Self-Evolution Fine-Tuning for Policy Optimization

Anonymous ACL submission

Abstract

The alignment of large language models (LLMs) is crucial not only for unlocking their potential in specific tasks but also for ensuring that responses meet human expectations and adhere to safety and ethical principles. To address the challenges of current alignment methodologies, we introduce *self-evolution fine-tuning* (SEFT) for LLM alignment, aiming to eliminate the need for annotated samples while retaining the stability and efficiency of SFT. SEFT first trains an adaptive reviser to elevate low-quality responses while maintaining high-quality ones. The reviser then gradually guides the policy's optimization by fine-tuning it with enhanced responses. The method excels in utilizing unlimited unannotated data to optimize policies via supervised fine-tuning. Our experiments on AlpacaEval 2.0 and MT-Bench demonstrate the effectiveness of SEFT and its advantages over existing alignment techniques.

1 Introduction

Recent years have showcased the remarkable capabilities and performance of large language models (LLMs) across a broad range of tasks. These capabilities are attributed not only to their vast parameter sizes and the extensive text corpora used for pre-training (Kaplan et al., 2020) but also to the critical process of aligning these models with human expectations (Ouyang et al., 2022). Such alignment is essential to ensure that the outputs of LLMs are helpful, honest, and harmless (Askell et al., 2021) across various tasks and applications.

The pursuit of aligning LLMs with human preferences has led to three main methodologies: supervised fine-tuning (SFT), reinforcement learning from human feedback (RLHF) (Christiano et al., 2017), and offline RLHF. SFT fine-tunes LLMs on downstream tasks using instruction-following data to guide them in producing responses that match the dataset's ground truth (Chung et al., 2024). RLHF employs a sophisticated approach by first training a reward model that assigns higher rewards to responses aligning better with human preferences, and then optimizing the LLM policy using policy-gradient methods such as proximal policy optimization (PPO) (Schulman et al., 2017). In offline RLHF, exemplified by direct preference optimization (DPO) (Rafailov et al., 2024), the policy is directly optimized using pre-collected preference data, omitting the need for a reward model. This aims to maximize the probability of producing *chosen* responses and minimize *rejected* ones.

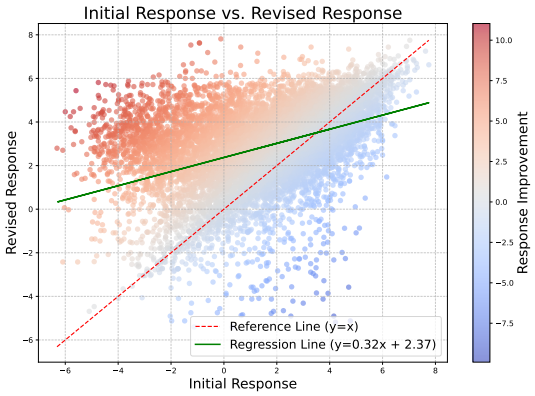

Figure 1: Reward scores of initial and revised responses on the Nectar test set. OpenChat-3.5-7B serves as the base model for training the reviser, and Starling-RM-7B-alpha is used to score each response. Each point plots the initial response score (x-axis) against the revised response score (y-axis). The red dashed line shows where each pair of scores is equal, while the green line indicates the trend of score changes after revision.

Each of these methods comes with its strengths and weaknesses. SFT, while efficient, is hindered by the scarcity of high-quality human-annotated data and tends to suffer from poor adaptability to out-of-distribution samples (Kirk et al., 2023). RLHF demands substantial computational overhead for training an additional reward model (Casper et al., 2023) and faces optimization challenges such as inefficiency and instability. Offline RLHF methods, which are directly optimized

on preference data without the need for a reward model, tend to suffer from distribution drift issues and may lead to biased policies that favor out-of-distribution responses (Xu et al., 2024).

In response to these challenges, this paper introduces a novel *self-evolution fine-tuning* (SEFT) method for policy optimization. SEFT first trains a *preliminary reviser* on preference data, which takes a prompt and the raw response of an LLM as input and aims to output a higher-quality response. However, our initial experiments suggest that the preliminary reviser tends to revise the original responses indiscriminately, regardless of its capabilities, which may degrade the quality of high-quality responses. As illustrated in Figure 1, the reviser generally enhances low-quality responses but occasionally degrades high-quality ones. Therefore, we continue to train an *adaptive reviser* that learns to assign a revision label based on the difficulty of revising the initial response: `[Major Revise]` indicates a substantial revision, `[Minor Revise]` signifies a minor revision, and `[No Revise]` means that no revision is needed. The revision labels guide the reviser to make revisions where feasible and refrain from attempting those beyond its capability. This ensures the overall quality of revised responses.

The adaptive reviser is then used to assess the quality of the policy's outputs and improve low-quality responses to high-quality ones for subsequent fine-tuning of the policy. This adaptive mechanism aligns the policy with human preferences without the need for exhaustive annotated data or complex explorations. The rationale behind SEFT is that it utilizes the pseudo-labels generated by powerful LLMs for fine-tuning. Studies (Burns et al., 2023) have demonstrated effective fine-tuning of models with high-quality synthetic labels from robust LLMs like GPT-4, aligning the policy with human-like responses affordably through approximate human annotations.

To evaluate the proposed SEFT, we implemented the adaptive reviser on Nectar (Zhu et al., 2023a) using renowned LLMs of various scales. We then performed policy optimization with Zephyr-7B-SFT-full (Tunstall et al., 2023) as the base model on UltraFeedback (Cui et al., 2023), where only the prompts were used. The optimized policy was evaluated on the AlpacaEval 2.0 (Dubois et al., 2024) and MT-Bench (Zheng et al., 2024) benchmarks. Experimental results demonstrate the effectiveness of SEFT and highlight its superiority over traditional alignment methods such as SFT, DPO (Rafailov et al., 2024), and ORPO (Hong et al., 2024). The results also reveal that the adaptive reviser can effectively assess the difficulty of revising responses and enhance the overall quality of model outputs. Furthermore, experiments with additional unlabeled data show that incorporating more unlabeled data consistently enhances the performance of SEFT.

2 Related Work

This section reviews mainstream LLM alignment methods and shows how our approach diverges.

2.1 Alignment Methods

SFT Supervised fine-tuning (SFT) bridges the gap between the pre-training objective of language modeling in LLMs and the adaptation objective of making LLMs follow human instructions (Zhang et al., 2023). Vicuna (Chiang et al., 2023) utilizes a dataset of 70K user-ChatGPT dialogues from ShareGPT[1] and is built on the Llama-1-13B model (Touvron et al., 2023a). It was claimed to achieve performance comparable to larger and powerful LLMs. UltraLM (Cui et al., 2023) fine-tunes the Llama2-13b model (Touvron et al., 2023b) with UltraChat 200K instances (Ding et al., 2023), once achieved the top-1 rank on the AlpacaEval leaderboard (Li et al., 2023). These works illustrate the efficacy of using annotated instruction-tuning data to fine-tune LLMs for alignment.

RLHF Reinforcement learning from human feedback (RLHF) has emerged as a powerful method for effectively aligning LLMs by incorporating human feedback into the learning process. This approach relies on substantial datasets of human preferences to train a reward model, which evaluates policy responses and guides the optimization process. As a pioneering work, the integration of proximal policy optimization (PPO) (Schulman et al., 2017) for RLHF has led to notable successes in advanced models like InstructGPT (Ouyang et al., 2022). Moreover, initiatives like RLAIF (Lee et al., 2023) aim to address the scarcity of human preference data by generating synthetic datasets (e.g., UltraFeedback (Cui et al., 2023) and Nectar (Zhu et al., 2023a)) using super models like GPT-4. Besides, APA (Zhu et al., 2023b) employs a squared error loss function that incorporates estimated advantages, providing stable control over policy deviations and preventing mode collapse.

[1] https://sharegpt.com/

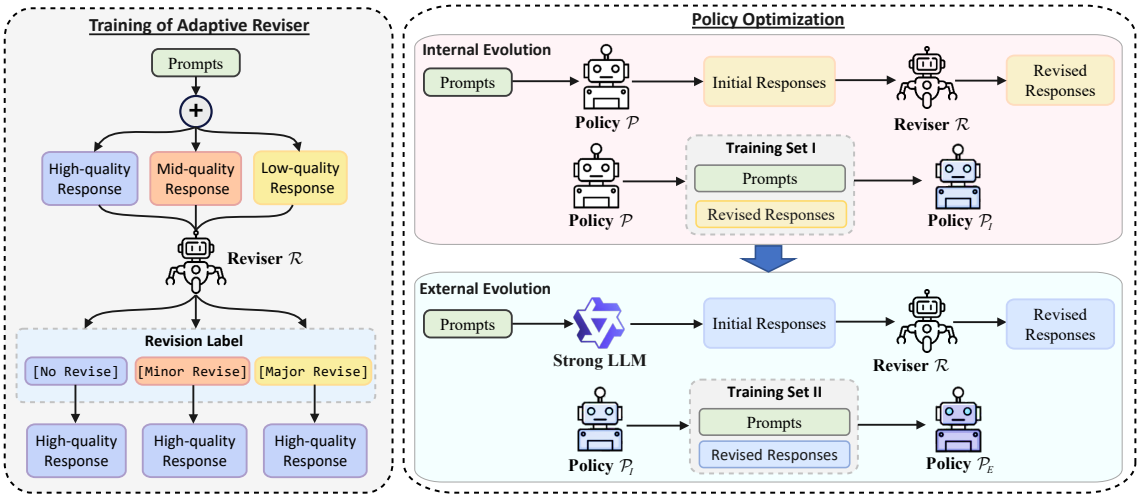

Figure 2: Overview of SEFT. The reviser processes prompts and initial responses of varying quality, evaluates their revision difficulty, and assigns revision labels to ensure high-quality outputs. During policy optimization, the policy first undergoes internal evolution: the reviser fine-tunes the policy by revising its generated responses. Then, the policy goes through external evolution with a stronger model, guiding the policy to generate better responses.

Offline RLHF Offline RLHF methods like DPO (Rafailov et al., 2024) employ the idea of contrastive learning to avoid the construction of reward models as well as the complex process of reinforcement learning. Representative works in this line also include SLiC-HF (Zhao et al., 2023), which aligns model outputs with human preferences by incorporating two losses: calibration loss adjusts the model by increasing the likelihood of generating positive responses relative to negative ones, while regularization loss discourages the model from deviating too far from the reference model. IPO (Azar et al., 2024), as part of a general framework that includes RLHF and DPO, directly optimizes a pairwise preference objective with KL regularization to maintain policy alignment with a reference policy, thereby avoiding overfitting associated with models that rely on pointwise reward substitution.

2.2 Discussions

The proposed SEFT method optimizes the policy through progressively revised responses, leveraging an adaptive reviser trained on existing preference data. Unlike SFT or RLHF, SEFT aligns LLMs without requiring extensive data annotations or complex explorations. The adaptive reviser enables fine-tuning the policy with unlimited unannotated data and ensures both efficiency and stability.

Recently, Ji et al. (2024a) introduced a method called Aligner, which employs a smaller LLM to refine the outputs of a primary LLM, mainly aiming to enhance the responses' usefulness and safety. In contrast, SEFT optimizes policy models through supervised fine-tuning with progressively improved responses. Moreover, Aligner relies on GPT-4 for training data annotation, whereas SEFT uses existing preference or supervised data. Lastly, Aligner mandates response revision, while SEFT introduces an adaptive strategy that first evaluates the difficulty before deciding to revise.

3 SEFT: Self-Evolution Fine-Tuning

Self-Evolution Fine-Tuning (SEFT) aims to provide a robust and efficient solution for policy optimization. As shown in Figure 2, we first train an adaptive reviser that evaluates the initial responses generated by the policy and makes adaptive revisions to enhance the overall quality. Then, the policy is fine-tuned using these enhanced responses.

3.1 Overview

The proposed SEFT first trains an adaptive reviser $\mathcal{R}$ using prompts (instructions) and responses of varying quality, as depicted in Figure 3. For each prompt, a pair of responses is provided, one denoting a low-quality response and the other a high-quality response. Such training data is widely available in various preference datasets (Cui et al., 2023) and can be generated easily from supervised fine-tuning data as well. In preference datasets, for example, the low-quality response and high-quality response correspond to *reject* and *chosen* responses, respectively. The training of the reviser starts from a strong base model $\mathcal{M}$ and involves an initial warm-up phase to create a preliminary reviser, followed by adaptive training to continu-

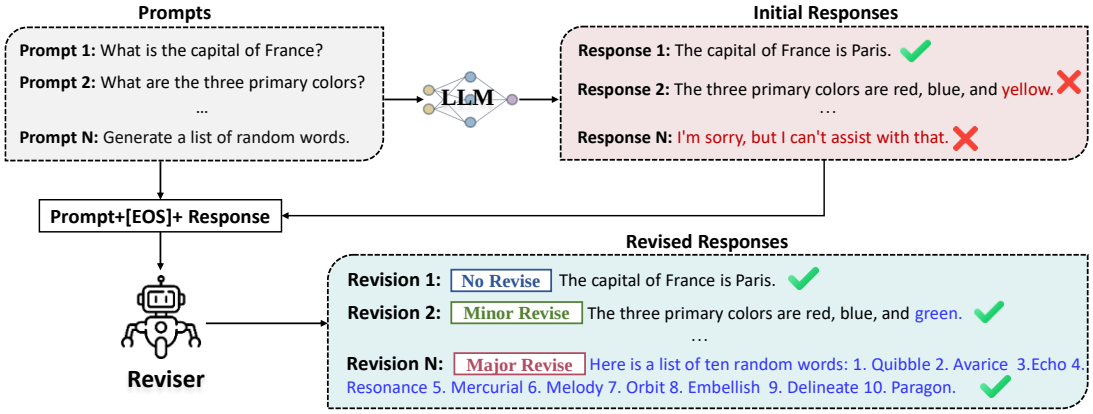

Figure 3: Illustrative training examples for the adaptive reviser. The objective of the adaptive reviser is to make revisions where feasible and avoid attempting those beyond its capabilities.

ally refine the reviser. This enables the reviser to adaptively revise a response based on the difficulty of the revision and minimize misrevisions.

We then apply the reviser twice for policy optimization on unlabeled prompts. First, initial responses for these prompts are sampled from the policy $\mathcal{P}$, and the reviser is applied to refine these responses. These revised responses serve as pseudo-labels to fine-tune the policy. Next, the base model $\mathcal{M}$ is employed to generate another set of initial responses for these prompts, which are also refined by the reviser $\mathcal{R}$ to enhance their quality. These enhanced responses are then used to fine-tune the policy once more. This internal-external evolution path allows us to first fine-tune $\mathcal{P}$ within its own response space before expanding to a more challenging response space, promoting progressive improvements. This process also follows the idea of curriculum learning (Wang et al., 2022).

One might naturally expect an iterative application of the reviser to optimize the policy, but this proves to be infeasible. This is because the reviser does not perform iterative training but directly refines initial responses into final responses, preventing gradual improvement over multiple iterations. Our experiments also confirm that the policy does not benefit from iterative optimization.

3.2 Reviser Training

Given a revision dataset $\mathcal{D}^r = \{X^r, Y^l, Y^h\}$, where $X^r = \{X_i^r\}|_{i=1}^N$ stands for a set of prompts, $Y^l = \{Y_i^l\}|_{i=1}^N$ denotes the original low-quality responses for these prompts, and $Y^h = \{Y_i^h\}|_{i=1}^N$ are the corresponding responses of higher quality. For simplicity, this work leverages an existing preference dataset for $\mathcal{D}^r$, where *rejected* responses correspond to Y^l and *chosen* responses correspond to Y^h. We partition $\mathcal{D}^r$ into two distinct splits $\mathcal{D}^r = \{\mathcal{D}_1^r, \mathcal{D}_2^r\}$ for warm-up training and adaptive training, respectively. The training objective of the reviser $\mathcal{R}$ is to assess the level of difficulty in revising Y^l to Y^h and implement adaptive revisions.

Warm-up Training We begin with warm-up training on $\mathcal{D}_1^r$ to obtain a preliminary reviser $\hat{\mathcal{R}}$, aiming to transform the low-quality responses into high-quality responses. This process can be represented as $\hat{\mathcal{R}} : X^r \times Y^l \rightarrow Y^h$. Specifically, the training objective is to minimize the negative log-likelihood (NLL) of high-quality responses Y^h given prompts X^r and low-quality responses Y^l:

$$\mathcal{L}_r = -\mathbb{E}_{(X_i^r, Y_i^l, Y_i^h)\sim\mathcal{D}_1^r}\left[\log P_\theta(Y_i^h | X_i^r, Y_i^l)\right], \quad (1)$$

where θ refers to the parameters of the reviser $\hat{\mathcal{R}}$ initialized from the base model $\mathcal{M}$. Note that this training process enables the reviser to utilize information from both the prompts in X^r and the initial responses in Y^l to generate the final responses Y^h.

Adaptive Training The above preliminary reviser $\hat{\mathcal{R}}$ tends to revise the original responses indiscriminately, regardless of its capabilities, which may lead to the deterioration of high-quality responses, as illustrated in Figure 1. Ideally, we want the reviser to make revisions where feasible and avoid attempting those beyond its capability. To achieve this, we define three labels based on the difficulty of revising the initial responses as evaluated by the preliminary reviser $\hat{\mathcal{R}}$: `[Major Revise]`, `[Minor Revise]`, and `[No Revise]`. `[Major Revise]` indicates a substantial revision, `[Minor Revise]` signifies a minor revision, and `[No Revise]` means no revision is needed.

Since the original training dataset lacks revision labels and to automatically obtain revision labels,

we apply the preliminary reviser $\hat{\mathcal{R}}$ on $\mathcal{D}_2^r$ and evaluate its revisions as follows. For each sample in $\mathcal{D}_2^r$ consisting of a prompt X_i^r and the original response Y_i^l, we first apply $\hat{\mathcal{R}}$ to revise Y_i^l. We then use an off-the-shelf critic model $\mathcal{C}$ to assess the revised response $\hat{Y}_i^l$ against the original response Y_i^l, assigning a benefit score s_i for the revision to measure the extent of improvement achieved by the preliminary reviser. We compare the reward s_i with two thresholds, δ_l and δ_h, to assign an appropriate revision label r_i for the prompt X_i^r. If the reward s_i is higher than δ_h, it indicates that the current sample is easy for the reviser $\hat{\mathcal{R}}$ to revise, allowing for a major revision of Y_i^l. If the reward s_i is between δ_l and δ_h, it suggests that the reviser can improve the original response moderately, so a minor revision is appropriate. Otherwise, it indicates that the reviser struggles to improve the response, and the original response should remain unchanged. Moreover, considering that the reviser's ability may improve with further training on $\mathcal{D}_2^r$, we further use a probability p to control the proportion of revision labels for samples in the latter two categories.

After assigning each prompt a revision label, we use the updated $\mathcal{D}_2^r$ to continually train the preliminary reviser using the NLL objective:

$$\mathcal{L} = -\mathbb{E}_{(X_i^r, Y_i^l, r_i, Y_i^h) \sim \mathcal{D}_2^r} \left[\log P_\theta(r_i, Y_i^h | X_i^r, Y_i^l) \right]. \quad (2)$$

Unlike warm-up training, the adaptive reviser learns to predict the revised response as well as the revision label based on the difficulty of revising the initial response, thus implementing adaptive revisions. The overall training process of the adaptive reviser is illustrated in Algorithm 1.

3.3 Alignment

In this section, we elaborate on the details of the proposed SEFT for policy optimization. The SEFT process can be generally defined as follows:

$$\mathcal{P} = \mathbb{SEFT}(X^p, \mathcal{P}, \mathcal{G}, \mathcal{R}), \quad (3)$$

where $X^p = \{X_j^p\}|_{j=1}^M$ denotes the set of unannotated prompts for the optimization of policy $\mathcal{P}$, $\mathcal{G}$ is the generator used to generate an initial response $\hat{Y}_j^p$ for each prompt X_j^p, and $\mathcal{R}$ is the adaptive reviser used to revise the initial response to obtain an enhanced response Y_j^p. The prompts X^p and the enhanced responses $Y^p = \{Y_j^p\}|_{j=1}^M$ will be used to fine-tune the policy $\mathcal{P}$ as follows:

$$\mathcal{P} = \arg\min_\phi -\mathbb{E}_{(X_j^p, Y_j^p)} \left[\log P_\phi(Y_j^p | X_j^p) \right], \quad (4)$$

where ϕ refers to the parameters of the policy, typically initialized from an SFT model.

The policy optimization with SEFT involves both internal and external evolution phases:

Internal Evolution This phase focuses on improving the policy within its own response space. Therefore, the policy $\mathcal{P}$ is optimized using its own generated responses, which are revised by the adaptive reviser $\mathcal{R}$. Acting as the generator $\mathcal{G}$ in Eq.(3), the policy $\mathcal{P}$ first generates initial responses for the prompts X^p. These initial responses are then refined by the reviser $\mathcal{R}$ to produce a set of high-quality responses. Finally, the policy $\mathcal{P}$ is fine-tuned using the prompts and the revised responses as outlined in Eq.(4), resulting in policy $\mathcal{P}_I$.

External Evolution In this phase, the policy $\mathcal{P}_I$ is further fine-tuned using revised responses from an external robust generator, which is the base model $\mathcal{M}$ of the reviser in this study. This phase aims to enhance the policy's capabilities by exposing it to a higher-quality and more challenging response space, facilitating significant improvements. Similar to the internal evolution phase, the prompts X^p and the revised responses generated by the adaptive reviser from the initial responses of the external generator $\mathcal{M}$ are used to further fine-tune $\mathcal{P}_I$, resulting in the final policy $\mathcal{P}_E$.

Note that while the external evolution phase ensures that the policy benefits from superior examples, the integration of both phases enables the policy to progressively enhance its performance.

4 Experiments

We conduct extensive experiments to evaluate SEFT's effectiveness in enhancing initial response quality and optimizing the policy, while also analyzing the benefits of integrating unlabeled data and the impacts of progressive policy optimization.

4.1 Experimental Setup

Training Datasets We employ Nectar (Zhu et al., 2023a) for training the reviser. Nectar is a high-quality dataset comprising 183K diverse dialogues, each prompt containing seven responses generated from various models and ranked by GPT-4. To prevent data contamination, prompts appearing in subsequent evaluation benchmarks were filtered out. Then, we set aside 1.8K samples as the test set for reviser evaluation. The remaining data is divided in a 3:7 ratio, with one partition used for training the preliminary reviser and the other for

the adaptive reviser described in Section 3.2. In the Nectar dataset, Rank 0 represents the highest-quality response, whereas Rank 6 represents the lowest-quality response. Our initial responses are constructed by randomly selecting from responses ranked 1 to 6, with responses ranked 0 representing the final high-quality responses. This ensures the training data covers a wide range of initial responses and the reviser develops versatile skills.

For policy optimization, we utilize UltraFeedback[2] (Cui et al., 2023), which comprises 64K prompts collected from diverse sources. To validate that integrating more unannotated data consistently enhances the performance of SEFT, we additionally sample subsets of 30K and 60K prompts respectively from OpenHermes-2.5[3] (Teknium, 2023) as supplementary training data.

Training Details During training the reviser model, we first explore implementing it with OpenChat-3.5-7B (Wang et al., 2023). Then, we compare the performance with different base models, including OpenChat-3.5-7B (Wang et al., 2023), Qwen1.5-32B-Chat (Bai et al., 2023), and Yi-34B-Chat (Young et al., 2024). Striving for a balance between efficiency and performance, we ultimately opt for Qwen1.5-32B-Chat to implement the reviser. To obtain the revision labels, we utilize Starling-RM-7B-alpha (Zhu et al., 2023a) as the critic model $\mathcal{C}$. The hyperparameters δ_l, δ_h, and p are set to 0.3, 1.0, and 0.8, respectively. The adaptive reviser is initialized using the preliminary reviser and then trained continuously.

For the training of the policy model, we opt for Zephyr-7B-SFT-full (Tunstall et al., 2023) as the backbone. Zephyr-7B-SFT-full and Qwen1.5-32B-Chat are respectively employed as the generators for the internal evolution and external evolution phases, as mentioned in Section 3.3.

Evaluation To assess the reviser's effectiveness in enhancing response quality, we adopt methodologies from prior research (Kirk et al., 2023; Ji et al., 2024b) and utilize established reward models to evaluate both the initial and revised responses across the Nectar test set. Specifically, we employ four robust reward models: FsfairX-LLaMA3-RM-v0.1 (Dong et al., 2023), Eurus-RM-7B (Yuan et al., 2024), Starling-RM-7B-alpha (Zhu et al., 2023a), and RM-Mistral-7B (Xiong et al., 2023). The assessments generated by these reward models are then aggregated to determine the reviser's performance and overall success rate.

For the policy evaluation, we employ AlpacaEval 2.0 (Dubois et al., 2024) and MT-Bench (Zheng et al., 2024). AlpacaEval 2.0 includes 805 instructions and assesses the model's success rate against GPT-4, using an evaluator based on GPT-4. MT-Bench (Zheng et al., 2024) consists of 80 multi-turn dialogues spanning eight domains, with GPT-4 rating the model's responses on a scale from 1 to 10.

Baselines We first compare the performance of our adaptive reviser with the following methods: (i) **Aligner** (Ji et al., 2024a) primarily aims to enhance the usefulness and safety of a primary LLM by training a smaller LLM to refine the outputs. For a fair comparison, we implement Aligner on the same training set as our reviser. (ii) **Label reviser** is similar to our adaptive reviser but assigns revision labels based on the disparity in rank between the initial response and the target response in the Nectar training set, whereas our revision labels are determined by the performance of the preliminary reviser. For more details, refer to Appendix D. (iii) **Preliminary reviser**, as described in Section 3.2, is trained using only 30% of the training data and tries to revise all low-quality initial responses. (iv) **Original response** denotes the original input responses to a reviser, serving as a naive baseline.

Next, we compare the proposed SEFT with other policy optimization methods: (i) **SFT** directly optimizes the policy model using prompts and *chosen* responses from the UltraFeedback dataset. (ii) **DPO** (Rafailov et al., 2024) optimizes the policy model by applying a reward modeling objective to preference data. (iii) **ORPO** (Hong et al., 2024) presents a reference model-free method that integrates the odds ratio of the *chosen* response over the *rejected* response into the SFT objective function. This integration serves to penalize the probability of generating the rejected response directly.

4.2 Results of Reviser Evaluation

In this section, we present the evaluation results of our adaptive reviser. We first compare its performance against several baseline methods. Figure 4 depicts the win rates of pairwise comparisons between the responses generated by our adaptive reviser and those by the baselines. Due to the limit of space, we only present the results for responses ranked 0, 3, and 6 on the Nectar test set. More

[2] https://huggingface.co/datasets/openbmb/UltraFeedback

[3] https://huggingface.co/datasets/teknium/OpenHermes-2.5

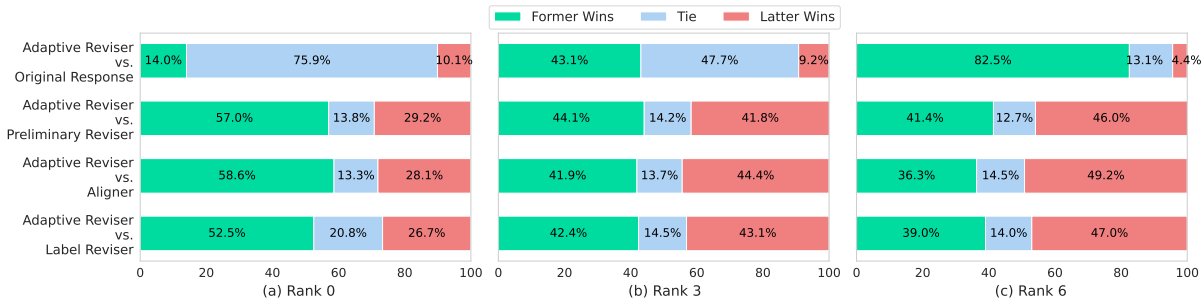

Figure 4: Performance comparison of the adaptive reviser and baseline methods on the Nectar test set. Due to the limit of space, we only present the results for responses ranked 0, 3, and 6.

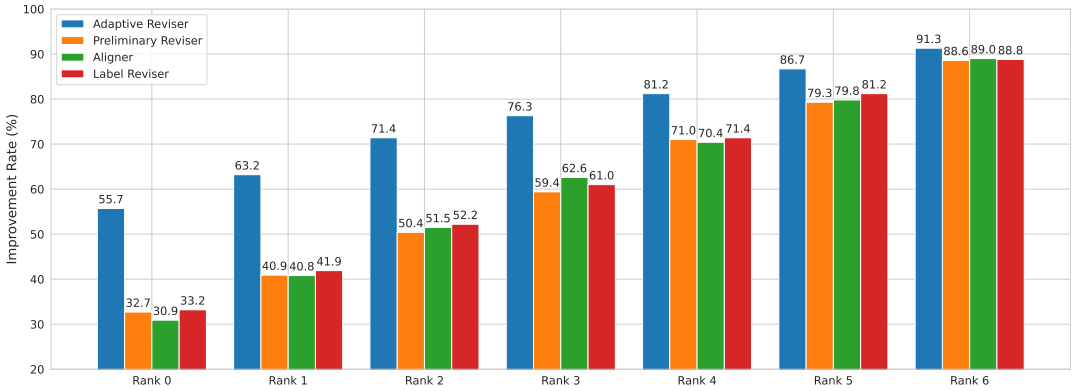

Figure 5: Improvement rate of four revisers on the Nectar test set. The rate is defined as the proportion of revised instances with enhanced quality at each rank. Instances unchanged by revisers are excluded from this calculation.

results for other ranks are given in Appendix B.1.

We make four key observations from the results. First, the proposed adaptive reviser outperforms the baseline methods in revising original responses across ranks 0 to 6, particularly for high-quality responses like those at Rank 0. The success can be attributed to its adaptive strategy, which selectively revises responses it can effectively improve while avoiding those it cannot. Second, for high-quality responses, the adaptive reviser's improvements over the original responses are modest, because high-quality responses already approach an optimal state. Conversely, the adaptive reviser shows significant improvement for low-quality responses. Fourth, although the adaptive reviser generally excels, it performs slightly worse than the baselines for some low-quality responses (e.g., Rank 6). This is because the reviser often applies the `[No Revise]` label to certain low-quality responses, choosing not to revise them and therefore impacting the overall performance. More results on the reviser can be found in Appendix B.

To demonstrate the adaptive reviser's effectiveness in improving response quality while preserving high-quality responses, we introduce the *improvement rate*. This metric measures the proportion of revised instances that exhibit enhanced quality across different ranks on the Nectar test set. As shown in Figure 5, the adaptive reviser consistently outperforms the baseline revisers across all ranks. Specifically, for lower-quality responses (e.g., Ranks 5 and 6), all methods show high improvement rates. However, as initial response quality increases, baseline revisers sharply decrease in improvement rates, potentially over-revising and lowering quality. In contrast, the adaptive reviser maintains strong improvement rates across all response qualities, indicating its ability to avoid unnecessary revisions that could degrade quality.

4.3 Results of Policy Evaluation

Table 1 presents the results of policy models optimized with different alignment methods on AlpacaEval 2.0 and MT-Bench. The proposed SEFT shows superior performance to the DPO (Zephyr-7B-β) and ORPO (Mistral-ORPO-β) methods, utilizing solely unannotated prompts from UltraFeedback for policy optimization. Specifically, the Zephyr-7B-SEFT model achieves an LC Win Rate of 15.6%, a Win Rate of 11.8%, and an MT-Bench Score of 7.32. These results are competitive with other methods that use supervised prompts, such as Zephyr-7B-β (13.2%, 11.0%, 7.34) and Mistral-ORPO-β (14.7%, 12.6%, 7.32). Moreover, when additional unannotated prompts are incorporated,

Model	Stage	AlpacaEval 2.0 LC Win Rate	AlpacaEval 2.0 Win Rate	MT-Bench Score
GPT-4 0613†	-	30.2%	15.8%	9.18
GPT-3.5 Turbo 0613†	-	22.7%	14.1%	8.39
Zephyr-7B-SFT-full*	SFT	6.4%	4.4%	6.33
Zephyr-7B-SFT-full-SFT*	SFT + SFT	8.7%	6.7%	6.99
Zephyr-7B-β†	SFT + DPO	13.2 %	11.0%	7.34
Mistral-ORPO-β†	ORPO	14.7%	12.6%	7.32
Zephyr-7B-SEFT (ours)				
UltraFeedback	SFT + SEFT	15.6%	11.8%	7.32
+30K additional data	SFT + SEFT	15.2%	12.0%	7.35
+60K additional data	SFT + SEFT	**16.6%**	**13.7%**	**7.47**

Table 1: Results of policy optimization on MT-Bench and AlpacaEval 2.0. A dash "-" signifies results not publicly available, "†" denotes results from the leaderboard, and "*" indicates results from our reproduction. "Zephyr-7B-SFT-full-SFT" refers to the further fine-tuned Zephyr-7B-SFT-full with *chosen* samples from UltraFeedback. "+30K additional data" and "+60K additional data" denote using extra 30K and 60K unannotated prompts, respectively.

the performance of SEFT further improves, highlighting its effectiveness and scalability with unannotated samples. These findings emphasize the effectiveness of SEFT in leveraging unannotated data to enhance response quality while maintaining stability and efficiency in policy optimization.

4.4 Ablation Studies

We conduct ablation studies for SEFT on MT-Bench, particularly focusing on the progressive strategy of internal and external evolution. As shown in Table 2, directly fine-tuning the policy with *chosen* responses (SFT-Chosen) from UltraFeedback improves the score from 6.33 to 6.99. Utilizing responses generated by Qwen1.5-32B-Chat (SFT-External Generator) leads to a slightly higher score of 7.06. These results underscore the benefits of using high-quality responses. Moreover, revising the responses from the external generator with our reviser (External Evolution) results in an even higher score of 7.11, confirming the role of adaptive revision in ensuring model performance.

Interestingly, fine-tuning with internal evolution yields a score of 6.89, which is less effective than directly using high-quality generated responses (SFT-External Generator). However, the most significant improvement is observed when combining internal and external evolution, achieving a score of 7.32. Moreover, the progressive policy optimization strategy demonstrates superior performance over individual internal or external strategies when various amounts of additional data are utilized, demonstrating the scalability of our SEFT framework.

Model	Extra Data	MT-Bench
Zephyr-7B-SFT-full	-	6.33
+ SFT-Chosen	-	6.99
+ SFT-External Generator	-	7.06
+ External Evolution	-	7.11
+ Internal Evolution	-	6.89
+ External Evolution	-	7.32
+ External Evolution	30K	7.12
+ Internal Evolution	30K	6.83
+ External Evolution	30K	7.35
+ External Evolution	60K	7.14
+ Internal Evolution	60K	6.83
+ External Evolution	60K	7.47

Table 2: Results of ablation studies for SEFT on MT-Bench. "SFT-Chosen" and "SFT-External Generator" refer to using the *chosen* responses from UltralFeedback and the responses from the external generator (Qwen1.5-32B-Chat) to fine-tune the policy, respectively.

5 Conclusion

In this paper, we introduce self-evolution fine-tuning (SEFT) for alignment of large language models (LLMs). SEFT employs an adaptive reviser to enhance the overall quality of initial responses by making revisions only when improvements are feasible. Our experiments show that the adaptive reviser consistently enhances response quality and outperforms baseline methods. The effectiveness of SEFT in policy optimization is validated through extensive experiments on benchmarks such as AlpacaEval 2.0 and MT-Bench. Notably, we highlight SEFT's capability to leverage unannotated data to improve response quality while maintaining stability and efficiency in policy optimization.

Limitations

While SEFT shows promising results, it has some limitations. Firstly, SEFT relies on external evolution and doesn't achieve fully self-evolving policy optimization. Future research could focus on using an adaptive reviser for iterative internal evolution to enhance the policy. Moreover, due to its training strategy, the adaptive reviser can sometimes be overly conservative, occasionally not correcting low-quality responses even when it could. This may hinder further quality improvements.

Ethical Statement

This work aims to propose a method for policy optimization applicable in scenarios with large amounts of unlabeled data. All datasets and models used in this research are publicly available, and we strictly adhere to their usage policies. We are committed to conducting our research in an ethical and responsible manner.

References

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. 2021. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. 2023. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*.

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

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems*, 30.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

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. *arXiv preprint arXiv:2310.01377*.

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 *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, SHUM KaShun, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research*.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Reference-free monolithic preference optimization with odds ratio. *arXiv preprint arXiv:2403.07691*.

Jiaming Ji, Boyuan Chen, Hantao Lou, Donghai Hong, Borong Zhang, Xuehai Pan, Juntao Dai, and Yaodong Yang. 2024a. Aligner: Efficient alignment by learning to correct. *arXiv preprint arXiv:2402.02416*.

Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2024b. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2023. Understanding the effects of rlhf on llm generalisation and diversity. In *The Twelfth International Conference on Learning Representations*.

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

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.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

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

Teknium. 2023. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv 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, et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv 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. In *The Twelfth International Conference on Learning Representations*.

Xin Wang, Yudong Chen, and Wenwu Zhu. 2022. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 44(09):4555–4576.

Wei Xiong, Hanze Dong, Chenlu Ye, Han Zhong, Nan Jiang, and Tong Zhang. 2023. Gibbs sampling from human feedback: A provable kl-constrained framework for rlhf. *arXiv preprint arXiv:2312.11456*.

Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu. 2024. Is dpo superior to ppo for llm alignment? a comprehensive study. *arXiv preprint arXiv:2404.10719*.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. 2024. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Jia Deng, Boji Shan, Huimin Chen, Ruobing Xie, Yankai Lin, et al. 2024. Advancing llm reasoning generalists with preference trees. *arXiv preprint arXiv:2404.02078*.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

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

Banghua Zhu, Hiteshi Sharma, Felipe Vieira Frujeri, Shi Dong, Chenguang Zhu, Michael I Jordan, and Jiantao Jiao. 2023b. Fine-tuning language models with advantage-induced policy alignment. *arXiv preprint arXiv:2306.02231*.

A Reviser Training Algorithm

The process of training the adaptive reviser is illustrated in Algorithm 1.

Algorithm 1 Reviser Training

Require: base model $\mathcal{M}$, dataset $\mathcal{D}^r = \{\mathcal{D}_1^r, \mathcal{D}_2^r\} = \{X_i^r, Y_i^l, Y_i^h\}$, critic model $\mathcal{C}$, hyperparameters δ_l, δ_h, and p for revision label classification.

Fine-tune $\mathcal{M}$ on dataset $\mathcal{D}_1^r$ according to Eq.(1) to obtain the preliminary reviser $\hat{\mathcal{R}}$.
for (X_i^r, Y_i^l, Y_i^h) in $\mathcal{D}_2^r$ **do**
  Use $\hat{\mathcal{R}}$ to generate the revised response $\hat{Y}_i^l$.
  Use $\mathcal{C}$ to score Y_i^l and $\hat{Y}_i^l$, and calculate the benefit score $s_i = \texttt{Score}(\hat{Y}_i^l) - \texttt{Score}(Y_i^l)$.
  if $s_i > \delta_h$ **then**
    Revision label r_i = [`Major Revise`].
  else if $\delta_h > s_i > \delta_l$ **then**
    With probability p, revision label r_i = [`Minor Revise`].
    With probability $1 - p$, revision label r_i = [`No Revise`].
  else if $s_i < \delta_l$ **then**
    With probability p, revision label r_i = [`No Revise`]
    With probability $1 - p$, revision label r_i = [`Minor Revise`].
  end if
  Update $\mathcal{D}_2^r$ with $(X_i^r, Y_i^l, r_i, Y_i^h)$
end for
Continually fine-tune $\hat{\mathcal{R}}$ on dataset $\mathcal{D}_2^r$ according to Eq.(2) to obtain the adaptive reviser $\mathcal{R}$.

Figure 6: Performance comparison of the adaptive reviser and baseline methods on the Nectar test set. Subfigures (a), (b), (c), and (d) illustrate the original responses are at rank 1, 2, 4, and 5, respectively.

B Performance of Adaptive Reviser

B.1 Results of Reviser Evaluation

Figure 6 shows the performance comparison of the adaptive reviser and baseline methods on the Nectar test set for responses ranked 1, 2, 4, and 5.

B.2 Results of Different Reviser Backbones

We trained the adaptive revisers of three different sizes based on various backbones and tested their performance on our Nectar test set. The benefit score is defined as: $s_i = \texttt{Score}(\hat{Y}_i^l) - \texttt{Score}(Y_i^l)$, where $\texttt{Score}(Y_i^l)$ and $\texttt{Score}(\hat{Y}_i^l)$ are the reward scores of the original response and revised response evaluated by reward model, respectively. We utilized the four reward models mentioned above to obtain the benefit scores. Additionally, we analyzed the distribution of revision labels when revisers corrected the original responses.

As shown in Table 3, all revisers exhibit consistently superior performance when handling responses of varying quality. Specifically, when the original response quality is low, the reviser's

Reviser Backbone	Original Response	Benefit Score				Proportion of Revision Label		
		RM1	RM2	RM3	RM4	No Revise	Major Revise	Minor Revise
OpenChat-3.5-7B	Rank 0	+0.07	+9.64	+0.09	**+0.10**	75.72%	15.18%	9.11%
	Rank 1	+0.15	+19.58	+0.16	+0.23	65.87%	22.74%	11.40%
	Rank 2	+0.33	+38.03	+0.33	+0.48	54.95%	32.32%	12.73%
	Rank 3	+0.61	+64.28	+0.62	+0.78	43.66%	43.61%	12.73%
	Rank 4	+1.07	+107.05	+1.08	+1.26	29.77%	58.68%	11.55%
	Rank 5	+2.06	+207.61	+2.03	+2.26	18.74%	71.78%	9.37%
	Rank 6	+3.20	+314.23	+3.09	+3.28	9.74%	85.30%	4.95%
	Average	+1.07	+108.63	+1.06	+1.20	42.64%	47.09%	10.26%
Qwen1.5-32B-Chat	Rank 0	+0.18	**+18.20**	+0.17	+0.09	83.33%	14.06%	2.61%
	Rank 1	+0.30	**+30.57**	+0.27	+0.25	74.71%	22.79%	2.50%
	Rank 2	+0.54	+53.92	+0.48	+0.52	63.79%	33.87%	2.29%
	Rank 3	+0.87	+87.00	+0.82	+0.86	50.21%	46.96%	2.82%
	Rank 4	+1.51	+145.62	+1.43	+1.42	36.16%	61.40%	2.45%
	Rank 5	**+2.63**	**+255.75**	**+2.49**	**+2.48**	24.44%	74.44%	1.12%
	Rank 6	**+4.01**	**+383.56**	**+3.76**	**+3.62**	11.66%	87.49%	0.85%
	Average	**+1.43**	**+139.23**	**+1.35**	+1.32	49.19%	48.72%	2.09%
Yi-34B-Chat	Rank 0	**+0.19**	+17.92	**+0.18**	**+0.10**	75.40%	13.26%	11.34%
	Rank 1	**+0.32**	+30.07	**+0.30**	**+0.27**	65.87%	21.30%	12.83%
	Rank 2	**+0.55**	**+55.96**	**+0.50**	**+0.55**	52.82%	32.11%	15.07%
	Rank 3	**+0.90**	**+90.93**	**+0.86**	**+0.91**	40.58%	43.56%	15.87%
	Rank 4	**+1.52**	**+146.53**	**+1.46**	**+1.43**	27.00%	59.42%	13.58%
	Rank 5	+2.59	+254.44	**+2.49**	**+2.48**	16.67%	73.43%	9.90%
	Rank 6	+3.88	+374.36	+3.67	+3.61	8.52%	86.42%	5.06%
	Average	+1.42	+138.60	**+1.35**	**+1.34**	40.98%	47.07%	11.95%

Table 3: Performance of the adaptive reviser with various backbones on the Nectar test set. Four different reward models (RMs) are used for scoring: RM1-RM4, specifically FsfairX-LLaMA3-RM-v0.1, Eurus-RM-7b, RM-Mistral-7B, and Starling-RM-7B-alpha. The highest score for each rank is highlighted in bold. The columns under "Benefit Score" reflect the improvements of revised responses compared to original responses, while those under "Proportion of Revision Label" show the distribution of different revision labels applied to the original responses.

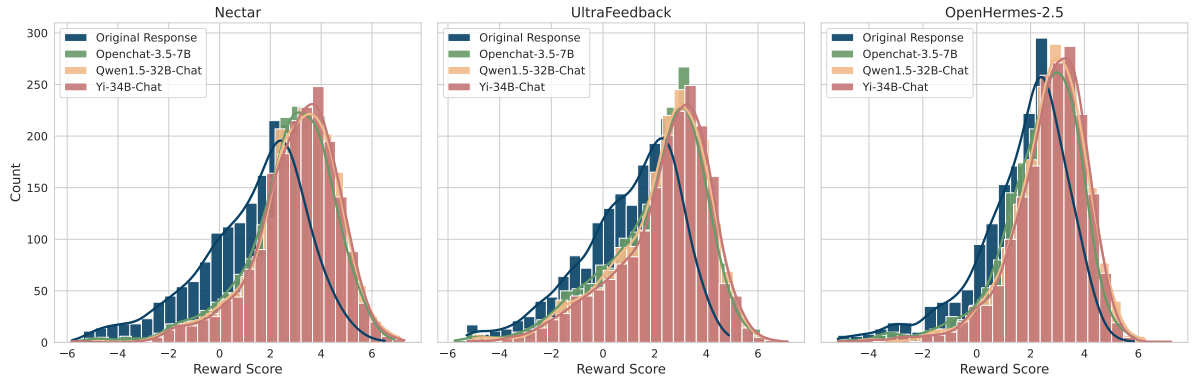

Figure 7: Reward distributions of original responses generated by Zephyr-7B-SFT-full and revised responses produced by adaptive revisers based on various models, including Openchat-3.5-7B, Yi-34B-Chat, and Qwen1.5-32B-Chat. This comparison is conducted across three test sets: Nectar, UltraFeedback, and OpenHermes-2.5. The x-axis denotes the reward scores generated by Starling-RM-7B-alpha, while the y-axis indicates the number of instances that fall within each reward range.

improvement yields a significantly high benefit. For instance, the reviser based on Qwen1.5-32B-Chat achieves benefit scores of +4.01, +383.56, +3.76, and +3.62 across the four RMs when revising Rank 6 responses. Conversely, when the initial response quality is high (GPT-4 level), the reviser still manages to bring about stable quality enhancements. Furthermore, we observed that even the 7B reviser (OpenChat-3.5-7B) could improve high-quality original responses, despite the original responses being generated by models larger than 7B. The distribution of revision labels indicates

Reviser	Original Response	RM1		RM2		RM3		RM4	
		Score	Benefit	Score	Benefit	Score	Benefit	Score	Benefit
Original Response	Rank 0	-1.52	-	64.81	-	5.08	-	3.42	-
	Rank 1	-1.85	-	35.98	-	4.79	-	3.04	-
	Rank 2	-2.24	-	-6.08	-	4.42	-	2.61	-
	Rank 3	-2.73	-	-50.74	-	3.94	-	2.16	-
	Rank 4	-3.41	-	-110.68	-	3.30	-	1.59	-
	Rank 5	-4.60	-	-227.37	-	2.18	-	0.50	-
	Rank 6	-5.95	-	-351.74	-	0.93	-	-0.64	-
	Average	-3.19	-	-92.26	-	3.52	-	1.81	-
Adaptive Reviser	Rank 0	**-1.45**	**+0.07**	**74.44**	**+9.64**	**5.17**	**+0.09**	**3.51**	**+0.10**
	Rank 1	**-1.70**	**+0.15**	**55.55**	**+19.58**	**4.95**	**+0.16**	**3.27**	**+0.23**
	Rank 2	**-1.92**	**+0.33**	**31.95**	**+38.03**	**4.75**	**+0.33**	**3.08**	**+0.48**
	Rank 3	**-2.12**	**+0.61**	13.55	+64.28	4.56	+0.62	2.94	+0.78
	Rank 4	-2.35	+1.07	-3.63	+107.05	4.38	+1.08	2.85	+1.26
	Rank 5	-2.54	+2.06	-19.76	+207.61	4.22	+2.03	2.76	+2.26
	Rank 6	-2.75	+3.20	-37.50	+314.23	4.02	+3.09	2.64	+3.28
	Average	**-2.12**	**+1.07**	**16.37**	**+108.63**	**4.58**	**+1.06**	**3.01**	**+1.20**
Preliminary Reviser	Rank 0	-2.04	-0.52	29.33	-35.48	4.68	-0.39	3.09	-0.33
	Rank 1	-2.11	-0.26	24.15	-11.82	4.61	-0.18	3.05	+0.01
	Rank 2	-2.17	+0.07	15.62	+21.70	4.54	+0.12	2.98	+0.37
	Rank 3	-2.26	+0.47	8.54	+59.27	4.48	+0.53	2.93	+0.77
	Rank 4	-2.41	+1.00	-1.59	+109.09	4.35	+1.05	2.88	+1.29
	Rank 5	-2.52	+2.08	-9.62	+217.75	4.26	+2.08	2.83	+2.32
	Rank 6	-2.66	+3.28	-25.45	+326.29	4.11	+3.19	2.73	+3.36
	Average	-2.31	+0.88	5.85	+98.11	4.43	+0.91	2.93	+1.11
Aligner	Rank 0	-2.04	-0.52	27.18	-37.63	4.66	-0.42	3.09	-0.33
	Rank 1	-2.10	-0.25	17.71	-18.26	4.60	-0.19	3.02	-0.02
	Rank 2	-2.10	+0.15	16.89	+22.96	4.58	+0.16	3.02	+0.41
	Rank 3	-2.14	+0.59	**15.71**	**+66.45**	**4.57**	**+0.63**	2.98	+0.82
	Rank 4	**-2.28**	**+1.14**	**6.44**	**+117.12**	**4.46**	**+1.16**	**2.92**	**+1.33**
	Rank 5	**-2.40**	**+2.20**	**-5.00**	**+222.36**	**4.37**	**+2.18**	2.86	+2.36
	Rank 6	**-2.48**	**+3.47**	**-12.04**	**+339.70**	**4.29**	**+3.36**	**2.82**	**+3.46**
	Average	-2.22	+0.97	9.55	+101.81	4.50	+0.98	2.96	+1.15
Label Reviser	Rank 0	-2.03	-0.52	29.38	-35.42	4.68	-0.40	3.16	-0.26
	Rank 1	-2.10	-0.24	24.53	-11.44	4.61	-0.18	3.06	+0.02
	Rank 2	-2.11	+0.14	20.29	+26.37	4.59	+0.17	3.03	+0.43
	Rank 3	-2.20	+0.53	10.30	+61.04	4.49	+0.55	**2.99**	**+0.83**
	Rank 4	-2.33	+1.08	1.09	+111.77	4.42	+1.11	**2.92**	**+1.33**
	Rank 5	-2.41	+2.19	-5.82	+221.55	4.35	+2.17	**2.89**	**+2.39**
	Rank 6	-2.55	+3.40	-18.12	+333.61	4.24	+3.32	2.79	+3.43
	Average	-2.25	+0.77	8.81	+85.06	4.48	+0.80	2.98	+1.02

Table 4: Performance comparison between our adaptive reviser and baseline methods, with Openchat-3.5-7B serving as the backbone where applicable. The highest score for each response is highlighted in bold. The specifications of RM1-RM4 are the same as in Table 3. The results show that our adaptive reviser consistently outperforms the baseline methods, especially for the top-ranked responses.

that as the original response quality increases, the proportion of `[No Revise]` labels output by the reviser also increases. This aligns with our primary goal for implementing revision labels: to ensure that revisions are made only when necessary, preserving high-quality responses as they are. On the other hand, as the quality of the initial response decreases, the proportion of `[Major Revise]` labels increases, signaling a more extensive revision effort by the reviser. The `[Minor Revise]` label appears more frequently for responses of moderate quality. This is expected because the `[Minor Revise]` label acts as a middle ground between `[No Revise]` and `[Major Revise]`.

B.3 Reward Distribution of Different Revisers

We analyze the reward distributions of the original responses generated by Zephyr-7B-SFT-full with those of the revised responses produced by adaptive revisers based on various models, including Openchat-3.5-7B, Yi-34B-Chat, and Qwen1.5-32B-Chat. This comparison is conducted

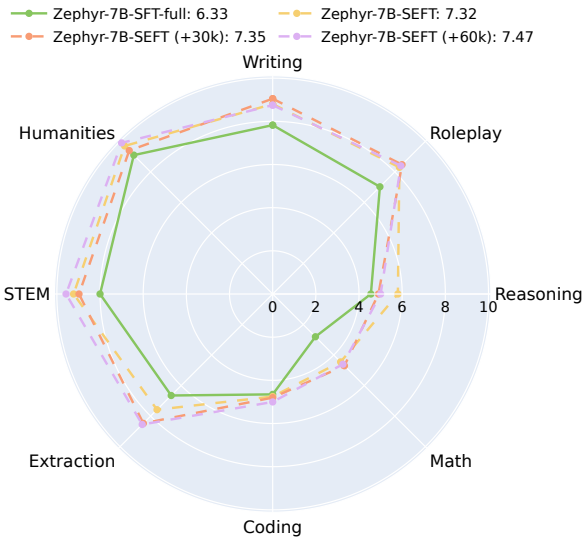

Figure 8: Comparison between Zephyr-7B-SFT-full and Zephyr-7B-SEFT on MT-Bench with varying amounts of additional data.

Hyperparameters	Preliminary Reviser	Adaptive Reviser	Internal (External) Evolution
epochs	3	3	3
max-length	2048	2048	2048
per-device-train-batch-size	2	2	4
gradient-accumulation-steps	8	8	4
learning-rate	2e-5	2e-5	5e-7
LR-scheduler-type	cosine	cosine	linear
LR-warmup-ratio	0.03	0.03	0.03
weight-decay	0.0	0.0	0.0
gradient-checkpointing	TRUE	TRUE	TRUE
seed	42	42	42
zero-stage	3	3	3
optimizer	AdamW	AdamW	AdamW
bf16	TRUE	TRUE	TRUE
tf32	TRUE	TRUE	TRUE
dataset-size	53K	124K	61K, 91K, 121K

Table 5: Hyperparameters of our training in different stages.

across three test sets: Nectar, UltraFeedback, and OpenHermes-2.5. The results in Figure 7 reveal a consistent improvement in the reward distributions for revised responses across these test sets. Furthermore, the magnitude of this shift tends to correlate with the scale of the reviser models, indicating that larger models are more effective at enhancing response quality.

B.4 Results of Different Reviser Baselines

The experimental results in Table 4 highlight the exceptional performance of the adaptive reviser compared to baseline methods. Across all reward models (RM1-RM4), the adaptive reviser achieves the highest average benefit scores, indicating its overall superiority. Notably, the adaptive reviser shows significant improvements over the original responses, particularly for lower-quality inputs, with greater benefit scores as the initial response quality declines. Furthermore, for high-quality original responses, the adaptive reviser consistently outperforms other revisers. For instance, under RM2, the adaptive reviser achieves benefit scores of +9.64 and +19.58 for Rank 0 and 1, respectively, while other methods show considerably lower or even negative scores. These results demonstrate that the adaptive reviser excels in refining both high and low-quality responses.

C Policy Evaluation on MT-Bench

Figure 8 shows consistent improvements in Zephyr-7B-SEFT's scores across various categories on MT-Bench as the volume of additional data increases. Specifically, the model with the 60K additional prompts demonstrates significant enhancements in categories such as *Extraction* and *STEM*, highlighting its improved capability to manage complex and diverse language tasks. The results clearly indicate that as more unannotated data is integrated, the model's performance not only becomes more robust across all categories but also significantly outperforms the baseline Zephyr-7B-SFT-full model.

D Implementation Details

The hyperparameters for our implementations are listed in Table 5. For the label reviser, we utilize three revision labels: `[No Revise]`, `[Major Revise]`, and `[Minor Revise]`. Specifically, 10% of the Nectar training samples with original responses ranked at 0 are labeled as `[No Revise]`. For the remaining samples, those with original response rankings between 1 and 3 receive the `[Minor Revise]` label, while all other samples are assigned the `[Major Revise]` label.

E Case Study

In this subsection, we provide examples of prompts and original responses from the Nectar training set. We illustrate how revision labels are assigned by comparing the scores given by the Starling RM to the original responses and those revised by our preliminary reviser. This comparison highlights the differences in response quality that guide the assignment of `[No Revise]`, `[Minor Revise]`, and `[Major Revise]` labels for the adaptive reviser.

Prompt	Human: Been listening to a lot of Aretha Franklin lately and really enjoying it. Do you know any female singing artists that are like her?\n \n Assistant: I've heard people mention Nina Simone, Ella Fitzgerald, and Patsy Cline.\n \n Human: Nina and Ella seem like good suggestions. Isn't Patsy Cline a bit country and western though?\n \n Assistant:
Original Response	She was known as a Country singer, but she also did some more traditional standards.
Revised Response	Yes, you're correct. Patsy Cline is often associated with country and western music. However, she did have a powerful and soulful voice, which might be similar to Aretha Franklin's in terms of emotional depth. If you're specifically looking for female singers with a soul and R&B style similar to Aretha Franklin, you might also enjoy artists like Etta James, Dionne Warwick, and Gladys Knight.
Scored by Starling RM	Original Response: -0.5859375; Revised Response: 5.84375
Revision Label	`[Major Revise]`

Table 6: A prompt and the original response from the Nectar test set, along with a revised response by our preliminary reviser. The revised response receives a significantly higher score ($s_i > 1.0$) than the original response, as evaluated by Starling RM. This substantial improvement demonstrates that the original response was easy for the preliminary reviser to enhance significantly. As a result, the revision label for this instance is assigned as `[Major Revise]`, demanding a substantial revision.

Prompt	Human: How do I tell my kid not to lie? \n \n Assistant:
Original Response	Here are a few tips for talking to your child about lying: \n \n 1. Explain the consequences: Let your child know that lying can have serious consequences, such as losing trust and damaging relationships. \n \n 2. Be a role model: Children often learn by example, so make sure you are setting a good example by being honest and truthful yourself. \n \n 3. Encourage honesty: Praise your child when they tell the truth, even if it's something you don't want to hear. This will reinforce the importance of honesty. \n \n 4. Talk about the difference between truth and lies: Help your child understand that there is a difference between telling the truth and telling a lie. Explain that telling the truth is always the right thing to do. \n \n 5. Discuss the reasons behind lying: Sometimes children lie because they are afraid of getting in trouble or they want to avoid disappointing someone. Talk to your child about these reasons and help them find alternative ways to handle these situations. \n \n 6. Set clear expectations: Let your child know that lying is not acceptable and that there will be consequences if they are caught lying. \n \n 7. Listen to your child: If your child does lie, try to understand why they did it. Listen to their side of the story and use it as an opportunity to teach them about honesty and trust. \n \n Remember to be patient and consistent when talking to your child about lying. It may take time for them to understand the importance of honesty, but with your guidance, they will learn to value the truth.
Revised Response	1. Lead by example: Children learn by observing their parents and caregivers. Make sure you are honest and truthful in your own actions and words. \n \n 2. Explain the consequences: Talk to your child about the negative consequences of lying, such as losing trust and damaging relationships. \n \n 3. Encourage open communication: Create a safe and open environment for your child to talk to you about anything without fear of punishment or judgment. \n \n 4. Teach the difference between right and wrong: Help your child understand the difference between telling the truth and telling a lie. \n \n 5. Praise honesty: When your child tells the truth, acknowledge and praise their honesty to reinforce positive behavior. \n \n 6. Address the root cause: If your child is lying, try to understand the reason behind it. Are they afraid of getting in trouble? Are they trying to impress others? Addressing the underlying issue can help prevent future lying. \n \n 7. Set clear expectations: Make it clear that lying is not acceptable and there will be consequences for doing so. \n \n 8. Use age-appropriate examples: Use stories or examples that your child can relate to in order to illustrate the importance of honesty. \n \n 9. Be a good listener: Sometimes, children lie because they feel unheard or ignored. Make sure to actively listen to your child and give them a chance to express themselves. \n \n 10. Seek professional help if needed: If your child continues to struggle with lying despite your efforts, it may be helpful to seek guidance from a child psychologist or therapist.
Scored by Starling RM	Original Response: 3.453125; Revised Response: 3.984375
Revision Label	`[Minor Revise]`

Table 7: A prompt and the original response from the Nectar test set, along with a revised response by our preliminary reviser. The revised response receives a slightly higher score ($1.0 > s_i = 0.53125 > 0.3$) than the original response, as evaluated by Starling RM. This improvement indicates that the preliminary reviser can enhance the original response, although with a little bit of difficulty. Therefore, with a probability of 0.8, the revision label is assigned as `[Minor Revise]`, signifying a minor revision.

Prompt	Human: What's a Greek mythology that is about evil parents creating a good child (like NAME_1?)\n \n Assistant:
Original Response	There are several Greek myths involving evil parents creating a good child. Here are a few examples: \n \n 1. The story of Persephone: In this myth, Hades, the god of the underworld, falls in love with Persephone, the daughter of Demeter, the goddess of agriculture. Hades kidnaps Persephone and takes her to the underworld, where she becomes his queen. Demeter is grief-stricken and refuses to let the earth produce any crops until her daughter is returned. Eventually, a deal is made between Hades and Demeter that allows Persephone to return to the world above, but she must spend part of each year in the underworld with Hades, which is why the seasons change. \n 2. The story of Dionysus: In this myth, the goddess Semele, who is the mother of Dionysus, is killed by her husband, King Cadmus, after he becomes jealous of her affair with Zeus. Zeus, who is the father of Dionysus, saves his son by placing him in his thigh, where he grows to maturity. Dionysus is known for his role in promoting fertility and the growth of crops, and his birth is seen as a symbol of the cycle of life and death. \n 3. The story of Hercules: In this myth, Hercules' parents, Zeus and Alcmene, are not evil, but Hercules' stepfather, Hera, is often depicted as a villainous figure. Hera is jealous of Hercules' divine parentage and seeks to harm him throughout his life. Despite this, Hercules is known for his heroism and his role in protecting the world from various monsters and evils. \n These are just a few examples of Greek myths involving evil parents creating a good child. There are many other myths that explore similar themes, and the specific details of each myth can vary depending on the source and interpretation.
Revised Response	One Greek myth that fits this description is the story of Persephone. In this myth, Persephone was the daughter of Demeter, the goddess of agriculture, and Zeus, the king of the gods. Despite her divine heritage, Persephone was kidnapped by Hades, the god of the underworld, and forced to become his wife. However, Persephone was able to maintain her innocence and purity, and she eventually became a beloved figure in Greek mythology known for her kindness and compassion.
Scored by Starling RM	Original Response: 4.25; Revised Response: 2.1875
Revision Label	`[No Revise]`

Table 8: A prompt and the original response from the Nectar test set, along with a revised response by our preliminary reviser. The revised response receives a much lower score ($s_i < 0.3$) than the original response, as evaluated by Starling RM. This means that the original response is of high quality and is difficult for the preliminary reviser to enhance. Therefore, with a probability of 0.8, the revision label for this instance is assigned as `[No Revise]`, indicating that no revision is needed.