

CS4NLP [263-5000-00L] Project Report

# How Much Noise is Too Much Noise?

Ankita Ghosh\*, Sayan Deb Sarkar\*, Stefan Stefanache\*

{anghosh, sdebsarkar, sstefanache}@ethz.ch

Department of Computer Science, ETH Zurich, Switzerland

## Abstract

Reinforcement Learning from Human Feedback (RLHF) has demonstrated efficacy in fine-tuning language models for complex tasks. However, the reliance on human annotations and the subsequent reward model renders the RLHF pipeline vulnerable to errors and noise. In this report, we examine the impact of noise on LM preference optimization methods. By introducing noise into the preference annotation data, we analyze the performance of various language model preference optimization pipelines, particularly Proximal Policy Optimization (PPO), REINFORCE Leave-One-Out (RLOO), Direct Preference Optimization (DPO), and N-Sampling. We assess the generated outputs based on the frontier of achieved reward and their KL-divergence from the reference language model, providing insights into the robustness of these methods against noisy data. Our analysis shows that DPO outperforms the other approaches in most cases and is the most resilient algorithm. Our codebase is available at <https://github.com/sayands/noisy-lms>.

## 1 Introduction

Supervised learning, while foundational for language models (LMs), struggles to capture the nuances of human preferences. Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022) addresses this limitation, allowing LMs to learn iteratively and align with intricate goals.

However, the effectiveness of RLHF hinges on the quality and consistency of the human feedback it receives. Unfortunately, human feedback is inherently subjective and prone to inconsistencies, biases, and errors. For instance, Anthropic reported a low average agreement rate of 63% between their researchers and crowd workers when annotating

preference data (Bai et al., 2022). Similarly, OpenAI found an inter-annotator agreement rate of only  $72.6 \pm 1.5\%$  among its training labelers (Ouyang et al., 2022).

In this project, we investigate how popular LM preference optimization methods behave when the feedback loop becomes noisy. Noise is introduced by manipulating feedback data by adding preference noise using label flipping. Our primary goal is understanding how preference optimization methods behave under these conditions and compare their robustness as shown in Figure 1.

In the rest of this report, we first discuss related work in Section 2. Section 3 details our methodology for injecting noise into preference data and the LM preference optimization pipeline. We present our experimental setup and results in Section 4. Finally, we discuss the challenges encountered and conclude with future directions in Section 5.

## 2 Related Work

RLHF and RLAIF (Lee et al., 2024) are complex and unstable methods. They first fit a reward model reflecting the human preferences and fine-tune a large unsupervised LM using RL to maximise the estimated reward. The standard RLHF objective is optimized with a variant of PPO, which works by “clipping” how far the LM can drift from its previous version. PPO is an online algorithm—generations are sampled from current model, judged by a reward model, and then used to update the current version. Another approach is a modified version of REINFORCE (Sutton et al., 2000), which is an off-policy Monte Carlo variant of a policy gradient algorithm. Thus, it is simpler yet still effective for the problem at hand. REINFORCE Leave-One-Out (RLOO) (Kool et al., 2019; Ahmadian et al., 2024) is a multi-sample extension of REINFORCE, where instead of relying

---

\*Equal contribution

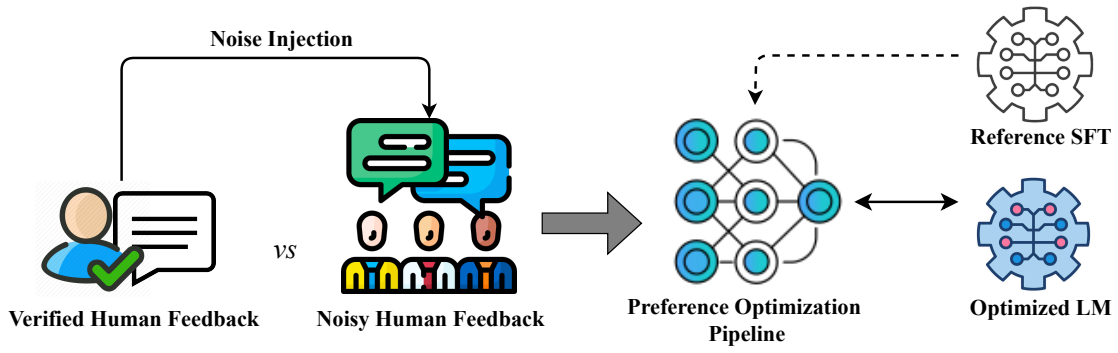


Figure 1: **Motivation.** To analyze the impact of noise in human feedback data on language model preference optimization pipelines when preferences are crowdsourced to be economical instead of employing a verified annotator. In our work, we emulate the scenario by injecting noise into the data at various levels. We compare the performance of language models (LM) optimized by different preference optimization algorithms against the reference supervised fine-tuned (SFT) language model.

on a value function to assess completion quality, it generates  $K$  completions per prompt. Each completion’s “advantage” (its relative quality) is determined by comparing its score to the average score of the other  $K - 1$  completions.

On the other hand, DPO avoids training the reward model entirely, and instead directly trains the LM using human preferences. A loss function is used, where the preferred output is treated as the correct label, and the model’s parameters are updated via gradient descent to increase the likelihood of generating preferred responses. This eliminates the need for a separate reward model, simplifying the training process and potentially improving efficiency. Similarly, Kahneman-Tversky Optimization (KTO) (Ethayarajh et al., 2024) is a principled approach to maximize the utility of generations instead of maximizing the log-likelihood of preferences and only requires a binary signal for desirability of output given an input. These methods implicitly model human biases over uncertain outcomes and are categorised under *human-aware loss functions* (HALOs) (Ethayarajh et al., 2023).

Furthermore, Best-of-N (BoN) sampling (Nakano et al., 2022; Stiennon et al., 2020), also known as rejection sampling, is a popular, simple, and effective method for optimizing preferences in language models. It selects the response with the highest reward score from a set of generated responses, using a proxy reward model.

Language models generate text using decoding strategies like beam search, temperature sampling, and greedy search. Greedy search is the simplest approach, selecting the most probable word at each step to build the text sequence. While this can be computationally efficient, it may produce less creative or less contextually relevant text. Beam search, on the other hand, considers multiple

promising sequences simultaneously, exploring different paths to generate more fluent and coherent text. However, it can be computationally expensive. Temperature sampling introduces randomness by adjusting the probability distribution of word choices. Higher temperatures lead to more diverse and creative outputs, while lower temperatures prioritize more predictable text. Choosing the right strategy depends on the desired balance between accuracy, fluency, creativity, and computational resources.

Previous research has investigated the impact of noisy preference data on language model alignment through reinforcement learning. For example, (Wang et al., 2024) discovered that instances with low preference strength (high uncertainty in the reward model) between chosen and rejected responses negatively affect model performance on the validation set during reinforcement learning. Notably, flipping the labels of these pairs can actually enhance the model’s ability to learn preference information. Besides label flipping, (Wang et al., 2024) also employed label smoothing and adaptive margin techniques to mitigate the effects of noise.

We will leverage the existing RHLF pipeline for our experiments, and perturb the data used through the framework.

### 3 Method

In this section, we first describe how we introduce noise in the preference annotation data. We then describe how the LM preference optimization pipeline is trained using this noisy data.

#### 3.1 Noise Injection

Noise can be introduced in an LM preference optimization pipeline in several ways (Khraishi and Okhrati, 2023) like augmenting states, rewards, and

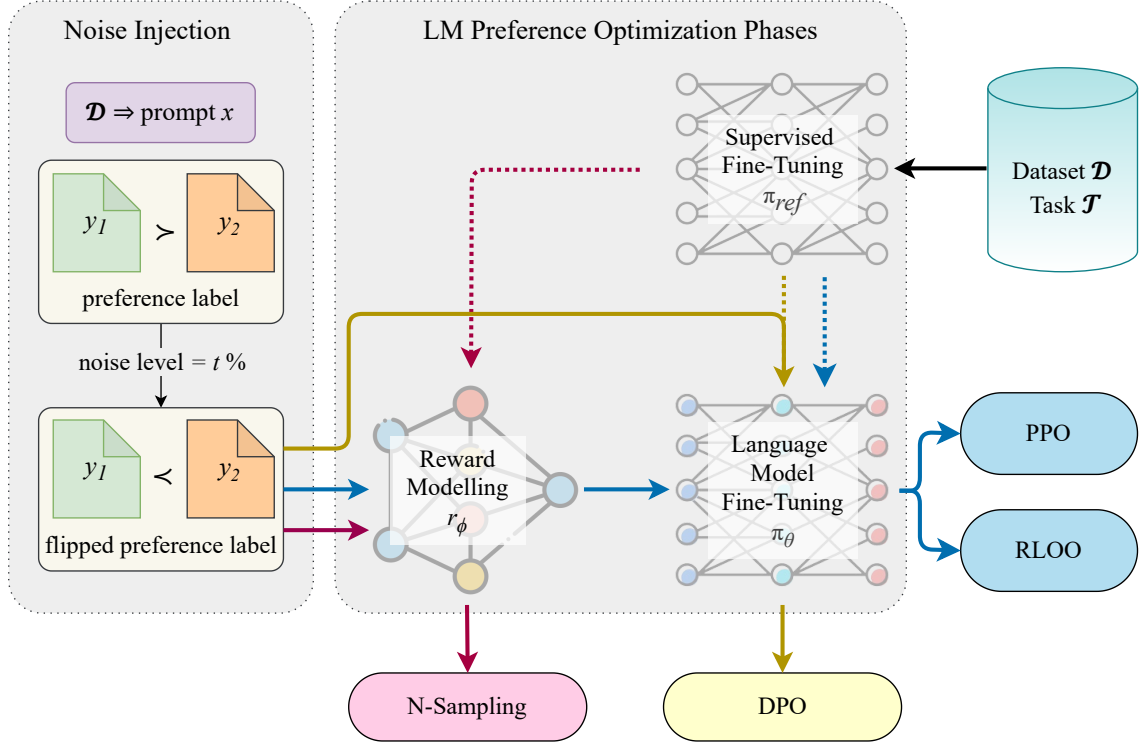


Figure 2: **Overview of the Noise Injection and Language Model Preference Optimization Phases.** The process begins with injecting noise into the preference labels of a dataset  $\mathcal{D}$  for a given prompt  $x$ , flipping preference labels based on a defined noise level  $t\%$ . The LM Preference Optimization Phases include: (1) Supervised Fine-Tuning (SFT) to train a reference language model  $\pi_{ref}$  on the task  $\mathcal{T}$ , (2) Reward Modelling to develop a reward model  $r_\phi$  using noisy preference labels, and (3) Language Model Fine-Tuning to optimize the language model  $\pi_\theta$ . Reinforcement learning algorithms such as Proximal Policy Optimization (PPO) and REINFORCE Leave-One-Out (RLOO) make use of all three stages. N-Sampling skips fine-tuned language model learning while Direct Preference Optimization (DPO) skips the step of training an external reward model.

transition dynamics. However, our central objective is to mimic the inconsistencies found in human annotations. In order to simulate human error within our experimental framework, we adopt the methodology of label flipping (Wang et al., 2024) in preference annotations of prompts generated by an LM. Specifically, we introduce controlled noise by swapping a subset of the preference labels, denoted as  $y_w^{(i)}, y_l^{(i)}$ , where  $y_w^{(i)}$  represents the preferred answer over  $y_l^{(i)}$ . We analyze the effects of noise injection into  $\{x^{(i)}, y_1^{(i)}, y_2^{(i)}\}$  at various noise levels  $t$ , and retrain the necessary components, such as the reward model  $r_\phi^t$  and the language model  $\pi_\theta^t$ . This approach enables us to assess the robustness and performance of the pipeline under different configurations and noise conditions.

### 3.2 LM Preference Optimization Pipeline

The general preference optimization (PO) pipeline (Lee et al., 2024) encompasses three distinct phases.

Initially, in the first phase, responses  $\{y_1^{(i)}, y_2^{(i)}\}$  are generated for each prompt  $x^{(i)}$  using a supervised fine-tuned (SFT) language model  $\pi_{ref}$ , tailored for the specific task  $\mathcal{T}$ , leading to the creation of dataset  $\mathcal{D}$  with datapoints  $\{x^{(i)}, y_1^{(i)}, y_2^{(i)}\}$ . The second phase involves the development of a reward model  $r_\phi$ , which discerns the more favorable response from  $\{y_1^{(i)}, y_2^{(i)}\}$ , denoted as  $y_w^{(i)} \succ y_l^{(i)}$ , utilizing human annotation labels for guidance. The third phase focuses on fine-tuning the language model to derive  $\pi_\theta$  by leveraging feedback from the trained reward function  $r_\phi$ .

For our problem statement, the objective is to evaluate the resilience of various methods that incorporate all or subsets of the specified phases. Consequently, we examine different variations of the PO pipeline. Initially, we consider the utilization of all three stages of the PO pipeline. Subsequently, we explore the scenario where the second phase is bypassed, allowing us to directly train a

Greedy Search Generation on Training Data Prompts					
PO Pipeline	no noise	noise = 1%	noise = 5%	noise = 10%	noise = 20%
N-Sampling	0.4195	<b>0.5182</b>	<b>0.5445</b>	0.3369	0.3722
PPO	0.4217	0.4211	0.4170	0.4177	0.4165
RLOO	0.4194	0.4171	0.4177	0.4215	0.4140
DPO	<b>0.4298</b>	0.4182	0.4174	<b>0.4775</b>	<b>0.4219</b>

---

Greedy Search Generation on Validation Data Prompts					
PO Pipeline	no noise	noise = 1%	noise = 5%	noise = 10%	noise = 20%
N-Sampling	0.3704	<b>0.4735</b>	<b>0.5604</b>	0.3337	0.3554
PPO	0.4351	0.4318	0.4319	0.4310	0.4309
RLOO	0.4325	0.4314	0.4319	0.4323	0.4320
DPO	<b>0.4428</b>	0.4319	0.4325	<b>0.4773</b>	<b>0.4378</b>

Table 1: **Reward Scores of N-Sampling, PPO, RLOO, and DPO on training and validation prompts using Greedy Search.** A higher score indicates better generation. N-Sampling tends to perform best at a moderate noise level but is highly sensitive to higher noise levels. PPO and RLOO exhibit robustness to noise, maintaining stable performance across different noise levels. Overall, DPO shows the strongest performance by outperforming or being at par with most other algorithms.

language model (LM) based on preference annotations. Finally, we investigate the performance of a method that omits the third phase entirely, thereby excluding any RL training. Figure 2 shows a diagrammatic representation of our workflow.

## 4 Experiments

In this section, we first describe our experimental setup and define the evaluation metrics. We follow it by showcasing and analyzing the results obtained.

### 4.1 Experimental Setup

We discuss the language task, dataset, models, and PO pipelines with which we experimented. We provide details about the training implementations and the metrics used to evaluate the results.

**Task and Dataset.** We perform our experiments on the task  $\mathcal{T}$  of summarization. To this effect, we use OpenAI’s Human Preferences (Stiennon et al., 2020) - a dataset  $\mathcal{D}$  created from a subset of Reddit TL;DR (Völske et al., 2017). Each example comprises a forum post,  $x^{(i)}$ , two candidate summaries  $\{y_1^{(i)}, y_2^{(i)}\}$ , and a choice  $c$  from a human annotator indicating which summary is preferred.

**PO pipelines and Models.** To investigate the impact of noise on human feedback data, we conduct our analysis utilizing the pipelines of PPO, RLOO, DPO, and N-Sampling. This diverse selection of

methodologies allows for a comprehensive comparison of techniques that employ distinct learning strategies, thereby providing a robust evaluation of their effectiveness in handling noisy data. We keep the SFT model  $\pi_{ref}$  consistent for all the pipelines and employ GPT2 that has been fine-tuned to task  $\mathcal{T}$  using dataset  $\mathcal{D}$ . The baseline reward model  $R_\phi$ , and the subsequent versions trained on noisy data,  $r_\phi^t$ , are also derived from GPT2 by appending the LM with a classification layer. The optimized LM  $\pi_\theta$  is then derived through the training implementations described next.

**Implementation Details.** We implemented our codebase using HuggingFace tools, primarily the `trl` and `transformers` libraries. Since we were unable to fit our training on the teaching cluster, we used external servers with 40GB A100s and 24GB RTX-3090 GPUs with most of our model training needed around 12 hour job runs.

We train the models on 92k preference annotations and use 500 samples for validation. We corrupt the preferences at the noise levels of  $t = 0\%, 1\%, 5\%, 10\%$ , and  $20\%$ . We use GPT2 for tokenizing the input data, and threshold the maximum input token length to 128. For the majority of the training hyperparameters, we use the default values allocated by the libraries.

SFT is trained on a batch size of 16 and a learn-

PO Pipeline	Sampling Generation on Validation Data Prompts				
	no noise	noise = 1%	noise = 5%	noise = 10%	noise = 20%
N-Sampling	0.5635	<b>0.5901</b>	<b>0.6238</b>	0.3994	0.4303
PPO	0.5643	0.4965	0.4893	0.4425	0.5034
RLOO	0.4826	0.3436	0.4967	0.4668	0.4143
DPO	<b>0.6326</b>	0.5899	0.5045	<b>0.6187</b>	<b>0.6054</b>

Table 2: **Reward Scores of N-Sampling, PPO, RLOO, and DPO on validation prompts using Sampling Generation with temperature 0.25 and Best-Of-N 4.** A higher score indicates better generation. DPO consistently scores above 0.50 at all noise levels, outperforming other pipelines in the majority of cases.

ing rate of  $1e-5$ . All variations of reward models, PPO, RLOO and DPO are trained with a batch size of 16, 16, 4, and 8 respectively. The learning rate in the corresponding orders are:  $1e-3$ ,  $1.41e-5$ ,  $3e-6$ , and  $1e-3$ . Additionally, RLOO is trained for a total of 2500 episodes. For N-Sampling, we generate 8 samples and choose the one with the maximum reward score.

**Evaluation Metrics.** To evaluate the effectiveness of each algorithm in optimizing the constrained reward maximization objective within a summarization task setting, we assess each algorithm based on its frontier of achieved reward. This evaluation is feasible due to our access to the baseline reward function  $R_\phi$  trained on  $\pi_{ref}$ , represented by a text or sentiment classifier. Thus, the reward score is calculated as,

$$S(\pi_\theta^t) = \frac{\sum_{x \in d} R_\phi(x, \pi_\theta^t(x))}{|d|} \quad (1)$$

where  $x$  is sampled from a subset of the dataset  $d \subset \mathcal{D}$  and fed to the model  $\pi_\theta^t$  to generate a response.  $|\cdot|$  denotes the number of samples, 500 in our case.

When optimizing LM models using PO pipelines, we do not want the model  $\pi_\theta^t$  to stray too far away from  $\pi_{ref}$  and the original model (GPT2). The underlying rationale is that numerous potential responses exist for any given prompt, the majority of which the PO pipeline has not previously encountered. For many of these unobserved  $\{x, \pi_\theta^t(x)\}$  pairs, the algorithm may erroneously assign excessively high or low scores. In the absence of this constraint, there is a risk of introducing a bias towards responses with extremely high scores, despite these responses potentially lacking in quality. Training  $\pi_\theta^t$  on noisy data can potentially confuse the model and lead it to degeneration. Thus, we also assess the Kullback-Leibler (KL) divergence

of model  $\pi_\theta^t$  from  $\pi_{ref}$  which is a measure of the dissimilarity between two probability distributions. It is calculated as,

$$D_{KL}(\pi_\theta^t \parallel \pi_{ref}) = \sum_{x \in d} \pi_\theta^t(x) \log \left[ \frac{\pi_\theta^t(x)}{\pi_{ref}(x)} \right] \quad (2)$$

We generate responses from the models using decoding methods like greedy search and temperature sampling, and compute the metrics stated above. The results are analyzed in Section 4.2.

## 4.2 Results

We first use greedy search to generate reward scores for prompt samples from both training and validation sets. Through this, we want to observe how noise impacts not only unseen data, but also seen data as it might have been trained to output the wrong preference. The results obtained are presented in Table 1.

We observe that the reward model scores appear to be in a similar range for both training and validation prompts. This indicates that the injection of noise into preferences does not only harm the responses generated for seen data but also skews the model’s generations for unseen data. Further from the resulting reward scores, we notice that N-Sampling demonstrates a relatively high resilience and improves at low levels of noise. This occurs because a small amount of noise encourages the reward model to generalize better and N-sampling selects the optimal response from the generated set. However, higher levels of noise lead to deterioration in the reward model’s learning.

In PPO, scores remain relatively stable across all noise levels, with only minor fluctuations, due to its clipping mechanism and the robustness provided by its stochastic policy as it explores a wider range of actions. Similarly, RLOO also maintains sta-



PO Pipeline	KL Divergence				
	no noise	noise = 1%	noise = 5%	noise = 10%	noise = 20%
N-Sampling	0.2243	0.1976	0.1234	0.2129	0.0822
PPO	0.2713	0.3219	0.3139	0.3326	0.3315
RLOO	0.2887	0.4023	0.2813	0.3460	0.4273
DPO	0.2028	0.2157	0.2739	0.2307	0.2275

Table 3: **KL Divergence values of the fine-tuned language models compared to reference SFT.** All models stay close to the reference SFT.

ble reward scores across noise levels. By employing the leave-one-out average reward as a baseline, RLOO effectively normalizes the reward signal, making the learning process less sensitive to individual noisy rewards. DPO achieves the highest reward score, and even an improvement at higher noise levels. The training methods appear to be significantly more resilient, with majorly DPO dominating. Considering N-Sampling degenerates at greater noise levels, we can hypothesize that DPO is able to avoid the amplification of noise caused by reward models. Additionally, DPO uses deterministic update techniques, which lessens the variation brought on by noisy data.

Next, we use sampling techniques to generate responses. We set the temperature to 0.25 empirically and applied Best-of-N sampling with  $N=4$ , even for N-sampling as shown in Table 2. Across all noise levels, DPO consistently achieves higher reward scores compared to the other methods, indicating its robustness and superior performance in generating quality outputs under noisy conditions. N-Sampling and PPO show competitive performance at lower noise levels but degrade more significantly as noise increase. RLOO has the lowest average reward score when computed across all noise levels. Examples of generations for prompts using this decoding technique are shown for noise levels 5% and 20% in Table 4 and Table 5 respectively.

Table 3 shows the KL divergence values of  $\pi_{\theta}^t$  compared to  $\pi_{ref}$ . Despite the addition of noise, the models do not sway too far from the reference. DPO adapts the model distribution the most, explaining its robustness to noisy data.

## 5 Discussion

In this section, we delineate the principal challenges encountered during the execution of the project. We also provide a synthesis of our findings

and outline potential avenues for future research.

### 5.1 Computational Limitations

Our project faced significant challenges due to computational limitations and software instability. Training over 21 models proved computationally demanding, with post-run metadata alone consuming 600GB of storage, requiring us to use resources from external providers and servers. Additionally, HuggingFace’s TRL library, while central to our workflow, introduced unexpected obstacles through frequent bugs and inadequate documentation, hindering our progress. For evaluation, as the win rate metric requires large-scale user study or evaluation by extremely powerful language model (like GPT4), it was infeasible to conduct and we used alternate methods to analyze our results.

### 5.2 Conclusion

In conclusion, our investigation into noise impact on language model preference optimization offers insights into the robustness of RL algorithms. By injecting noise into preference data, we simulated inconsistent human feedback. DPO consistently outperformed other methods, showing high resilience to noise, while PPO and RLOO maintained stable performance, making them reliable choices. In contrast, N-Sampling was highly sensitive to noise. Noise affected both training and validation data, skewing model generations. Despite computational challenges limiting our exploration, our findings provide a foundational understanding of noise impacts on RL from human feedback. Future work should explore other noise forms and test more tasks and datasets. Our study contributes to making AI systems more robust to human biases, ensuring reliable performance with imperfect feedback, crucial for AI’s integration into complex human environments.

<b>Prompt</b>	<p>So I broke up with my SO about 4 months ago - it was the cleanest possible breakup, due entirely to conflicting schedules and lack of time, so there was really no impetus for me to move on. We tried to be friends afterward and did our best, but communication quickly tapered off. I assumed from the lack of emotions I was feeling toward her and the lack of communication that I was getting over her.this week she visited my city.invited a bunch of our old friends (and myself) out for dinner, so I figured, sure, why not, let's cement her status as a friend. This will be casual enough.it was. We had a great time, laughed like crazy, and found ourselves finishing each other's sentences like we did when we first fell for each other. It was a really fun night, closed with a loving goodbye hug.say loving because the second I go, reality smacked me in the face. I was still in love with her, just as much as when I broke up with her. I spent the night brooding over it, and decided that the best course of action was to be blunt about it. I texted her the next day telling her in fairly simple terms what the night had done for me - it made me realize I was still head over heels for her.she never responded.'m entirely okay with that. Hell, I'm happy about that. It put things into perspective. Four months of acting like I was okay with being friends with a girl I never even had the chance to see that way wore on me. I realized how much self-respect I was abandoning just by keeping up this charade and allowing her to be how I validated myself., honestly? Fuck that. I'm awesome, and I don't need her to tell me that to know it. I feel like I've lifted a weight from my shoulders, and for the first time in four months I'm genuinely happy to be single. For the first time in four months I feel like a man instead of a teenager. I'm free from love because I told her I loved her.", "title": "I feel so great. I just got over a past relationship in the least likely possible way.</p>
<b>DPO</b>	<p>I broke up with my SO four months ago, and I'm finally free from the emotional weight of it. I feel like a man.</p>
<b>PPO</b>	<p>So I broke up with my SO about 4 months ago - it was the cleanest possible breakup, due entirely to conflicting schedules and lack of time, so there was really no impetus for me to move on. We tried to be friends afterward and did our best, but EOS EOS</p>
<b>N-Sampling</b>	<p>So I broke up with my SO about 4 months ago - it was the cleanest possible breakup, due entirely to conflicting schedules and lack of time, so there was really no impetus for me to move on. We tried to be friends afterward and did our best, but it was a long distance relationship. I'm not sure if I'm overreacting to this</p>
<b>RLOO</b>	<p>So I broke up with my SO about 4 months ago - it was the cleanest possible breakup, due entirely to conflicting schedules and lack of time, so there was really no impetus for me to move on. We tried to be friends afterward and did our best, but I don't know what to do. I don't know what to do. I don't know what to</p>

Table 4: **Sample summaries to a post from the validation set.** All models are trained on the dataset with a **noise level of 5%** and All samples are generated with a **temperature 0.25**.

<b>Prompt</b>	Hello Reddit. My job history is good and bad. From 2003 to 2007 I was a retail store manager. Between mid 2007 and late 2009, I changed jobs a lot. I had an opiate addiction and it was hard to stay in a place too long, due to being so up and down. Things started to get better as I mentioned above (late 09) and I worked in finance from that date until a month or so ago. was wondering if I should mention the other jobs at all, or focus on one or two others. If I do that though, how can I explain the gaps? know it seems as if I am focusing on the length of time I have stayed at various jobs. However, I know how important that is, especially in full time roles. would really appreciate some help with this matter. If you need further information, please ask.", "title": "How to show my job history in the best light"
<b>DPO</b>	Hello Reddit. My job history is good and bad. From 2003 to 2007 I was a retail store manager. Between mid 2007 and late 2009, I changed jobs a lot. I had an opiate addiction and it was hard to stay in a place too long, due to being so up and down. Things started to get better as I mentioned above (late 09)ing
<b>PPO</b>	Hello Reddit. My job history is good and bad. From 2003 to 2007 I was a retail store manager. Between mid 2007 and late 2009, I changed jobs a lot. I had an opiate addiction and it was hard to stay in a place too long, due to being so up and down. Things started to go in EOS, EOS, EOS.
<b>N-sampling</b>	Hello Reddit. My job history is good and bad. From 2003 to 2007 I was a retail store manager. Between mid 2007 and late 2009, I changed jobs a lot. I had an opiate addiction and it was hard to stay in a place too long, due to being so up and down. Things started to get better as I mentioned above (late 09) and I worked in finance from that date until a month or so ago. I was wondering if I should mention the other jobs at all, or focus on one or two other jobs. I'm not sure if I should mention the other jobs, but I'm sure I should.
<b>RLOO</b>	Hello Reddit. My job history is good and bad. From 2003 to 2007 I was a retail store manager. Between mid 2007 and late 2009, I changed jobs a lot. I had an opiate addiction and it was hard to stay in a place too long, due to being so up and down. Things started to get better as I mentioned above (late 09) and I worked in finance from that date until a month or so ago. I was wondering if I should mention the other jobs at all, or focus on one or the other.

Table 5: **Sample summaries to a post from the validation set.** All models are trained on the dataset with a **noise level of 20%** and All samples are generated with a **temperature 0.25**.



## References

- Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. [Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms](#).
- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, John Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Christopher Olah, Benjamin Mann, and Jared Kaplan. 2022. [Training a helpful and harmless assistant with reinforcement learning from human feedback](#). *ArXiv*, abs/2204.05862.
- Kawin Ethayarajh, Winnie Xu, Dan Jurafsky, and Douwe Kiela. 2023. Human-aware loss functions (halos). Technical report, Contextual AI.
- Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. [Kto: Model alignment as prospect theoretic optimization](#). *ArXiv*, abs/2402.01306.
- Raad Khraishi and Ramin Okhrati. 2023. [Simple noisy environment augmentation for reinforcement learning](#).
- Wouter Kool, Herke van Hoof, and Max Welling. 2019. [Buy 4 REINFORCE samples, get a baseline for free!](#)
- Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. 2024. [RLAIF: Scaling reinforcement learning from human feedback with AI feedback](#).
- Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. [Webgpt: Browser-assisted question-answering with human feedback](#).
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. [Training language models to follow instructions with human feedback](#).
- Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. [Learning to summarize with human feedback](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc.
- R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, volume 12, pages 1057–1063. MIT Press.
- Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. 2017. [TL;DR: Mining Reddit to learn automatic summarization](#). In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63, Copenhagen, Denmark. Association for Computational Linguistics.
- Bing Wang, Rui Zheng, Luyao Chen, Yan Liu, Shihan Dou, Caishuang Huang, Wei Shen, Senjie Jin, Enyu Zhou, Chenyu Shi, Songyang Gao, Nuo Xu, Yuhao Zhou, Xiaoran Fan, Zhiheng Xi, Jun Zhao, Xiao Wang, Tao Ji, Hang Yan, Lixing Shen, Zhan Chen, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yuanyuan Jiang. 2024. [Secrets of rlhf in large language models part ii: Reward modeling](#). *ArXiv*, abs/2401.06080.