{w}_h \rangle, \quad (5)$$

where $\mathbf{w}_h \in \mathbb{R}^d$ and $\langle \cdot, \cdot \rangle$ denotes inner product. We can write the personalised version of the Bradley-Terry model based on (1) and (5) as:

$$\begin{aligned} p(y \succ y' | x, h) &= \sigma(r_h(x, y) - r_h(x, y')) \\ &= \sigma(\langle \phi(x, y), \mathbf{w}_h \rangle - \langle \phi(x, y'), \mathbf{w}_h \rangle) \\ &= \sigma(\langle \phi(x, y) - \phi(x, y'), \mathbf{w}_h \rangle). \end{aligned} \quad (6)$$

In analogy with (2), we can then write

$$\begin{aligned} p(z | x, y, y', h) &= \sigma(\langle \phi(x, y) - \phi(x, y'), \mathbf{w}_h \rangle)^z \times \\ &\quad \times \sigma(\langle \phi(x, y') - \phi(x, y), \mathbf{w}_h \rangle)^{1-z}. \end{aligned}$$

Following the steps taken in Section 2.1, we now define a parametric form for r_h that reflects (5). We first define a parameterised function $\phi_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$, with $\theta \in \mathbb{R}^c$. Note that θ is the set of shared parameters defined above; the parameters associated with a specific individual $h \in \mathcal{H}$ are simply a vector $\mathbf{w}_h \in \mathbb{R}^d$. Putting it all together, our parameterised model is $r_{\theta, \mathbf{w}_h}(x, y) = \langle \phi_\theta(x, y), \mathbf{w}_h \rangle$. We call this architecture a *reward-feature model* (RFM).

4.1. Training

We now derive an appropriate optimisation objective for the training phase of an RFM r_{θ, \mathbf{w}_h} . Let $\mathbf{W} \in \mathbb{R}^{|\hat{\mathcal{H}}| \times d}$ be a matrix formed by stacking $|\hat{\mathcal{H}}|$ vectors $\mathbf{w}_{\hat{h}}$. To define the likelihood function of θ and \mathbf{W} with respect to \mathbf{z} , we let $\mathcal{L}(\theta, \mathbf{W}) = p(\mathbf{z} | x, y, y', \hat{h}; \theta, \mathbf{W}) = \sigma(\langle \delta_{xyy'z}^\theta, \mathbf{w}_{\hat{h}} \rangle)$, with $\delta_{xyy'z}^\theta = \phi_\theta(x, y) - \phi_\theta(x, y')$ if $z = 1$ and $\delta_{xyy'z}^\theta = \phi_\theta(x, y') - \phi_\theta(x, y)$ otherwise. Since the data in D^+ is i.i.d., we have that

$$\mathcal{L}(\theta, \mathbf{W} | D^+) = \prod_{i \in D^+} \sigma(\langle \delta_{x_i y_i y'_i z_i}^\theta, \mathbf{w}_{\hat{h}_i} \rangle).$$

As before, the problem of learning θ and \mathbf{W} can be formalised as a maximum-likelihood estimation:

$$\max_{\theta, \mathbf{W}} \sum_{i \in D^+} \log \sigma(\langle \delta_{x_i y_i y'_i z_i}^\theta, \mathbf{w}_{\hat{h}_i} \rangle) \quad (\text{training}). \quad (7)$$

Note that \hat{h}_i works like an index to select the appropriate row of \mathbf{W} used to compute the reward function of that particular user. The solution of (7) yields a feature function $\phi_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$ and a matrix $\mathbf{W} \in \mathbb{R}^{|\hat{\mathcal{H}}| \times d}$. Each dimension of ϕ_θ , $\phi_{\theta i}$, can be interpreted as a criterion used by humans to express their preferences. The i -th element of $\mathbf{w}_{\hat{h}_i}$, $w_{\hat{h}_i i}$, represents how much human rater \hat{h}_i values (or does not value) criterion $\phi_{\theta i}$. Importantly, learning the features ϕ_θ does not depend on raters being able to articulate the criteria underlying their preferences; the only thing that is required from them is to rank pairs of candidate responses.

Features ϕ_θ plus matrix \mathbf{W} allow for intra-user generalisation, since each row of \mathbf{W} specialises the RFM to one of the raters $\hat{h} \in \hat{\mathcal{H}}$. To get inter-user generalisation we will need to combine ϕ_θ with a new set of weights \mathbf{w} whose computation we will discuss in the next section. The question arises as to how well the reward features ϕ_θ will be able to capture the preferences of the entire group \mathcal{H} . We need ϕ_θ to have enough representational capacity—a proxy for which is the number of parameters c —and its dimension d to be sufficient to capture the preferences in D^+ . We may want c and d not to be too large, though. When this is the case, an RFM resulting from (7) may effectively become $|\hat{\mathcal{H}}|$ independent reward models, one for each rater $\hat{h} \in \hat{\mathcal{H}}$. We usually do not want that. Instead, we want ϕ_θ to capture the features that are *common* to the raters in $\hat{\mathcal{H}}$, since these should also reflect the preferences of the users in \mathcal{H} more generally. This means that c and d have to be appropriately set or controlled through regularisation.

4.2. Adaptation

We are interested in using features ϕ_θ learned during training (7) to specialise r_{θ, w_h} to users beyond the raters $\hat{h} \in \hat{\mathcal{H}}$. We first note that we can get a reward model specialised to \hat{h} by selecting the appropriate row of \mathbf{W} : $r_{\theta, w_{\hat{h}}}(x, y) \equiv \langle \phi_\theta(x, y), \mathbf{w}_{\hat{h}} \rangle$.

With that in mind, it is not difficult to see that the problem of adapting r_{θ, w_h} to an unseen user can be formulated as a small modification of training in which the log likelihood function is maximised with respect to a new set of coefficients: $\max_{\mathbf{w}} \log \mathcal{L}(\mathbf{w} | D_h, \theta)$. Explicitly, we have

$$\max_{\mathbf{w}} \sum_{i \in D_h} \log \sigma \left(\langle \delta_{x_i y_i y'_i z_i}^\theta, \mathbf{w} \rangle \right) \quad (\text{adaptation}). \quad (8)$$

Adaptation is particularly simple with an RFM: since θ is frozen, optimisation (8) comes down to logistic regression, which is a well understood convex optimization problem whose sample complexity has been characterised for different scenarios (see, for example, Long, 1995, Plan et al., 2016, Hsu and Mazumdar, 2024 and references therein).

The solution of (8) yields an RFM r_{θ, w_h} specialised to a

user h not necessarily in D^+ . Here another question naturally comes up: how to adapt the outputs of an LLM π to reflect the preferences encoded in r_{θ, w_h} , which is what we ultimately want? One way of doing it is to adopt r_{θ, w_h} as a criterion to select responses coming from π . An obvious example is the procedure known as “best-of- n ”: given a context x , we sample n candidate responses $y_i \sim \pi(x)$ and select $\arg \max_{y_i} r_{\theta, w_h}(x, y_i)$. Alternatively, we can use \mathbf{w} resulting from (8) to directly modulate the output of a conditional LLM; we will come back to this point later.

5. Experiments

We now present experiments illustrating how an RFM performs in practice. Here we describe the crucial aspects of our experimental setup; for further details, see Appendix A.

We used Google DeepMind’s (2024a) Gemma 1.1 2B model to implement both RFM and the baselines.¹ We defined three baselines: the *non-adaptive baseline* is a reward model that neither distinguishes between raters nor performs adaptation, as is common practice. It was trained using gradient ascent to solve (3) starting from the pre-trained parameters of Gemma. The *adaptive baseline* uses (3) with a user-specific dataset D_h to adapt the parameters of the non-adaptive baseline to h . For a more direct comparison with RFM, we also defined an *adaptive linear baseline*. This baseline is similar to the adaptive baseline, but only its final layer is adapted. We implemented an RFM using the same Gemma model, with the final layer replaced by d counterparts corresponding to the features ϕ_θ . Its training and adaptation were carried out using gradient ascent to solve (7) and (8), respectively.

We used Cui et al.’s (2023) UltraFeedback, a dataset carefully curated to ensure the quality and diversity of the responses. The version of the dataset we adopted has a training set with 60,829 examples and a test set with 985 examples.²

5.1. Feature-based raters

Although UltraFeedback’s data is not rater-annotated, it does come with the features used to compute preferences. Associated with each (x_i, y_i) and each (x_i, y'_i) in the dataset, we have four features $\{\phi_j(x, y)\}_{j=1}^4$ indicating the level of *helpfulness*, *honesty*, *instruction following*, and *truthfulness* of response y with respect to context x . These features were computed using OpenAI’s (2024b) GPT-4.

The preferences in the original dataset were defined based on a single feature ϕ_j selected per example. We used instead the more general notion of a linear combination of features.

¹huggingface.co/google/gemma-1.1-2b-it

²huggingface.co/datasets/allenai/ultrafeedback_binarized_cleaned

Specifically, given (x_i, y_i, y'_i) and $\mathbf{w}_{\hat{h}} \in \mathbb{R}^4$, we set

$$z_i = \mathbb{1}\{\langle \phi(x_i, y_i), \mathbf{w}_{\hat{h}} \rangle > \langle \phi(x_i, y'_i), \mathbf{w}_{\hat{h}} \rangle\}, \quad (9)$$

where $\mathbb{1}\{\cdot\}$ is the indicator function. We created three scenarios using different sets $\hat{\mathcal{H}}$ defined through the raters' preference coefficients $\mathbf{w}_{\hat{h}}$. When an example was used during training (7), a rater \hat{h} was sampled uniformly at random from $\hat{\mathcal{H}}$ and concatenated to the example together with the corresponding preference z_i computed through (9).

5.1.1. SCENARIO 1

In our first set of experiments we used rater preferences $\mathbf{w}_{\hat{h}}$ sampled from four normal distributions $\{\mathcal{N}_i(\mathbf{e}_i, 0.3\mathbf{I})\}_{i=1}^4$, where $\mathbf{e}_i \in \mathbb{R}^4$ is the “one-hot” vector whose i -th component is 1 and all other components are 0, and \mathbf{I} is the 4×4 identity matrix. We sampled 30 preference vectors $\mathbf{w}_{\hat{h}}$ from each distribution, resulting in a set $\hat{\mathcal{H}}_1$ with 120 raters. This way of generating preference vectors $\mathbf{w}_{\hat{h}}$ results in raters \hat{h} that mostly care about a single feature.

We assess RFM's performance using different numbers of reward features. The notation $\text{RFM}(d)$ indicates that a d -dimensional feature vector $\phi_\theta \in \mathbb{R}^d$ was used. It is worth emphasising that RFM did *not* have access to the real features ϕ underlying the data in (9). Instead, it *learned* feature functions ϕ_θ parameterised by $\theta \in \mathbb{R}^c$ using (7).

We performed two independent training runs for each model. Figure 1 shows the intra-user test accuracy obtained by the baseline and RFM throughout training. This corresponds to the fraction of examples $(x_i, y_i, y'_i, z_i, \hat{h}_i)$ in the test set for which the models can correctly predict the preference $z_i \in \{0, 1\}$. That is, it is an estimate of the models' intra-user generalisation. Not surprisingly, RFM significantly outperforms the rater-agnostic baseline.

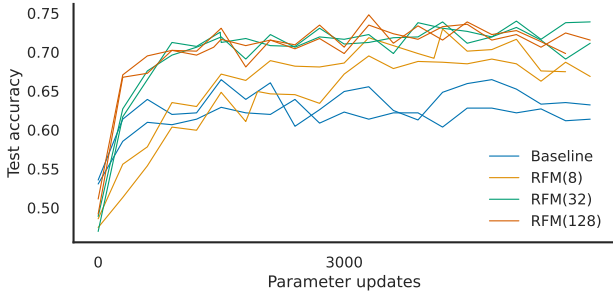


Figure 1. **Intra-user test accuracy:** accuracy in predicting the preferences of raters on test set during training in Scenario 1 (estimate of intra-user generalisation). Showing two runs per model.

But what about the inter-user generalisation, which is what we really care about? To assess the ability of the models to adapt to unseen individuals, we defined three

“held-out users” $\{h_i\}_{i=1}^3$ with the following preference weights: $\mathbf{w}_{h_1} = [1, 0, 0, 0]^\top$, $\mathbf{w}_{h_2} = [1, 0, 0, 1]^\top$, and $\mathbf{w}_{h_3} = [1, 0, -1, 1]^\top$. These users were defined somewhat arbitrarily and are supposed to represent different preference profiles in the population \mathcal{H} .³ Looking at the distributions \mathcal{N}_j defined above, we see that the held-out users \mathbf{w}_{h_i} get increasingly “out-of-distribution” with i —that is, they become increasingly less likely to be sampled from any of the distributions $\{\mathcal{N}_j\}_{j=1}^4$.

We assessed RFM's inter-user generalisation using different numbers of examples m to do the adaptation. For each held-out user h_i , we defined an initial dataset D_{h_i} by sampling 10 examples uniformly at random from D^+ and labelling them using (9) with the corresponding \mathbf{w}_{h_i} . We then solved (8) using this data and assessed its accuracy on the test set (also properly relabelled). We iterated this process until we had a dataset D_{h_i} with 90 examples. For each of the two feature functions ϕ_θ computed during training (7), and each $d \in \{8, 32, 128\}$, we repeated all the steps above 5 times.

Figure 2 (top row) shows the performance of the baselines and RFM in predicting the preferences of the held-out users. Surprisingly, the preferences of held-out users h_1 and h_2 seem to reasonably agree with the average preference among the raters $\hat{\mathcal{H}}$, as indicated by the non-adaptive baseline's results on these raters. In these cases adaptation does not provide much of a benefit—neither for the adaptive baselines nor for RFM. In contrast, for held-out user h_3 , the most “out-of-distribution” of the three, adaptation provides a clear boost in performance, especially for the linear models.

5.1.2. SCENARIO 2

UltraFeedback features ϕ give rise to homogeneous raters when used in isolation. To give an idea, the “one-hot raters” obtained by plugging $\{\mathbf{e}_i\}_{i=1}^4$ into (9) agree on 54% of the training set; if we count the examples in which at least three such raters agree, this number increases to 85%. Crucially, the overlapping preferences of one-hot raters happen to be highly predictive of the preferences of held-out users h_1 and h_2 : the opinion of these users coincide with the majority vote among raters $\{\mathbf{e}_i\}_{i=1}^4$ on 68% and 70% of the test set, respectively. Observe how these numbers are similar to the results in the top row of Figure 2.

The benefits of an adaptive reward model should be clearer in a situation where human raters disagree significantly

³We use negative weights in \mathbf{w}_{h_3} to illustrate that the desirability of a feature may change from user to user. To ensure that the model is aligned with prevalent ethical norms and scientific consensus, we want to either constrain the features themselves or the associated weights. This can be accomplished through appropriate data collection and curation, as well as explicit constraints imposed on the reward model or the downstream LLM. For further discussion, see the Impact Statement section.

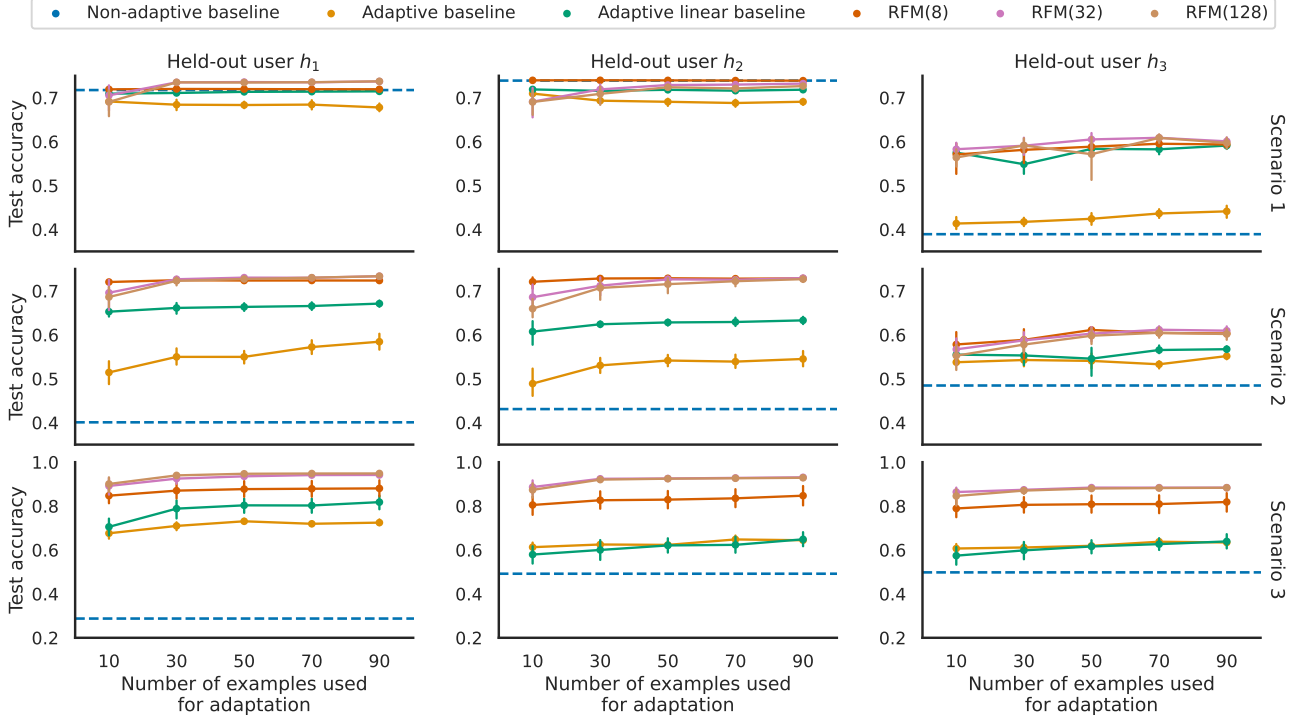


Figure 2. Inter-user test accuracy: accuracy in predicting the preferences of held-out users on test set after adaptation (estimate of inter-user generalisation). **Scenario 1:** UltraFeedback features and raters sampled from normal distributions with one-hot means. **Scenario 2:** UltraFeedback features and raters sampled from normal distributions whose means are all permutations of the vector $[1, -1, 0, 0]^T$. **Scenario 3:** Simple functions used as features and raters as in Scenario 2. The scale of the y-axis is defined per row. Error bars are 95% confidence intervals over 10 runs (5 adaptation runs on top of 2 training runs).

with each other (which should be the case in some realistic scenarios). To simulate this, we replaced the normal distributions used above with 12 counterparts whose means are all possible permutations of the vector $[1, -1, 0, 0]^T$ (see Appendix A). This ensures that for each mean vector there is another one pointing in the opposite direction—*i.e.*, we have 6 pairs of reference raters with opposite preferences. We then sampled 10 raters \hat{h} from each distribution, resulting in a set \mathcal{H}_2 with 120 raters. Finally, we re-ran the previous experiment with $\hat{\mathcal{H}}_1$ replaced by $\hat{\mathcal{H}}_2$.

The middle row of Figure 2 shows the results of the baselines and RFM in this new scenario. Note that we are trying to predict the preferences of the same three held-out users as before, but now using a different set of raters. Because the baseline cannot distinguish between raters during training, opposing preferences become contradictory learning signals, making it difficult to capture any trends in the data. This explains why the non-adaptive baseline’s performance reduces to chance. The adaptive baselines perform significantly above chance, but not as well as in Scenario 1. This is because they either have to adapt a very large number of parameters or adapt a few parameters on top of features learned in a rater-agnostic way. In contrast, by distinguish-

ing between raters during training, RFM can detect the signal in the data and learn useful features whose adaptation yields roughly the same performance as in Scenario 1. We call attention to how well RFM performs on held-out user h_1 , whose preferences are based on the first feature ϕ_1 only. This suggests that, using (7), RFM can capture the reward features underlying the raters’ preferences even when all the raters are based on combinations of such features (that is, even when the effect of features is not observed in isolation).

Comparisons. We now compare RFM with adaptive counterparts. Perhaps the most natural baselines here are models that perform “in-context” adaptation. We implemented such baselines using two prominent LLMs, Google DeepMind’s (2023) Gemini 1.5 Pro and OpenAI’s (2024a) GPT-4o. To assess the LLMs’ prediction accuracy for held-out user h_i , we provided $m = 10$ training examples ranked by h_i together with the test example to be ranked (the precise prompt with LLM instructions can be found in Appendix A). For reference, we also show the “zero-shot” performance of Gemini, obtained with $m = 0$.

Figure 3 shows the results of the two LLMs side-by-side with those obtained by the non-adaptive baseline and

RFM(32) trained under Scenario 2 and also using $m = 10$ examples for adaptation (*cf.* Figure 2). RFM outperforms its in-context counterparts in predicting the preferences of held-out users h_1 and h_2 , and essentially matches their performance on h_3 . We highlight the fact that both LLMs used for comparison are highly capable models.

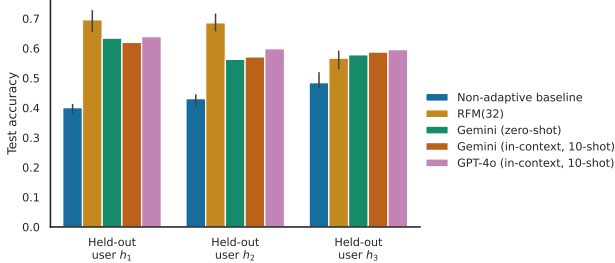


Figure 3. Accuracy in predicting the preferences of held-out users on test set under Scenario 2. All models used $m = 10$ examples for adaptation. The LLMs went over the test set once; for the non-adaptive baseline and RFM error bars are 95% confidence intervals over 10 runs (5 adaptation runs on top of 2 training runs).

We also compared RFM with Poddar et al.’s (2024) *variational preference learning* (VPL), described in Section 6. Even though in principle VPL allows for inter-user generalisation, Poddar et al. have only assessed VPL’s intra-user generalisation. They report an estimate of this type of generalisation on the UltraFeedback dataset of 61.49%. We tried to reproduce their experimental protocol as closely as possible, as explained in Appendix A. The resulting estimate of RFM(32)’s intra-user generalisation is 61.61%, which is on par with VPL’s performance.

5.1.3. SCENARIO 3

In this section we assess how well the good performance of RFM in predicting preferences transfers to the scenario where it is used to steer the behaviour of an LLM. We use best-of- n over $n = 40$ responses to each context x in the test set. The responses were generated by Google DeepMind’s (2024b) Gemma 2 9B and Gemma 2 27B (20 responses each). To be able to easily score the responses in an unequivocal way, we replaced the UltraFeedback features $\phi(x, y)$ with simple functions $\phi'(x, y)$ of the context x and the response y . Specifically, we defined four functions:

- $\phi'_1(x, y)$: the length of y .
- $\phi'_2(x, y)$: the number of adjectives in y .
- $\phi'_3(x, y)$: the number of alliterations in y .
- $\phi'_4(x, y)$: the number of words in y that also occur in x .

All the features were normalised to fall in the interval $[0, 1]$. These features were defined somewhat arbitrarily as simple illustrations of possibly conflicting subjective criteria. They were also kept simple to ensure that they can be computed with the architecture adopted for RFM.

The bottom row of Figure 2 shows how well the baselines and RFM can predict the preferences of the held-out users using the new features ϕ' . Observe how RFM’s prediction accuracy lies between 80% and 90%, a significant improvement over the results shown in the middle row of Figure 2 referring to Scenario 2. This indicates that, when the features underlying the data can be computed with RFM’s architecture, training (7) does indeed recover them.

Modulating the LLM’s output. For each context x in the test set, we scored all 40 responses using the non-adaptive baseline and RFM(32) adapted with $m = 30$ examples. Next, we selected the best-of- n response to context x according to the baseline, y_b , and RFM, y_r . Finally, we compared $\langle \phi'(x, y_b), \mathbf{w}_{h_i} \rangle$ and $\langle \phi'(x, y_r), \mathbf{w}_{h_i} \rangle$, for each $\{h_i\}_{i=1}^3$, and declared either a winning model or a draw. In words, we decided which model was the winner by comparing the “ground-truth” score of their selected responses.

Figure 4 shows results when best-of- n is applied with increasing n . We highlight that the candidate responses used with best-of- n are all qualitatively similar (and thus not easy to distinguish), and they originate from a distribution that is different from the one used for training. Yet, RFM consistently outperforms the baseline for all held-out users, and the fraction of times its selected response is preferred grows with n . One way to interpret this is as follows. Suppose we were to replace the non-adaptive baseline, which represents the current practice, with RFM(32) adapted with $m = 30$ pairwise comparisons. If we ignore ties, this means that the response delivered to, say, h_2 would be an improvement 66% of the time, in expectation, according to h_2 ’s own taste.

5.2. Reward models as raters

In the previous experiments we had access to the features ϕ underlying the raters’ preferences; although this is useful to investigate the models’ performance under different conditions, in a more realistic scenario we would only have the examples $\{(x_i, y_i, y'_i, z_i, h_i)\}_{i=1}^n \in D^+$. To simulate this situation, we used publicly available reward models as the raters \hat{h} . Specifically, we used 8 reward models to score and rank all the test-set examples (details in Appendix A).

We draw attention to the fact that the reward models were trained with human preference data, and hence they reflect the opinion of real people. However, as these models do not distinguish between raters—much like the non-adaptive baseline—, they reflect an average over preferences, and thus tend to “agree” considerably with each other. To avoid

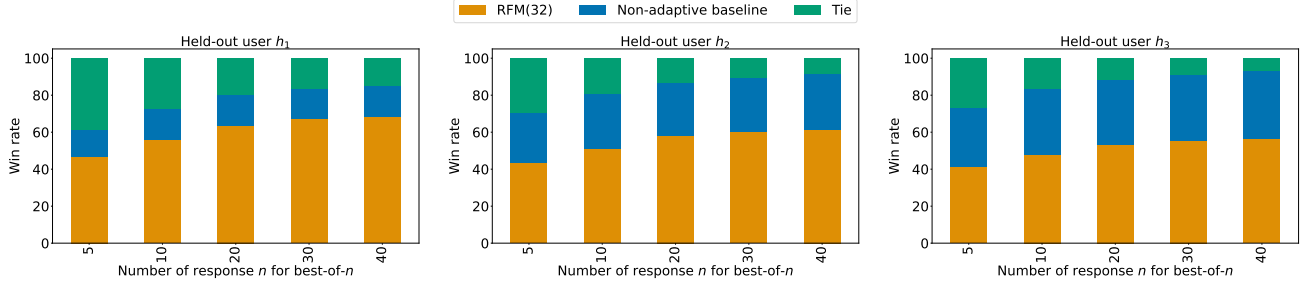


Figure 4. Fraction of times the best-of- n response computed based on the scores of each model has the highest ground-truth score in Scenario 3. RFM used $m = 30$ examples for adaptation. Statistics computed over 10 runs (5 adaptation runs on top of 2 training runs).

such overlapping, which may render the distinction of raters unnecessary (*cf.* Figure 2, top), we filtered out all the examples in the training and test sets in which 2 or fewer raters disagreed with the majority. This resulted in 23,614 training examples and 401 test examples. We then carried out a “leave-one-out” cross-validation composed of 8 rounds in which 7 of the models played the role of the raters $\hat{\mathcal{H}}$ and the remaining model played the role of the held-out user.

Results are shown in Figure 5. Given enough (but still few) adaptation examples, RFM’s performance either matches or significantly surpasses that of the non-adaptive baseline. This suggests that RFM can be useful in real scenarios. In particular, it can work as a form of “safety net” to make sure that minority preferences—like those of reward models 1, 3, and 4 in Figure 5—are also properly represented.

6. Related work

The field of personalised RLHF is fast growing due to the popularity of LLMs. We redirect the reader to Zhang et al.’s (2024) recent survey for a broad overview of different approaches. Here we focus on methods designed to specialise the reward model to specific users.

We distinguish between two categories of methods. In the first category we have methods whose ultimate goal is to compute a single policy representing some form of compromise across users (Dumoulin et al., 2024; Chakraborty et al., 2024; Siththaranjan et al., 2024; Conitzer et al., 2024; Dai et al., 2024; Yao et al., 2024). These methods take individual idiosyncrasies into account with the aim to build more accurate reward models, and therefore they do not necessarily provide a mechanism for adaptation to new users.

Dumoulin et al. (2024) present results on “rater misspecification”, in which a model trained on the preferences of a single rater which favors two characteristics (*e.g.*, short responses or long responses, but not middling responses) behaves very differently from a model trained on the preferences of two raters, each of which favors one character-

istic (short responses for one rater, long responses for the other). The authors frame the issue as a misspecified generative process for pairwise preferences and show that under correct assumptions—namely, the preference dataset is a mixture of individual raters’ preferences—it is possible to accurately capture multiple raters’ preferences. Chakraborty et al. (2024) propose to learn a user-specific reward function by using an expectation-maximisation algorithm to define a mixture of preference distributions. To learn a single policy that better represent diverse human preferences, the authors propose a “max-min” alignment objective inspired by the egalitarian principle in social choice theory.

In the second category we have methods designed to support the downstream specialisation of a policy (or LLM) to a specific user. This category can be further divided into two. In the first sub-category we have methods that learn a mapping from user information to a reward. Poddar et al.’s (2024) VPL encodes tuples $\{x_i, y_i, y'_i, z_i\}_{i=1}^m$ previously ranked by a user h into a latent variable, and then conditions the reward model and the policy on that variable (we compare VPL with RFM in Section 5.1). Zhao et al. (2024) cast personalised preference learning as a few-shot learning problem and tackle it with in-context meta-learning. Their *group preference optimization* (GPO) approach trains a transformer to predict target preferences for a given homogeneous group of users based on in-context examples of pairs of responses ranked by the same group.

Other methods in this sub-category encode different types of information about users, like demographic data, either in isolation or together with examples of ranked pairs (Fleisig et al., 2023; Li et al., 2024; Kim and Yang, 2025). For example, Li et al. (2024) propose a personalised RLHF (P-RLHF) framework in which a learnable user model computes an embedding for each user as the concatenation of an implicit embedding (that depends on the user’s unique identifier) and an optional explicit embedding (that depends on textual information about the user). The user embedding is then used to condition a base LLM through soft prompting.

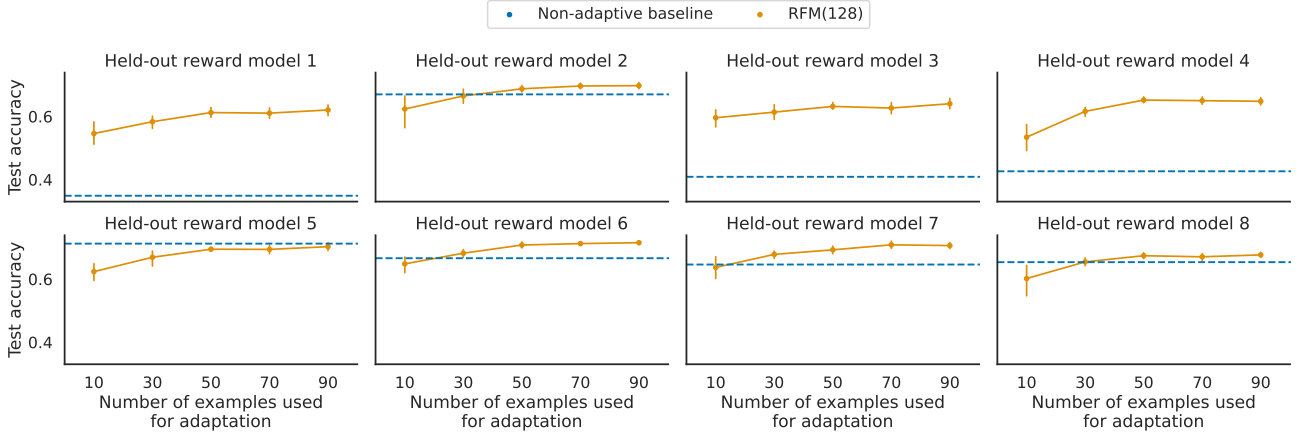


Figure 5. **Inter-user test accuracy:** accuracy in predicting the preferences of held-out “reward-models users” on test set after adaptation (estimate of inter-user generalisation). Error bars are 95% confidence intervals over 10 runs (5 adaptation runs on top of 2 training runs).

New users are accommodated by using a generic implicit embedding and computing an explicit embedding based on the new user’s textual information (if available).

The second sub-category is composed of methods that use the examples $\{x_i, y_i, y'_i, z_i\}_{i=1}^m$ to learn a set of weights specific to a user, like RFM. Chen et al. (2024)’s work is a notable example, and perhaps the closest to ours in the literature. The authors propose to replace the Bradley-Terry model with Coombs’s (1950) *ideal point model*. This model posits that people make assessments of a given object based on the distance between the representation of the object in a latent space and an “ideal reference point” in the same space. While the inner product between RFM’s $\phi_\theta(x, y)$ and \mathbf{w}_h does not strictly qualify as a distance function, we can think of \mathbf{w}_h as being akin to an “ideal point” for user h against which $\phi_\theta(x, y)$ is compared. Chen et al. (2024) propose two different instantiations of the ideal point model—Model A and Model B—which differ from RFM in a few aspects. Like RFM’s $\phi_\theta(x, y)$, Model A computes features from the context and response jointly, but the ideal point is constructed as a convex combination of “prototypical” ideal points and the comparison is made using the Euclidean distance. Model B defines an ideal point that depends on the context only, also as a convex combination of prototypical ideal points. Unlike RFM, Model B computes response features separately and compares them to the context ideal point using the cosine similarity. Both Model A and Model B constrain the user-specific parameters to lie on a simplex whose number of vertices becomes an extra hyper-parameter (in addition to the dimension d of the ideal points). This presents an additional optimisation challenge and, as the authors themselves point out, does not generalise to users whose ideal point would fall outside of the convex hull. Like RFM, Go et al.’s (2024) *compositional preference model*

(CPM) expresses the learned reward function as a linear combination of features computed from the context and response. However, unlike RFM—which learns features using (7)—, the CPM approach uses handcrafted ordinal features computed by LLMs using pre-specified prompts.

Although there certainly are trade-offs between the two sub-categories above, we argue that learning weights associated with a user has some clear advantages over learning a mapping from user information to rewards. Since adaptation can be carried out using iterative methods that refine the weights with every example, these methods can be applied incrementally and incorporate from very little to a virtually infinite amount of data. We also argue that, among the methods belonging to its sub-category, RFM has the simplest architecture, as well as the easiest-to-solve and most well-understood adaptation phase (8). In addition, RFM has arguably been subjected to a higher level of experimental scrutiny at scale than its counterparts.

7. Discussion

This paper aims at providing a conceptual scaffolding for RFMs. As such, we deliberately refrained from delving too much into the technical aspects of the proposed approach, focusing instead on presenting the ideas in their simplest and clearest form. We leave open many opportunities for developing the more technical aspects of this work. For example, instead of solving (7) using vanilla gradient ascent, one could exploit the structure of the problem by using different learning rates for the features ϕ_θ and the matrix of coefficients \mathbf{W} . Another possibility is to hold ϕ_θ fixed for some time while optimising \mathbf{W} , and then do the opposite—an algorithm reminiscent of block coordinate descent (Wright, 2015). In our experiments adaptation used data uniformly

sampled from the training set, but it can probably be made more efficient if the process is formulated as an active learning problem (Baumler et al., 2023). Here, again, the fact that RFM’s adaptation can be interpreted as a logistic regression may come in handy, as one can exploit all the know-how accumulated around this problem (Yang and Loog, 2018).

We used best-of- n as a simple way to illustrate the benefits of using an RFM to modulate the outputs of an LLM. A more promising approach would be to take RFM’s features into account when training the LLM. Specifically, we can train multiple LLMs induced by different linear combinations of RFM’s features ϕ_θ resulting from (7), and later use vector w resulting from (8) to decide how to combine the LLMs’ outputs (Schaul et al., 2015; Barreto et al., 2017; 2018; 2020; Borsa et al., 2019; Jang et al., 2023).

One unsurprising point that became evident in our experiments is that the usefulness of adaptive reward models is proportional to the level of disagreement in the training data. Chen et al. (2024) argue that some of the apparent overlapping of opinions may be due to rigid rubrics in the data collection. One possibility to get around such homogeneity, regardless of its source, is to model coinciding opinions as an “average user”, and use adaptation to focus on responses with sufficient (and legitimate) disagreement. Note that the possibility of measuring agreement depends on multiple raters assessing the same examples, which may have to either be considered during data collection or circumvented using techniques like collaborative filtering (Su and Khoshgoftaar, 2009).

Throughout the paper we considered the case in which RFM was being adapted to a specific person, but the concept of “user” can be easily extended to a group of people. In fact, the concept is even more fluid than that, for the same person can play different roles across time (weekends/weekdays, holidays/working months, *etc.*) and space (office/home, different jurisdictions, *etc.*). A related point is that RFM’s adaptation procedure (8) makes it easy to continuously adapt the model to a given user. This facilitates the incremental specialisation of the model to the user, thus diluting the burden of data collection over time, and also allows for the tracking of changes in the user’s preferences.

As one last point, we note that the method presented in this paper is agnostic to which aspects of the responses RFM is adapted to. Since both training and adaptation only require pairwise response comparisons, the resulting model can in principle assess responses based on different criteria: their content, the style of the prose, the formatting of the text, the adoption of visual aids like images and videos, as well as their style, and so on. In general, this is a desirable property, since we can capture users’ preferences as a whole, without arbitrary material categorisations or the need for explicit deliberations. If we want to restrict which aspects

of responses should be used to assess them, we can enforce it by appropriately curating the pairs of responses presented to the users during training and adaptation (also see Impact Statement section).

8. Conclusion

We have introduced RFM, a reward-model architecture for RLHF specifically designed for fast adaptation to new users. RFM can be trained using pairwise response comparisons provided by humans. This results in a set of reward features that can be linearly combined to represent a user, even if their preferences are not reflected in the training data. Such an adaptation process can be formulated as a simple logistic regression—a well-understood convex classification problem. We showed how RFM can be personalised to an unknown user using a few dozen pairs of examples ranked by them. In the context of LLMs, this means that by responding to a few questions a user can have an otherwise generic model specialised to their taste.

We compared RFM with a non-adaptive baseline, which is today’s prevailing practice, and also several adaptive counterparts. When there was enough disagreement in the data, RFM either outperformed the baselines or matched their performance with a simpler architecture and more reliable training and adaptation.

In this paper we focused on the use of RFM in the context of LLMs, but the model is readily applicable to other modalities beyond language, like images, sound, and video. In fact, RFM can be used as a reward function in any scenario that can be formalised as an RLHF problem, which includes not only more general state and action spaces but also multi-turn interactions of the policy with the environment.

Impact statement

Beyond an improved user experience, adaptive reward models have great potential for positive impact. For example, they may allow for the inclusion of minority opinions and the representation of diverse viewpoints in model outputs. However, as with most technological advances, this does not come without risks. If not implemented with ethical implications in mind, the specialisation of LLMs to user preferences may result in models behaving in undesirable ways. It may also reinforce existing points of view through sycophantic behaviour, contributing to the polarisation of opinions and the creation of “echo chambers”.

There are ways to anticipate and mitigate these undesirable outcomes on the methodological, technical, and societal fronts. Methodologically, it is important to define rubrics for data collection and curation that clearly distinguish between genuine disagreement on subjective matters and denial of facts or deviations from prevalent ethical norms and scientific consensus. From a technical standpoint, the fact that a user-specialised reward model will be used to modulate the outputs of an LLM renders the LLM itself a safeguard mechanism. For example, the best-of- n approach used in this paper re-ranks the LLM’s outputs based on the adapted reward model. Consequently, if the LLM generates factually correct and ethical text, the re-ranking primarily prioritises or de-prioritises aspects of a subjective nature—like the style of the prose, for instance. Finally, on a societal level, the personalisation of LLMs should be part of, and would directly benefit from, the wider ongoing discussion regarding the deployment of this new technology.

On the positive side, adaptive reward models and the consequent personalisation of LLMs offer a great opportunity for the inclusion of diverse points of view into model outputs. Current reward models reflect the average preferences of the target population, which excludes under-represented or “outlier” preferences. In using specialised reward models to adapt LLM outputs to individuals, we create systems that can more accurately and reliably reflect the perspectives of users who hold minority views, potentially empowering them, together with everyone else, to participate more fully in social debate.

Acknowledgements

We would like to thank Tom Schaul and Benjamin Van Roy for their insightful comments and useful feedback. We are also thankful to Canfer Akbulut and Arianna Manzini for their help in the discussion regarding the potential societal impact of our work.

References

- M. G. Azar, Z. D. Guo, B. Piot, R. Munos, M. Rowland, M. Valko, and D. Calandriello. A general theoretical paradigm to understand learning from human preferences. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 4447–4455, 2024. URL <http://dblp.uni-trier.de/db/conf/aistats/aistats2024.html#AzarGPMRVC24>.
- A. Barreto, W. Dabney, R. Munos, J. Hunt, T. Schaul, H. van Hasselt, and D. Silver. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4055–4065, 2017. URL <https://arxiv.org/abs/1606.05312>.
- A. Barreto, D. Borsa, J. Quan, T. Schaul, D. Silver, M. Hessel, D. Mankowitz, A. Zidek, and R. Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 501–510, 2018. URL <https://arxiv.org/abs/1901.10964>.
- A. Barreto, S. Hou, D. Borsa, D. Silver, and D. Precup. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117(48):30079–30087, 2020. ISSN 0027-8424. doi: 10.1073/pnas.1907370117. URL <https://www.pnas.org/doi/10.1073/pnas.1907370117>.
- C. Baumler, A. Sotnikova, and H. Daumé III. Which examples should be multiply annotated? Active learning when annotators may disagree. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10352–10371, 2023. URL <https://aclanthology.org/2023.findings-acl.658/>.
- D. Borsa, A. Barreto, J. Quan, D. J. Mankowitz, H. van Hasselt, R. Munos, D. Silver, and T. Schaul. Universal successor features approximators. In *International Conference on Learning Representations (ICLR)*, 2019. URL <https://openreview.net/forum?id=S1VWjiRcKX>.
- R. A. Bradley and M. 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>.
- S. Chakraborty, J. Qiu, H. Yuan, A. Koppel, F. Huang, D. Manocha, A. S. Bedi, and M. Wang. MaxMin-RLHF: Towards equitable alignment of large language models with diverse human preferences. In *Proceedings of the International Conference on Machine Learning*, 2024.

- D. Chen, Y. Chen, A. Rege, and R. K. Vinayak. Modeling the plurality of human preferences via ideal points. In *ICML 2024 Workshop on Theoretical Foundations of Foundation Models*, 2024. URL <https://openreview.net/forum?id=qfhBieX3jv>.
- P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.
- V. Conitzer, R. Freedman, J. Heitzig, W. H. Holliday, B. M. Jacobs, N. Lambert, M. Mossé, E. Pacuit, S. Russell, H. Schoelkopf, et al. Social choice should guide AI alignment in dealing with diverse human feedback. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.
- C. H. Coombs. Psychological scaling without a unit of measurement. *Psychological Review*, 57:145–158, 1950.
- G. Cui, L. Yuan, N. Ding, G. Yao, W. Zhu, Y. Ni, G. Xie, Z. Liu, and M. Sun. UltraFeedback: Boosting language models with high-quality feedback, 2023. URL <https://arxiv.org/abs/2310.01377>.
- J. Dai, X. Pan, R. Sun, J. Ji, X. Xu, M. Liu, Y. Wang, and Y. Yang. Safe RLHF: Safe reinforcement learning from human feedback. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.
- V. Dumoulin, D. D. Johnson, P. S. Castro, H. Larochelle, and Y. Dauphin. A density estimation perspective on learning from pairwise human preferences. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=YH3oERVYjF>. Expert Certification.
- E. Fleisig, R. Abebe, and D. Klein. When the majority is wrong: Modeling annotator disagreement for subjective tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 6715–6726, 2023. doi: 10.18653/v1/2023.emnlp-main.415. URL <https://aclanthology.org/2023.emnlp-main.415/>.
- D. Go, T. Korbak, G. Kruszewski, J. Rozen, and M. Dymetman. Compositional preference models for aligning LMs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=tiiAzqi60l>.
- Google DeepMind. Gemma: Open models based on Gemini research and technology, 2024a. URL <https://arxiv.org/abs/2403.08295>.
- Google DeepMind. Gemma 2: Improving open language models at a practical size, 2024b. URL <https://arxiv.org/abs/2408.00118>.
- Google DeepMind, Gemini Team. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. URL <https://arxiv.org/abs/2312.11805>. [Online; accessed December 2024].
- N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2790–2799, 2019. URL <https://proceedings.mlr.press/v97/houlsby19a.html>.
- D. Hsu and A. Mazumdar. On the sample complexity of parameter estimation in logistic regression with normal design. In *Annual Conference on Learning Theory*, 2024.
- J. Jang, S. Kim, B. Y. Lin, Y. Wang, J. Hessel, L. Zettlemoyer, H. Hajishirzi, Y. Choi, and P. Ammanabrolu. Personalized soups: Personalized large language model alignment via post-hoc parameter merging. *arXiv preprint arXiv:2310.11564*, 2023. URL <https://arxiv.org/abs/2310.11564>.
- J. Kim and Y. Yang. Few-shot personalization of LLMs with mis-aligned responses, 2025. URL <https://arxiv.org/abs/2406.18678>.
- X. Li, Z. C. Lipton, and L. Leqi. Personalized language modeling from personalized human feedback. *arXiv preprint arXiv:2402.05133*, 2024. URL <https://arxiv.org/abs/2402.05133>.
- P. M. Long. On the sample complexity of PAC learning half-spaces against the uniform distribution. *IEEE Transactions on Neural Networks*, 6(6):1556–1559, 1995. ISSN 1045-9227. doi: 10.1109/72.471352. URL <https://doi.org/10.1109/72.471352>.
- OpenAI. GPT-4o system card, 2024a. URL <https://arxiv.org/abs/2410.21276>.
- OpenAI. GPT-4 technical report, 2024b. URL <https://arxiv.org/abs/2303.08774>.
- Y. Plan, R. Vershynin, and E. Yudovina. High-dimensional estimation with geometric constraints. *Information and Inference: A Journal of the IMA*, 6(1):1–40, 09 2016. ISSN 2049-8764. doi: 10.1093/imaiai/iaw015. URL <https://doi.org/10.1093/imaiai/iaw015>.
- S. Poddar, Y. Wan, H. Ivison, A. Gupta, and N. Jaques. Personalizing reinforcement learning from human feedback with variational preference learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL <https://arxiv.org/abs/2408.10075>.

- C. Poth, H. Sterz, I. Paul, S. Purkayastha, L. Engländer, T. Imhof, I. Vulić, S. Ruder, I. Gurevych, and J. Pfeiffer. Adapters: A unified library for parameter-efficient and modular transfer learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 149–160, 2023. URL <https://aclanthology.org/2023.emnlp-demo.13/>.
- A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.
- R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- P. Ramachandran, P. J. Liu, and Q. V. Le. Unsupervised pre-training for sequence to sequence learning. *arXiv preprint arXiv:1611.02683*, 2016. URL <http://arxiv.org/abs/1611.02683>.
- S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8119–8127, 2018.
- T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal Value Function Approximators. In *International Conference on Machine Learning (ICML)*, pages 1312–1320, 2015.
- A. Siththaranjan, C. Laidlaw, and D. Hadfield-Menell. Distributional preference learning: Understanding and accounting for hidden context in RLHF. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.
- X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009(1):421425, 2009. doi: <https://doi.org/10.1155/2009/421425>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1155/2009/421425>.
- R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018. URL <https://mitpress.mit.edu/books/reinforcement-learning-second-edition>. 2nd edition.
- J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned language models are zero-shot learners, 2022. URL <https://arxiv.org/abs/2109.01652>.
- S. J. Wright. Coordinate descent algorithms, 2015. URL <https://arxiv.org/abs/1502.04759>.
- Y. Yang and M. Loog. A benchmark and comparison of active learning for logistic regression. *Pattern Recognition*, 83:401–415, 2018. ISSN 0031-3203. doi: 10.1016/j.patcog.2018.06.004. URL <http://dx.doi.org/10.1016/j.patcog.2018.06.004>.
- B. Yao, Z. Cai, Y.-S. Chuang, S. Yang, M. Jiang, D. Yang, and J. Hu. No preference left behind: Group distributional preference optimization. *arXiv preprint arXiv:2412.20299*, 2024. URL <https://arxiv.org/abs/2412.20299>.
- Z. Zhang, R. A. Rossi, B. Kveton, Y. Shao, D. Yang, H. Zamani, F. Dernoncourt, J. Barrow, T. Yu, S. Kim, et al. Personalization of large language models: A survey. *arXiv preprint arXiv:2411.00027*, 2024. URL <https://arxiv.org/abs/2411.00027>.
- S. Zhao, J. Dang, and A. Grover. Group preference optimization: Few-shot alignment of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

A. Appendix: details of the experiments and additional results

We tried to keep our experimental setup as simple as possible to provide a realistic estimate of out-of-the-box performance. In particular, we did not carry out an extensive search over network architectures or hyper-parameters, keeping most of them at sensible defaults from the outset. We also did not specialise the training procedure to RFM’s architecture, solving both (7) and (8) using standard gradient ascent.

As mentioned in the main text, we used Google DeepMind’s (2024a) Gemma 1.1 2B to implement RFM and the three main baselines: the non-adaptive baseline, the adaptive baseline, and the adaptive linear baseline. The maximum context and response lengths were set to $\ell_x = \ell_y = 1,525$ tokens. Training and adaptation were carried out using gradient ascent with a learning rate of 10^{-5} . We evaluated the models using the weights computed at the end of training. The only exception is the experiment with reward models as raters (Section 5.2): in this case we carried out a random 90%–10% split of the training set and used the error in the smaller subset as a criterion to select the model to undergo adaptation.

Training was carried out for 6,000 parameter updates with a batch size of 32. This means that the training procedure went over the entire UltraFeedback training set approximately three times. Each time the example (x_i, y_i, y'_i) was encountered a new rater $\mathbf{w}_{\hat{h}_k}$ was sampled uniformly at random from $\hat{\mathcal{H}}$, as explained below.

A.1. Feature-based raters

Scenario 1. We used training raters sampled from four Gaussian distributions whose means are the “one-hot raters”:

$$\begin{aligned} \boldsymbol{\mu}_1 &= [1, 0, 0, 0], \boldsymbol{\mu}_2 = [0, 1, 0, 0], \\ \boldsymbol{\mu}_3 &= [0, 0, 1, 0], \text{ and } \boldsymbol{\mu}_4 = [0, 0, 0, 1]. \end{aligned} \quad (10)$$

The covariance of all four normal distributions was set to $0.3\mathbf{I}$, where \mathbf{I} is the 4×4 identity matrix. That is, we have $\mathcal{N}_j \equiv \mathcal{N}(\boldsymbol{\mu}_j, 0.3\mathbf{I})$ for $j = 1, 2, 3, 4$, where \mathcal{N} is the normal distribution.

We sampled 30 raters $\mathbf{w}_{\hat{h}_k} \sim \mathcal{N}_j$, $k = 1, 2, \dots, 30$, for each $j = 1, 2, 3, 4$, totaling 120 raters. During training, for each example (x_i, y_i, y'_i) , we drew a number k uniformly at random from $\{1, 2, \dots, 120\}$ and set $z_i^k = \mathbb{1}\{\langle \phi(x_i, y_i), \mathbf{w}_{\hat{h}_k} \rangle > \langle \phi(x_i, y'_i), \mathbf{w}_{\hat{h}_k} \rangle\}$.

The metric we report for training (Figure 1) is the *intra-user test accuracy*, an estimate of the fraction of examples in the test set correctly classified by the models. To compute it, we went over the test set once and, for each example, we sampled an integer k uniformly at random from $\{1, 2, \dots, 120\}$

and checked if

$$\begin{aligned} \mathbb{1}\{\langle \phi_\theta(x_i, y_i), \mathbf{w}_k \rangle > \langle \phi_\theta(x_i, y'_i), \mathbf{w}_k \rangle\} &= \\ = \mathbb{1}\{\langle \phi(x_i, y_i), \mathbf{w}_{\hat{h}_k} \rangle > \langle \phi(x_i, y'_i), \mathbf{w}_{\hat{h}_k} \rangle\} &= z_i^k, \end{aligned} \quad (11)$$

where \mathbf{w}_k is the current value of the k -th row of the matrix \mathbf{W} used for RFM during training (cf. Section 4.1).

We repeated the training process twice for the baseline and RFM(d) with $d = 8, 32, 128$. All the numbers reported are averages over these two runs.

To assess the ability of the models to adapt to unseen individuals, we used three held-out users:

$$\mathbf{w}_{h_1} = [1, 0, 0, 0], \mathbf{w}_{h_2} = [1, 0, 0, 1], \mathbf{w}_{h_3} = [1, 0, -1, 1]. \quad (12)$$

The metric reported for adaptation (Figures 2, 3, and 5) is the *inter-user test accuracy*. It was calculated analogously to the intra-user test accuracy described above, but using held-out users instead of raters. That is, for each held-out user h_k in (12), we went over the test set once and computed (11) using \mathbf{w}_{h_k} instead of $\mathbf{w}_{\hat{h}_k}$.

We assessed RFM’s inter-user test accuracy using different numbers of examples to do the adaptation, as explained in the main text. During training, for each $d \in \{8, 32, 128\}$ we computed two feature functions ϕ_θ . For each $m \in \{10, 30, 50, 70, 90\}$ we carried out 5 independent adaptation runs with each of the 6 feature functions computed during training. We carried out 1,000 parameter updates and estimated the inter-user test accuracy using the resulting weights.

Scenario 2. For Scenario 2 we used the same protocol used in Scenario 1, but now we sampled training raters from different distributions. We defined a mean vector for each possible instantiation of a 4-dimensional vector with one element equal to 1, one element equal to -1 , and the remaining elements equal to zero, that is:

$$\begin{aligned} \boldsymbol{\mu}_1 &= [1, -1, 0, 0], \boldsymbol{\mu}_2 = [1, 0, -1, 0], \\ \boldsymbol{\mu}_3 &= [1, 0, 0, -1], \boldsymbol{\mu}_4 = [-1, 1, 0, 0], \dots \\ &\dots, \boldsymbol{\mu}_{12} = [0, 0, -1, 1]. \end{aligned} \quad (13)$$

Analogously to what we did before, we defined 12 normal distributions $\mathcal{N}_j(\boldsymbol{\mu}, 0.3\mathbf{I})$. We sampled 10 training raters $\mathbf{w}_{\hat{h}_k} \sim \mathcal{N}_j(\boldsymbol{\mu}, 0.3\mathbf{I})$ from each normal distribution \mathcal{N}_j , totaling again 120 training raters.

The rationale of this experiment is to see how the methods perform with a slightly more complex set of training raters whose preferences take more than one feature into account.⁴

⁴The negative weights are used here for didactic purposes only. We want to illustrate how disagreement between the raters may result in poor performance of methods that do not distinguish be-

If we think of the means μ in (13) as vectors, we see that for each μ_j there is a counterpart $-\mu_j$ pointing in the opposite direction. This induces considerable disagreement among the resulting raters w_{h_k} , making the problem very difficult for the baseline. We expect this type of disagreement to occur in practice, at least in some contexts.

Scenario 3. The experiments in Scenario 3 were carried out exactly as those in Scenario 2, except that the UltraFeedback features ϕ were replaced by simple-to-compute feature functions ϕ' . We defined four functions:

- $\phi'_1(x, y)$: the length of y .
- $\phi'_2(x, y)$: the number of adjectives in y .
- $\phi'_3(x, y)$: the number of alliterations in y .
- $\phi'_4(x, y)$: the number of words in y that also occur in x .

All the features were normalised to fall in the interval $[0, 1]$. These features were defined somewhat arbitrarily as simple illustrations of possibly conflicting subjective criteria.

Comparison with in-context methods. We present the prompt used to evaluate the in-context capabilities of Gemini 1.5 Pro and GPT-4o in Figure 6. It has three main parts: (i) the system instructions, delimited by [System] and [End of system]; (ii) a sequence of 10 previous user ratings (whose formatting is described in the system instructions); and (iii) the prompt, first response, and second response to be assessed by the LLM.

Our in-context evaluation loops over each held-out user h_i , and for each example in the test set we draw 10 examples of responses previously ranked by h_i . The previously ranked responses are inserted in part (ii) of the prompt template, and the test example itself is inserted in part (iii) of the prompt template. We then compare the predicted comparison outcome against user h_i 's ranking for that test example.

We also evaluate Gemini's zero-shot agreement with the held-out users (*Gemini (zero-shot)*) to help assess whether adding previously ranked responses as context makes a difference in terms of performance. In that setting, we omit the paragraph starting with "We will provide a few examples" in part (i) of the prompt template and remove part (ii) altogether.

Comparison with VPL. Poddar et al.'s (2024) evaluation protocol for VPL allows to measure intra-user generalization, but not inter-user generalization. This is because the

tween them, while methods that do make this distinction perform well. In a real scenario the raters' responses will reflect their preferences and the data collection rubric. See the Impact Statement section for a discussion on preventive measures against misaligned reward models and LLMs.

preferences used for evaluation are obtained from the same four raters that were used to train the model. In terms of evaluating intra-user generalization, VPL relies on an episodic protocol: for every rater \hat{h}_i and for every tuple (x, y, y', z) in the rater's test set, the authors simulate a new adaptation problem by drawing between two to eight other tuples sampled at random from the rater's test set to provide as context for the inference network to make a prediction. This means that the rater's identity \hat{h}_i is never explicitly revealed to the model, but only contextually through the two to eight other tuples. In contrast, our evaluation methodology is more akin to transfer learning evaluation: for each rater, we adapt RFM on a held-out set of examples. The *same* held-out set is used for all test examples, that is, we learn w once using the held-out data and then use it to process all the test tuples (x, y, y', z) .

The VPL authors provide the exact data used for evaluation on UltraFeedback.⁵ For simplicity, we adapt our evaluation protocol so as to get an unbiased estimate of the episodic metric used to evaluate VPL. We hold out 250 examples from VPL's training set to sample adaptation sets. After RFM has been trained on the four one-hot UltraFeedback raters, we loop over raters, number $n \in \{2, 3, 4, 5, 6, 7, 8\}$ of examples used for adaptation, and 5 random seeds, each time drawing n examples from the held-out set for that rater, discarding the pre-trained set of weights w for that rater, and learning a new w on the sampled adaptation set. We then aggregate the accuracies measured on the test set across adaptation set sizes, raters, and random seeds (see Figure 7 for non-aggregated results). As a result, the examples used for adaptation for each individual test example do not correspond exactly to the ones used by VPL (and are shared across test examples), but they are equally disjoint from the training set and their distribution is i.i.d. with respect to the distribution VPL samples from.

We obtain an averaged test accuracy of 61.61% against VPL's 61.49%. Given the methodological caveats outlined above, the conclusion we draw is that, despite its simplicity, RFM achieves an intra-user generalization performance comparable with VPL's. We again highlight that this notion of generalisation is distinct from inter-user generalisation, which is the main one we are targeting in this work, but for which VPL does not provide metrics to compare against.

A.2. Reward models as raters

In the experiments described in Section A.1 we had control over the definition of the training raters. In the experiments with reward models we removed this assumption. We used the following 8 publicly-available reward models to emulate

⁵https://github.com/WEIRDLabUW/vpl_llm?tab=readme-ov-file#data-and-pretrained-models

```
[System]
Please act as an impartial judge and evaluate the quality of the responses provided by
two AI assistants to the user question displayed below.

A rating starts with the tag [User question] followed by the context and the tag [End of
user question]. After that, we have the tag [The Start of Model A's answer] followed by
the first response. The first response ends with the tag [The End of Model A's answer].
We then have the tag [The Start of Model B's answer] followed by the second response. The
second response ends with the tag [The End of Model B's answer].

We will provide a few examples of previous ratings to help you understand the task. The
example ratings will have the structure above followed by the tag [Verdict], the verdict
("[[A]]" if assistant A is better, and "[[B]]" if assistant B is better), and the tag
[End of verdict].

We are interested in your evaluation of the last two responses. Begin your evaluation by
comparing these two responses and provide a short explanation. Avoid any position biases
and ensure that the order in which the responses were presented does not influence your
decision. Do not allow the length of the responses to influence your evaluation. Do not
favor certain names of the assistants. Be as objective as possible. After providing your
explanation, output your final verdict by strictly following this format: "[[A]]" if
assistant A is better, and "[[B]]" if assistant B is better.
[End of system]

[User question]
<...>
[End of user question]

[The Start of Model A's answer]
<...>
[The End of Model A's answer]

[The Start of Model B's answer]
<...>
[The End of Model B's answer]

[Verdict]
<...>
[End of verdict]

<...>

[User question]
<...>
[End of user question]

[The Start of Model A's answer]
<...>
[The End of Model A's answer]

[The Start of Model B's answer]
<...>
[The End of Model B's answer]
```

Figure 6. The prompt template used for in-context evaluation. When evaluating Gemini’s zero-shot capabilities we omit the paragraph starting with “We will provide a few examples” in the prompt.

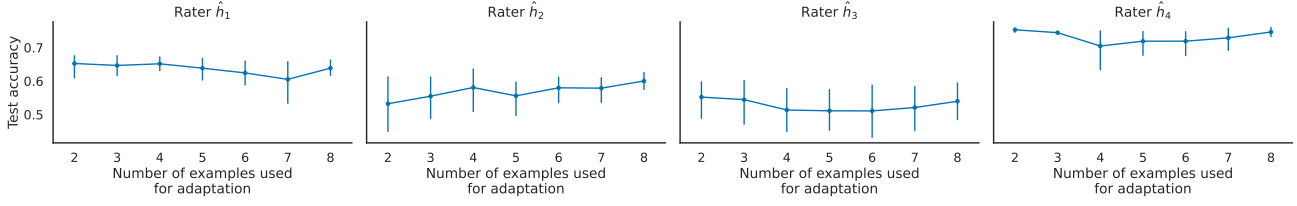


Figure 7. Test accuracies for RFM(32) on the UltraFeedback-derived UF-P-4 task introduced by Poddar et al. (2024), broken down by rater and number of examples used for adaptation. The error bars represent 95% confidence intervals over 5 randomly-sampled adaptation sets.

raters and users:

- OpenAssistant_reward-model-deberta-v3-large-v2,
- weqweasdas_RM-Mistral-7B,
- OpenAssistant_oasst-rm-2.1-pythia-1.4b-epoch-2.5,
- Ray2333_GRM-Gemma-2B-sftreg,
- Ray2333_reward-model-Mistral-7B-instruct-Unified-Feedback,
- weqweasdas_RM-Gemma-7B,
- internlm_internlm2-7b-reward,
- openbmb_Eurus-RM-7b.

All the models above are available in the Hugging Face website.⁶ For each reward model r_k and each example (x_i, y_i, y'_i) in the training and test sets, we defined $z_i^k = \mathbb{1}\{r_k(x_i, y_i) > r_k(x_i, y'_i)\}$. We then performed leave-one-old cross validation using the 8 resulting raters. That is, we carried out 8 rounds of evaluation in which 7 of the reward models played the role of the raters $\hat{\mathcal{H}}$ and the remaining reward model played the role of the held-out user.

⁶<https://huggingface.co/>.