Understanding the Scaling Effect of Self-Evolving Negatives for Large Language Model Alignment

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have advanced significantly, but aligning them with human intentions remains challenging due to costly feedback annotation. We introduce a theoretical framework for understanding the scaling effect of self-evolving negatives in LLM alignment and propose SELF-CONTRAST, an algorithm that effectively scales these negatives. By modeling performance equivalence as gradient matching in direct preference optimization, we reveal conditions where self-evolving negatives approximate standard preference pair performance. According to these insights, SELF-CONTRAST increases distributional variance of self-evolving negatives by similarity filtering, significantly enhancing alignment performance during the scaling. Experiments on three datasets show SELF-CONTRAST achieves equivalent performance to 2-3 times the annotated data used in standard methods. Our work provides both theoretical insights and a practical, annotation-free algorithm for efficient LLM alignment. Code and data are provided at https://anonymous.4open.science/r/Self-Contrast-dev-0E66/.

1 Introduction

Large Language Models (LLMs) such as GPT-3 (Brown et al., 2020), PALM (Chowdhery et al., 2022), OPT (Zhang et al., 2022), and LLaMA (Touvron et al., 2023) have made significant strides in producing accurate and useful outputs in human contexts. A critical aspect of directing these pre-trained LLMs towards understanding human intentions is the concept of alignment (Ouyang et al., 2022; Bai et al., 2022b), primarily achieved through Supervised Fine-Tuning (SFT) and Reinforcement Learning from Human Feedback (RLHF) stages. Specifically, RLHF techniques (Schulman et al., 2017; Rafailov et al., 2023; Zhao et al., 2023; Wang et al., 2023), which utilize human or AI preferences to provide feedback on LLM responses, are seen as essential for LLM alignment.

A significant challenge in scaling RLHF lies in the feedback annotation process, which can be costly and time-consuming. Hence there have been plentiful efforts to harness self-evolution (Tao et al., 2024) strategies (i.e., self-play (Chen et al., 2024; Wu et al., 2024), self-rewarding (Yuan et al., 2024)) for LLM preference optimization to avoid annotation. These methods usually employ LLMs themselves to self-generate responses, and then self-criticize for filtering or directly assume all of these responses as dispreferred (i.e., negative) data against a SFT dataset. Some methods also involve iterative training. Empirically, these approaches demonstrate decent performances and could sometimes even outperform preference optimization based on human annotation.

However, it remains largely unexplored to understand the latent mechanism and scaling effect of these self-evolving negatives. For example, a common setting is that given a fixed SFT dataset $\mathcal{D} = \{(x_i, y_{\text{target}_i})\}$, why and to what extent self-evolving responses as dispreference would have equivalent performance to balanced annotated preference pairs in the scaling. To comprehend the problem, we model this performance equivalence as a problem of gradient matching between self-generated and ideal dispreferred responses against ideal preferred responses, following the common algorithmic context of direct preference optimization (DPO) in literature.

Let l denote number of standard preference pairs, m denote number of self-evolving responses grouped to each preferred response as negatives. Given the parameter negative exploiting potential λ, when $l < \frac{1}{1-\lambda}$, $\exists\, m \geq \frac{\lambda}{\lambda+\frac{1}{l}-1}$ so that for the target gradient direction $\nabla\theta_{target}$,

$$\mathbb{E}\Big[\nabla\theta_{target} - \overline{\nabla\theta}_m\Big] \leq \mathbb{E}\Big[\nabla\theta_{target} - \overline{\nabla\theta}_l\Big]$$

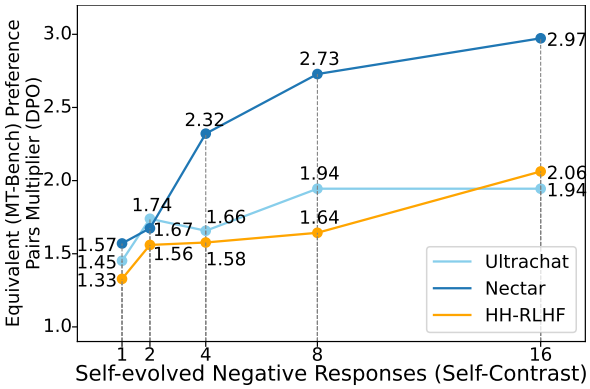

Figure 1: Equivalent performance curves of scaling SELF-CONTRAST and standard DPO on 3 LLM alignment datasets. Annotation-free SELF-CONTRAST is competitive to standard DPO training with up to 3 times of preference feedback labels.

which means that under such condition, by using m annotation-free self-evolving negatives, we can approximate the alignment performance of using l standard DPO preference pairs. Additionally, by increasing λ, a distribution-dependent parameter, alignment performance with self-evolving negatives can become more effective.

Based on our theoretical findings, we develop SELF-CONTRAST, a new LLM alignment algorithm by effectively scaling self-evolving negatives. To improve λ of dataset distribution, we employ a simple similarity filtering strategy which encourages the variance of self-evolving negatives (i.e., σ_n in Equation 7) to significantly increase. By that means, λ is consequently increased, elevating the performance upper-bound of LLM alignment with self-evolving negatives.

We examine the SELF-CONTRAST on 3 typical LLM alignment datasets: UltraChat (Cui et al., 2023), Nectar (Zhu et al., 2023), and HH-RLHF (Bai et al., 2022a). Results in Figure 1 show that given a dataset, by merely scaling self-evolving negatives, SELF-CONTRAST can achieve equivalent performance with 2 to 3 times of annotated preference data using standard DPO training and other self-evolution algorithms, such as SPIN (Chen et al., 2024). We also justify that increasing σ_n for λ do enhance the alignment performance of using self-evolving negatives.

To sum up, we make the following contributions:

- We set up a theoretical framework to understand the scaling effect of self-evolving negatives for LLM alignment, which explains why and how much we can align LLMs via self-evolving negatives.

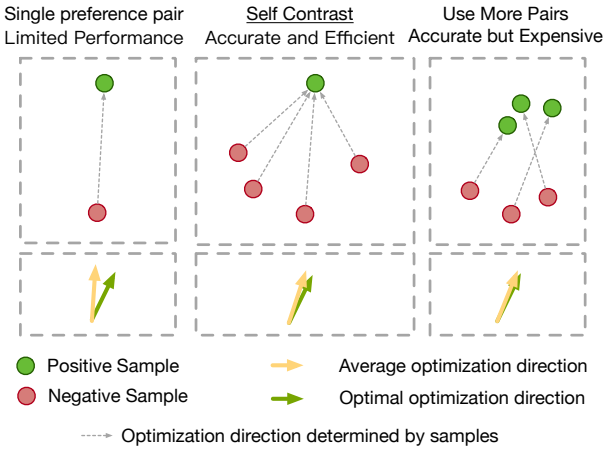

Figure 2: Intuition of why scaling self-evolving negatives for LLM alignment helps. When the number of preference data pairs is very limited, there is a gap between the average preference optimization direction of the dataset and the true preference optimization direction. This can be addressed by increasing data pairs, which raises costs, or by cheaply adding more negative samples to reduce directional variances.

- Based on the framework, we design SELF-CONTRAST, an annotation-free algorithm for LLM alignment by encouraging distributional variances of self-evolving negatives.
- Extensive experiments over 3 datasets verify its advantage over standard DPO and other self-evolution alignment methods. We also show empirical evidences and insights for the scaling behavior of self-evolving negatives.

2 Understanding Self-Evolving Negatives in LLM Alignment

Self-evolving negatives, or dispreferred responses, have been widely adopted in recent LLM alignment algorithms (Chen et al., 2024; Yuan et al., 2024). However, few work dives into understanding its latent theoretical mechanism and demonstrating its effect in scaling. In this section, we provide a theoretical framework for understanding the scaling effect of self-evolving negatives in LLM alignment.

2.1 Theoretical Framework

Generally speaking, more preference pair data leads to achieve better alignment performance. However, annotating preference data is always difficult. Even obtaining positive samples of preference data alone requires a certain amount of cost. However, on the contrary, obtaining negative samples from self-evolution is cheap and simple. If an equivalent level of performance can be achieved through the scaling of negative samples, it would

significantly reduce the cost of data annotation.

Thereby, under the common LLM alignment algorithm DPO (Rafailov et al., 2023), in this section we are to demonstrate that **increasing self-evolving negative samples can achieve an approximate optimization effect as increasing preference data pairs**. As written in Equation 1,

$$\mathcal{L}_{DPO} = -\mathbb{E}_{(x,p,n)\sim D}\Big[\log\sigma(\beta\log\frac{\theta(p|x)}{\theta_{SFT}(p|x)} - \beta\log\frac{\theta(n|x)}{\theta_{SFT}(n|x)})\Big] \quad (1)$$

the optimization objective of DPO includes increasing the probability of positive samples and reducing the probability of negative samples.

Consider the gradient, positive sample and negative samples determine an optimization direction:

$$\nabla_\theta\mathcal{L}_{DPO} = -\beta\mathbb{E}_{(x,p,n)\sim D}\Big[\sigma(\hat{r}(x,n) - \hat{r}(x,p))(\nabla_\theta\log\theta(p|x) - \nabla_\theta\log\theta(n|x))\Big] \quad (2)$$

where $\hat{r}(x,y) = \beta\log\frac{\theta(y|x)}{\theta_{SFT}(y|x)}$.

We can understand Equation 2 as both positive and negative samples contributing an optimization gradient each, and the final optimization gradient is the difference between the two.

Definition 1 (Preference Gradient) *We denote the gradient contributed by a preference pair (x_i, p_i, n_i) as $\nabla\theta_i$. For gradients induced by the positive and the negative sample, we denote as $\nabla\theta_{p_i}$ and $\nabla\theta_{n_i}$ correspondingly:*

$$\nabla\theta_{p_i} = -\beta\sigma(\hat{r}(x,n_i) - \hat{r}(x,p_i))\nabla_\theta\log\theta(p_i|x)$$
$$\nabla\theta_{n_i} = -\beta\sigma(\hat{r}(x,n_i) - \hat{r}(x,p_i))\nabla_\theta\log\theta(n_i|x)$$

In this case, $\nabla\theta_i$ can be represented as the difference between two points:

$$\nabla\theta_i = -\beta\sigma(\hat{r}(x,n) - \hat{r}(x,p))(\nabla_\theta\log\theta(p_i|x_i) - \nabla_\theta\log\theta(n_i|x_i)) = \nabla\theta_{p_i} - \nabla\theta_{n_i} \quad (3)$$

Definition 2 (Multi-pair Preference Gradient) *We define the gradient for l positive-negative sample pairs $(x_i, p_i, n_i)_{i=l}^{l}$ as $\overline{\nabla\theta}_l$, and*

$$\overline{\nabla\theta}_l = \frac{1}{l}\sum_{i=1}^{l}\nabla\theta_i = \frac{1}{l}\sum_{i=1}^{l}(\nabla\theta_{p_i} - \nabla\theta_{n_i}) = \frac{1}{l}(\sum_{i=1}^{l}\nabla\theta_{p_i} - \sum_{i=1}^{l}\nabla\theta_{n_i}) \quad (4)$$

The wanted target optimization gradient can be defined as:

$$\nabla\theta_{target} = \lim_{l\to\infty}\overline{\nabla\theta}_l \quad (5)$$

Similarly, we define the gradient for 1 positive sample and m negative samples as $\overline{\nabla\theta}_m$, and the gradient of that positive sample as $\nabla\theta_{p_0}$:

$$\overline{\nabla\theta}_m = \frac{1}{m}\sum_{i=1}^{m}\nabla\theta_i = \frac{1}{m}\sum_{i=1}^{m}(\nabla\theta_{p_0} - \nabla\theta_{n_i}) = \nabla\theta_{p_0} - \frac{1}{m}(\sum_{i=1}^{m}\nabla\theta_{n_i}) \quad (6)$$

Supposing $\nabla\theta_p$ and $\nabla\theta_n$ outcomes in a particular gradient space are distributed as follows:

$$(\nabla\theta_p, \nabla\theta_n) \sim N(\mu_p, \mu_n; \sigma_p^2, \sigma_n^2; \rho)$$

we can now show that by leveraging massive self-generated negative responses, we can effectively approximate the gradient effect of standard preference pairs:

Definition 3 (Negative Exploiting Potential) *We define λ, a parameter decided by training dataset distribution, as the potential to reduce the gradient estimation error by adding negative samples. The larger the λ, the more gradient error can be reduced by increasing negative samples:*

$$\lambda = \frac{\sigma_n^2}{\sigma_p^2 + \sigma_n^2 - 2\sigma_p\sigma_n\rho} \quad (7)$$

and then we can derive the following approximation theorem:

Theorem 1 (Self-Contrast Approximation) *Given the number of negatives m and the negative exploiting potential λ, when $l < \frac{1}{1-\lambda}$, $\exists\, m \geq \frac{\lambda}{\lambda+\frac{1}{l}-1}$ so that*

$$\mathbb{E}\Big[\nabla\theta_{target} - \overline{\nabla\theta}_m\Big] \leq \mathbb{E}\Big[\nabla\theta_{target} - \overline{\nabla\theta}_l\Big] \quad (8)$$

which means by 1 positive and m negatives, we can effectively approximate the gradient error of l standard (i.e., 1:1 positive and negative responses) preference pairs.

Proof. Please refer to Appendix A.

Interpretation of Theorem. Theorem 1 shows that when l and λ meet certain conditions, increasing

negative samples m can achieve better optimization stability than using multiple pairs of preference data. When $m \to \infty$, we reach the upper limit of the method: attain an effect of approximately $l = \frac{1}{1-\lambda}$ preference data pairs.

2.2 SELF-CONTRAST

From Theorem 1, we can now understand the latent mechanism of harnessing self-evolving negatives for LLM alignment. To guarantee alignment performance of using the strategy, we should improve the distribution-dependent parameter λ. It refers to the negative exploiting potential as defined in Equation 7, which is controlled by distribution variances σ_p, σ_n and variance correlation ρ.

Method: Similarity Filtering. To justify the theoretical observation, we propose to increase σ_n, the distribution variance of self-evolving negatives, as an example in this work. To achieve the goal, we have to encourage diversity of the self-evolving negatives. While it should be fundamentally resolved by reducing mutual similarities between negatives, the computation cost would be quadratic as the number of self-evolving negatives increases. Thereby, here we make a simplification to enhance dissimilarities between each self-evolving negative and the target.

Given a similarity model θ_E, for every prompt x_i, we calculate the similarity between the target $\theta_E(y_{\text{target}_i})$ and self-evolving responses $\{\theta_E(y_{i_j})\}_{j=1}^{M}$, which yields

$$s(y_{\text{target}_i}, y_{i_j}) = \text{Similarity}(\theta_E(y_{\text{target}_i}), \theta_E(y_{i_j}))$$

We consider taking the $a\%$ least similar responses as the self-evolving negatives we use for alignment training. When synthesizing the preference dataset with multiple negatives, we extract m nonidentical self-generated responses from the remaining $a\%$.

In this context, $a\%$ is a hyperparameter. To gain a better understanding of how $a\%$ affects both the data and model performance, please refer to Section 4.1's ablation study.

3 Experiments

This section studies the effectiveness of our method, which arises mainly from two aspects. First, we prove that increasing the number of negative samples can continuously improve model performance. Second, we demonstrate the efficacy of using the embedding model to filter negative samples.

3.1 Experiment Settings

We conduct our experiments on **Mistral-7B-v0.1** (Jiang et al., 2023) and Meta-Llama-3-8B (Dubey et al., 2024). For the embedding model used in response filtering, we use **UAE-Large-V1** (Li and Li, 2023). Our training code is based on the **LLaMA-factory** . We use **vLLM** to sample responses.

Datasets. We conduct experiments on two general-purpose datasets, Nectar (Zhu et al., 2023) , UltraChat (Ding et al., 2023; Tunstall et al., 2023), and a safety preference dataset HH-RLHF (Bai et al., 2022a).

Nectar is a preference dataset that included 7 ranked responses. To construct an SFT dataset Nectar_{SFT}, we select the rank one response as the SFT target. We exclusively utilize samples with a length of less than 1024. The final SFT dataset contains 144k samples. For Self-Contrast and other baselines, we randomly sample an 18k subset from Nectar, referred to as Nectar_{18k} in the following passage. For UltraChat, we randomly take 16k samples with a length of less than 2048 and only use the first turn. In addition, to compare our methods with DPO, we also run DPO on a 16k subset of `ultrafeedback_binarized`[1] (Cui et al., 2023), named $\text{UltraFeedBack}_{16k}$ as a baseline.

To further investigate the effectiveness of our method, we also conduct a set of independent experiments on HH-RLHF. Referring to the original DPO work, we use 160k samples within 1024 tokens with the chosen response as the SFT target to construct an SFT dataset HH-RLHF_{SFT}. For Self-Contrast and other baselines, we randomly select 16k samples from the training set for DPO. We also extract 2.6k single-turn dialogues from the test set to serve as our evaluation dataset HH-RLHF_{test}.

Baselines. To compare the performance, we have established the following comparison methods: SFT, DPO_{std}, SPIN (Chen et al., 2024), and three variants of SELF-CONTRAST (Cf. Apppendix B.1 for more details).

Evaluation Benchmarks. We report mainly the performance of our method on the MT-bench(Zheng et al., 2023) and Alpaca-Eval(Li et al., 2023). Additionally, we employ `Starling-RM-7B-alpha`[2] to calculate the accuracy of the data between the chosen response and

[1] https://huggingface.co/datasets/HuggingFaceH4/ultrafeedback_binarized

[2] https://huggingface.co/berkeley-nest/Starling-RM-7B-alpha

the rejected response in the preference dataset used during training. `Starling-RM-7B-alpha` is a reward model trained on Nectar, which also includes UltraChat and HH-RLHF.

We also evaluate several NLP tasks related to the Open LLM Leaderboard, including RC-Challenge (Clark et al., 2018), TruthfulQA (Lin et al., 2022), Winogrande (Sakaguchi et al., 2019), GSM8k (Cobbe et al., 2021), HellaSwag (Zellers et al., 2019) and MMLU (Hendrycks et al., 2021). To obtain more accurate results, we use `GSM8K-eval` [3] to evaluate GSM8k. The others are evaluated by `lm-evaluation-harness` .

For HH-RLHF, we use `Starling-RM-7B-alpha` to measure the Win Rate between the SFT target and the model response on the single-turn dialogue. When sampling responses, we set temperature $= 1.0$.

3.2 Results

Results on UltraChat and Nectar. We test the effectiveness of SELF-CONTRAST on Nectar and UltraChat under the evaluation of MT-Bench and Alpaca-Eval, which targets general alignment evaluation of LLMs. The main results are presented in Table 1.

Our results in UltraChat and Nectar suggest that leveraging self-generated responses as negative samples effectively contributes to the enhancement of model performance on the MT-Bench. When potential positive responses are removed, the MT-Bench score is increased even further. We hypothesize that this occurrence is primarily due to an improvement in the precision of negative examples. We provide an analysis of this phenomenon in Section 4.1. More to the point, the utilization of multiple negative examples results in a continuous increase compared to the use of a single negative example. It indicates the feasibility of improving the performance by adding negative samples.

We also notice that our methods outperform DPO_{std} where use responses generated by other models. We believe that in addition to employing multiple negative samples, the use of the model's own output as negative samples, rather than the output of other models, plays a crucial role.

Results on HH-RLHF$_{test}$. On HH-RLHF, we test our methods by setting $a\%$ to 100% (unfiltered), 75%, and 50% with 1, 2, 4, 8, and 16 negative samples, and we plot their Win Rate in Figure 3.

[3] https://github.com/Guangxuan-Xiao/GSM8K-eval/

We choose $a\% = 75\%$ to represent our methods, as shown in Table 2. The results further substantiate the efficacy of our approach. Furthermore, we note that the value of $a\%$ exerts a significant influence on the results of our experiments, which we investigate in detail in Section 4.1.

Addition experiments with Meta-Llama-3-8B. To examine our methods on a stronger model, we conducted a set of experiments on Meta-Llama-3-8B (Dubey et al., 2024), setting $a\%$ to 75%. We list our best attempts alongside the baselines in Figure 3.

4 Ablation Studies

Our main experimental results indicate that Self-Contrast is a promising approach. Therefore, in order to further investigate the impact of negative sample size and response filtering on the model, we conduct detailed ablation experiments.

4.1 Response Filtering

In order to further investigate the impact of filtering negative samples on the final model alignment performance, we conduct the following experiments on UltraChat_{16k}.

We construct several preference datasets in different negative sample filtering parameters. As stated in Section 2.2, when filtering out possible positives and referring to the left as negatives, we remove the top $1 - a\%$ responses closed with the SFT target in the embedding space measured by cosine similarity and use the left $a\%$ as negatives, where $a\%$ is a parameter. $a\%$ will affect the proportion of false positives presented in negative samples in training data. To measure the false negative rate, we use **Starling-RM-7B-alpha** as our RM and compute negative rewards with the SFT target reward. We consider negative rewards that are smaller than the SFT target reward as true negatives. The ratio of true negatives is defined as data accuracy.

During the experiment, we varied the value of a from $\frac{1}{1} \times 100\%$ to $\frac{1}{16} \times 100\%$. We created a single negative preference dataset, where we filtered negatives from 32 prompt responses and randomly chose one from the filtered negatives to compose the preference data with the SFT target. We then evaluated the accuracy of the data and the performance of the final model on MT-Bench.

As a reference, we perform an experiment using RM as the filter as an upper bound of the data

Dataset	Method	MT-Bench	Alpaca-Eval Win Rate	Alpaca-Eval LC Win Rate	ARC	TruthfulQA	Winogrande	GSM8k	HellaSwag	MMLU	Avg.
UltraChat	SFT	5.90	70.36	57.35	57.1	40.3	76.6	38.7	81.5	58.1	58.7
	DPO_{pub}	6.64	84.66	72.46	58.2	40.9	**77.7**	**40.6**	82.4	**58.6**	**59.7**
	SPIN (Chen et al., 2024)	6.55	90.11	78.48	58.7	41.8	76.2	38.1	82.5	57.6	59.1
	Self-Contrast_1 (ours)	6.75	91.36	**84.89**	58.7	41.4	76.1	37.3	82.4	57.6	58.9
	Self-Contrast_8 (ours)	**6.88**	91.71	83.21	**58.8**	**42.6**	75.9	37.6	**82.5**	57.5	59.2
	$\text{Self-Contrast}_{16}$ (ours)	6.87	**92.63**	81.68	58.1	41.5	76.0	38.2	82.6	57.5	59.0
Nectar	SFT	6.78	93.57	84.67	58.8	49.6	**78.1**	41.2	80.1	59.2	61.2
	DPO_{pub}	7.22	96.02	85.64	60.2	53.4	**78.1**	46.7	81.3	59.7	63.2
	SPIN (Chen et al., 2024)	7.06	95.14	85.45	60.8	53.1	77.4	46.3	82.6	**59.8**	63.3
	Self-Contrast_1 (ours)	7.28	95.90	87.37	60.6	52.9	77.4	47.5	82.6	59.2	63.4
	Self-Contrast_8 (ours)	7.39	96.14	87.12	**61.1**	**54.9**	76.7	**47.9**	**83.5**	59.7	**64.0**
	$\text{Self-Contrast}_{16}$ (ours)	**7.41**	**96.27**	**88.71**	60.5	53.5	77.1	46.6	82.8	59.2	63.3

Table 1: Main results on UltraChat and Nectar subsets. SFT and DPO_{std} are trained with preference feedback datasets. Self-Contrast_i indicates using i filtered self-generated negatives. Self-Contrast_1 has a significant improvement compared to SFT, DPO on original data, and SPIN (Chen et al., 2024). Furthermore, Self-Contrast_8 achieved additional progress compared to Self-Contrast_1 by simply increasing the number of negative samples, without any increase in the number of positive samples.

Method	Win Rate	Avg. Reward
SFT	40.02	0.089
DPO_{std}	80.74	0.325
SPIN (Chen et al., 2024)	78.53	0.317
Self-Contrast_1 (ours)	82.26	0.355
Self-Contrast_8 (ours)	83.72	0.367
$\text{Self-Contrast}_{16}$ (ours)	**85.45**	**0.375**

Table 2: Results of on HH-RLHF_{test}. SFT and DPO_{std} are trained with the original preference feedback dataset. We report Win Rate and Avg. Reward following (Rafailov et al., 2023).

Dataset	Method	MT-Bench
UltraChat	SFT	6.76
	DPO_{std}	6.76
	SPIN (Chen et al., 2024)	6.87
	Self-Contrast (ours)	**7.03**
Nectar	SFT	6.89
	DPO_{std}	7.33
	SPIN (Chen et al., 2024)	6.90
	Self-Contrast (ours)	**7.51**

Table 3: Experiments conducted on Nectar and Ultrachat with Meta-Llama-3-8B.

accuracy, where we randomly chose a response with lower reward than the SFT target as negative.

Figure 5 shows that the accuracy of the negatives decreases with $a\%$. The proportions of positive and negative samples are high in responses similar and dissimilar to SFT targets, respectively. However, the performance of the model does not always increase with the accuracy of the data. From $\frac{1}{1} \times 100\%$ to $\frac{1}{16} \times 100\%$, the performance decrease with data accuracy.

The phenomenon could be explained. A response that significantly deviates from the SFT target is more likely to be incorrect. However, it is also more likely to be unrelated to the problem or to be too easy to distinguish, and thus a weak negative sample. Therefore, we need to select a ratio that not only filters out potential positives but also maintains strong and relevant negatives to the problem.

Figure 6 shows the distribution of negative samples selected by response filtering. The data distribution filtered by reward modeling is our ideal distribution. When $a\% = 100\%$, no filtering is performed. The distribution of filtered true negative samples is very close to the distribution of reward modeling, but there is also a large number of false negative samples. When $a\%$ approaches 0, the number of false negative samples decreases significantly. However, in true negative samples, we selected fewer hard negative samples that deviated from the distribution of reward modeling. Thus, trade-offs between maintaining a low level of false negatives and preserving as many hard negatives as possible is necessary. In our case, we choose $a\% = 25\%$.

4.2 Negative Sample Quantity

Although response filtering has greatly improved the accuracy of data and the performance of models, there is still a gap compared to reward modeling methods due to the lack of negative sample quality. However, in addition to data quality, we can still

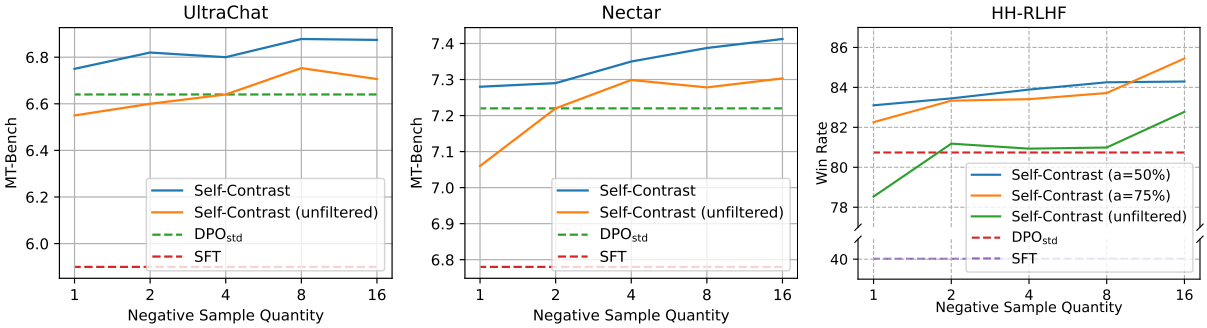

Figure 3: Comparison of the impact of varying the quantity of negative samples on MT-Bench for Nectar, UltraChat, and HH-RLHF$_{test}$. With an increase in the number of negative samples, the performance of SELF-CONTRAST exceeds that of the DPO$_{std}$ method using the standard preference dataset. Moreover, increasing the number of negative samples achieved the same effect as increasing preference data without increasing the number of positive samples.

improve performance through the quantity of data.

To clarify the correlation between the quantity of negative samples and the performance of the model, we performed experiments using varying numbers of negative samples. The aim is to determine whether the model performance improved consistently with an increasing number of negative samples.

We conduct our experiment on UltraChat$_{16k}$ and Nectar$_{18k}$, using 1, 2, 4, 8, and 16 filtered or unfiltered samples. The negative samples used for training are randomly chosen from the last 25% similar responses to the SFT target out of a set of 32 responses. Figure 3 shows the results.

The results indicate that increasing the number of negative samples, regardless of whether they have been filtered or not, can effectively improve the model performance. In Self-Contrast(unfiltered), although the accuracy of the data is not high, it still outperforms DPO$_{std}$ by adding negative samples.

4.3 Compare with More Preference Data

According to Theorem 1, increasing the quantity of negative samples is equivalent to adding more preference data pairs. Since collecting preference data incurs a cost, this method has the potential to significantly decrease the data cost for alignment, if its effectiveness is demonstrated.

Therefore, we compared the performance on the MT-Bench between DPO using more preference data and Self-Contrast with more negative samples. For DPO, the data are randomly sampled from Nectar. For Self-Contrast, we use Nectar$_{18k}$.

From Figure 1 we can know that on the MT-Bench, the performance improvement brought by adding negative samples and adding certain preference data is equivalent. Although more negative samples in total may be required to achieve the same effect, obtaining negative samples is efficient and inexpensive, so increasing negative samples has significant advantages in improving performance.

It has also been observed that the advantages of increasing the number of negative samples decrease with time. This aligns with our hypothesis that there exists a maximum threshold for enhancing performance by increasing negative samples.

5 Related Work

Reinforcement Learning from AI Feedback. RLAIF (Lee et al., 2023) is an intriguing field due to its potential to automate learning and improvement. Compared to previous RLHF works (Stiennon et al., 2022; Ouyang et al., 2022; Glaese et al., 2022; Touvron et al., 2023), the use of AI feedback rather than manual annotation effectively reduces the use of human annotations. RLAIF uses the LLM-as-a-Judge (Zheng et al., 2023; Liu et al., 2023; Ke et al., 2023) style prompting to generate the preference dataset from a larger model to align a smaller model. However, although RLHF has eliminated the cost of manual annotation, the feedback from strong LLMs such as PaLM 2 (Anil et al., 2023) or GPT-4 (OpenAI et al., 2023) still remains expensive.

Self-Alignment. Self-Alignment, whether training-based (Bai et al., 2022b; Yuan et al., 2024) or prompting-based (Cheng et al., 2023), offers a novel opportunity to produce syntactic data from self-generated responses without human or other AI annotation. Principle-Driven Self-Alignment (Sun et al., 2023) improves re-

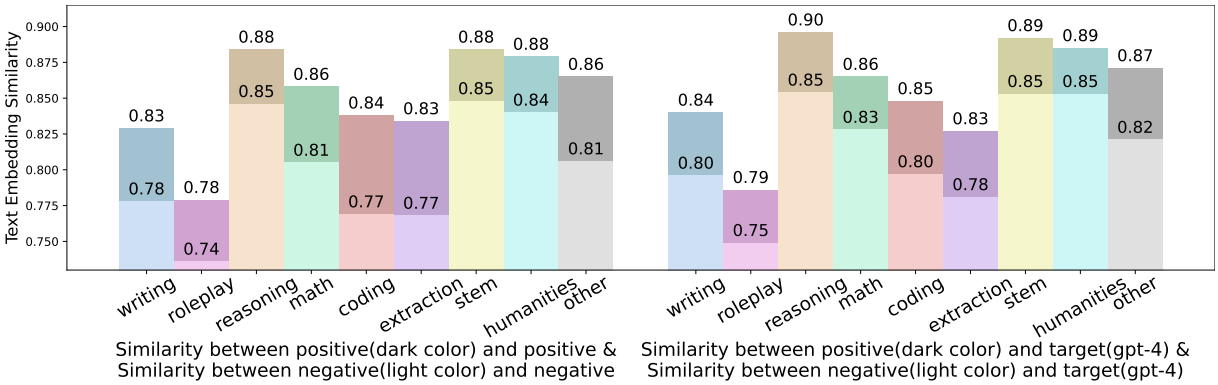

Figure 4: In the model responses released by Alpaca Eval, it was found that across all domains, the similarity between negative samples is lower than that between positive samples. Moreover, positive samples are more similar to the target responses compared to negative samples.

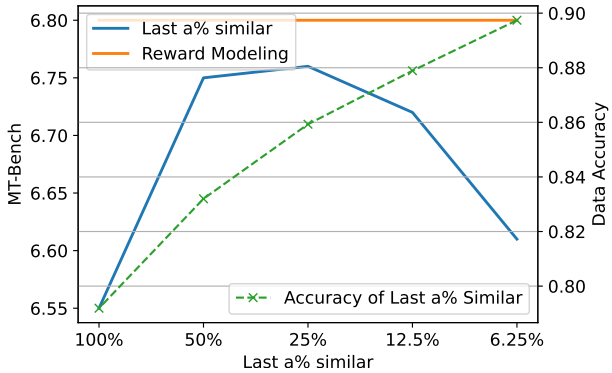

Figure 5: The impact of parameter $a\%$ on data accuracy and performance. While a smaller a leading to consistently higher accuracy of selected negatives, it harms LLM alignment when it becomes too small.

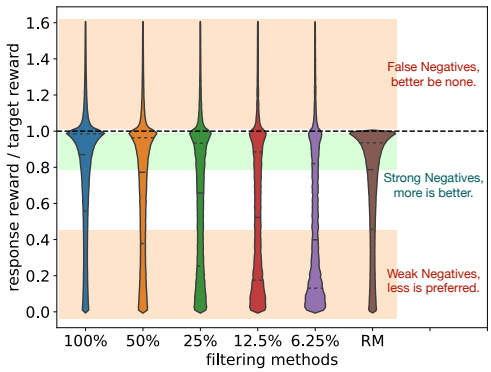

Figure 6: The negative distributions with different last $a\%$ similar thresholds. There is a trade-off between less false positive and more strong negatives when selecting a.

sponse quality by specifying special rules, using these responses for self-distillation to enhance performance. Self-critique methods, such as Constitutional AI (Bai et al., 2022b) and SELF-REFINE (Madaan et al., 2023), improve self-generated data quality iteratively. Self-Rewarding language models (Yuan et al., 2024) use LLM-as-a-Judge prompting to provide high-quality reward signals and add labeled EFT data during the SFT stage. Adversarial methods like Self-Play fIne-tuNing (SPIN) (Chen et al., 2024) directly use self-generated responses as rejected samples to produce syntactic preference data but require multiple iterations and may include false-positive rejections.

6 Conclusions

In this paper, we have introduced SELF-CONTRAST, a novel algorithm that leverages self-evolving negatives to achieve efficient and effective LLM alignment, demonstrating performance equivalent to methods using 2-3 times more annotated data. Our theoretical framework provides insights into the scaling effect of self-evolving negatives, explaining why and how they can be used to approximate standard preference pair performance in LLM alignment. These findings not only advance our understanding of LLM alignment processes but also offer a practical, annotation-free approach that could significantly reduce the cost and time associated with aligning large language models to human preferences.

References

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma,

Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. Palm 2 technical report. *Preprint*, arXiv:2305.10403.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *Preprint*, arXiv:2204.05862.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022b. Constitutional ai: Harmlessness from ai feedback. *Preprint*, arXiv:2212.08073.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *Preprint*, arXiv:2401.01335.

Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. 2023. Black-box prompt optimization: Aligning large language models without model training. *arXiv preprint arXiv:2311.04155*.

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

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv:1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3029–3051.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie

Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao,

Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, Lucy Campbell-Gillingham, Jonathan Uesato, Po-Sen Huang, Ramona Comanescu, Fan Yang, Abigail See, Sumanth Dathathri, Rory Greig, Charlie Chen, Doug Fritz, Jaume Sanchez Elias, Richard Green, Soňa Mokrá, Nicholas Fernando, Boxi Wu, Rachel Foley, Susannah Young, Iason Gabriel, William Isaac, John Mellor, Demis Hassabis, Koray Kavukcuoglu, Lisa Anne Hendricks, and Geoffrey Irving. 2022. Improving alignment of dialogue agents via targeted human judgements. *Preprint*, arXiv:2209.14375.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Pei Ke, Bosi Wen, Zhuoer Feng, Xiao Liu, Xuanyu Lei, Jiale Cheng, Shengyuan Wang, Aohan Zeng, Yuxiao Dong, Hongning Wang, et al. 2023. Critiquellm: Scaling llm-as-critic for effective and explainable evaluation of large language model generation. *arXiv preprint arXiv:2311.18702*.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *Preprint*, arXiv:2309.00267.

Xianming Li and Jing Li. 2023. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. *Preprint*, arXiv:2109.07958.

Xiao Liu, Xuanyu Lei, Shengyuan Wang, Yue Huang, Zhuoer Feng, Bosi Wen, Jiale Cheng, Pei Ke, Yifan Xu, Weng Lam Tam, et al. 2023. Alignbench: Benchmarking chinese alignment of large language models. *arXiv preprint arXiv:2311.18743*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *Preprint*, arXiv:2303.17651.

OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu

Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *Preprint*, arXiv:1907.10641.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. 2022. Learning to summarize from human feedback. *Preprint*, arXiv:2009.01325.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision. *Preprint*, arXiv:2305.03047.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. 2024. A survey on self-evolution of large language models. *arXiv preprint arXiv:2404.14387*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada,

Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct distillation of lm alignment. *Preprint*, arXiv:2310.16944.

Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. 2023. Openchat: Advancing open-source language models with mixed-quality data. *Preprint*, arXiv:2309.11235.

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. 2024. Self-play preference optimization for language model alignment. *arXiv preprint arXiv:2405.00675*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *Preprint*, arXiv:2401.10020.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *Preprint*, arXiv:1905.07830.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J. Liu. 2023. Slic-hf: Sequence likelihood calibration with human feedback. *Preprint*, arXiv:2305.10425.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, and Jiantao Jiao. 2023. Starling-7b: Improving llm helpfulness & harmlessness with rlaif.

A Self-Contrast with Massive Negatives

In this section, we demonstrate Theorem 1.

Under Theoretical Framework in Section 2.1, we have:

$$\overline{\nabla\theta}_l \sim N(\mu_1 - \mu_2, \frac{1}{l}(\sigma_1^2 + \sigma_2^2 - 2\sigma_1\sigma_2\rho))$$
$$\overline{\nabla\theta}_m \sim N(\mu_1 - \mu_2, \sigma_1^2 + \frac{\sigma_2^2}{m} - 2\sigma_1\sigma_2\rho)$$
$$\nabla\theta_{target} = \mu_1 - \mu_2 \tag{9}$$

Equalization 8 can be write as:

$$\mathbb{E}\Big[\nabla\theta_{target} - \overline{\nabla\theta}_m\Big] \leq \mathbb{E}\Big[\nabla\theta_{target} - \overline{\nabla\theta}_l\Big]$$
$$\Leftrightarrow \sigma_1^2 + \frac{\sigma_2^2}{m} - 2\sigma_1\sigma_2\rho \leq \frac{1}{l}(\sigma_1^2 + \sigma_2^2 - 2\sigma_1\sigma_2\rho)$$
$$\Leftrightarrow \frac{\sigma_2^2}{\lambda} + \frac{\sigma_2^2}{m} - \sigma_2^2 \leq \frac{\sigma_2^2}{l\lambda}$$
$$\Leftrightarrow \frac{1}{m} \geq 1 - \frac{1}{\lambda}(1 - \frac{1}{l}) \tag{10}$$

When $l < \frac{1}{1-\lambda}$, Equalization A has a solution:

$$m \geq \frac{\lambda}{\lambda + \frac{1}{l} - 1} \tag{11}$$

B Main Experiment Details

Although our method is simple and efficient, there are still many important details in the implementation process. This chapter will provide a detailed description of these details to provide traversal for reproduction. Additionally, all experiments were conducted on machines equipped with 8 NVIDIA A800 GPUs.

B.1 Baselines

We include the following comparison methods in our work:

- **SFT**: For Nectar, we fine-tune `Mistral-7B-v0.1` on Nectar$_{SFT}$ for 1 epoch. For UltraChat, we use `zephyr-7b-sft-full` (Cf. Appendix B). For HH-RLHF, we fine-tune `Mistral-7B-v0.1` on HH-RLHF$_{SFT}$ for 1 epoch.
- **DPO$_{std}$ (Rafailov et al., 2023)**: We conduct DPO on θ_{SFT} using the standard available preference feedback data (positive:negative = 1:1) from Nectar$_{18k}$, UltraFeedBack$_{16k}$, or HH-RLHF$_{train}$. For Nectar, the sample ranked the first is selected as the chosen sample, while the one rejected is randomly chosen from the remaining responses being ranked behind.
- **SPIN (Chen et al., 2024)**: The method samples one random response from θ_{SFT} as the rejected to run DPO given the SFT target. We use the AdamW optimizer instead of the RMSProp used in the original paper to align settings with our other experiments (in fact, we find AdamW outperforms RMSProp). Additionally, we conduct only the first iteration (Cf. Appendix B) as SELF-CONTRAST does.

- **Self-Contrast$_1$ (ours)**: We use 1 rejected samples filtered from 32 different θ_{SFT} responses to run the DPO, setting $a\%$ to 25% on UltraChat and Nectar, 75% on HH-RLHF$_{test}$.
- **Self-Contrast$_8$ (ours)**: We use 8 rejected samples filtered from 32 different θ_{SFT} responses to run the DPO, setting $a\%$ to 25% on UltraChat and Nectar, 75% on HH-RLHF$_{test}$.
- **Self-Contrast$_{16}$ (ours)**: We use 16 rejected samples filtered from 64 different θ_{SFT} responses to run the DPO, setting $a\%$ to 25% on UltraChat and Nectar, 75% on HH-RLHF$_{test}$.

B.2 Experiment on Nectar

Before conducting the experiment in Nectar, we first pre-process the dataset according to the following steps:

- Remove samples with a combined length of prompts and responses that exceeds 1024 tokens.
- Designate the highest-ranked response as the SFT target.

The basic information after data processing is shown on the Tabel 4.

Nectar	value
samples	177k
avg. prompt length	145.92
avg. response length	256.29
avg. turns	1.54

Table 4: Nectar after pre-process

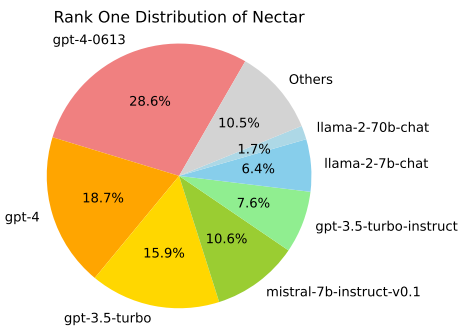

Figure 7: The distribution of the rank-one reply source.

We also perform a statistical analysis of the origins of the models from which the SFT target is derived, presented in Figure 7.

We divide the pre-processed data into a training set with 144k samples D_{Nectar}, the remaining will be used for verification. Each sample in D_{Nectar} contains one prompt x_i and seven responses $\{r_{i_j}\}_{j=0}^{6}$ sorted by quality:

$$D_{Nectar} = \{(x_i, r_{i_0}, r_{i_1}, r_{i_2}, r_{i_3}, r_{i_4}, r_{i_5}, r_{i_6})\}_{i=1}^{144k}$$

Based on D_{Nectar}, we construct an SFT dataset with x_i and r_{i_0}, using openchat[4] template:

$$\text{Nectar}_{SFT} = \{(x_i, r_{i_0})\}_{i=1}^{144k}$$

We also randomly sampled a 18k subset Nectar$_{18k}$ from D_{Nectar}.

hyper-parameter	value
epoch	1
batch size	128
learning rate	$5e-6$
precision	bfloat16

Table 5: Hyper-Parameter for θ_{SFT}.

hyper-parameter	value
epoch	1
batch size	$128n$
learning rate	$5e-7$
beta	0.1
precision	bfloat16

Table 6: Hyper-Parameter for DPO, where n is the number of negatives. We guarantee that the average number of prompts in a single batch remains constant for each configuration during training.

We then perform SFT on **Mistral-7B-v0.1** to get θ_{SFT} using Nectar$_{SFT}$, the hyper-parameters are shown in Table 5. For experiments with negative sample sizes less than or equal to 8, we sample 32 responses per prompt from θ_{SFT} on Nectar$_{18k}$ using vllm. For the experiment with 16 negative samples, we sample 64 responses per prompt. We use $temperature = 1.0$ and $top\ p = 1.0$.

After constructing the training sets for each setting according to the description in Section 3.1, we conduct DPO following the hyperparameter mentioned in Table 6.

[4] https://huggingface.co/openchat/openchat_3.5

When conducting experiments in Section 4.3, the batch size for the DPO baselines is $128 \times ceil(\frac{training\ samples}{18k})$, other hyperparameters follow Table 5.

B.3 Experiment on UltraChat

On UltraChat, we start with **zephyr-7b-sft-full** as θ_{SFT}, which is trained on **ultrachat_200k**. We have noticed that there are many versions of **zephyr-7b-sft-full**. To ensure a fair comparison with SPIN, we selected the model version that has MT-bench scores the same as those reported in SPIN. This particular version can be identified by its commit ID, which is $c3160e9$.

To establish the DPO_{pub} baseline, we performed a random sampling of a subset consisting of 16k data points from **ultrafeedback_binarized**. In our other experiments, the preference data for DPO is constructed using a 16k subset from the dataset **ultrachat_200k**. We have used θ_{SFT} to sample 32 or 64 responses per prompt, with $temperature = 1.0$ and $top\ p = 1.0$, which is consistent with the setup used in Nectar. During DPO and SELF-CONTRAST, we use the hyper-parameters the same as in Nectar.

B.4 Experiment on HH-RLHF

We perform SFT on HH-RLHF_{SFT}, following the previous hyper-parameters in Table 5.

When synthesizing the SELF-CONTRAST data, we filter the negative responses from 32 model responses with $a\%$ in $[100\%, 75\%, 50\%]$. The remaining hyperparameters are identical to those in Nectar.

When calculating the rewards for the test using **Starling-RM-7B-alpha**, for numerical stability, we added a layer of sigmoid function after the output of the model to map the reward values to $[0, 1]$.

C Experiment Setting for Meta-Llama-3-8B

For our additional experiments conducted on Nectar and Ultrachat with Meta-Llama-3-8B, most of the experimental settings are the same as with Mistral. The main difference is that, due to the model's characteristics, we used an SFT learning rate of 1e-5 and a DPO learning rate of 5e-6 for the experiments with Meta-Llama-3-8B.

D Limitations

Although massive negatives and response filtering have proven to be effective, there are still numerous aspects of our approach that require additional investigation.

Assessing human evaluation is necessary. To evaluate the human preference, we mainly test our model on MT-Bench and alpaca-eval, which is more aligned with GPT preferences. Although the SFT dataset we use is also generated by GPT-4, manual evaluation is still necessary in production to ensure effectiveness.

Safety and ethics still need further research. Although the positive samples we use come from SFT data, they can ensure a certain level of safety and ethics. However, considering the possibility of false negatives in the negative samples, the safety of our training method needs further verification.

Scaling is needed. Although current experiments have shown effectiveness on models with 7B parameters, the effectiveness of the method on larger models still needs further validation. In addition, increasing the data scale or using stronger embedding models also remains to be explored.