Personalized Language Modeling from Personalized Human Feedback

Anonymous authors
Paper under double-blind review

Abstract

Reinforcement Learning from Human Feedback (RLHF) is the current dominating framework to fine-tune large language models to better align with human preferences. However, the underlying premise of algorithms developed under this framework can be problematic when user preferences encoded in human feedback are diverse. In this work, we aim to address this problem by developing methods for building personalized language models. We propose a general Personalized-RLHF (P-RLHF) framework, which requires one to jointly learn a user model and a language (or reward) model. We develop new learning objectives for personalized reward modeling and personalized Direct Preference Optimization. To demonstrate the efficacy of our method, we test it on real-world text summarization data with annotated preferences and annotator information. We fine-tune GPT-J 6B to obtain personalized language (and reward) models, which outperform non-personalized models in terms of aligning with individual preferences.

1 Introduction

Reinforcement Learning from Human Feedback (RLHF) is a widely adopted framework to align pre-trained large language models (LMs) with human preferences (Ziegler et al., 2019). In the RLHF pipeline, human users provide their feedback consisting of their rankings of two (or more) responses for different prompts. The LM is subsequently fine-tuned using this data either directly (e.g., through Direct Preference Optimization (Rafailov et al., 2023)) or by first learning a user preference model (a.k.a. reward model) and then optimizing the LM against it. The goal for RLHF is to align LM behaviors with user preferences encoded in their feedback.

Human preferences are inherently diverse and subjective. Current dominating RLHF approaches *implicitly* assume that all crowdsourced human feedback (and preference) comes from the same distribution and obscure the inter-user variations when modeling human preferences (Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022; Rafailov et al., 2023). This assumption will potentially lead to reward models prioritizing the preferences of the majority group in the data while neglecting the preferences of the minorities (Casper et al., 2023; Feffer et al., 2023; Fleisig et al., 2023; Prabhakaran et al., 2021). Consequently, LMs fine-tuned with such reward models may provide responses that do not align with the preferences of users outside the majority (Santurkar et al., 2023).

To prevent such misalignments, the diversities in human preferences need to be explicitly accounted for in RLHF (Kirk et al., 2023). In this paper, we propose a general *personalized RLHF (P-RLHF)* framework. Our proposed framework jointly learns a user model that captures individual user preference and a language (or reward) model for personalized language generation. Under the P-RLHF framework, we develop new learning objectives for performing personalized reward modeling (P-RM, Section 2.2) and personalized direct preference optimization (P-DPO, section 2.3). Using a real-world preference dataset for text summarization Stiennon et al. (2020) and GPT-J 6B as the base LM, we show that P-RM and P-DPO can effectively learn personalized LMs, improving the alignment between LM behavior and individual user preferences (Section 3).

2 Learning from Personalized Human Feedback

A personalized human feedback (or preference) dataset $\mathcal{D}_{\mathrm{p}} = \{(x_i, y_{i,1}, y_{i,2}, u_i)\}_{i=1}^{n}$ consists of n samples where $u_i \in \mathcal{U}$ is the information of the user who annotates the data or provides the

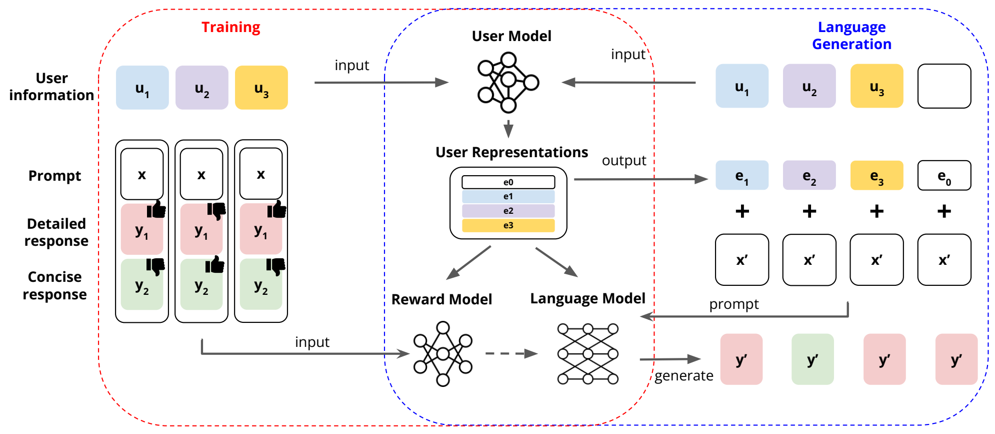

Figure 1: Our **Personalized RLHF** framework. For training, user information (u_1, u_2, u_3) and preference data (e.g. the preferences for detailed or concise responses) are collected from each user. The user model maps the user information into user representations (user-specific embedding e_1, e_2, e_3 and generic e_0 for unknown user information), which are learned jointly with the reward model (or the language model in DPO) using the preference data. During language generation, for users seen during training, the responses tailored to their individual preferences are generated based on their own user representations (e_1, e_2, e_3) and the prompt (x'), while for new users unseen during training, the responses are generated using the generic user representation (e_0) and the prompt (x').

preferences, x_i is the prompt, $y_{i,1}$ and $y_{i,2}$ are two generated texts such that $y_{i,1}$ is preferred over $y_{i,2}$ (i.e., $y_{i,1} \succ y_{i,2}$) under the prompt x_i. In general, the space of user information $\mathcal{U}$ can contain the user identifiers, user demographics, their historical texts, etc. As a running example, we consider cases where the user information u_i is the user id, i.e., $\mathcal{U} = \{0\} \cup [m]$ where m is the maximum number of known user ids, and we set $u_i = 0$ when the user id is unknown.[1]

Given $\mathcal{D}_\text{p}$, to obtain a personalized LM π_p via personalized RLHF (P-RLHF), one may take different approaches including: (1) first learn a personalized reward model (RM) and then optimize the LM against it, similar to the vanilla RLHF pipeline (Section 2.2); or (2) directly learn a personalized LM by adapting from the vanilla DPO procedure (Section 2.3). We denote their corresponding methods P-RM and P-DPO, respectively. Similar to the vanilla RLHF, when building personalized RMs or LMs, we start with a base LM, often times, a supervised fine-tuned LM π^{SFT} that takes in texts (e.g., x for the LM or (x, y) for the RM). There are two additional components to specify:

- a learnable **User Model** f_P that extracts a user embedding e_u, which is a tensor, from the user information u (e.g., user identifiers). For all $u \in \mathcal{U}$, a user embedding is given by $e_u = f_\text{P}(u)$;
- an **Aggregator** function that combines the user embedding e_u with the text embedding e_x (or $e_{x,y}$ depending on whether the input is x or (x, y)). The text embedding is given by the base LM. For example, it can be the input embedding or the last hidden state for the text. The aggregator outputs a combined embedding for the user and text, to generate personalized language or rewards.

Below we first provide some examples of user models and then separately discuss how the aggregator may be specified for P-RM and P-DPO . We will also present new learning objectives for them.

2.1 User Models

The structure of a user model f_P encodes one's *preference assumptions* on how different users' preferences are related to each other. In the following, we illustrate how f_P can be defined.

Example 1 (Uniform Preference). *For all $u \in \mathcal{U}$, the user model $f_\text{P}(u) = e$ outputs the same embedding (e can be an empty tensor).*

Example 2 (Individualized Preference). *Let $\mathcal{U} = \{0\} \cup [m]$ be the set of user indices. The user model outputs $f_P(0) = e_0$ for (unknown) users indexed by 0. For all $u \in [m]$, the user model outputs $f_P(u) = e_0 + o_u$ where o_u is a user-specific offset tensor.*

[1]We are only able to find publicly available *human* annotated preference learning datasets where $\mathcal{U}$ are the user ids.

The individualized user model assumes that each user u has their individualized preference offset o_u while maintaining a component e_0 shared across users . The common tensor e_0 can be understood as the commonality across user preferences concerning topics like factuality and safety.

Example 3 (Cluster-based Preference)**.** *Let $\mathcal{U} = \{0\} \cup [m]$ be the set of user indices. For all $u \in \mathcal{U}$, the user model output $f_P(u) = V \cdot w_u$ where $V \in \mathbb{R}^{K \times d}$, K is the number of clusters, and $w_u \in \mathbb{R}^K$ is a weight vector for each user.*

Inspired by the crowdsourcing literature (Imamura et al., 2018), we develop this clustering-based user model that assumes user embeddings (and hence preferences) span a common set of vectors given by V; each user embedding is a weighted combination of these vectors.

2.2 Personalized RM for Personalized LM

Given the *learnable* user model f_P, we have a user embedding e_u for each user $u \in \mathcal{U}$. Our next task is to decide how we want to include it into the personalized RM r_p. We discuss two approaches: (1) use e_u as a soft prompt; or (2) when e_u is a vector, use e_u as a linear head.

In the case of soft prompting, the aggregator prepends e_u to the input (text not positional) embedding $e_{x,y} \in \mathbb{R}^{T_{x,y} \times d}$ given by the base LM, where $T_{x,y}$ is the token length and d is the token-wise embedding dimensionality. The user embedding $e_u \in \mathbb{R}^{T_u \times d}$ is a tensor with T_u being its corresponding user token length. One factor that controls the expressivity of user embeddings is the size of their corresponding user token length T_u. Then a linear layer is added to map the last hidden state of the base LM (under the new input embedding $(e_u, e_{x,y})$) to a scalar reward value, as in the RM in the vanilla RLHF. In our experiments, we worked with this implementation.

In the case where e_u is a linear head, the aggregator function can be taken as an inner product between e_u and the hidden state $e_{x,y}$ of the last transformer layer of the base LM, thus outputting a scalar reward value. Here, the user embedding e_u serves as the additional linear head.

To learn the RM (including the user model f_P), we use the following objective:

$$\min_{r_\text{P}} -\mathbb{E}_{x,y_1,y_2,u \sim \mathcal{D}_\text{P}} \Big[\alpha \log \sigma(r_\text{P}(x, y_1, u) - r_\text{P}(x, y_2, u)) + (1-\alpha) \log \sigma(r_\text{P}(x, y_1, u_0) - r_\text{P}(x, y_2, u_0)) \Big],$$

where $\alpha \in [0, 1]$. Recall that u_0 indicates empty user information. The loss can be viewed as a combination of a user-specific loss term that relies on explicit user identifier u and a user-agnostic loss term that depends on u_0. The user-agnostic loss uses the same preference data but without any user identifier. The hyper-parameter α is used to balance between the two loss components.

2.3 Personalized DPO for Personalized LM

To directly learn a personalized LM without the reward learning step, we propose a Personalized DPO (P-DPO) procedure that adapts from vanilla DPO. Similar to the above, we need to first specify how we want to aggregate the user embedding e_u with the text embedding e_x for the personalized LM π_P. Following how we build personalized RM, we propose two possible ways: (1) treating e_u as a soft prompt (or extra tokens) by prepending it to the input texting embedding e_x; or (2) aggregate e_u and the last hidden state e_x using a linear layer for a new hidden state before outputting the final next-token probability. We adopt the first approach in our experiments.

Given the personalized LM π_P (including the user model f_P), we use the following learning objective:

$$\begin{aligned} \min_{\pi_\text{P}} -\mathbb{E}_{x,y_1,y_2,u \sim \mathcal{D}_\text{P}} \Bigg[& \alpha \log \sigma \left(\beta \log \frac{\pi_\text{P}(y_1|x,u)}{\pi^\text{SFT}(y_1|x)} - \beta \log \frac{\pi_\text{P}(y_2|x,u)}{\pi^\text{SFT}(y_2|x)} \right) \\ & + (1-\alpha) \log \sigma \left(\beta \log \frac{\pi_\text{P}(y_1|x,u_0)}{\pi^\text{SFT}(y_1|x)} - \beta \log \frac{\pi_\text{P}(y_2|x,u_0)}{\pi^\text{SFT}(y_2|x)} \right) \Bigg], \end{aligned}$$

where $\beta > 0$ controls the deviance from the policy π^SFT. The hyper-parameter $\alpha \in [0, 1]$ balances between the user-specific loss terms that depend on the user identifier u and the user-agnostic loss terms that replace u by the generic user identifier u_0 under the same user feedback data.

3 Experiments

We empirically evaluate the performance of our methods on the Reddit TL;DR summarization dataset[2] curated by Stiennon et al. (2020). In the dataset, each comparison includes a Reddit post x, two summaries y_1 and y_2, with $y_1 \succ y_2$ determined by a worker u represented by a unique worker id. As we do not have access to the SFT model used by Stiennon et al. (2020), we initialize both the personalized RM and the personalized LM using an open-source SFT[3]—a GPT-J 6B model Wang & Komatsuzaki (2021) supervised fine-tuned using the TRLX library Havrilla et al. (2023). For both P-RM and P-DPO, we experimented with user models that encode (1) individualized preference assumption (Example 2), or (2) cluster-based preference assumption with $K = 5$ (Example 3). To evaluate the effect of the user-agnostic loss, we experimented with (1) $\alpha = 1$ where only user-specific loss is used, and (2) $\alpha = 0.5$ where user-specific loss and user-agnostic loss are equally weighted.

We evaluate the RMs by their accuracy of correctly assigning higher rewards to the chosen summaries y_1 than to the rejected summaries y_2 for the validation set. For P-DPO, we evaluate the performance using the accuracy of its implicit reward function $r_{\mathrm{P}}(x, y, u) = \beta \log \frac{\pi_{\mathrm{P}}(y|x,u)}{\pi^{\mathrm{SFT}}(y|x)}$ (the learning objective of DPO can be viewed as deriving the optimal policy under this implicit reward function). We compare their performances against the vanilla RLHF RM and vanilla DPO baselines which assume preference uniformity across users. Specifically, we evaluate three accuracies:

- **Accuracy-top**: the accuracy of all comparisons annotated by known workers in the training set.
- **Accuracy-average**: the average and standard deviation of per-user accuracy of known workers.
- **Accuracy-generic**: the accuracy of comparisons annotated by unknown workers (id 0).

The performances of all RMs and DPO models are listed in Table 1. With $\alpha = 0.5$ and individualized user model, the P-RM and P-DPO models achieve the highest accuracy for known workers, outperforming the vanilla RM and DPO baselines. A similar trend is observed for accuracy-average. P-RM and P-DPO also achieve higher accuracies on unknown workers than the baselines, and the two RMs with cluster-based user models also achieve performances on par or better than the baseline. These results demonstrate that based on our user preference assumption, P-RM and P-DPO are able to effectively accommodate the personal preferences of known users, while also learning a meaningful generic user representation which improves the alignment between the LM and unknown users.

Table 1: P-RM and P-DPO performances. All accuracies (ACC) are in %.

		P-RM Performance			P-DPO Performance		
User Model	α	Acc top	Acc average	Acc generic	Acc top	Acc average	Acc generic
Vanilla	N/A	60.27	61.10 ± 3.44	59.20	60.48	60.57 ± 3.14	60.99
Individual	1.0	58.59	58.76 ± 2.91	57.10	60.07	60.13 ± 3.75	60.53
Individual	**0.5**	**61.72**	$\mathbf{62.09 \pm 3.51}$	**60.65**	**61.33**	$\mathbf{61.61 \pm 3.49}$	**61.97**
Cluster 5	1.0	58.79	59.72 ± 3.76	59.94	59.24	60.20 ± 4.69	59.58
Cluster 5	0.5	59.66	60.19 ± 2.95	59.90	60.98	61.46 ± 4.77	61.27

For both the individualized and cluster-based user models, $\alpha = 0.5$ achieve a better performance than $\alpha = 1.0$. This demonstrates the importance of the user-agnostic loss term in our learning objective, which emphasizes the commonality in user preferences and serves as a natural regularization for preventing the user model from overfitting to any individual user. Further details of the dataset, experiments, and additional results are provided in Appendix A.

4 Conclusions

To enable building personalized LMs accounting for diverse human preferences, we propose P-RLHF—a personalized RLHF framework for working with preference data that may contain user information. Our framework jointly learns a user model which captures structural preference assumptions and a LM (or RM). Through our experiments on a real-world preference dataset for text summarization, we have demonstrated the effectiveness of our framework in aligning LMs with individual user preferences seen during training and generalizing to unseen users with the generic user representation.

[2]`https://huggingface.co/datasets/openai/summarize_from_feedback`
[3]`https://huggingface.co/CarperAI/openai_summarize_tldr_sft`

REFERENCES

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28 (1):20–28, 1979.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*, 2022.

Michael Feffer, Hoda Heidari, and Zachary C Lipton. Moral machine or tyranny of the majority? *arXiv preprint arXiv:2305.17319*, 2023.

Eve Fleisig, Rediet Abebe, and Dan Klein. When the majority is wrong: Modeling annotator disagreement for subjective tasks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6715–6726, 2023.

Amelia Glaese, Nat McAleese, Maja Trebacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*, 2022.

Hayit Greenspan, Bram Van Ginneken, and Ronald M Summers. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE transactions on medical imaging*, 35(5):1153–1159, 2016.

Alexander Havrilla, Maksym Zhuravinskyi, Duy Phung, Aman Tiwari, Jonathan Tow, Stella Biderman, Quentin Anthony, and Louis Castricato. trlX: A framework for large scale reinforcement learning from human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8578–8595, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.530. URL `https://aclanthology.org/2023.emnlp-main.530`.

EunJeong Hwang, Bodhisattwa Prasad Majumder, and Niket Tandon. Aligning language models to user opinions. *arXiv preprint arXiv:2305.14929*, 2023.

Hideaki Imamura, Issei Sato, and Masashi Sugiyama. Analysis of minimax error rate for crowdsourcing and its application to worker clustering model. In *International Conference on Machine Learning*, pp. 2147–2156. PMLR, 2018.

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

Hannah Rose Kirk, Bertie Vidgen, Paul Röttger, and Scott A Hale. Personalisation within bounds: A risk taxonomy and policy framework for the alignment of large language models with personalised feedback. *arXiv preprint arXiv:2303.05453*, 2023.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

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

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. Memprompt: Memory-assisted prompt editing with user feedback. 2022.

Joshua Maynez, Priyanka Agrawal, and Sebastian Gehrmann. Benchmarking large language model capabilities for conditional generation. *arXiv preprint arXiv:2306.16793*, 2023.

Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, et al. Teaching language models to support answers with verified quotes. *arXiv preprint arXiv:2203.11147*, 2022.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

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.

Vinodkumar Prabhakaran, Aida Mostafazadeh Davani, and Mark Diaz. On releasing annotator-level labels and information in datasets. *arXiv preprint arXiv:2110.05699*, 2021.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. Grips: Gradient-free, edit-based instruction search for prompting large language models. *arXiv preprint arXiv:2203.07281*, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 2023.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11(43): 1297–1322, 2010. URL `http://jmlr.org/papers/v11/raykar10a.html`.

Filipe Rodrigues and Francisco Pereira. Deep learning from crowds. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Shibani Santurkar, Esin Durmus, Faisal Ladhak, Cinoo Lee, Percy Liang, and Tatsunori Hashimoto. Whose opinions do language models reflect? *arXiv preprint arXiv:2303.17548*, 2023.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

Rion Snow, Brendan O'connor, Dan Jurafsky, and Andrew Y Ng. Cheap and fast–but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 conference on empirical methods in natural language processing*, pp. 254–263, 2008.

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

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`, 2020.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`, May 2021.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. Fine-grained human feedback gives better rewards for language model training. *arXiv preprint arXiv:2306.01693*, 2023.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Experiment Details and Additional Results

A.1 OpenAI Summarize from Feedback Dataset

We use the Reddit TL;DR summarization dataset[4] curated by Stiennon et al. (2020) for our experiments. In the dataset, each comparison includes a Reddit post x, two summaries y_1 and y_2, with $y_1 \succ y_2$ determined by a worker u represented by a unique worker id. The summaries were sampled from a wide variety of policies and the preferences over summary pairs were crowdsourced from multiple workers. In addition, the dataset also includes how y_1 and y_2 are sampled, e.g., from prior SFT or PPO checkpoints. To ensure that the summaries are close to the distribution of the SFT we use, we only include the comparisons where both y_1 and y_2 are noted as sampled from the SFT models in the dataset, and exclude comparisons which contain summaries sampled from other policies such as different PPO checkpoints.

To ensure effective learning of individual user preferences with ample samples, we rank the workers based on the number of annotated comparisons in the train split of the dataset and include the top 10 workers for training, resulting in $23,292$ comparisons in the training set. For evaluation, we utilize all $16,294$ comparisons in the validation split, among which $4,921$ are annotated by the same 10 workers in the training set and the remaining $11,373$ are annotated by unseen workers. The number of comparisons annotated by each worker in the training set and the validation set are listed in Table 2.

Table 2: Number of comparisons annotated by each worker in the training set and the validation set.

Worker id	Num of training comparisons	Num of validation comparisons
1	$5,244$	$1,175$
2	$2,919$	643
3	$2,435$	381
4	$2,366$	604
5	$2,200$	586
6	$2,125$	360
7	$1,823$	321
8	$1,606$	263
9	$1,365$	177
10	$1,209$	411
0	0	$11,373$

A.2 User Model Implementation

In our experiments, we take the worker id as user information u. In descending order of the number of annotated comparisons, workers in the training set are denoted by ids $1, 2, \dots, 10$, and all unknown workers in the validation set are denoted by id 0.

We implement the user model as a look-up (embedding) table which maps a user id to a single-token user embedding $e_u \in \mathbb{R}^d$, where d is the same as the dimensionality of the input text embedding of π^{SFT}. The P-RM and P-DPO both prepend the user embedding to the text embedding similarly as "soft prompts" Lester et al. (2021), and then the concatenated input embeddings flow through the transformer as usual. We leave other implementations of the user model, such as taking textual user description or historical data as user information input, and other methods to aggregate the user embedding and the text embedding as interesting directions for future work.

A.3 P-RM Experiment Details

All the reward models (RMs) are initialized from an SFT with a randomly initialized linear reward head. The SFT we use is an open-source GPT-J 6B model[5] fine-tuned with the Reddit TL;DR reference summaries. In our user model, we utilize $T_u = 1$ user tokens for each user, and the user

[4]`https://huggingface.co/datasets/openai/summarize_from_feedback`

[5]`https://huggingface.co/CarperAI/openai_summarize_tldr_sft`

embedding $e_u \in \mathbb{R}^d$ is prepended to the input text embeddings $e_{x,y} \in \mathbb{R}^{T_{x,y} \times d}$, where $T_{x,y}$ and d are the tokenized input sequence length and the dimension of the embedding space of the language model (LM). This results in a concatenated input embedding $(e_u, e_{x,y}) \in \mathbb{R}^{(T_{x,y}+1) \times d}$. Following the initialization of the soft prompt tokens in Lester et al. (2021), we initialize the user embeddings to word embeddings randomly sampled from the vocabulary of the LM, which we empirically verify leads to faster convergence than random initialization.

All RMs, including the P-RM models and the vanilla baseline, are trained using AdamW Loshchilov & Hutter (2017) optimizer with learning rate $5e$-6 and batch size 64 for 1 epoch. The learning rate is linearly warmed up from 0 to $5e$-6 over 150 steps. All reward models are trained using a PyTorch based, personalized Reward Trainer we develop based on the Reward Trainer in the TRL library von Werra et al. (2020).

A.4 P-DPO Experiment Details

All the LMs in P-DPO experiments are initialized to the same SFT as in P-RM experiments, and the user embeddings are prepended and initialized in the same fashion. Following the experiment setup in Rafailov et al. (2023), we set $\beta = 0.5$ for the TL;DR dataset, and all the DPO models are trained with learning rate $1e$-6 and batch size 64 for 1 epoch, with the learning rate warmed up from 0 to $1e$-6 over 150 steps. All DPO models are trained with a PyTorch based, personalized DPO Trainer we develop by extending the DPO Trainer in the TRL library von Werra et al. (2020).

A.5 Additional Experiment Results

In addition to the **Accuracy-average** we reported in Section 3, the accuracy of each individual worker with id $1, \ldots, 10$ for P-RM and P-DPO are provided in Tables 3 and 4. The **Accuracy-average** reported in Section 3 are computed as the mean and standard deviation of the 10 individual worker accuracies.

Table 3: The accuracies of P-RM on each individual worker. All accuracies are in $\%$.

Worker id	Vanilla RM	Individual ($a = 1.0$)	Individual ($a = 0.5$)	Cluster 5 ($a = 1.0$)	Cluster 5 ($a = 0.5$)
1	57.45	56.85	60.77	56.34	58.98
2	61.74	62.05	61.90	60.50	59.41
3	65.88	61.15	64.83	63.78	62.47
4	55.79	57.12	58.11	54.80	55.63
5	62.80	58.19	63.65	58.70	60.24
6	66.11	65.00	70.28	66.39	66.39
7	61.06	55.76	60.75	56.70	58.57
8	62.36	58.94	59.70	63.50	63.12
9	61.58	57.06	63.28	61.02	59.89
10	56.20	55.47	57.66	55.47	57.18

B Related Work

Reinforcement learning from human feedback (RLHF) RLHF optimizes LMs as RL policies to generate responses aligned with human preferences, using reward models learned from human feedback. In the RLHF literature, human feedback is typically collected by asking annotators to compare or rank multiple candidate responses. This type of data is easier to collect than demonstration data, especially for tasks where the ground truth responses may not exist due to inherent subjectivity among annotators. RLHF has been utilized to improve the LM performances on a variety of NLP tasks, including summarization (Ziegler et al., 2019; Stiennon et al., 2020), question answering (Nakano et al., 2021; Menick et al., 2022), instruction following Ouyang et al. (2022) and improving helpfulness and harmlessness (Bai et al., 2022; Glaese et al., 2022). While vanilla RLHF tends to model the reward of a whole sequence using a scalar score output by a single reward model, recent studies have imposed more sophisticated structures on reward learning, e.g., training separate reward models for

Table 4: The accuracies of P-DPO on each individual worker. All accuracies are in %.

WORKER ID	VANILLA DPO	INDIVIDUAL ($a = 1.0$)	INDIVIDUAL ($a = 0.5$)	CLUSTER 5 ($a = 1.0$)	CLUSTER 5 ($a = 0.5$)
1	59.40	59.49	59.32	54.89	56.60
2	61.90	58.79	61.43	59.56	62.83
3	64.83	66.67	66.67	63.25	64.04
4	57.62	56.79	59.11	57.12	58.94
5	60.07	61.77	62.63	60.58	64.33
6	61.39	62.50	65.28	65.83	65.28
7	60.75	61.99	60.12	63.86	65.73
8	65.02	63.12	66.16	67.68	66.92
9	53.67	52.54	54.80	52.54	50.85
10	61.07	57.66	60.58	56.69	59.12

different targets (Glaese et al., 2022), assigning fine-grained rewards to small text segments (Wu et al., 2023), or training individual reward models to capture pre-defined dimensions of user preferences and then merging the fine-tuned LMs for personalization Jang et al. (2023). The need for separate reward modeling and policy optimization makes RLHF a complex procedure and prone to instabilities during training. Direct Preference Optimization (DPO) has emerged as an RL-free algorithm. DPO circumvents the reward modeling step, directly fine-tunes the language model using the preference data, and significantly improves the training efficiency of RLHF Rafailov et al. (2023).

Our work differs from previous RLHF approaches in two ways: firstly, we model user-specific preferences instead of assuming that all users share the same preference distribution (over responses). Secondly, our personalized reward and language models are learned directly using personalized feedback data, rather than requiring additional fine-grained learning signals or pre-defined preference dimensions.

Crowdsourcing When collecting large sets of labeled data (like in the preference data collection phase of RLHF), crowdsourcing is often adopted by first dispatching the unlabeled samples to multiple annotators and then estimating the ground-truth labels by aggregating the noisy annotations (Snow et al., 2008; Greenspan et al., 2016). The observed annotations are often modeled as the confused outputs for the hidden ground-truth labels and the confusion of each annotator is characterized by an individual confusion matrix (Dawid & Skene, 1979; Raykar et al., 2010; Rodrigues & Pereira, 2018). Recent research has introduced novel methods to better capture real-world annotator behaviors. For instance, Imamura et al. (2018) modeled the confusion matrices at a cluster level to capture the shared confusion patterns among annotators. Inspired by the behavioral assumptions (on annotators) in crowdsourcing literature, we design analogous strategies to model user preferences at the population, cluster, and individual levels through different user model structures.

Conditional Natural Language Generation With the advent of autoregressive pre-trained LMs such as GPT-3 Brown et al. (2020) and PaLM (Chowdhery et al., 2022), natural language generation tasks are often performed via prompting or in-context learning approaches Maynez et al. (2023); Shin et al. (2020); Deng et al. (2022); Prasad et al. (2022). To personalize language generations without re-training the LM, prompts with relevant historical data are used to align the LM outputs with user intents Madaan et al. (2022) or opinions Hwang et al. (2023). The methods most closely related to our work include prefix-tuning Li & Liang (2021) and soft-prompt learning Lester et al. (2021), which prepend task-specific continuous embeddings to the transformer layers or the embedded inputs to adapt the pre-trained LMs to specific downstream tasks. While the previous approaches learn task-specific embeddings from datasets with reference outputs, our approach instead focuses on the personalization setting by learning user-specific representations from preference datasets (instead of traditional text generation or labeling datasets).