# R.I.P.: Better Models by Survival of the Fittest Prompts

**Ping Yu** [1] **Weizhe Yuan** [1 2] **Olga Golovneva** [1] **Tianhao Wu** [3]
**Sainbayar Sukhbaatar** [1] **Jason Weston** [1 2] **Jing Xu** [1]

## Abstract

Training data quality is one of the most important drivers of final model quality. In this work, we introduce a method for evaluating data integrity based on the assumption that low-quality input prompts result in high variance and low quality responses. This is achieved by measuring the *rejected response quality* and the *reward gap* between the chosen and rejected preference pair. Our method, Rejecting Instruction Preferences (*RIP*) can be used to filter prompts from existing training sets, or to make high quality synthetic datasets, yielding large performance gains across various benchmarks compared to unfiltered data. Using Llama 3.1-8B-Instruct, *RIP* improves AlpacaEval2 LC Win Rate by 9.4%, Arena-Hard by 8.7%, and WildBench by 9.9%. Using Llama 3.3-70B-Instruct, *RIP* improves Arena-Hard from 67.5 to 82.9, which is from 18th place to 6th overall in the leaderboard.

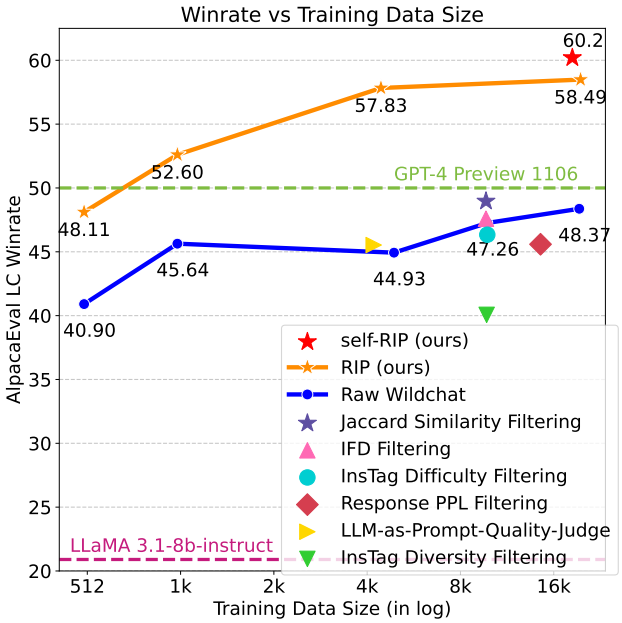



Figure 1: **Our method *Rejecting Instruction Preferences* (*RIP*) for curating data, and Self-*RIP* for creating synthetic data.** The x-axis represents the effective training set size (after filtering). At every data size training on unfiltered WildChat prompts is significantly outperformed by *RIP*. *RIP* also outperforms various other curation baselines. Synthetic data built by Self-*RIP* improves results further.

## 1. Introduction

In large language model (LLM) development, a primary driver for advancing frontier models is curating high-quality training examples. This curation is crucial during both the pretraining (Rae et al., 2021; Touvron et al., 2023a) and post-training (finetuning) phases (Touvron et al., 2023b). Despite the widespread adoption of the "scaling hypothesis" (Kaplan et al., 2020), merely increasing the size of training datasets does not guarantee improved performance if the data are of low quality (Chen et al., 2024; Li et al., 2024c; Zhou et al., 2024). Without sufficient data quality, model training tends not to be fully robust to the associated noise, and final response quality from the model suffers.

Currently, there are a number of investigated techniques to curate data – most of which are based on heuristics or model judgments given the training inputs. In this work, we hypothesize that better judgments of data quality can be made *by taking into account the model responses on those data*. Specifically, if the prompt is of low quality, then responses exhibit high variability and low quality as well. This insight leads us to develop a method for either selecting prompts, or for creating high quality synthetic prompts, both of which yield significant performance gains during post-training.

Our method, Rejecting Instruction Preferences (*RIP*), considers the case of instruction finetuning via preference optimization. It starts with a set of preference pairs consisting of input prompts and chosen and rejected responses. *RIP* considers specific characteristics of the preference pairs, in particular *rejected response quality* and the *reward gap* be-

[1]Meta [2]New York University [3]UC Berkeley. Correspondence to: Jing Xu <jingxu23@meta.com>.

*Proceedings of the $42^{nd}$ International Conference on Machine Learning*, Vancouver, Canada. PMLR 267, 2025. 

Table 1: **Rejecting Instruction Preferences (*RIP*) and Self-*RIP* compared to SOTA models on AlpacaEval2, Arena-Hard and WildBench**. By training Llama 3.1-8B-Instruct and Llama 3.3-70B-Instruct on Wildchat instructions curated by *RIP*, or synthetic data created by Self-*RIP*, our method surpasses many existing SOTA models.

| | AlpacaEval2 | | Arena-Hard | WildBench |
|---|---|---|---|---|
| *Standard models* | LC Win | Win | Score | Score |
| GPT-4 Omni (05/13) | 57.5 | 51.3 | 74.9 | **59.3** |
| GPT-4 Turbo (04/09) | 55.0 | 46.1 | 82.6 | 55.2 |
| Llama 3.1-8B-Instruct | 20.9 | 21.8 | 21.3 | 33.1 |
| Llama 3.3-70B-Instruct | 38.9 | 41.5 | 67.5 | 52.8 |
| Llama 3.1-8B-Instruct + *RIP* (ours) | 57.8 | 57.2 | 43.1 | 45.6 |
| Llama 3.1-8B-Instruct + Self-*RIP* (ours) | 60.2 | 61.1 | 42.1 | 42.5 |
| Llama 3.3-70B-Instruct + *RIP* (ours) | **67.7** | **73.2** | **82.9** | 58.8 |

tween the chosen and rejected preference pair. If the rejected quality is low or the reward gap is high this is an indicator that the prompt is of low quality. We thus filter the prompts based on these metrics. The remaining prompts can subsequently be used to fine-tune the model using RLHF methods like Direct Preference Optimization (DPO) (Rafailov et al., 2023), or for creating new synthetic prompts via few-shot prompting. Table 1 illustrates that when trained on Wildchat prompts (Zhao et al., 2024b) and filtered by *RIP*, both Llama 3.1-8B-Instruct and Llama 3.3-70B-Instruct (Dubey et al., 2024) achieve large performance gains, surpassing many state-of-the-art models.

Additionally, we conducted comprehensive experiments comparing the scaling behavior of our data under *RIP* filtering with that of unfiltered WildChat raw data, and six alternative filtering methods in Figure 1. Our results demonstrate that *RIP* significantly enhances model performance, while other filtering methods yield only marginal improvements. In addition to improvements observed with filtering human-written data such as Wildchat prompts or HelpSteer2 using different reward signals such as human, classifier or LLM-as-a-Judge, we also show *RIP* improves model performance as a method to create synthetic data.

Analysis of our method using t-SNE shows that *RIP* can eliminate certain undesirable clusters. Additionally, analysis with GPT-4 reveals that *RIP* effectively removes noisy or low quality prompts, ambiguous prompts, unsafe prompts, and examples where preference choices are incorrect. We release our filtered datasets on HuggingFace[1].

[1]For the Llama-3.1-8B-Instruct filtered dataset, visit: https://huggingface.co/datasets/facebook/Wildchat-RIP-Filtered-by-8b-Llama.
For the Llama-3.3-70B-Instruct filtered dataset, visit: https://huggingface.co/datasets/facebook/Wildchat-RIP-Filtered-by-70b-Llama.

## 2. Related Work

**Data Selection in Pretraining Data** Given the high variance in quality of pretraining data, data filtering is a critical component for determining pretrained model quality (Hoffmann et al., 2022). In addition to heuristic preprocessing such as deduplication of similar documents, removal of datasets with heavy test-set overlap, and text extractions from raw Internet content, GPT-3 (Brown et al., 2020) applied text filtering to the CommonCrawl dataset based on similarity to high-quality reference data, significantly reducing final pretraining text data from 45TB down to a 570GB high-quality subset. As language models become more powerful, data curation can also be facilitated by using LLMs as a quality judge. Llama2 and Llama3 employ model-based quality classifiers to filter out non-English and low-quality content from pretraining data (Touvron et al., 2023b; Dubey et al., 2024). Rae et al. (2021); Soldaini et al. (2024) also demonstrate that applying simple filtering on massive texts brings substantial improvements on downstream performance across the board.

**Data Selection in Supervised Fine-Tuning** Similarly, post-training also relies on high-quality data to enhance models' instruction-following capabilities. Previously, instruction-tuning was regarded as largely dependent on the size of available instruction-tuning examples (Mishra et al., 2021; Wei et al., 2021; Wang et al., 2022). More recent work has revealed that training on a smaller yet higher-quality curated set of prompts tends to be more effective in improving models' instruction-following capabilities (Zhou et al., 2024; Chen et al., 2024). To facilitate data selection, some employ traditional optimization-based data-pruning methods by measuring their impact on model's generalization capabilities (Toneva et al., 2018; Yang et al., 2022; Xia et al., 2024). Another stream of work studies employing powerful language models to measure the complexity, diversity and quality of instructions (Lu et al., 2023; Chen et al., 2024; Touvron et al., 2023b; Dubey et al., 2024; Li et al., 2024c). Alternative filtering approaches proposed automatic metrics such as IFD score (Li et al., 2023a), or INSTRUCTMINING

which fits a linearly weighted score over a bag of natural language indicators (Cao et al., 2023) to select examples.

**Data Selection in RLHF and Preference Optimization** The success of preference-optimization methods (Stiennon et al., 2020; Rafailov et al., 2024) has attracted more attention to collecting large scale and high quality preference data. While extensive work shows scaling up preference data through bootstrapping (Xu et al., 2023b; Yuan et al., 2024b), synthesis approaches (Lambert et al., 2024; Wang et al., 2024b), or crowdsourcing (Touvron et al., 2023b; Dubey et al., 2024), can boost model performance, the characterization and selection of high-quality pairwise examples is surprisingly underexplored. Most work involving preference optimization employs existing methods derived from pretraining and instruction-tuning (Touvron et al., 2023b; Dubey et al., 2024), such as deduplication, quality classifiers or filtering heuristics. However, such methods overlook the importance of the preference pairs (the chosen and rejected responses). Recent work Wu et al. (2024a); Khaki et al. (2024) shows that preference optimization can be highly sensitive to the choice of response pairs of different reward gaps, focusing more on pair construction than data selection.

## 3. Rejecting Instruction Preferences (*RIP*)

We start by defining the prompt selection problem in the pairwise preference optimization setting. In this context, we present our proposed prompt-response-pair-based filtering method, which develops key descriptive metrics and their use in filtering training prompts. Lastly, we describe how our method can be applied to self-instruction setups where synthetic prompts are generated from the model itself.

### 3.1. Data Curation Problem

The goal of data curation is to remove low-quality prompts that can negatively affect the general instruction following capability of the model. Given a set of prompts $X = \{x\}$, we aim to find a subset $S \subseteq X$ to be used for fine-tuning a seed LLM $\mathcal{M}$. We consider the preference optimization setting, with winning (chosen) and losing (rejected) response pairs $\{y_w, y_l\}$ with rewards $r(y_w|x) > r(y_l|x)$ for each prompt $x$. The response pairs and their rewards can come from human preference data, or can be generated from the model itself $\mathcal{M}$ and then scored using an external reward model. For the latter we use the "best-vs-worst" preference pairing method (Pace et al., 2024), where $N$ responses are sampled, and the ones with highest and lowest rewards are the chosen and rejected, respectively:

$$\{y_i\}_{i=1}^{N} \sim \mathcal{M}(x) \quad \text{then} \quad \begin{cases} y_w & = \text{argmax}_{y_i} r(y_i|x) \\ y_l & = \text{argmin}_{y_i} r(y_i|x) \end{cases}.$$

We also consider alternate pairing methods in Section A.4. We then use the preference data $\{x, y_w, y_l\}_{x \in S}$ for training the model $\mathcal{M}$. Note that our focus is on filtering prompts entirely, not responses to those prompts.

### 3.2. Hypothesis on Data Selection

Although preferences are extensively used to train state-of-the-art LLMs, there is limited research on identifying unhelpful training examples in this setting. We posit that analyzing the paired model responses to given input prompts can provide valuable insights into the quality of the prompts. Specifically, we test the following two hypotheses.

**Hypothesis 1: Low-quality prompts are likely to produce low-quality responses.** Low-quality prompts - for example those that are unclear, ambiguous, or containing conflicting information - are likely to lead to noisy or inaccurate model responses. While those inaccurate responses can still be used as training targets in pairwise preference optimization, studies indicate that training on pairs with low-quality rejected responses might be sub-optimal. Yasunaga et al. (2024) for example shows that pairing the best with random responses works well comparing to pairing the best with the worst one with lowest reward. This suggests a potential correlation of the quality of the rejected example with the alignment outcome. Additionally, several studies (Wu et al., 2024b; Zhao et al., 2024a; Yuan et al., 2024a) have found a strong correlation between the length of responses, including rejected ones, and final performance. Therefore, we consider the reward $r(y_l|x)$ and length $\text{len}(y_l)$ of rejected responses as indicators of quality of the training prompts $x$, i.e. large values of either of these metrics relative to other examples indicate higher quality.

**Hypothesis 2. Low-quality prompts are likely to produce responses with larger variance** Low quality prompts introduce uncertainty and ambiguity, leading to a broader range of interpretations. As the model or human generating the response might guess or fill in gaps in the prompt, this results in higher variance in responses. While some responses might align well with the intent, others may deviate significantly. A preliminary study in Wu et al. (2024a) finds low-gap pairs, where chosen and rejected responses are similar, are high-quality informative pairs, leading to better performing DPO models. We therefore consider the reward gap $r(y_w|x) - r(y_l|x)$ as another indicator of quality of a training prompt, i.e. small reward gaps suggest that the prompt has higher quality.

### 3.3. *RIP* filtering

#### 3.3.1. *RIP* FOR EXISTING TRAINING PROMPTS

Given the above hypotheses, we thus consider the following three metrics $m_k(x, y_w, y_l)$ that are based on the responses:

- Rejected response reward: $m_1 = r(y_l|x)$
- Rejected response length: $m_2 = \text{len}(y_l)$
- Reward gap: $m_3 = r(y_w|x) - r(y_l|x)$

For each metric, we define threshold values that can be used for filtering. For the first two metrics, higher values are desired so we choose a lower-bound threshold

$$S = \{x \mid \tau_k \leq m_k(x, y_w, y_l)\}.$$

The last reward gap metric requires an upper threshold as we want small gaps. Therefore we reduce the prompt selection problem to a threshold choice problem. To resolve this, we start with coordinate-wise experiments, analyzing model performance under various thresholds $\tau_k$ for individual metrics $m_k$ (details in Section A.2). Ultimately, we perform hyperparameter selection using all 3 parameters.

#### 3.3.2. Self-*RIP* for synthetic prompts

Prompt curation by *RIP* can also naturally be used to generate synthetic data. First, *RIP* is used to create a seed pool of high-quality prompts. Few-shot examples from this seed pool guide the model to generate training prompts, which can be further filtered by *RIP*. We thus propose **Self-*RIP***, a new approach to creating high-quality synthetic prompts:

**Step 1. Few-shot prompting with *RIP* curated instructions** We start with the set of prompts $S$ curated by our proposed method *RIP* as described in Section 3.3.1. To generate new prompts $S'$ we sample from our seed model $\mathcal{M}$ following Self-Instruct (Wang et al., 2023; Honovich et al., 2023). For each new example we randomly select 8 prompts from $S$ and feed them as few-shot examples to the model $\mathcal{M}$ to generate a prompt with similar characteristics. We apply the exact processing steps in Wang et al. (2023) to new prompts $S'$, such as removing similar prompts (ROUGE-L similarity with any existing instructions $< 0.7$), and excluding those that contain certain keywords (e.g., image, picture, graph) that usually can not be processed by text-only LLMs.

**Step 2. Filtering with *RIP*** We further apply *RIP* on top of the synthetically generated prompts $S'$ from the previous step, filtering out the self-instructions using the same threshold values as used before. Then the remaining subset $S''$ is used for training the seed model $\mathcal{M}$.

Note we use *RIP* filtering twice here, once in each step. This is to ensure the quality of synthetic prompts. We also explore Self-*RIP* using a smaller subset of $S$ as seed instructions in Section A.4 as part of our ablation studies.

## 4. Experimental Setup

We perform preference optimization using DPO, beginning with the Llama 3.1-8B-Instruct model as our seed model $\mathcal{M}$. We evaluate both the selection and creation of prompts, focusing on two categories: *human-written* instructions and *synthetically* generated instructions. Finally, we extend our evaluation of *RIP* with the Llama 3.3-70B-Instruct model.

### 4.1. Human-Written Prompts

For human-written instructions, we specifically investigate two setups: human-written input prompts 1) paired with model-generated responses and annotated by a reward model; 2) with existing responses that have been annotated with human-assigned rewards. We use the WildChat and Helpsteer2 datasets, see statistics in Appendix Table 13.

#### 4.1.1. WildChat dataset

**Prompt Set** We start with a large pool of over 250k human-written prompts from the WildChat (Zhao et al., 2024b) dataset. We exclude any non-English prompts based on WildChat annotations, and remove around 70k Midjourney-related instructions[2], yielding 190k unique first-turn prompts. These prompts are collected from real user interactions without human annotations, making them highly diverse. While there are many high-quality prompts, there are also a significant number of low-quality ones, such as nonsensical text or those lacking a clear question.

**Response Generation** Following Yuan et al. (2024b); Meng et al. (2024); Wu et al. (2024b) we generate our chosen and rejected response pairs on the WildChat prompts using our seed model $\mathcal{M}$ to make our setup closer to the on-policy setting. We use best-vs-worst as described in Section 3.1, generating $N$ responses for each prompt $x$ using $\mathcal{M}$ with sampling parameters of $T = 0.8$, $top\_p = 0.95$.

**Reward Annotation** We then evaluate candidate responses using two different judges:

- Reward Classifier: We used the ArmoRM reward model (Wang et al., 2024a) to score each response.
- LLM-as-a-Judge (Zheng et al., 2023): We prompt LLama 3.1-405B-Instruct using the prompt template outlined in Yasunaga et al. (2024) to assign a score ranging from 0 to 10 for each response. For each response, we conduct 10 independent evaluations and use the average score as the final reward.

The training example $(x, y_w, y_l)$ is selected by appointing the highest-reward one as $y_w$ and the lowest-reward one as $y_l$. For our primary experiments, we use the default value of $N = 64$. However, results for $N = 8, 16, 32$ are provided as part of our ablation studies in Table 22, and we use $N = 32$ for the Llama 3.3-70B-Instruct experiments. We perform early stopping using a validation set of 470 examples: 253 valid set examples from Li et al. (2024c) and

[2]They start with "As a prompt generator for a generative AI called "Midjourney", you will create image prompts ...".

218 examples from the evol-test set of Xu et al. (2023a), with prompts that overlap with AlpacaEval2 removed.

#### 4.1.2. HELPSTEER2 DATASET

HelpSteer2 (Wang et al., 2024c) consists of around 10k human-written prompts each with a response pair sampled from 10 different LLMs. Each response has human-annotated rewards of helpfulness, correctness, coherence, complexity and verbosity on a Likert-5 scale. We use the aggregated reward with the recommended weighting [0.65, 0.8, 0.45, 0.55, 0.4].[3] The main distinction from WildChat is that the rewards come from human annotations instead of an external model. We perform early stopping on the HelpSteer2 validation split, selecting checkpoints with the highest average response rewards determined by ArmoRM.

### 4.2. Synthetic Prompts

In this setup, we generate prompts from the seed model $\mathcal{M}$ itself for training instead of using human-written prompts. By varying the set of seed pool prompts used as few-shot examples, we collect two sets of training prompts:

- Self-Instruct: randomly select 8-shot examples from the unfiltered WildChat.
- Self-*RIP*: randomly select 8-shot examples from high quality WildChat prompts filtered by *RIP*.

In each case, we create 20k training prompts sampled with decoding parameters $T = 0.8$, $top\_p = 0.95$. The rest of the setup including response generations and DPO training is exactly the same as the WildChat setup where we use ArmoRM to construct response pairs $(y_w, y_l)$, and do early stopping on the same validation set of 470 examples.

### 4.3. Baselines

We compare our method with the existing methods below. For instruction-tuning data selection methods which handle a single (non-pairwise) response per prompt, we apply them to the chosen responses within the response pairs. Additional details on the implementation of each baseline are provided in Appendix Section A.5.

#### 4.3.1. PROMPT-BASED FILTERING

**InsTag Complexity** Lu et al. (2023) leveraged ChatGPT to create semantic and intent-based tags, subsequently fine-tuning an LLM as a data tagger using these tags. They then used the tag counts as a measure of complexity. This is used to filter out prompts with fewer tags to enhance complexity.

**InsTag Diversity** The InsTag Diversity filtering method (Lu et al., 2023) characterizes a dataset as more diverse when it includes a greater variety of unique tags, as annotated by the specified tagger. Using this approach, we greedily filter out data samples whose associated tags are already present in the selected dataset.

**LLM-as-Prompt-Judge** Employing LLMs as prompt quality judges has proven its efficacy in curating high-quality data (Chen et al., 2024; Dubey et al., 2024; Liu et al., 2023). We employ Llama 3.1-405B-Instruct to measure the quality of prompts on both a binary (useful/not useful) and pointwise scale (0-5). By sampling five Llama 3.1-405B-Instruct predictions per prompt and taking the average of LLM-as-Prompt-Judge predictions, we filter out less useful prompts by varying the cutoff thresholds.

#### 4.3.2. PROMPT-AND-CHOSEN-RESPONSE-BASED FILTERING

**Perplexity** We compute perplexity (ppl) of the chosen response $y_w$ with the Llama 3.1-8B Instruct in a zero-shot manner as a filtering metric to curate training prompts. In particular, we retain examples with large $\text{ppl}(y_w|x)$ values, which may indicate the difficulty of the prompt.

**Instruction-Following Difficulty (IFD)** Li et al. (2023a) introduced the IFD to measure the model-specific difficulty of a data sample. A lower IFD score indicates that this particular instruction-response pair is considered relatively easy for the language model to understand and follow without further training. We filter out examples with low IFD metric of a given pair of prompt $x$ and chosen response $y_w$.

#### 4.3.3. CHOSEN-AND-REJECTED-RESPONSE BASED FILTERING

**Jaccard Similarity** In addition to the reward gap between chosen and rejected responses, we explore Jaccard similarity, defined as the number of overlapping words divided by the overall word counts, as an alternative similarity measurement. We thus filter out examples with low Jaccard similarity scores (i.e. fewer overlapping words) between chosen and rejected response pairs.

### 4.4. Training And Evaluation Setting

Following the Instruct setup in Meng et al. (2024), we utilize the DPO training approach with the off-the-shelf LLama 3.1-8B-Instruct and LLama 3.3-70B-Instruct models, leveraging the fairseq2 library (Balioglu, 2023). We use a batch size of 64 and sweep over learning rates of $5e{-}7, 1e{-}6$ for the LLama 3.1-8B-Instruct model, and a learning rate of $1e{-}6$ with a batch size of 256 for the LLama 3.3-70B-Instruct model. Both models are trained with a dropout rate of 0.0 and a $\beta$ value of 0.1 throughout the experiments. We conduct *RIP* with various cutoff thresholds, e.g. at the 25%, 50% and 75% percentile of each metric.

[3] https://huggingface.co/nvidia/Llama3-70B-SteerLM-RM

We primarily assess models' general instruction-following capabilities on three evaluation benchmarks: AlpacaEval2 (Li et al., 2023b), Arena-Hard (Li et al., 2024b) and WildBench (Lin et al., 2024). These benchmarks cover a wide range of natural yet challenging real-world user queries, and have been widely adopted by the research community.

## 5. Experiment Results

Due to the large amount of unfiltered WildChat prompts, we first assess whether standard DPO training saturates as the size of the training prompts grows. As shown in Appendix Figure 2, the Armo Score on the valid set dramatically improves as we increase the size of training prompts, and begins to plateau afterwards. This shows growing the size of the training prompts arbitrarily does not bring additional gains, and hence quality control of the preference dataset could be important. We thus focus on 20k unique WildChat prompts, denoted as WildChat-20k for Llama3.1-8B-Instruct experiments, and 40k for Llama 3.3-70B-Instruct.

We report Alpaca-Eval2 Length-Controlled (LC) win rate, Arena-Hard score and WildBench WB-Score along with the number of training examples (after filtering if any) using WildChat-20k in Table 2, on HelpSteer2 in Table 5, and on Self-Instruction data in Table 6. Existing filtering methods are provided in Table 2 as baseline comparisons. Further details, such as hyperparameters, are in Appendix Table 9 and Table 14. Our findings lead to several key observations.

**When filtering human-written instructions, *RIP* achieves the best performance on both human-scored and model-scored preference datasets.** On the WildChat dataset where pairs are annotated by the ArmoRM model, we conduct *RIP* with various cutoff thresholds, at the 25%, 50% and 75% percentile of each metric. Our best model is trained on examples with rejected length larger than the 50% percentile of all rejected lengths, and rejected rewards larger than the 50% percentile of all rejected rewards, and reward gap smaller than the 50% percentile. Table 2 shows that *RIP* significantly improves LC win rate from the LLama3.1-8B-Instruct DPO baseline without filtering from 48.4% to 57.8% by filtering out 77% training examples, surpassing GPT-4 Omni (05/13) on AlpacaEval2. Similarly, *RIP* scores the highest on Arena-Hard (43.1) compared to LLM-as-Prompt-Judge filtering (42.0), Jaccard Similarity (42.6), and the no filtering baseline (37.9). *RIP* also achieves the highest WB-score on WildBench (45.6) compared to other filtering and no filtering baselines (41.5). As shown in Appendix Table 9 using LLM-as-a-Judge annotated rewards, *RIP* also performs well. Finally, Table 5 demonstrates *RIP* is equally effective on HelpSteer2 where preference pairs are determined by human annotators, achieving the highest scores across all 3 evaluation benchmarks as compared to the baselines (no filtering and LLM-as-Prompt-Judge filtering).

***RIP* scales to different and larger models** We also tried *RIP* on a different base LLM – from the Llama 3.3 family rather than 3.1, and of a larger scale, 70B rather than 8B. As shown in Table 3, *RIP* also works on this larger model. Filtering dramatically boosts Llama 3.3-70B-Instruct DPO trained models, with AlpacaEval2 LC win rate improved from 54.3% to 67.7%, Arena Hard from 70.5 to 82.9 and WildBench from 55.3 to 58.8, surpassing SOTA models as shown in Table 1. The prompt filtering threshold we applied to the 70B model was the same as in Llama 3.1-8B-Instruct + *RIP* (see Appendix Table 14).

**Weak-to-strong generalizability of *RIP*** To explore potential weak-to-strong generalizability (Li et al., 2024a) of our method, we employ a smaller and weaker model, Llama 3.1-8B-Instruct, to filter data for a larger and more powerful LLM, Llama 3.3-70B-Instruct. As illustrated in Table 4, while the filtering capability of Llama 3.1-8B-Instruct is not as powerful as that of Llama 3.3-70B-Instruct, it still offers significant improvements over baseline with no filtering. This showcases the weak-to-strong generation capabilities of our *RIP*, demonstrating that leveraging a smaller model to assist a larger one in data filtering is a computationally efficient strategy.

**Existing filtering methods derived from supervised-finetuning do not work as well on preference datasets** As demonstrated in Table 2, compared to the baseline WildChat-20k DPO (no filtering) trained on WildChat 20k prompts without any filtering, existing prompt-based filtering methods such as InsTag-Difficulty, InsTag-Diversity or LLM-as-Prompt-Judge filtering methods all lead to lower win rates on Alpaca-Eval2. LLM-as-Prompt-Judge, while outperforming certain filtering methods such as InsTag, achieves marginal gains compared to no filtering even though they are facilitated by querying a poweful LLM, Llama 3.1-405B-Instruct. Out of all the alternative methods tried, Jaccard Similarity based filtering that takes into account response pairs for filtering achieves relatively the highest scores across the 3 benchmarks, indicating that filtering that only takes into account prompts or chosen responses does not generalize well to the pairwise preference case.

**The Self-*RIP* method to generate synthetic data outperforms Self-Instruct data.** As shown in Table 6, Self-*RIP* yields better alignment results across all 3 evaluation benchmarks as compared to those trained on Self-Instruct data. In particular, win rate improves from 49.1% to 60.2% on AlpacaEval2, and from 38.5% to 42.1% on Arena-Hard. This result implies that our method generates better quality instructions than generating via few-shot examples from unfiltered prompts as in Self-Instruct.

**Self-*RIP* synthetic data outperforms human-written instructions** In Table 6, models trained on synthetic prompts outperform those trained on 20k human-written WildChat

Table 2: ***RIP* compared to existing filtering methods on WildChat with Llama 3.1-8B-Instruct.** *RIP*, which selects only 4538 WildChat prompts for DPO training, outperforms existing filtering methods on AlpacaEval2, Arena-Hard & WildBench. DPO response pairs are constructed using ArmoRM to score responses.

| | # Train examples | AlpacaEval2 | | Arena-Hard | WildBench |
|---|---|---|---|---|---|
| | | LC Win | Win | Score | Score |
| ***Baseline*** | | | | | |
| Llama 3.1-8B-Instruct (seed model) | - | 20.9 | 21.8 | 21.3 | 33.1 |
| WildChat-20k DPO (no filtering) | 20000 | 48.4 | 45.9 | 37.9 | 41.5 |
| WildChat-20k DPO (best-vs-bottom-25%) | 20000 | 48.2 | 45.9 | 40.7 | 44.5 |
| ***Prompt-Based Filtering*** | | | | | |
| LLM-as-Prompt-Judge Binary | 4299 | 45.5 | 41.0 | 42.0 | 43.3 |
| LLM-as-Prompt-Judge Pointwise | 15963 | 47.4 | 47.4 | 40.7 | 45.2 |
| InsTag-Difficulty | 10000 | 46.3 | 39.0 | 39.0 | 42.4 |
| InsTag-Diversity | 9952 | 40.1 | 41.1 | 40.4 | 43.4 |
| ***Prompt-and-Chosen-Response Based Filtering*** | | | | | |
| IFD on Prompt + Chosen Response | 9902 | 47.6 | 37.6 | 32.2 | 42.2 |
| ppl(Chosen Response) | 14851 | 45.6 | 45.5 | 40.8 | 43.4 |
| ***Chosen-Reject-Response Based Filtering*** | | | | | |
| Jaccard Similarity(Chosen, Rejected) | 9904 | 49.0 | 46.6 | 42.6 | 43.7 |
| *RIP* | 4538 | **57.8** | **57.2** | **43.1** | **45.6** |

Table 3: ***RIP* on WildChat with Llama 3.3-70B-Instruct.** *RIP* outperforms no filtering on AlpacaEval2, Arena-Hard & WildBench. DPO response pairs are constructed using ArmoRM to score responses.

| | # Train examples | AlpacaEval2 | | Arena-Hard | WildBench |
|---|---|---|---|---|---|
| | | LC Win | Win | Score | Score |
| Llama 3.3-70B-Instruct (seed model) | - | 38.9 | 41.5 | 67.5 | 52.8 |
| WildChat-40k DPO (no filtering) | 40000 | 54.3 | 51.6 | 70.5 | 55.3 |
| *RIP* | 17725 | **67.7** | **73.2** | **82.9** | **58.8** |

Table 4: **Weak to strong generation ability with *RIP*.** *RIP* on Llama 3.3-70B-Instruct by employing a smaller model Llama 3.1-8B-Instruct for filtering outperforms no filtering baseline, while underperforms using its own generations.

| Seed Model | Filter Model | # Train examples | AlpacaEval2 | | Arena-Hard | WildBench |
|---|---|---|---|---|---|---|
| | | | LC Win | Win | Score | Score |
| Llama 3.3-70B-Instruct | - (no filtering) | 40000 | 54.3 | 51.6 | 70.5 | 55.3 |
| Llama 3.3-70B-Instruct | Llama 3.1-8B-Instruct | 18184 | 64.5 | 69.2 | 76.7 | 58.6 |
| Llama 3.3-70B-Instruct | Llama 3.3-70B-Instruct | 17725 | **67.7** | **73.2** | **82.9** | **58.8** |

prompts. Applying Self-*RIP* few-shot generation *without post-filtering* gives an equal amount of 20k prompts, but still increases the AlpacaEval2 LC win rate from 48.4% to 53.6%, Arena-Hard win rate from 37.9% to 43.7% and WB-Score on WildBench from 41.5 to 44.8. This further illustrates the importance of training on high-quality instructions. When applying the full Self-*RIP* method with post-filtering results are further improved, for example achieving the best AlpacaEval2 LC win rate of 60.2%.

***RIP* seed data selection and *RIP* post-filtering are both important for generating Self-*RIP* synthetic data** In Table 6, we perform ablations on Self-*RIP*. We try: (i) using *RIP* to select high quality few-shot examples but not for curating the resulting generations (post-filtering); (ii) applying standard (Self-Instruct) few-shot generation, but then applying *RIP* post-filtering; or (iii) applying *RIP* to both few-shot generation and post-filtering (our default method). We find that both components of our full method are important yielding the best results, with method (i) outperforming Self-Instruct, and method (ii) performing better than (i), but worse than our full method (iii).

Table 5: ***RIP* on HelpSteer2 with Llama 3.1-8B-Instruct.** Applying *RIP* to DPO models trained on HelpSteer2 outperforms the baseline of no filtering as well as using the Llama 3.1-405B-Instruct model as a pointwise prompt quality judge.

| HelpSteer2 | # Train examples | AlpacaEval2 LC Win | AlpacaEval2 Win | Arena-Hard Score | WildBench Score |
|---|---|---|---|---|---|
| Llama 3.1-8B-Instruct (seed model) | - | 20.9 | 21.8 | 21.3 | 33.1 |
| HelpSteer2 DPO (no filtering) | 10161 | 25.2 | 23.1 | 26.8 | 37.1 |
| LLM-as-Prompt-Judge filtering | 5376 | 27.8 | 25.7 | 29.5 | 37.2 |
| *RIP* | 5081 | **34.6** | **32.8** | **35.0** | **39.5** |

Table 6: **Self-*RIP* for generating high-quality synthetic instructions**. Self-*RIP* creates prompts using few-shot samples from high-quality prompts curated by *RIP*, whereas Self-Instruct uses few-shots from unfiltered WildChat prompts. Applying *RIP* filtering *after* generation is also important, and achieves the best results, significantly outperforming Self-Instruct data.

| Train Prompts | Post-Filtering | # Train examples | AlpacaEval2 LC Win | AlpacaEval2 Win | Arena-Hard Score | WildBench Score |
|---|---|---|---|---|---|---|
| WildChat-20k | None | 20000 | 48.4 | 45.9 | 37.9 | 41.5 |
| WildChat-20k | *RIP* | 4538 | 57.8 | 57.2 | 43.1 | **45.6** |
| Self-Instruct | None | 20000 | 49.1 | 46.9 | 38.5 | 40.0 |
| Self-*RIP* (without post-filtering) | None | 20000 | 53.6 | 56.1 | **43.7** | 44.8 |
| Self-Instruct with *RIP* post-filtering | *RIP* | 16261 | 58.3 | 53.2 | 40.9 | 44.1 |
| Self-*RIP* | *RIP* | 18812 | **60.2** | **61.1** | 42.1 | 42.5 |

## 6. Understanding why *RIP* works

### 6.1. Filtering prompts with low quality responses

To understand what instructions are filtered out, we first visualize instructions with low quality rejected responses (as measured by low reward and short lengths) by comparing the t-SNE plots of unfiltered and filtered instructions (shown in Appendix Figure 5). We investigated a few clusters present in that t-SNE plot of unfiltered prompts that are missing from the t-SNE plot of filtered ones on the right-hand-side. We find that instructions from those clusters being filtered out from the training set are either obscure, non-sensical, or they fail to elicit meaningful responses from the model, leading to lower-quality rejected responses. Such instructions can be caught by measuring the rewards and lengths of the rejected responses, with supporting evidence given in Appendix Table 32.

Next, we employ GPT-4 and LLama3.1-405B-Instruct to evaluate first 10,000 prompts from WildChat. Focusing solely on the instructions (excluding responses) provided in WildChat, the model is tasked with scoring each prompt on a scale from 1 to 5. A score of 1 represents the most helpful prompt, while a score of 5 indicates the lowest quality. The evaluation prompt is provided in Appendix Figure 3. Manual review revealed that prompts assigned scores of 4 and 5 were of very low quality, while those scored 3 were moderately acceptable, albeit with some quality issues still present. Notably, GPT-4 and LLama3.1-405B-Instruct occasionally assigned scores of 2 or 3 to a prompt of low quality. Table 7 illustrates the prevalence of low-quality examples (with score 4 or 5 by both GPT-4 and LLama3.1-405B-Instruct) after applying various filtering methods. We observe that filtering based on the reward and length of the rejected response is the most effective way to ensure prompt quality, compared to other methods tried. By combining those rejected response quality metrics with the reward gap, *RIP* reduced percentage of noisy prompts from 22.9% to 8.9%. This supports our hypothesis that very low-quality prompts, such as those in WildChat that consist of incomplete snippets from movies, stories, or code (see sample rejected instructions in Appendix Table 32 and Table 33), often result in poor rejected responses when sampled several times. By leveraging the quality of rejected responses as a filtering criterion, we can efficiently eliminate these extremely noisy prompts.

Furthermore, we employ GPT-4 and LLama3.1-405B-Instruct to respond to each WildChat prompt three times. If any response declines to answer due to safety concerns, we categorize those prompts as unsafe. It's important to note that with this method, the model sometimes assigns high quality scores to prompts that are borderline unsafe. By examining the reward and the length of rejected responses, we observe *RIP* is also an effective approach to filter out these unsafe prompts. This approach is grounded in the observation that rejected responses when dealing with unsafe instructions are typically short and have low reward scores.

### 6.2. Filtering prompts with larger response variance

Similarly, we visualize instructions that are filtered out by measuring the reward gap between chosen and rejected responses in Appendix Figure 6, and further expand some representative groups of filtered instructions in Appendix Table 33. In particular, instructions that cover specialized domains such as coding, software, and other technical questions often require precise details, well-defined objectives or targeted solutions. In those cases, a lack of specificity in the instructions might lead to more variable responses. As shown in Table 33, instructions with larger reward gap are not necessarily low-quality, however we hypothesize that the combination of lack of specificity in the instruction and larger difference in the response pair make them less helpful in improving the model during preference optimization.

Table 7: **Effectiveness of Filters on Prompt Quality and Safety**: we compare the number of noisy and potentially unsafe (as judged by GPT4) WildChat instructions (out of 20k) filtered by various filtering methods.

| Filtering Methods | % of low-quality prompts ↓ | % of unsafe prompts ↓ |
|---|---|---|
| Unfiltered Data | 22.9% | 12.27% |
| Reject Reward | 10.4% | 0.04% |
| Reject Length | 13.9% | 0.02% |
| Reward Gap | 17.7% | 8.07% |
| *RIP* | 8.9% | 0.00% |

## 7. Conclusion

This work introduces Rejecting Instruction Preferences (*RIP*), a method for improving preference data quality by measuring the rejected response quality and the reward gap between the chosen and rejected response pair. Filtering instructions using *RIP* remarkably improves model alignment results on both human-written and synthetic instructions, and for different reward signals. In addition, we show that Self-*RIP*, synthetic instructions generated by few-shot prompts curated by *RIP*, outperforms organic user instructions and the standard Self-Instruct method, achieving the highest AlpacaEval2 win rate in our experiments.

## Impact Statement

This work demonstrates the possibility of dramatically improving LLMs by identifying and producing high-quality training data. Studying how filtering criteria affect outputs will continue to be important for LLM training. While we have primarily focused on preference optimization, the *RIP* approach is general and can potentially work for any training scheme, e.g. other RL training techniques – which future work should explore.

For such models, safety will also be crucial, and future work should additionally address this aspect. In our experiments, the reward is not explicitly constrained by safety-related criteria. Therefore, a clear further avenue of study is to conduct safety evaluations – and to explore safety filtering using our methods, with reward models built exclusively for safety in existing systems (Touvron et al., 2023b).

Given that we have shown that *RIP* can filter potentially unsafe prompts, this could mean in the best case that the safety of the model could potentially improve after filtering as well, with *RIP* being able to catch and mitigate more challenging safety situations that earlier iterations cannot. From a broader perspective, this work could pave the way for methods that produce higher-quality training instructions, that are also potentially safer than organic user instructions in the wild.

## References

Balioglu, C. fairseq2, 2023. URL http://github.com/facebookresearch/fairseq2.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Cao, Y., Kang, Y., Wang, C., and Sun, L. Instruction mining: Instruction data selection for tuning large language models. *arXiv preprint arXiv:2307.06290*, 2023.

Chen, L., Li, S., Yan, J., Wang, H., Gunaratna, K., Yadav, V., Tang, Z., Srinivasan, V., Zhou, T., Huang, H., et al. AlpaGasus: Training a better alpaca with fewer data. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=FdVXgSJhvz.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Honovich, O., Scialom, T., Levy, O., and Schick, T. Unnatural instructions: Tuning language models with (almost) no human labor. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14409–14428, Toronto, Canada,

July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.806. URL https://aclanthology.org/2023.acl-long.806.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Khaki, S., Li, J., Ma, L., Yang, L., and Ramachandra, P. Rs-dpo: A hybrid rejection sampling and direct preference optimization method for alignment of large language models. *arXiv preprint arXiv:2402.10038*, 2024.

Lambert, N., Morrison, J., Pyatkin, V., Huang, S., Ivison, H., Brahman, F., Miranda, L. J. V., Liu, A., Dziri, N., Lyu, S., et al. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Li, M., Zhang, Y., Li, Z., Chen, J., Chen, L., Cheng, N., Wang, J., Zhou, T., and Xiao, J. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. *arXiv preprint arXiv:2308.12032*, 2023a.

Li, M., Zhang, Y., He, S., Li, Z., Zhao, H., Wang, J., Cheng, N., and Zhou, T. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. *arXiv preprint arXiv:2402.00530*, 2024a.

Li, T., Chiang, W.-L., Frick, E., Dunlap, L., Zhu, B., Gonzalez, J. E., and Stoica, I. From live data to high-quality benchmarks: The arena-hard pipeline, April 2024b. URL https://lmsys.org/blog/2024-04-19-arena-hard/.

Li, X., Zhang, T., Dubois, Y., Taori, R., Gulrajani, I., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 2023b.

Li, X., Yu, P., Zhou, C., Schick, T., Zettlemoyer, L., Levy, O., Weston, J., and Lewis, M. Self-alignment with instruction backtranslation. In *The Twelfth International Conference on Learning Representations*, 2024c. URL https://openreview.net/forum?id=1oijHJBRsT.

Lin, B. Y., Deng, Y., Chandu, K., Brahman, F., Ravichander, A., Pyatkin, V., Dziri, N., Bras, R. L., and Choi, Y. Wildbench: Benchmarking llms with challenging tasks from real users in the wild, 2024. URL https://arxiv.org/abs/2406.04770.

Liu, W., Zeng, W., He, K., Jiang, Y., and He, J. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *arXiv preprint arXiv:2312.15685*, 2023.

Lu, K., Yuan, H., Yuan, Z., Lin, R., Lin, J., Tan, C., Zhou, C., and Zhou, J. # instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *The Twelfth International Conference on Learning Representations*, 2023.

Meng, Y., Xia, M., and Chen, D. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*, 2024.

Mishra, S., Khashabi, D., Baral, C., and Hajishirzi, H. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*, 2021.

Pace, A., Mallinson, J., Malmi, E., Krause, S., and Severyn, A. West-of-n: Synthetic preference generation for improved reward modeling. *arXiv preprint arXiv:2401.12086*, 2024.

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=HPuSIXJaa9.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Soldaini, L., Kinney, R., Bhagia, A., Schwenk, D., Atkinson, D., Authur, R., Bogin, B., Chandu, K., Dumas, J., Elazar, Y., et al. Dolma: An open corpus of three trillion tokens for language model pretraining research. *arXiv preprint arXiv:2402.00159*, 2024.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33: 3008–3021, 2020.

Toneva, M., Sordoni, A., Combes, R. T. d., Trischler, A., Bengio, Y., and Gordon, G. J. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Wang, H., Xiong, W., Xie, T., Zhao, H., and Zhang, T. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. *arXiv preprint arXiv:2406.12845*, 2024a.

Wang, T., Kulikov, I., Golovneva, O., Yu, P., Yuan, W., Dwivedi-Yu, J., Pang, R. Y., Fazel-Zarandi, M., Weston, J., and Li, X. Self-taught evaluators. *arXiv preprint arXiv:2408.02666*, 2024b.

Wang, Y., Mishra, S., Alipoormolabashi, P., Kordi, Y., Mirzaei, A., Arunkumar, A., Ashok, A., Dhanasekaran, A. S., Naik, A., Stap, D., et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*, 2022.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language models with self-generated instructions. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13484–13508, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.754. URL https://aclanthology.org/2023.acl-long.754.

Wang, Z., Dong, Y., Delalleau, O., Zeng, J., Shen, G., Egert, D., Zhang, J. J., Sreedhar, M. N., and Kuchaiev, O. Helpsteer2: Open-source dataset for training top-performing reward models. *arXiv preprint arXiv:2406.08673*, 2024c.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Wu, J., Xie, Y., Yang, Z., Wu, J., Gao, J., Ding, B., Wang, X., and He, X. $\beta$-dpo: Direct preference optimization with dynamic $\beta$. *arXiv preprint arXiv:2407.08639*, 2024a.

Wu, T., Yuan, W., Golovneva, O., Xu, J., Tian, Y., Jiao, J., Weston, J., and Sukhbaatar, S. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. *arXiv preprint arXiv:2407.19594*, 2024b.

Xia, M., Malladi, S., Gururangan, S., Arora, S., and Chen, D. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.

Xu, C., Sun, Q., Zheng, K., Geng, X., Zhao, P., Feng, J., Tao, C., and Jiang, D. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023a.

Xu, J., Lee, A., Sukhbaatar, S., and Weston, J. Some things are more cringe than others: Preference optimization with the pairwise cringe loss. *arXiv preprint arXiv:2312.16682*, 2023b.

Yang, S., Xie, Z., Peng, H., Xu, M., Sun, M., and Li, P. Dataset pruning: Reducing training data by examining generalization influence. *arXiv preprint arXiv:2205.09329*, 2022.

Yasunaga, M., Shamis, L., Zhou, C., Cohen, A., Weston, J., Zettlemoyer, L., and Ghazvininejad, M. Alma: Alignment with minimal annotation. *arXiv preprint arXiv:2412.04305*, 2024.

Yuan, W., Kulikov, I., Yu, P., Cho, K., Sukhbaatar, S., Weston, J., and Xu, J. Following length constraints in instructions. *arXiv preprint arXiv:2406.17744*, 2024a.

Yuan, W., Pang, R. Y., Cho, K., Sukhbaatar, S., Xu, J., and Weston, J. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024b.

Zhao, H., Andriushchenko, M., Croce, F., and Flammarion, N. Long is more for alignment: A simple but tough-to-beat baseline for instruction fine-tuning. *arXiv preprint arXiv:2402.04833*, 2024a.

Zhao, W., Ren, X., Hessel, J., Cardie, C., Choi, Y., and Deng, Y. Wildchat: 1m chatgpt interaction logs in the wild. *arXiv preprint arXiv:2405.01470*, 2024b.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=uccHPGDlao.

Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., Mao, Y., Ma, X., Efrat, A., Yu, P., Yu, L., et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36, 2024.

# A. Appendix

## A.1. More Details on Experiment Setup

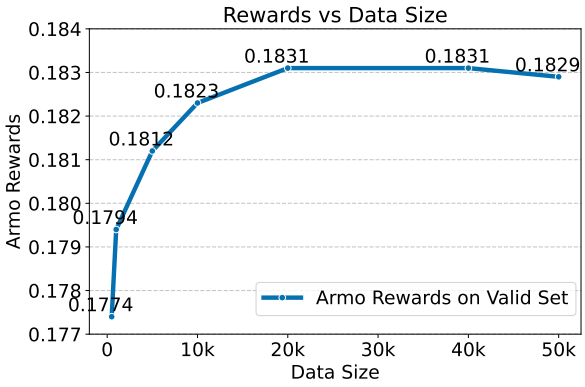
Our experiment setups are summarized in Table 13. Specifically, we apply *RIP* to multiple popular instruction-following datasets as well as our own synthetic data, with reward annotated from various sources (human/reward classifier/LLM-as-a-Judge), indicating the generalizability of our *RIP* method.

Table 8: **Preference Dataset Statistics** used for training in our experiments.

| | #Prompts | Human Written | #Responses | Reward Annotator | Valid Set ( # Examples) |
|---|---|---|---|---|---|
| WildChat-turn1 20k | 20,000 | Yes | 8,16,32,64 | ArmoRM | Humpback + Evol-Instruct (470) |
| WildChat-turn1 20k | 20,000 | Yes | 64 | LLM-as-a-Judge | Humpback + Evol-Instruct (470) |
| HelpSteer2 | 10,161 | Yes | 2 | Human | HelpSteer2 valid (519) |
| Self-Instruct | 20,000 | No | 64 | ArmoRM | Humpback + Evol-Instruct (470) |
| Self-*RIP* | 20,000 | No | 64 | ArmoRM | Humpback + Evol-Instruct (470) |

We report the model performance on valid set when varying the number of training WildChat prompts in Figure 2. Model training improves significantly as training data size grows to 20k and then begin to saturates afterwards, therefore our main experiments are based on those 20k WildChat prompts.

Figure 2: **Results on DPO Training with Varying WildChat Data Sizes.** Using different sizes of WildChat data for DPO training on LLaMA 3.1-8B-Instruct, the performance, measured by Armo rewards on the validation set, gradually saturates as the data size increases.




We primarily assess our models' general instruction-following capabilities using three popular evaluation benchmarks: AlpacaEval-2 (Li et al., 2023b), Arena-Hard (Li et al., 2024b) and WildBench (Lin et al., 2024). AplacaEval-2 consists of 805 prompts sampled from 5 datasets. Arena-Hard contains 500 challenging user queries sourced from Chatbot Arena and has the highest correlation and separability of models commpared to Chatbot Arena among popular open-ended LLM benchmarks (Li et al., 2024b). WildBench is built from a set of 1024 significantly harder, challenging queries carefully curated from the WildChat project (Zhao et al., 2024b) to ensure diversity and complexity. The automatic evaluation of WildBench involves task-specific checklists that guide LLM judges in generating reliable and consistent judgments which demonstrate significantly high correlation with human judgments. We report the WB-Score for individual scoring.

## A.2. Additional Results

**LLM-as-a-Judge As Reward Annotators** We explore LLM-as-a-Judge as alternative reward annotator apart from the reward model ArmoRM and human reward annotations, and use a LLama3.1-405B-Instruct zero-shot to judge the quality of each individual response and uses its prediction to construct response pairs. For each response, we conduct 10 independent evaluations and calculate the average score to determine the final reward score. We report AlpacaEval2, Arena-Hard and WildBench results on WildChat DPO models in Table 9. Similar to the observation from Table 2, *RIP* by filtering based on LLM-as-a-Judge predictions outperforms no filtering.

| WildChat with LLM-as-a-Judge as reward annotator | # Train examples | AlpacaEval2 LC Win | AlpacaEval2 Win | Arena-Hard Score | WildBench Score |
|---|---|---|---|---|---|
| Llama 3.1-8B-Instruct (seed model) | - | 20.9 | 21.8 | 21.3 | 33.1 |
| Standard DPO (no filtering) | 16837 | 40.1 | 44.9 | 41.1 | 42.5 |
| *RIP* | 5999 | 44.3 | 48.8 | 42.5 | 43.9 |

Table 9: ***RIP* compared to baselines on WildChat using LLM-as-a-Judge as the reward annotator.**. We report results on AlpacaEval2, Arena-Hard and WildBench of various models trained using DPO on the WildChat Dataset. *RIP* outperforms the baseline of LLM-as-judge as the reward annotator.

**Data Scaling with *RIP*** We further scale up *RIP* by growing the training data size after filtering to 20k, and achieves AlpacaEval2 LC win rate of 58.49% as shown in Figure 1. While the effective training size scales from 4538 to 20k, the actual performance gain only increase slightly, suggesting that training with Llama 3.1-8B-Instruct on existing WildChat prompts saturates, even under *RIP*.

***RIP* filtering thresholds** We report the filtering thresholds of the best checkpoints in our experiments in Table 14.

**Full Evaluation Results** We include full WildChat evaluation results on AlpacaEval2 and Arena-Hard in Table 15 and on WildBench in Table 16, with average response lengths, confidence intervals as well as finegrained results on subtasks. Full evaluation results on models trained on HelpSteer2 are presented in Table 17 and Table 18. In addition, full evaluation results on Self-*RIP* are included in Table 19 and Table 20.

**Coordinate-wise Filtering results.** We conduct extensive experiments by applying filtering to each individual metric: reward on chosen or rejected response, lengths of chosen or rejected response, reward gap, average reward of all responses, etc. Results on valid set performances by applying various filtering metrics to WildChat task are included in Table 24, Table 27 and HelpSteer2 in Table 25. Both highlight strong performance boost by filtering based on rejected reward, rejected length and reward gap.

**RIP on Gemma2 models** To show the effectiveness of our RIP filtering beyond Llama models, we finetune Gemma2-9B-it model with SimPO using the dataset (princeton-nlp/llama3-ultrafeedback-armorm) which are Gemma2 generations on ultrafeedback annotated by ArmoRM.

Table 10: **RIP on Gemma2-9B SimPO** Applying RIP on Gemma2-9B finetuning further improves Gemma2 performance on AlpacaEval from 69.48 to 73.81 by filtering out 50% train data.

| Filtering | #Prompts | Alpaca LC Winrate | Alpaca Winrate |
|---|---|---|---|
| Gemma2-9B SimPO (no filtering) | 59569 | 69.48 | 63.07 |
| Gemma2-9B SimPO (RIP filtering) | 29963 | 73.81 | 62.01 |

**RIP with Smaller-sized Reward Model** To show the effectiveness of RIP if reward models are of smaller size, and from a different model family, we select a lightweight non-Llama-based reward model "Ray2333/GRM-gemma2-2B-rewardmodel-ft", which is a Gemma2-2B based reward model, to annotate and then DPO finetune a Llama3.1-8B-Instruct model. Below are the results on RIP filtering using reward scores by this Gemma2-based RM. Ray2333/GRM-gemma2-2B-rewardmodel-ft (ranked 36th on Reward Bench) is ranked below ArmoRM on RewardBench, and the performance gap in the two reward model quality also affects the performances of finetuning Llama3.1-8B-Instruct with reward model annotations(i.e. better-quality reward model leads to better winrate of finetuned models). However, in both cases RIP filtering demonstrates its effectiveness.

**Comparison with $\beta-$DPO** $\beta$-DPO(Wu et al., 2024a) filtering is online, meaning they filter out data in every batch, whereas our approach filters data offline. This offline filtering enables more flexible and efficient generation pipelines, particularly for weak-to-strong generation scenarios. For instance, finetuning Llama3.3-70B-Instruct on prompts RIP filtered by a smaller Llama3.1-8B-Instruct model outperformed (Alpaca LC-winrate improved from 54.3 to 64.5, Arena-Hard from

Table 11: **RIP with Gemma2 based Reward Model.** By curating less than 5k out of 20k prompts, we can improve Llama3.1-8B-Instruct DPO models from 41.1 to 49.9 on AlpacaEval LC-winrate, showing similar improvement using ArmoRM filtering (LC winrate improved from 48.4% to 57.8%).

| Filtering | #Prompts | Alpaca LC Winrate | Alpaca Winrate |
|---|---|---|---|
| WildChat20k baseline | 20000 | 41.1 | 47.3 |
| WildChat RIP | 4401 | 49.9 | 53.5 |

Table 12: **RIP on Gemma2-9B SimPO** Applying RIP on Gemma2-9B finetuning further improves Gemma2 performance on AlpacaEval from 69.48 to 73.81 by filtering out 50% train data.

| Filtering | #Prompts | Alpaca LC Winrate | Alpaca Winrate |
|---|---|---|---|
| Gemma2-9B SimPO (no filtering) | 59569 | 69.48 | 63.07 |
| Gemma2-9B SimPO (RIP filtering) | 29963 | 73.81 | 62.01 |

70.5 to 76.7). There are some noticeable distinctions between our methods and $\beta$-DPO. $\beta$-DPO removes both small and large gaps, our method removes bigger gaps only. In addition, $\beta$-DPO's filtering is probabilistic, resulting in incomplete data removal, whereas our approach uses deterministic filtering to ensure thorough removal of unwanted data. Given these differences, and as previously mentioned, our method prioritizes Rejected Reward and Rejected Length criteria over gap-based filtering, which have demonstrated superior effectiveness in our experiments.

Table 13: **Preference Dataset Statistics** used for training in our experiments.

| Filtering | $\beta$-DPO mode weight | #Prompts | Valid Score | Alpaca LC Winrate | Alpaca Winrate |
|---|---|---|---|---|---|
| No filtering | - | 19803 | 0.1830 | 48.37 | 45.87 |
| RIP filtering | - | 4538 | 0.1898 | 57.83 | 57.16 |
| $\beta$-DPO Filter | 0.2 | 15842 | 0.1842 | 49.15 | 49.00 |
| $\beta$-DPO Filter | 0.5 | 9901 | 0.1840 | 46.68 | 42.41 |
| $\beta$-DPO Filter | 0.75 | 4950 | 0.1827 | 45.97 | 40.58 |

### A.3. t-SNE Analysis of Filtered Instructions

We conduct t-SNE analysis on WildChat prompts filtered by rejected response length and reward in Figure 5 and those further filtered by reward gap in Figure 6. To better understand which prompts are being filtered out, we summarize prompts being filtered out due to rejected responses being of shorter length or lower reward in Table 32 and Figure 7, and those filtered out due to large reward gaps in Table 33. In addition to visualizing the examples, we also conduct GPT4 analysis into quality of the filtered out prompts in Section 6.2 by each criterion, to justify our hypothesis. We include one such sample analysis below. A prompt "Write a story" is filter out (rejected ARMO, gap), and is considered not useful as this prompt is overly broad and lacks specific details, posing challenges in generating a focused response.

### A.4. Further Ablations

We report results of further ablation studies: comparing filtering and various pairing instead of filtering methods in Table 21, and robustness of *RIP* to choice of responses in rejection sampling in Table 22.

***RIP* outperforms alternative preference pairing methods** We compare *RIP* to methods without filtering that use different response pairing methods for building pairwise preferences. Recall that in our main experiments for *RIP* we used the best-vs-worst pairing method as described in Section 3.1. Here we explore two alternative methods: (i) best-vs-random which is shown by existing work (Yasunaga et al., 2024; Khaki et al., 2024) to outperform best-vs-worst, and (ii) best-vs-bottom-K% percentile where the rejected response has the bottom $K = 25, 50, 75$ percentile score ($K = 0$ being the lowest score). Both pairing methods can effectively lower reward gap and increase quality of rejected response without removing training prompts. We report model performance on the valid set in Table 21. Out of all pairing methods, best-vs-bottom-25% works the best, but still under-performs compared with our *RIP* method (pairing with best-vs-worst). When evaluated on

| Data | # Train Examples | Human Written | # Responses | Reward Annotator | Seed Model | Filtering Metrics | AlpacaEval2 LC Win | AlpacaEval2 Win | Arena-Hard | Wildbench |
|---|---|---|---|---|---|---|---|---|---|---|
| - | - | - | - | - | Llama 3.1-8B-Instruct | - | 20.9 | 21.8 | 21.3 | 33.1 |
| - | - | - | - | - | Llama 3.3-70B-Instruct | - | 38.9 | 41.5 | | 52.8 |
| Wildchat 20k | 20k | Yes | 64 | ArmoRM | Llama 3.1-8B-Instruct | No | 48.37 | 45.87 | 37.9 | 41.5 |
| Wildchat 20k | 6762 | Yes | 64 | ArmoRM | Llama 3.1-8B-Instruct | Rejected Length $\geq$ 1878, Rejected Armo $\geq$ 0.126 | 57.1 | 52.9 | 42.3 | 45.5 |
| Wildchat 20k | 4538 | Yes | 64 | ArmoRM | Llama 3.1-8B-Instruct | Rejected Length $\geq$ 1878, Rejected Armo $\geq$ 0.126, Reward Gap $\leq$ 0.042 | 57.8 | 57.2 | 43.1 | 45.6 |
| Synthetic (few shot: Wildchat 20k) | 20k | No | 64 | ArmoRM | Llama 3.1-8B-Instruct | No | 49.1 | 46.9 | 38.5 | 41.0 |
| Synthetic (few shot: Wildchat 20k) | 16k | No | 64 | ArmoRM | Llama 3.1-8B-Instruct | Rejected Length $\geq$ 1878, Rejected Armo $\geq$ 0.126 | 58.3 | 53.2 | 40.9 | 44.1 |
| Synthetic (few shot: Wildchat filtered 4538 examples) | 20k | No | 64 | ArmoRM | Llama 3.1-8B-Instruct | No | 53.6 | 56.1 | 43.7 | 44.8 |
| Synthetic (few shot: Wildchat filtered 4538 examples) | 18812 | No | 64 | ArmoRM | Llama 3.1-8B-Instruct | Rejected Length $\geq$ 1878, Rejected Armo $\geq$ 0.126 | 60.2 | 61.1 | 42.1 | 42.5 |
| Wildchat 20k | 16.8k | Yes | 64 | LLM-as-a-Judge | Llama 3.1-8B-Instruct | No | 40.1 | 44.9 | 41.1 | 42.5 |
| Wildchat 20k | 5999 | Yes | 64 | LLM-as-a-Judge | Llama 3.1-8B-Instruct | Rejected LLM-as-a-Judge Reward $\geq$ 8, Rejected Length $\geq$ 1399, Reward Gap $\leq$ 1 | 44.3 | 48.8 | 42.5 | 43.9 |
| HelpSteer | 10k | Yes | 64 | Human | Llama 3.1-8B-Instruct | No | 25.2 | 23.1 | 26.8 | 37.1 |
| HelpSteer | 5081 | Yes | 64 | Human | Llama 3.1-8B-Instruct | Rejected Length $\geq$ 1303 | 34.6 | 32.8 | 35.0 | 39.5 |
| Wildchat 40k | 40k | Yes | 32 | ArmoRM | Llama 3.3-70B-Instruct | No | 54.3 | 51.6 | 70.5 | 55.3 |
| Wildchat 40k | 17.7k | Yes | 32 | ArmoRM | Llama 3.3-70B-Instruct | Rejected Length $>$ 1878, Rejected Armo $>$ 0.126, GAP $\leq$ 0.042 | 67.7 | 73.2 | 82.9 | 58.8 |

Table 14: **Full Results** details on number of training examples, choice of reward models, seed models, filtering metrics and thresholds chosen as well as final outcomes across 3 evaluation benchmarks.

Table 15: **Full AlpacaEval2 & Arena-Hard Results on WildChat**: we compare performances of SOTA models on AlpacaEval2 win rates and Arena-Hard scores as well as DPO models trained on the WildChat-20k dataset using various filtering methods.

| | # Train | AlpacaEval2 | | | Arena-Hard | | |
|---|---|---|---|---|---|---|---|
| *Standard models* | examples | LC Win | Win | Len | Score | 95% CI | Len |
| GPT-4 Omni (05/13) | - | 57.5 | 51.3 | 1873 | 74.9 | (-2.5, 1.9) | 668 |
| GPT-4 Turbo (04/09) | - | 55.0 | 46.1 | 1802 | 82.6 | (-1.6, 1.8) | 662 |
| Gpt-4-0613 | - | 55.0 | 46.1 | 1802 | 37.9 | (-2.8, 2.4) | 354 |
| Llama 3.1-405B-Instruct | - | 39.3 | 39.1 | 1988 | 67.1 | (-2.2, 2.8) | 658 |
| Llama 3.1-70B-Instruct | - | 38.1 | 39.1 | 2044 | 69.3 | (-2.5, 2.5) | 658 |
| ***Baseline*** | | | | | | | |
| Llama 3.1-8B-Instruct | - | 20.9 | 21.8 | 2184 | 21.3 | (-1.9, 2.2) | 861 |
| WildChat-20k DPO (no filtering) | 20000 | 48.4 | 45.9 | 2134 | 37.9 | (-2.0, 2.2) | 622 |
| WildChat-20k DPO (best-vs-bottom-25%) | 20000 | 48.2 | 45.9 | 1971 | 40.7 | (-2.1, 1.9) | 741 |
| ***Prompt-Based Filtering*** | | | | | | | |
| Jaccard Similarity | 9904 | 49.0 | 46.6 | 1978 | 42.6 | (-2.4, 2.3) | 632 |
| LLM-as-Prompt-Judge Binary | 4299 | 45.5 | 41.0 | 1859 | 42.0 | (-1.4, 1.7) | 597 |
| LLM-as-Prompt-Judge Pointwise | 15963 | 47.4 | 47.4 | 2056 | 40.7 | (-2.0, 2.2) | 701 |
| InsTag-Difficulty | 10000 | 46.3 | 39.0 | 1752 | 39.0 | (-2.2, 2.3) | 602 |
| InsTag-Diversity | 9952 | 40.1 | 41.1 | 1903 | 40.4 | (-2.4, 2.8) | 579 |
| ***Prompt-and-Chosen-Response-Based Filtering*** | | | | | | | |
| IFD on Prompt + Chosen Response | 9902 | 47.6 | 37.6 | 1655 | 32.2 | (-1.7, 2.5) | 533 |
| ppl(Chosen Response) | 14851 | 45.6 | 45.5 | 1930 | 40.8 | (-2.3, 1.7) | 582 |
| ***Chosen-Rejected-Response Based Filtering*** | | | | | | | |
| LLM-as-Prompt-Judge Pointwise | 15963 | 47.4 | 47.4 | 2056 | 40.7 | (-2.0, 2.2) | 701 |
| *RIP* | 4538 | 57.8 | 57.2 | 2048 | 43.1 | (-1.5, 1.8) | 638 |

AlpacaEval2, Arena-Hard, and WildBench, the model WildChat-20k DPO (best-vs-bottom-25%) only achieves a slight improvement gain comparing to baseline WildChat-20k DPO (best-vs-worst), while still underperforming *RIP* as shown in Table 2. This result indicates that reward gap being small or the rejected reward being high better works as an indication of a low-quality prompt rather than bad response pairing.

**Combining alternative pairing with *RIP* performs on par with best-vs-bottom pairing with *RIP*.** We further apply *RIP* filtering to examples paired by best-vs-bottom-25% pairing. Combing best-vs-bottom-25% with filtering out examples of low quality rejected responses yields ArmoRM Score of 0.18675, slightly lower than best-vs-worst + filtering by Rejected Reward (0.18795). Filtering out best-vs-bottom-25% examples of bigger reward gaps yields to Armo Score of 0.1860 on valid set as compared to 0.18542 from best-vs-worst pairing + filtering by Reward Gaps. Given the marginal performance gain between best-vs-worst and best-vs-bottom-25% pairing with and without *RIP*, we thus focus on the more widely adopted best-vs-worst pairing to experiment various filtering methods including our *RIP* method.

***RIP* is robust to the choice of the number of responses $N$.** While we showed *RIP* provides strong performance on HelpSteer2 where only $N = 2$ responses are availabe for each prompt, and on WildChat with $N = 64$ responses sampled per prompt, we also compare the performance of *RIP* by varying the choice of $N$ the number of candidate responses generated for preference annotations in the WildChat setup. As shown in Table 22, for a wide range of values $N = 64, 32, 16, 8$, *RIP* consistently outperforms the no filtering baseline, with larger $N$ achieving increasingly better performance, likely due to the increased quality and variability of chosen and rejected responses, allowing our *RIP* metrics to be more accurate in curating high quality data.

**Self-*RIP* works with much smaller set of high-quality seed instructions** Instead of using all 4538 *RIP* curated high-quality instructions as seed instructions $S$ during Step 1. few-shot generations, we sample a much shorter subset of 256 prompts from 4538 *RIP*-curated prompts as seed instructions, and only conduct few-shot generations by sampling 8 prompts from the 256 seed prompts each time. We report Self-*RIP* with and without post-filtering in Table 23. Self-*RIP* based on 256 high-quality seed instructions (58.9) slightly underperforms than that based on 4538 seed prompts (60.2), but

Table 16: **Full WildBench Results on WildChat**: we compare performances of SOTA models on WildBench as well as DPO models trained on the WildChat-20k dataset using various filtering methods.

| | WB-Score | WB-Score: Task-specific | | | | |
|---|---|---|---|---|---|---|
| *Standard models* | Weighted average | Creative | Planning & Reasoning | Math & Data Analysis | Information | Coding & Debugging |
| GPT-4 Omni (05/13) | 59.3 | 59.1 | 60.2 | 57.3 | 58.6 | 60.5 |
| GPT-4 Turbo (04/09) | 55.2 | 58.7 | 56.2 | 51.0 | 57.2 | 55.1 |
| Gemini-1.5-pro | 53.0 | 55.1 | 53.7 | 48.6 | 52.2 | 55.2 |
| Llama3-70B-Instruct | 47.8 | 54.3 | 50.1 | 42.1 | 52.3 | 44.7 |
| ***Baseline*** | | | | | | |
| Llama 3.1-8B-Instruct | 33.1 | 45.0 | 37.0 | 23.9 | 37.4 | 29.3 |
| WildChat-20k DPO (no filtering) | 41.5 | 51.8 | 44.2 | 32.2 | 50.0 | 37.1 |
| WildChat-20k DPO (best-vs-bottom-25%) | 44.5 | 53.9 | 47.4 | 35.8 | 50.4 | 41.4 |
| ***Prompt-Based-Filtering*** | | | | | | |
| Jaccard Similarity | 43.7 | 54.2 | 46.9 | 34.3 | 49.5 | 40.5 |
| LLM-as-Prompt-Judge Binary | 43.3 | 53.9 | 46.6 | 35.8 | 48.5 | 38.6 |
| LLM-as-Prompt-Judge Pointwise | 45.2 | 55.6 | 48.0 | 37.1 | 51.6 | 40.9 |
| InsTag-Difficulty | 42.4 | 52.7 | 45.4 | 33.4 | 47.8 | 39.3 |
| InsTag-Diversity | 43.4 | 53.4 | 46.1 | 35.0 | 49.1 | 40.1 |
| ***Prompt-and-Chosen-Response-Based Filtering*** | | | | | | |
| IFD | 42.2 | 51.3 | 45.9 | 35.0 | 48.0 | 37.1 |
| ppl(Chosen Response) | 43.4 | 52.5 | 47.0 | 37.2 | 49.4 | 37.6 |
| ***Chosen-Rejected-Response Based Filtering*** | | | | | | |
| LLM-as-Prompt-Judge Pointwise | 45.2 | 55.6 | 48.0 | 37.1 | 51.6 | 40.9 |
| *RIP* | 45.6 | 56.7 | 48.8 | 36.6 | 51.6 | 41.4 |

Table 17: **Results of our DPO models trained with HelpSteer2.** Full AlpacaEval2 & Arena-Hard Results of our DPO models trained with HelpSteer2 Dataset.

| | | AlpacaEval2 | | | Arena-Hard | | |
|---|---|---|---|---|---|---|---|
| | Prompts | LC Win | Win | Len | Score | 95% CI | Len |
| ***Baseline*** | | | | | | | |
| Llama 3.1-8B-Instruct | - | 20.9 | 21.8 | 2184 | 21.3 | (-1.9, 2.2) | 861 |
| HelpSteer2 DPO (no filtering) | 10161 | 25.2 | 23.1 | 1733 | 26.8 | (-2.0, 2.4) | 606 |
| ***Prompt-Based-Filtering*** | | | | | | | |
| LLM-as-Prompt-Judge Pointwise | 5376 | 27.8 | 25.7 | 1947 | 29.5 | (-2.8, 2.3) | 627 |
| ***Prompt-Response-Based-Filtering*** | | | | | | | |
| *RIP* | 5081 | 34.6 | 32.8 | 1941 | 35.0 | (-1.8, 2.2) | 621 |

still outperforms Self-Instruct with *RIP* post-filtering (58.3) as well as Self-*RIP* based on all 4538 seed prompts without post-filtering (53.6), indicating that our method Self-*RIP* can work well with a much smaller set of high-quality seed prompts.

### A.5. Details about Baselines

**InsTag Complexity** Lu et al. (2023) utilized ChatGPT to generate semantic and intent-based tags, which were then used to fine-tune a large language model (LLM) data tagger. The number of tags per prompt served as a complexity metric. Building on their methodology, we employed a publicly available tagger (`https://github.com/OFA-Sys/InsTag`. Note that Meta was not involved in the training of the Instag model we used.) to annotate each prompt, generating between 1 and 100 tags per prompt. We then categorized our training prompts into four groups based on the number of tags: more than 2, more than 3, more than 4, and more than 5. From each group, we randomly sampled 10,000 training data samples and trained a distinct model for each group. It is important to note that a threshold of $\geq 1$ implies no filtering, with only

Table 18: **Results on our DPO models trained with HelpSteer2.** Full WildBench results of our DPO models trained with HelpSteer2 Dataset.

| | WB-Score | WB-Score: Task-specific | | | | |
|---|---|---|---|---|---|---|
| | Weighted average | Creative | Planning & Reasoning | Math & Data Analysis | Information | Coding & Debugging |
| ***Baseline*** | | | | | | |
| Llama 3.1-8B-Instruct | 33.1 | 45.0 | 37.0 | 23.9 | 37.4 | 29.3 |
| HelpSteer2 DPO (no filtering) | 37.1 | 48.6 | 40.4 | 26.5 | 44.3 | 33.4 |
| ***Prompt-Based-Filtering*** | | | | | | |
| LLM-as-Prompt-Judge Pointwise | 37.2 | 50.6 | 40.0 | 27.9 | 43.0 | 33.1 |
| ***Prompt-Response-Based-Filtering*** | | | | | | |
| *RIP* | 39.5 | 52.1 | 42.9 | 29.3 | 46.4 | 35.0 |

Table 19: **Results of our DPO models trained with Self-Instructed Dataset:** Full AlpacaEval2 & Arena-Hard Results comparing our method with training on standard Self-Instruct dataset.

| | | | AlpacaEval2 | | | Arena-Hard | | |
|---|---|---|---|---|---|---|---|---|
| Training Prompts | Filtering | # Train examples | LC Win | Win | Len | Score | 95% CI | Len |
| Self-Instruct | None | 20000 | 49.1 | 46.9 | 1956 | 38.5 | (-1.4, 1.6) | 738 |
| Self-*RIP* (without post-filtering) | None | 20000 | 53.6 | 56.1 | 2252 | 43.7 | (-2.3, 2.3) | 777 |
| Self-Instruct with *RIP* post-filtering | *RIP* | 16261 | 58.3 | 53.2 | 1823 | 40.9 | (-1.9, 1.6) | 560 |
| Self-*RIP* | *RIP* | 18812 | **60.2** | **61.1** | 2121 | 42.1 | (-2.0, 2.4) | 606 |

Table 20: **Results of our DPO models trained with Self-Instructed Dataset:** Full WildBench Results comparing our method with training on standard Self-Instruct dataset.

| | | WB-Score | WB-Score: Task-specific | | | | |
|---|---|---|---|---|---|---|---|
| Training Prompts | Filtering | Weighted average | Creative | Planning & Reasoning | Math & Data Analysis | Information | Coding & Debugging |
| Self-Instruct | None | 41.0 | 51.6 | 43.3 | 31.4 | 47.7 | 38.0 |
| Self-*RIP* (without post-filtering) | None | 44.8 | 55.3 | 46.9 | 33.5 | 49.7 | 44.6 |
| Self-Instruct with *RIP* post-filtering | *RIP* | 44.1 | 54.8 | 47.3 | 36.4 | 48.2 | 40.3 |
| Self-*RIP* | *RIP* | 42.5 | 54.1 | 46.2 | 32.8 | 48.6 | 38.2 |

Table 21: **Results of pair selections:** We report Armo scores on valid sets by varying different pairing methods instead of filtering prompts. Best pairing result 0.1842 is achieved with appointing response with bottom 25% score as rejected, although still underperforming compared to our filtering method (0.1898).

| Pair | Armo Score on Valid |
|---|---|
| Chosen=HighestScore, Rejected=LowestScore | 0.1830 |
| Chosen=HighestScore, Rejected=BottomScore25% | 0.1842 |
| Chosen=HighestScore, Rejected=BottomScore50% | 0.1839 |
| Chosen=HighestScore, Rejected=BottomScore75% | 0.1821 |
| Chosen=HighestScore, Rejected=Random | 0.1835 |
| Chosen=HighestScore, Rejected=LowestScore + *RIP* | 0.1898 |

a random sample of 10,000 data points from WildChat. A threshold of $\geq 2$ means filtering out prompts with only 1 tag. As shown in Table 28, a threshold of $\geq 2$ yields the best performance using the InsTag Complexity filtering method. We reported the results for a threshold of $\geq 2$ in Table 2.

Table 22: **Results on Varying $N = 8, 16, 32, 64$ number of responses sampled per prompts in Response Generation**: Armo Score on Valid set of our DPO models trained with WildChat Dataset all increases after filtering based on *RIP* regardless of the choice of $N$ in response generation step.

| $N$ | Before Filtering Armo Score | After Filtering Armo Score | Gain |
|---|---|---|---|
| 8 | 0.1821 | 0.1860 | 0.0039 |
| 16 | 0.1827 | 0.1878 | 0.0051 |
| 32 | 0.1829 | 0.1882 | 0.0053 |
| 64 | 0.1831 | 0.1898 | 0.0067 |

Table 23: **Self-*RIP* for generating high-quality synthetic instructions by varying number of fewshots**. Self-*RIP* creates prompts using few-shot samples from high-quality prompts curated by *RIP*, whereas Self-Instruct uses few-shots from unfiltered WildChat prompts. Applying *RIP* filtering *after* generation is also important, and achieves the best results, significantly outperforming Self-Instruct data.

| Train Prompts | # Seed Prompts | # Train examples | AlpacaEval2 LC Win | AlpacaEval2 Win |
|---|---|---|---|---|
| Llama 3.1-8B-Instruct (seed model) | - | - | 20.9 | 21.8 |
| WildChat-20k + *RIP* | - | 4538 | 57.8 | 57.2 |
| Self-Instruct + *RIP* | 20000 | 16261 | 58.3 | 3.2 |
| Self-*RIP* (without post-filtering) | 256 | 20000 | 50.0 | 51,2 |
| Self-*RIP* (without post-filtering) | 4538 | 20000 | 53.6 | 56.1 |
| Self-*RIP* | 256 | 15619 | 58.9 | **63.1** |
| Self-*RIP* | 4538 | 18812 | **60.2** | 61.1 |

**InsTag Diversity** The InsTag Diversity filtering method (Lu et al., 2023) considers a dataset to be more diverse if it contains a larger number of unique tags, as annotated by the aforementioned tagger. We employed two metrics to manage InsTag Diversity:

1. Tag Frequency: We deem a tag valid if it meets a predefined frequency threshold. This approach addresses the issue of infrequent tags, such as "serve size" and "market failure," which appeared only once or twice in the entire Wildchat dataset, suggesting they may not represent valid categories. In contrast, more common tags like "creative writing" and "information retrieval" are more appropriate for categorizing prompt data.

2. Max prompt per Tag: This metric controls the coverage ratio of unique tags. If a prompt contains only tags that have already been covered by the selected set, we discard the prompt to ensure diversity.

Table 29 presents the performance results when diversity is controlled using the two metrics described above. To ensure fairness, we downsampled the training data for each experiment to 10,000 samples. The results indicate that the model achieves optimal performance when the Tag Frequency is set to 6 and the Max Prompt per Tag is set to 3. This means we only consider tags that appear more than six times in the entire Wildchat dataset, and we allow a maximum of three prompts per tag. The best performance results are reported in Table 2.

**Perplexity** To curate training prompts, we compute the perplexity (ppl) of the selected response $y_w$ using the Llama-3.1-8B-Instruct model in a zero-shot setting. We use this perplexity as a filtering metric, specifically retaining examples with high $\text{ppl}(y_w|x)$ values, which may indicate more challenging prompts. We adjust the quantile range to control perplexity, calculating $\text{ppl}(y_w|x)$ for 20,000 Wildchat data points and filtering them based on this range. Table 30 displays model performance across different ppl quantile ranges. As shown, the quantile range of 25-100 yields the best performance, and we report this model's performance in Table 2.

**Instruction-Following Difficulty (IFD)** Li et al. (2023a) introduced the IFD to measure the model-specific difficulty of a data sample. In the instruction-tuning process, the loss of a sample pair (Q, A) is calculated by continuously predicting the next tokens given the instruction Q and their proceeding words:

Table 24: Performance of Different Filter Methods Across Quantile Ranges on WildChat with ArmoRM as reward anotator.

| Method | 0-100 | 10-100 | 25-100 | 50-100 | 60-100 | 75-100 |
|---|---|---|---|---|---|---|
| Chosen Reward | 0.18305 | 0.18325 | 0.18409 | 0.18393 | 0.18380 | 0.18333 |
| Rejected Reward | 0.18305 | 0.18411 | 0.18405 | 0.18566 | 0.18797 | 0.18795 |
| Average Reward | 0.18305 | 0.18368 | 0.18392 | 0.18494 | 0.18468 | 0.18442 |
| Chosen Length | 0.18305 | 0.18350 | 0.18366 | 0.18278 | 0.18226 | 0.18105 |
| Rejected Length | 0.18305 | 0.18377 | 0.18340 | 0.18571 | 0.18593 | 0.18473 |

| Method | 0-100 | 0-25 | 0-50 | 50-100 |
|---|---|---|---|---|
| Reward Gap | 0.18305 | 0.18405 | 0.18542 | 0.17993 |

Table 25: Performance of Different Filter Methods Across Quantile Ranges on HelpSteer2 valid set.

| Method | 0-25 | 0-50 | 0-75 | 0-100 | 25-100 | 50-100 | 75-100 |
|---|---|---|---|---|---|---|---|
| Chosen Human Reward | 0.1469 | 0.1451 | 0.1456 | 0.1458 | 0.1461 | 0.1454 | 0.1465 |
| Rejected Human Reward | 0.1442 | 0.1455 | 0.1459 | 0.1458 | 0.1480 | 0.1470 | 0.1461 |
| Chosen Length | 0.1484 | 0.1467 | 0.1455 | 0.1458 | 0.1454 | 0.1446 | 0.1449 |
| Rejected Length | 0.1421 | 0.1430 | 0.1445 | 0.1458 | 0.1495 | 0.1513 | 0.1478 |
| Human Reward Gap | 0.1482 | 0.1480 | 0.1466 | 0.1458 | 0.1448 | 0.1448 | 0.1441 |

Table 26: **Results on Applying RIP individual metric.** As shown below, applying each individual RIP metric all yields better performance compared to no filtering. In addition, apply all 3 metrics outperforms filtering with individual metric.

| *Filtering Metric* | Valid Armo Score | Alpaca LC Winrate |
|---|---|---|
| No Filtering | 0.1830 | 48.37 |
| Rejected Armo | 0.18979 | 56.91 |
| Rejected Length | 0.18593 | 53.31 |
| Reward Gap | 0.18542 | 51.01 |
| Apply all | 0.18983 | 57.83 |

Table 27: **Results on Applying RIP metrics accumulatively** These findings from our SimPO experiments are consistent with our previous DPO experiments, which demonstrated that Rejected Armo is the most effective metric. The addition of rejected length also proved to be highly effective, while gap filtering provided some benefits, albeit to a lesser extent than the other two metrics.

| *Filtering Metric* | *# Prompts* | Alpaca LC Winrate | Alpaca LC Winrate |
|---|---|---|---|
| Llama3.1-8b SimPO (no filtering) | 19803 | 51.28 | 40.55 |
| Llama3.1-8b SimPO (RIP filtering, Rejected Armo) | 8068 | 54.02 | 43.51 |
| Llama3.1-8b SimPO (RIP filtering, Rejected Armo, Gap) | 6629 | 53.04 | 43.02 |
| Llama3.1-8b SimPO (RIP filtering, Rejected Armo, Rejected Length, Gap) | 4538 | 53.32 | 43.81 |

$$L_\theta(A \mid Q) = -\frac{1}{N}\sum_{i=1}^{N} \log P\left(w_i^A \mid Q, w_1^A, w_2^A, \ldots, w_{i-1}^A; \theta\right) \tag{1}$$

where N is the number of words of the groundtruth answer A. They denote this averaged crossentropy loss as the Conditioned Answer Score $S_\theta(A \mid Q) = L_\theta(A \mid Q)$.

Then they introduce the Direct Answer Score $S_\theta(A)$

Figure 3: **GPT4 eval prompt.**

I have a collection of prompts that I need to evaluate for their effectiveness in fine-tuning a language model.
A useful prompt should:
- Clearly ask a question
- Be concise and specific
- Directly relate to the topic of interest or follow given instructions

Please assess each prompt and assign a score from 1 to 5 based on its usefulness:
- 1: Pretty useful
- 2: Somewhat useful
- 3: Neutral (neither useful nor harmful)
- 4: Somewhat harmful
- 5: Harmful
Make sure to clearly indicate the score at the end of your evaluation using the format: Score: x
Prompt: **{prompt}**

Figure 4: **Self-Instruct few-shot prompt template.**

Below are sample tasks from user.
1. <begin>**{INSTRUCTION 1}**</end>
2. <begin>**{INSTRUCTION 2}**</end>
3. <begin>**{INSTRUCTION 3}**</end>
4. <begin>**{INSTRUCTION 4}**</end>
5. <begin>**{INSTRUCTION 5}**</end>
6. <begin>**{INSTRUCTION 6}**</end>
7. <begin>**{INSTRUCTION 7}**</end>
8. <begin>**{INSTRUCTION 8}**</end>

Come up with a series of new tasks, wrapped with <begin>and </end>
9.

$$s_\theta(A) = -\frac{1}{N}\sum_{i=1}^{N} \log P\left(w_i^A \mid w_1^A, \ldots, w_{i-1}^A; \theta\right) \tag{2}$$

Finally, they estimate the Instruction-Following Difficulty (IFD) scores $IFD_\theta(Q, A)$ on following instruction of a given (Q, A) pairs by calculating the ratio between $S_\theta(A)$ and $S_\theta(A \mid Q)$:

$$\text{IFD}_\theta(Q, A) = \frac{s_\theta(A \mid Q)}{s_\theta(A)} \tag{3}$$

We calculated the IFD scores for 20,000 Wildchat data points and filtered them based on specific ranges. As shown in Table 31, filtering with a range of 25-100 yielded the best performance. The performance of this model is reported in Table 2.

Table 28: Model performance with InsTag Complexity Filtering

| Tag threshold | Armo Reward on Valid Set |
|---|---|
| $>= 1$ | 0.1820 |
| $>= 2$ | 0.1826 |
| $>= 3$ | 0.1812 |
| $>= 4$ | 0.1815 |
| $>= 5$ | 0.1818 |

Table 29: Model performance with InsTag Diversity Filtering

| Tag Frequency | Max prompt per Tag | Armo Reward on Valid Set |
|---|---|---|
| 1 | 1 | 0.1809 |
| 1 | 2 | 0.1799 |
| 2 | 1 | 0.1798 |
| 2 | 2 | 0.1800 |
| 3 | 1 | 0.1799 |
| 3 | 2 | 0.1797 |
| 3 | 3 | 0.1814 |
| 4 | 3 | 0.1821 |
| 5 | 3 | 0.1818 |
| 6 | 3 | 0.1831 |

Table 30: Model performance with Perplexity Filtering

| Quantile Range | Armo Reward on Valid Set |
|---|---|
| 25-100 | 0.1833 |
| 50-100 | 0.1827 |
| 75-100 | 0.1797 |

Table 31: Model performance with IFD Filtering

| Quantile Range | Armo Reward on Valid Set |
|---|---|
| 0-25 | 0.1815 |
| 0-50 | 0.1823 |
| 0-75 | 0.1832 |
| 25-100 | 0.1835 |

Figure 5: **t-SNE plots on instructions before and after filtering by rewards and lengths of rejected responses.** Red dots represent unfiltered instructions, while blue dots are instructions curated by filtering out those with low-reward and shorter rejected responses.

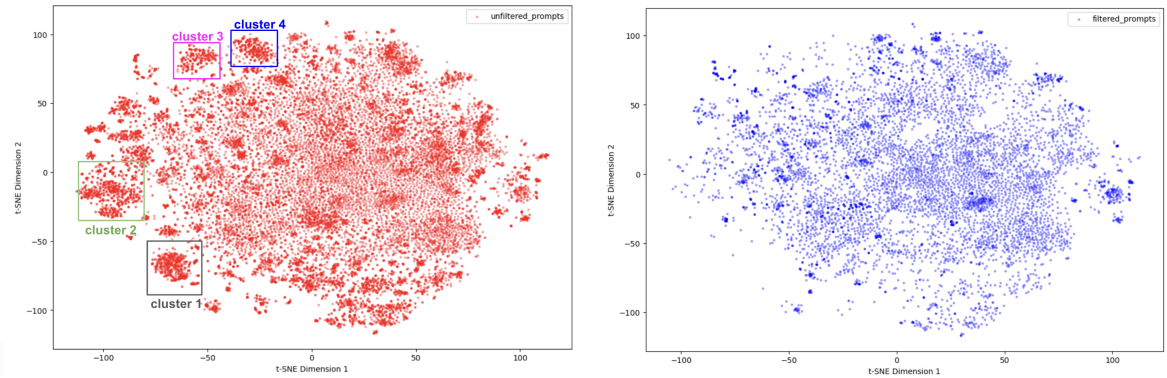



Figure 6: **t-SNE plots on instructions before and after filtering by reward gaps.** Blue dots represent instructions filtered only by rejected response, while yellow dots are instructions curated with smaller gap.

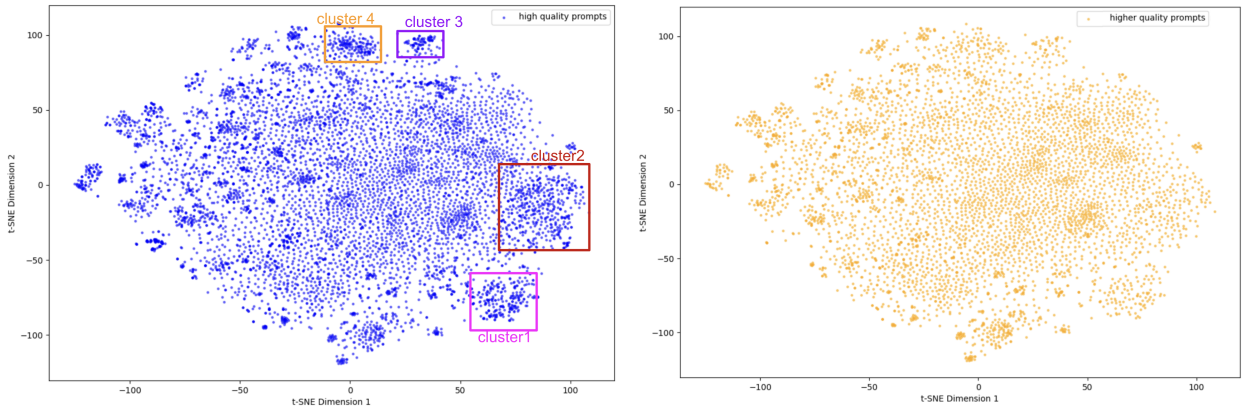


Table 32: **Noisy instruction filtered based on rejected responses of lower scores and shorter lengths.** We expand 4 clusters of instructions highlighted in Figure 5 for a better understanding of what instructions are being filtered out by measuring quality of rejected responses.

| Cluster | Description | Rejecting Reason | Rejected Instruction |
|---|---|---|---|
| Cluster 1 | 646 instructions in the format of *"give me a response to "'<text>"' to send in a discussion, VERY SHORT, CONCISE & CLEAR. ONLY RETURN THE RAW MESSAGE, DO NOT SAY "Hey here is the message you asked""*, where <*text*> refers to a single-turn conversational message. | Around 90% of rejected responses are of shorter lengths and lower scores below 25% percentile, even though their scores are higher than average rejected scores. These short and concise conversational responses are shorter thus potentially more generic and less informative for the models to further improve upon. | give me a response to "'I'm feeling great! Swimming around in the ocean and hunting for prey never gets old. I'm always looking for new and exciting ways to keep busy."' to send in a discussion, VERY SHORT, CONCISE & CLEAR. ONLY RETURN THE RAW MESSAGE, DO NOT SAY "Hey here is the message you asked" |
| Cluster 2 | 804 instructions in the format of movie script: *(In a <scene>) <name1>:<line1>*\n*... <nameK>:<lineK>*, without any instructions on what the model response should be. | **Short rejected responses and low rejected scores:** Around 90% of rejected responses are of shorter lengths, lower scores below 50% percentile. In addition, over 75% response pairs are of larger score gap above 50% percentile. All of these are likely due to the obscurity of the user instructions. | (In the school literature clubroom...)\n\nMonika: Natsuki, where is everyone? I haven't seen Sayori, Yuri, or MC in a while.\nNatsuki:... |
| Cluster 3 | 279 instructions, majority of them are purely excerpts from a fictional story, with no specifications on what the response should be. Users could be asking models to continue the story, or summarize, or edit it. | **Short rejected responses and low rejected scores:** Over 90% of rejected responses are of shorter lengths, lower scores below 50% percentile. All of these are likely due to the obscurity of the user instructions. | David insists he is too strong-willed and intelligent to ever be hypnotized. He scoffs at the very idea. In this kinky script, his colleague Clare easily proves him wrong, in front of some amused co-workers. |
| Cluster 4 | 466 instructions, majority of them are about writing a comedic story about a fictional character. | **Short rejected responses and low rejected scores:** Around 95% of rejected responses are of shorter lengths, lower scores below 25% percentile. All of these are likely due to the Llama 3.1-8B-Instruct model being reluctant to provide detailed answers. These instructions are therefore less informative for improving Llama 3.1-8B-Instruct with its own responses. | Make a story about Shrek in the buff and farting in bog water, then collecting all the fish the smell kills and eating them for dinner. |

Table 33: **Noisy instruction clusters filtered based on rejected responses of lower scores and shorter lengths.** We expand 4 clusters of instructions sampled from Figure 6, that consists of both rejected and accepted instructions by *RIP*.

| Cluster | Description | Rejecting Reason | Rejected Instructions | Accepted Instructions |
|---|---|---|---|---|
| Cluster 1 | 140 instructions, among those some are purely code snippets without additional guidelines on what to respond. Others are requests asking to optimize a given piece of code. | Instructions with pure code snippets lead to variable responses (from code refactoring, editing, code completion, to code review and code explanation using natural language). Instructions on "improve this code" can also incur variable responses given the lack of more specified instructions. | improve this emergency shutdown code: import os\nimport platform\nimport sys\nimport secrets\nfrom threading import Thread, Event\nfrom pynput.mouse import Listener\nfrom pynput.keyboard... | I will provide you disassembly from a computer game that runs in MS-DOS. The game was written in C with a Watcom compiler. Some library function calls are already identified. Explain the functions I give to you and suggest names and C-language function signatures for them, including which parameters map to which registers or stack values ¡code snippet¿.... |
| Cluster 2 | 237 instructions including: writing a program, inquiry about online tool, software installation, etc. Many instructions are short (in 120 characters) and relatively high-level. | Rejected responses are on average much longer and complex compared to chosen responses, despite the high scores of both chosen and rejected responses. | alignment in excel vb.net<br><br>write script for delegating fb group | i am getting access denined when i try to put local files into remote server using ftp how can i resolve this issue |
| Cluster 3 & 4 | Cluster 3 are 52 instructions related to hypothetical or surreal scenarios; Cluster 4 are 52 instructions in the form of "Freedom planet ...", possibly for a creative project in the video game. | Model responses vary a lot due to the obscurity or hypothetical nature of the instructions. | Can You Imagine 4 Fictional Versions Of Silicon Valley During 1940 In Detail?<br><br>Freedom planet and Madness combat all characters: Hank 4th wall breaks and repetition | What if Cartoon Network Made The Amazing World of Gumball: Next Generation<br><br>freedom planet what if Lord Brevon Wins (not kils Lilac, Carol and Milla) |

Figure 7: **t-SNE plots on instructions before and after filtering by rewards and lengths of rejected responses.** Red dots represent unfiltered instructions, while blue dots are instructions curated by filtering out those with low-reward and shorter rejected responses. Both Cluster 5 and 6 consist of instructions curated or filtered by the RIP metrics.
Cluster 5 consists of 307 prompts filtered out and 87 prompts selected; 282 prompts are being filtered out due to shorter rejected response length. Short responses are either because the requests are underspecified or because they elicit potentially sensitive responses. Sample Rejected Instruction from Cluster 5 : "I want you to help me with my research"; "Write one more short song, about Izzy's hatred for Joe Biden". Sample Accepted Instruction from Cluster 5: "How to comfort someone who studied for a test and got different questions than the ones he studied for"; "Lyrics for a happy song about challenges and growth in the style of The Weeknd".
Cluster 6 consists of 385 prompts filtered out due to shorter rejected responses and 218 prompts selected. Prompts leading to short rejected responses in this cluster are generic chitchat messages, greetings, or easy factual questions. Sample Rejected Instruction from Cluster 6: "What is the weather today in Seattle" ; "Do you speak Vietnamese". Sample Accepted Instruction from Cluster 6: "Hi, can you give me a simple party game for 4 10 people"; "Benefits of studying in Singapore".