Aligning Large Language Models by On-Policy Self-Judgment

Anonymous ACL submission

Abstract

To align large language models with human preferences, existing research either utilizes a separate reward model (RM) to perform on-policy learning or simplifies the training procedure by discarding the on-policy learning and the need for a separate RM. In this paper, we present a novel alignment framework, SELF-JUDGE that is (1) on-policy learning and 2) parameter efficient, as it does not require an additional RM for evaluating the samples for on-policy learning. To this end, we propose Judge-augmented Supervised Fine-Tuning (JSFT) to train a single model acting as both a policy and a judge. Specifically, we view the pairwise judgment task as a special case of the instruction-following task, choosing the better response from a response pair. Thus, the resulting model can judge preferences of on-the-fly responses from current policy initialized from itself. Experimental results show the efficacy of SELF-JUDGE, outperforming baselines in preference benchmarks. We also show that self-rejection with oversampling can improve further without an additional evaluator.

1 Introduction

Research on aligning Large Language Models (LLMs) with human preference has increasingly gained attention in recent years (Askell et al., 2021; Ouyang et al., 2022; Bai et al., 2022a; Rafailov et al., 2023). Reinforcement Learning from Human Feedback (RLHF) is the most dominant approach for the alignment of LLMs for human preference (Ziegler et al., 2020). It utilizes a reward model (RM) to estimate human preference scores for the generated responses from policy. The policy is updated with *on-policy* Reinforcement Learning (RL) to maximize the estimated rewards of sampled responses regularized with the KL divergence between the current policy and reference policy. However, RLHF requires a complex setup for on-policy updates because of the simultaneous use of an RM with the reference model.

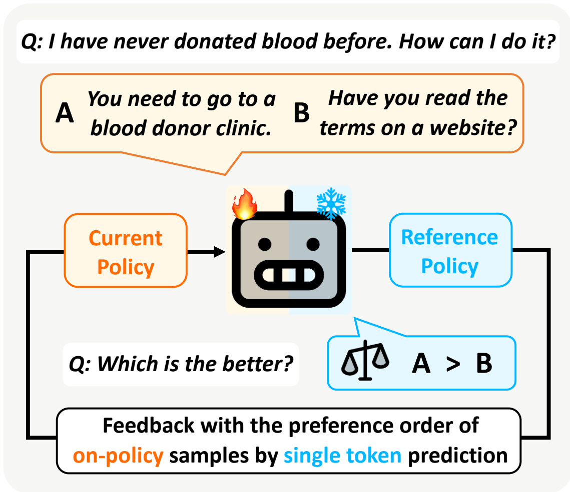

Figure 1: In our framework, SELF-JUDGE, a single model is trained not only to generate responses but also to perform a judgment task, where it selects the better from two responses through a single token prediction. This enables on-policy self-training by performing judgments on current policy initialized from itself.

In contrast, a line of research (Rafailov et al., 2023; Azar et al., 2023) proposes discarding on-policy updates and optimizing from preference orders in static datasets without RMs (i.e., *offline* learning). Similarly, other studies propose constructing datasets with responses sampled from the policy and labeled by separated evaluators (i.e., *off-policy* learning) (Zhao et al., 2023; Liu et al., 2023; Xu et al., 2023; Gulcehre et al., 2023; Dong et al., 2023). However, offline learning can lead to suboptimal results due to the lack of exploration (Mediratta et al., 2023), and off-policy learning could cause degeneration if an appropriate replay buffer strategy is not used (Zhang and Sutton, 2017).

In this paper, we present **SELF-JUDGE**, a simplified *on-policy* training scheme that trains a *single* model to perform on-the-fly feedback for updated policy initialized from *itself*. To this end, we introduce a Judge-augmented Supervised Fine-tuning (JSFT) to obtain a model that can both generate a re-

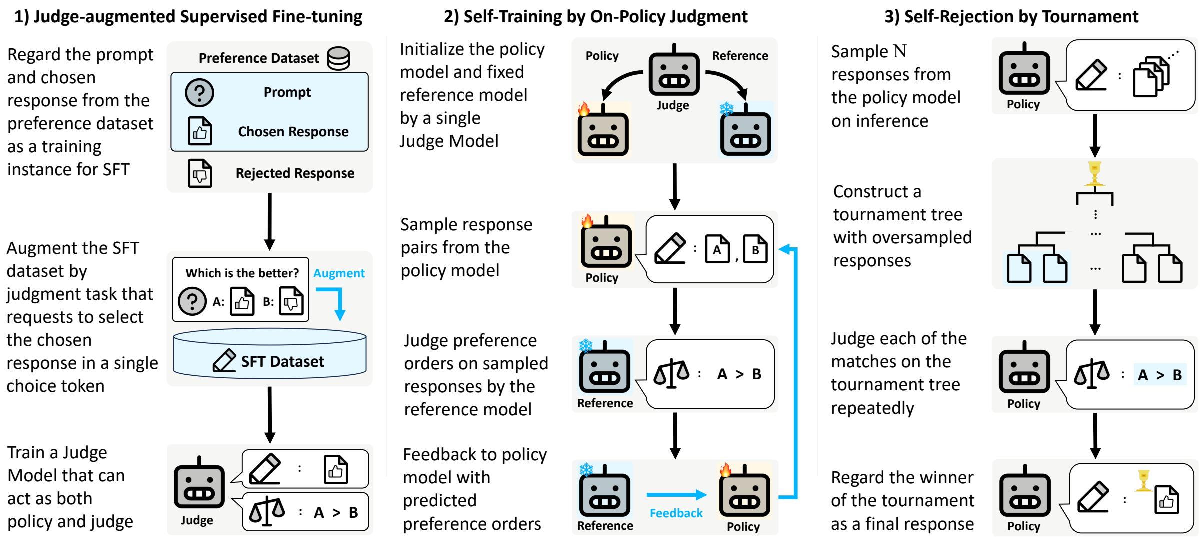

Figure 2: An overview of SELF-JUDGE. 1) We train a LLM to act as a Judge Model (JM), which can both generate responses and compare response pairs. We train the JM with a SFT dataset augmented with the judgment task where the better response can be selected by a single token. 2) We initialize a policy and a fixed reference mode from the trained JM. Then, the policy model samples response pairs, and the reference model performs judgments on the pairs for giving feedback with preference orders. 3) We perform a rejection sampling by a tournament on responses from the resulting policy model through the judgments by itself for further improvements on inference time.

sponse and perform pairwise comparison, as shown in Figure 1. Specifically, we regard the pairwise judgment task, choosing the better response from a response pair, as a special case of the instruction-following task, which can be answered in a *single token* and optionally with a rationale. SELF-JUDGE initializes the current policy and reference policy by the resulting model. Then, it samples a response pair from the current policy and chooses a better response in the pair by the reference policy as a judge for updating the current policy with preference orders without an RM (Rafailov et al., 2023; Zhao et al., 2023; Azar et al., 2023).

Experimental results show that SELF-JUDGE outperforms RLHF and other offline and off-policy approaches (Rafailov et al., 2023; Liu et al., 2023; Gulcehre et al., 2023; Dong et al., 2023) on preference benchmarks. Unlike existing approaches, SELF-JUDGE leverages on-policy sampling while not introducing an additional evaluator. These results demonstrate the effectiveness and parameter efficiency of SELF-JUDGE performing on-policy *self-training* for LLM alignment. Furthermore, we show that SELF-JUDGE can maximize performance through the *self-rejection* that selects the best response from its own responses using its judgment capabilities learned through JSFT. In particular, the performance gains are significant when the pairwise judgment task of JSFT involves comparisons based on principles with rationale for the decision.

In summary, our main contributions are:

- We propose a parameter-efficient on-policy alignment framework, SELF-JUDGE, introducing JSFT for the initial policy that can judge.
- We analyze the efficacy of the JSFT for judgment and suggest the best practices to exploit the improved judgment ability.
- We show resulting models from JSFT can self-improve by acting as a judge: on-policy self-training and self-rejection at inference time.

2 Preliminaries

Aligning by Reinforcement Learning To align LLMs with human preferences, RLHF (Ziegler et al., 2020) follows three stages: 1) obtaining an initial policy π_{ref} through SFT from the pre-trained LLM, 2) training an RM from human preference triplets (x, y_w, y_l), where y_w is a chosen response, and y_l is a rejected response for a given prompt x, 3) fine-tuning the initial policy by Proximal Policy Optimization (PPO) (Schulman et al., 2017), with KL divergence between current policy π and reference policy π_{ref} as a regularization for the reward maximization objective. Generally, RMs are trained by Bradley-Terry model (Bradley and Terry, 1952), which minimizes negative log-likelihood of score difference, $\log\sigma(r_\theta(x, y_w) - r_\theta(x, y_l))$, to compute a pointwise scalar reward of response sampled from the current policy. This *on-policy* rollout procedure provides the frequent *exploration* of responses but introduces an additional training stage and memory usage during policy updates for RMs.

Aligning from Preference Orders It has been observed that RM is easily susceptible to language biases such as the response length, preferring longer responses over shorter ones (Shen et al., 2023). Askell et al. (2021) suggest a language modeling loss on preferred context (x, y_w) during the training of the RM to address this issue. This approach, however, necessitates another pre-training stage of the RM. Direct Policy Optimization (DPO) (Rafailov et al., 2023) and Sequence Likelihood Calibration with Human Feedback (SLiC-HF) (Zhao et al., 2023) introduce objectives that can be trained by only preference orders $y_w \succ y_l$ without scalar rewards and need for RMs. However, *offline* learning methods optimized on static datasets inherently reach sub-optimal results compared to *online* learning (Mediratta et al., 2023). Xu et al. (2023); Yuan et al. (2024) propose an online learning approach that iteratively constructs datasets by responses sampled from the policy. However, these *off-policy* approaches with a large buffer size can induce performance degeneration (Zhang and Sutton, 2017).

3 SELF-JUDGE

In this section, we describe our proposed alignment framework, **SELF-JUDGE**, which utilizes a *single* model that acts as both policy and judge: sampling responses and judgment over response pairs. It can perform feedback on the response pairs sampled from current policy initialized from itself as an *on-policy* manner and also perform rejection sampling by itself, as depicted in Figure 2.

3.1 Judge Model

Using LLMs to evaluate responses of another LLM, *LLM-as-a-judge*, is shown promising (Zheng et al., 2023; Bai et al., 2022b; Kim et al., 2023). Inspired by these studies, we leverage the generative pairwise evaluator, which we refer to as **Judge Model (JM)** for aligning LLMs with human preference (Ethayarajh et al., 2023; Zhao et al., 2023; Liu et al., 2023). Unlike RM, which produces a scalar score for a single response, JM simply chooses a better response between the two responses. Specifically, the JM π is trained by maximizing the log-

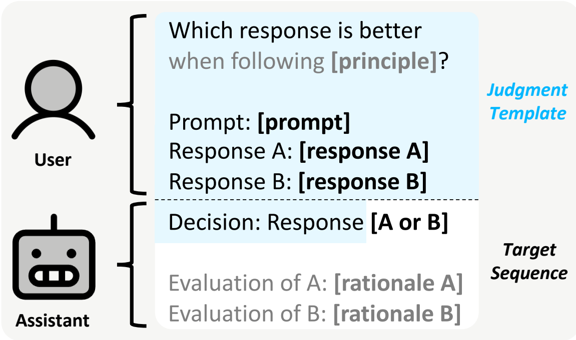

Figure 3: An example of judgment template $\mathcal{C}$. The judgment template asks which of two responses is better on a given prompt and requests to complete the choice token $\mathcal{J} \in \{\mathcal{A}, \mathcal{B}\}$ corresponding to the better response. Optionally, it is possible to change requests to match the specific principle for judgment, and the rationale can be included in the target sequence for training.

likelihood of the choice token $\mathcal{J}$ corresponding ground-truth preference on a given judgment template $\mathcal{C}$, typically a single token such as 'A' or 'B' to predict the better response efficiently. Figure 3 illustrates an example of the judgment template.

Judge-augmented Supervised Fine-tuning Typically, the chosen response y_w is used for target sequence of prompt x in training preference dataset $\mathcal{D}$ on the SFT stage, i.e., $\mathcal{D}_{\text{SFT}} = \{(x, y_w) | (x, y_w, y_l) \in \mathcal{D}\}$. In addition, we treat the pairwise judgment task as a special case of instruction-tuning, e.g., $\mathcal{D}_{\text{Judge}} = \{(\mathcal{C}(x, y_w, y_l), \mathcal{J}) | (x, y_w, y_l) \in \mathcal{D}\}$, where $\mathcal{J} \in \{\mathcal{A}, \mathcal{B}\}$. As a result, we train the pairwise judgment task as a response generation using the augmented dataset $\mathcal{D}^{+}_{\text{SFT}} = \mathcal{D}_{\text{SFT}} \cup \mathcal{D}_{\text{Judge}}$. We refer to the process of fine-tuning the LLMs on the augmented dataset $\mathcal{D}^{+}_{\text{SFT}}$ as **Judge-augmented Supervised Fine-tuning (JSFT)**. With JSFT, we can obtain a JM that can not only compare pairwise preferences but also generate responses. Also, we can expect a better understanding of language-relevant features on preference comparison since the JM is trained to mimic *good* behavior y_w similar to the observation found on RMs (Askell et al., 2021).

Principle-aware Judgment with Rationale One advantage of JM compared to a conventional RM is that the JM can adapt the judgment template $\mathcal{C}$ according to predict multiple aspects of human preference reflecting diverse principles (Sun et al., 2023; Cui et al., 2023). Also, the generative nature of JM makes it easy to utilize rationale, a justification of the judgment in natural language, for more precise judgments similar to observations on instruction-tuning of LLMs (Mukherjee et al., 2023). For a given principle $p \in \mathcal{P}$ where the $\mathcal{P}$ is a principle set, we devise different judgment templates $\mathcal{C}_p$ for each p. Also, we include a rationale $\mathcal{R}$ in the target sequence for the judgment task when it is available. The JM first produces a choice token $\mathcal{J} \in \{\mathcal{A}, \mathcal{B}\}$ which corresponds to the choice label of better response, then rationale $\mathcal{R}$ is followed. Formally, $\pi(\mathcal{J}, \mathcal{R} \mid \mathcal{C}_p(x, y_w, y_l))$. As a result, principle-aware judgment can be conducted by simply adjusting the judgment template.

Judging Self-Generated Responses Since JM is trained by JSFT, it can judge its own response by playing the roles of the policy and judge: this is the intuition behind the name **SELF-JUDGE**. As a policy, JM first samples responses y_a, y_b for a given x. Then, the JM, as a judge, chooses a better response between the two responses, $\pi(\mathcal{J} \mid \mathcal{C}(x, y_a, y_b))$. More precisely, it averages the likelihoods of corresponding choice tokens, $\mathcal{J}$, utilizing a position-swapped judgment template to mitigate position bias (Chiang et al., 2023). From the relative likelihoods of these choice tokens for each response, a pseudo-preference triplet label $(x, \hat{y}_w, \hat{y}_l)$ for the *self-training* and *self-rejection* can be deduced. On principle-aware judgments, we can check the winning rate across principles or the mean likelihood of choice tokens across principles when a tie happens.

3.2 Self-Training by On-policy Judgment

Ethayarajh et al. (2023) adopt JM's predicted likelihood of label tokens as the reward for on-policy evaluation in RLHF framework, comparing with a *blank* baseline response [1]. However, this approach has significant limitations; the likelihood inferred by a language model cannot be regarded as confidence without calibration (Zhou et al., 2023; Zhu et al., 2023). Hence, we adopt objectives that perform optimizations on preference orders (Rafailov et al., 2023; Zhao et al., 2023; Azar et al., 2023) to leverage JMs properly since the objectives do not require pointwise scalar scores. In our framework, we regard the reference policy π_{ref} as a judge to evaluate the preference order between generated samples from current policy π_θ, as follows:

$$
\begin{aligned}
&y_a \sim \pi_\theta(\cdot \mid x), y_b \sim \pi_\theta(\cdot \mid x),\\
&(x, \hat{y}_w, \hat{y}_l) \leftarrow \pi_{\text{ref}}(\mathcal{J} \mid \mathcal{C}(x, y_a, y_b)).
\end{aligned}
$$

[1] huggingface.co/stanfordnlp/SteamSHP-flan-t5-xl

That is, a single JM with JSFT is initialized for both π_θ and π_{ref}, but the latter one is frozen for the likelihood normalization (Liu et al., 2023) and on-policy judgments. This setup enjoys *on-policy* training scheme without introducing an additional model for the evaluation as RLHF. If we adopt the DPO objective, the following is used for optimization, where σ is a sigmoid function, and β is a coefficient for KL divergence regularization,

$$
-\mathbb{E}\big[\log \sigma\big(\beta \log \frac{\pi_\theta(\hat{y}_w|x)}{\pi_{\text{ref}}(\hat{y}_w|x)} - \beta \log \frac{\pi_\theta(\hat{y}_l|x)}{\pi_{\text{ref}}(\hat{y}_l|x)}\big)\big].
$$

3.3 Self-Rejection by Tournament

RMs can also be utilized at inference time for rejection sampling such as *Best-of-N* sampling (Stiennon et al., 2020), selecting the best response among oversampled N responses according to the reward score of each response. However, this approach requires a separate RM in addition to the policy model during inference. In contrast, JM trained within SELF-JUDGE does not require an additional RM for evaluating responses, as it can evaluate the self-generated responses by itself.

However, judging all the possible comparison pairs to get average winning rates requires $\mathcal{O}(N^2)$ forward processes for $\binom{N}{2}$ comparisons. Thus, we adopt the *tournament* (Zhao et al., 2023; Liu et al., 2023) for rejection sampling on inference time. We construct a tournament tree whose leaf nodes are sampled responses, and the non-leaf nodes are chosen by the winner on judgment between the child nodes. Since the tournament tree has less than $N-1$ non-leaf nodes, we can find the best response by $\mathcal{O}(N)$ forward processes, the same as the Best-of-N sampling with a separated RM.

4 Experimental Setups

In this section, we discuss the experimental setup for validating our proposed framework, SELF-JUDGE. We use two datasets: Anthropic-HH (Yuan et al., 2023) and UltraFeedback (Cui et al., 2023). In the experiments, we choose DPO (Rafailov et al., 2023), RSO (Liu et al., 2023), ReST (Gulcehre et al., 2023), RAFT (Dong et al., 2023), and RLHF (Ziegler et al., 2020) as baselines for comparing with SELF-JUDGE. We evaluate the resulting models from each experiment by AlpacaEval (Li et al., 2023), VicunaEval (Chiang et al., 2023), and MT-Bench (Zheng et al., 2023). The implementation details are in Appendix A.

Method	Policy	Evaluator	On-Policy Sampling	On-Memory Parameters	AlpacaEval (% Win)	VicunaEval (% Win)	MT-Bench (Score)
SFT	✗	✗	✗	p	24.75	50.00	4.63
DPO	SFT	✗	✗	$2p$	<u>35.14</u>	60.63	<u>4.73</u>
RSO	SFT	JM	✗	$2p$	34.27	<u>64.38</u>	4.42
ReST	SFT	RM	✗	p	27.43	55.00	4.53
RAFT	SFT	RM	✗	p	32.50	59.38	4.43
RLHF	SFT	RM	✓	$3p$	33.46	53.75	4.29
SELF-JUDGE (Ours)	JM		✓	$2p$	**44.88**	**76.25**	**4.80**

Table 1: Evaluation results of models trained on HH-Helpful (Bai et al., 2022a). The best result and second best result on each benchmark are represented as bold and underline. We report theoretical memory usage on training for model parameters required for each method where p denotes the number of parameters on the base model. We use `base` and `online` splits but used the `online` split only for constructing the training instances of SFT. SELF-JUDGE outperforms baselines on all benchmarks with a single JM, which can act as both policy and judge.

4.1 Datasets

Anthropic-HH Anthropic-HH (Bai et al., 2022a) is a human preference dataset on 170k dialogues, which consists of two subsets, HH-Helpful and HH-Harmless, which are labeled by the helpfulness and harmlessness principle. We focus on the HH-Helpful to better isolate and understand the benefits of SELF-JUDGE due to the conflicting nature of helpfulness and harmlessness principles (Bai et al., 2022a). HH-Helpful contains various data splits corresponding to the development stages of an AI assistant. We use the `base` split and include the pair (x, y_w) from the `online` split to the SFT dataset for analyzing transition effects on JSFT.

UltraFeedback As obtaining human-labeled feedback is costly, utilizing AI feedback as an alternative is widely investigated (Bai et al., 2022b; Sun et al., 2023; OpenAI, 2023). UltraFeedback (Cui et al., 2023) is one of the datasets that consists of AI feedback where GPT-4 (OpenAI, 2023) rates responses obtained from four different language models for the 64k prompts, based on four principles of helpfulness, instruction-following, truthfulness, and honesty. Each rating contains the rationale obtained from GPT-4, which represents an explanation for the quality and rating of the corresponding response based on the given principle. We randomly split 10% of the prompts of the dataset and use them as a test set for further analysis.

4.2 Baselines

We choose **DPO** (Rafailov et al., 2023) as an offline learning baseline and use DPO objective for self-training in SELF-JUDGE. We include **RSO** (Liu et al., 2023) with the DPO objective as an off-policy learning baseline, which utilizes the JM as a separate evaluator. We include baselines which utilize RMs, **ReST** (Gulcehre et al., 2023) and **RAFT** (Dong et al., 2023) for off-policy approach and **RLHF** (Ziegler et al., 2020) for on-policy approach. We choose LLaMa-2-7B (Touvron et al., 2023) as a base model for all experiments.

4.3 Evaluations

We evaluate the resulting models based on the three benchmarks, AlpacaEval (Li et al., 2023), VicunaEval (Chiang et al., 2023), and MT-Bench (Zheng et al., 2023). **AlpacaEval** is an alignment benchmark that compares the quality of two responses on 805 questions sampled from a diverse dataset, rated by GPT-4 (OpenAI, 2023). We use the `text-davinci-003` (Ouyang et al., 2022) as the baseline models for measuring winning rates. **VicunaEval** is another benchmark utilizing GPT-4 as a judge for comparing two models' responses from 80 questions on various topics. We use the SFT model on each dataset as the baseline model for measuring winning rates. **MT-Bench** is a multi-turn benchmark consisting of 80 instances of 2-turn questions from 8 different domains, which is evaluated by GPT-4 on a scale of 1 from 10. We report the average scores obtained on each turn.

5 Experimental Results

5.1 Main Results

SELF-JUDGE is a strong alignment method. In Table 1, we can see that SELF-JUDGE outperforms all baselines. Notably, SELF-JUDGE consistently shows the highest performance on all three benchmarks while not introducing additional parameters compared to DPO, which is trained in an offline setting. In addition, SELF-JUDGE also shows significant strength compared to all of the baselines utilizing separate evaluators for off-policy and on-policy optimization (Liu et al., 2023; Gulcehre et al., 2023; Dong et al., 2023; Ziegler et al., 2020). Qualitative examples can be found in Appendix B.

5.2 Analysis of Judge Models

We conduct an ablation study with HH-Helpful to verify two hypotheses about SELF-JUDGE: 1) JSFT improves JM's judgment ability, and 2) JM's judgment ability can be fully utilized through on-policy learning with direct optimization on preference orders. To this end, we train three JMs with different strategies: 1) training solely on judgment task induced from `base` split, 2) JSFT on `base` split, and 3) JSFT on `base` split with additional SFT data from `online` split. We report the performance of each JM as a policy and a judge. We further examine the resulting models on RLHF regarding the predicted likelihood of label tokens on JM as rewards similar to Ethayarajh et al. (2023). We also investigate how different sampling strategies on self-training influence the final performance.

JSFT boosts the performance as a judge. Table 2 shows the results regarding the first hypothesis. We can see that JM has only a marginal difference in prediction accuracy on the test split of HH-Helpful compared to RM when it is not trained by JSFT. From this observation, we confirm that the transition effects of imitation learning to judgment task occur when judgment task is trained with canonical SFT task. On the other hand, including the `online` split for JSFT significantly improves the performance as a policy, but performance as a judge is slightly decreased compared to the JM that only trained on the `base` split. We conjecture that the distribution gap between `online` split and `base` split influences the transition effects, as `online` split is obtained from a language model already aligned with human preferences.

JM is more compatible with direct preference optimization than RL. In Table 3, we observe that RLHF with JM as an evaluator results in a lower performance compared to conventional RM producing scalar rewards, which implies that the choice token likelihood obtained from a judgment could not be regarded as a pointwise preference score on the Bradley-Terry model (Bradley and Terry, 1952). Also, Table 4 shows that on-policy learning outperforms offline and off-policy approaches using a single JM on self-training. Therefore, we can conclude that JM is more compatible with direct optimization on preference orders, such as DPO objective in an on-policy manner, rather than RL approaches such as PPO or offline and off-policy approaches in order to leverage JM's superior performance on judgments.

Type	JSFT (+ $\mathcal{D}_{\text{SFT}}$)	Judge (% Accuracy)	Policy (% Win)
RM	✗	66.11	✗
JM	✗	66.49	✗
	+ `base`	**68.32**	5.78
	+ `base` / `online`	67.84	**20.26**

Table 2: Prediction accuracy on test split of HH-Helpful and winning rates on AlpacaEval using JM as a judge or a policy. JSFT improves not only the performance as a policy but also the performance as a judge of JM.

Policy	Evaluator	JSFT (+ $\mathcal{D}_{\text{SFT}}$)	AlpacaEval (% Win)
SFT	RM	✗	**33.46**
	JM	✗	26.18
		+ `base`	29.63
		+ `base` / `online`	28.46

Table 3: Prediction accuracy with different evaluator types. With JMs, we regard the likelihood of choice token comparing with chosen response y_w as a reward[2]. JM's token likelihoods are not appropriate for the pointwise reward function in RLHF.

5.3 Effects of Principle and Rationale

In this subsection, we compare JMs trained by three different JSFT strategies to verify the effect of principle-aware judgment and rationale through UltraFeedback (Cui et al., 2023). The three different strategies include 1) judgment derived from overall scores across the principles, 2) principle-aware judgments, and 3) principle-aware judgments with rationales. We check the performance of each JM as a policy and a judge. We also examine how principle-aware judgment and rationale affect the performance as a judge after a self-training stage. We further check the statistics of selected responses on self-rejection according to the sampled number

[2] We found that using the *blank* response fails on the assessment of response since the reward is always close to 1.

Sampling	AlpacaEval (% Win)	VicunaEval (% Win)	MT-Bench (Score)
Offline	28.57	58.75	4.68
Off-Policy	32.03	64.38	4.59
On-Policy	**44.88**	**76.25**	**4.80**

Table 4: Effect of sampling strategy on self-training. On-policy learning yields the best performance.

of responses. Additionally, we investigate the feasibility of iterative training, which regards the JM after the self-training as the initial policy and conducts several iterations of self-training.

Type	P	R	Judge (% Accuracy)	Policy (% Win)
RM	✗	✗	79.9	✗
JM	✗	✗	80.5	67.4
JM-P	✓	✗	81.6	59.3
JM-PR	✓	✓	**84.0**	65.7

Table 5: Evaluation results of performance as a judge or a policy by test split of UltraFeedback and the AlpacaEval according to the usage of principle (**P**) and rationale (**R**) for pairwise judgment task. Principle-aware judgment with rationale (JM-PR) boosts the performance as a judge while slightly sacrificing the ability as a policy.

Involving principles and rationale improves performance as a judge but not as a policy. Table 5 shows that the performance as a judge increases when the JM is trained for principle-aware judgment (JM-P) and further increases when the rationale is also used for training (JM-PR). This confirms that the judgment task can be treated as instruction, inheriting the benefits of instruction-tuning. However, we observe that degeneration in performance as a policy occurs when JM is trained solely on principle-aware judgment but recovered when JM is trained with rationale. We presume this trade-off comes from the bias on task distribution caused by increased judgment task instances for training principle-aware judgment. However, we speculate that rationale can mitigate performance degradation caused by response distribution mismatch between two tasks.

JM-PR excels in self-improvement. In Table 6, we can observe that JM-PR achieves comparable performance on benchmarks even though JM-PR shows the degradation as an initial policy model. When the self-rejection is applied on JM-PR, the winning rate on AlpacaEval is reliably improved up to 8.7% with shorter response lengths and fewer repetitions compared to JM as Figure 4 even LLM-as-a-judge is likely to prefer longer responses (Zheng et al., 2023). We further experiment with an iterative training scheme that regards the JM-PR after self-training as an initial policy. We observe that the performance as a policy improves while the capability as a judge is maintained, as shown in Figure 5. We find that the increase in performance as a policy diminishes with more iterations, but the performance degeneration as an initial policy can be overcome with iterative training.

Method	AlpacaEval (% Win)	MT-Bench (Score)
SFT	68.99	5.08
DPO	80.25	5.74
RSO	77.67	5.69
ReST	72.95	5.47
RAFT	71.08	5.32
RLHF	71.68	5.40
Self-Judge (JM)	**80.75**	6.00
+ *Self-Rejection* ($N = 16$)	84.47	6.08
Self-Judge (JM-PR)	79.98	**6.12**
+ *Self-Rejection* ($N = 16$)	88.39	6.14

Table 6: Evaluation results of models trained on UltraFeedback by AlpacaEval and MT-Bench. Note that applying *self-rejection* on resulted models from Self-Judge does not require a separate evaluator.

6 Related Work

Learning from Preference Scores There are several approaches utilizing an RM for alignment learning without RL, as opposed to RLHF, which utilizes an RM for on-policy learning (Ziegler et al., 2020). RRHF (Yuan et al., 2023) maximizes the margin of log-likelihood by the rank of responses determined by the score from RM and human annotators. RAFT (Dong et al., 2023) and ReST (Gulcehre et al., 2023) apply rejection sampling on sampled responses through the RM to perform self-imitation learning. SALMON (Sun et al., 2023) trains the LLM to generate scores for responses through principle-driven synthetic preference data utilizing the SFT model. However, all these approaches require a separate RM for the alignment procedure, unlike our work.

Optimizing on Preference Orders From preference orders in the static dataset, DPO (Rafailov

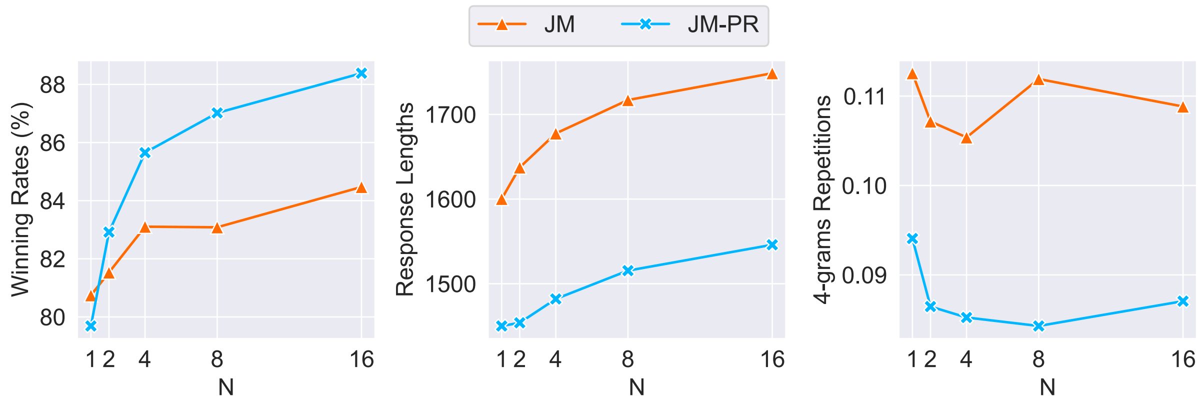

Figure 4: Tendency of winning rates, average response lengths, and 4-gram repetitions on AlpacaEval according to the number of sampling (N) for self-rejection on JM and JM-PR after the self-training. Even though LLM-as-a-judge tends to favor verbose responses (Zheng et al., 2023), JM-PR reliably improves winning rates as N increases, with smoother increments of response lengths and lower repetitions compared to JM.

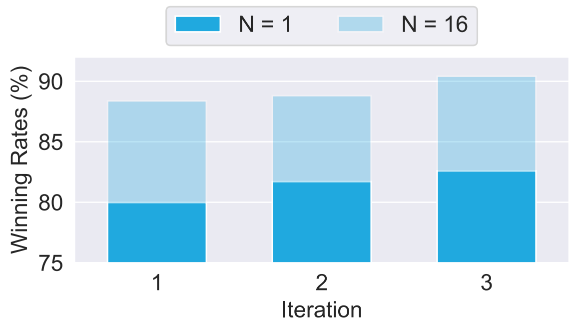

Figure 5: Result of iterative self-training on AlpacaEval using JM-PR. Performance as a policy increases as iterations proceed without losing the capacity as a judge for applying self-rejection.

et al., 2023) optimizes LLMs by implicit rewards without a separated RM. IPO (Azar et al., 2023) proposes a modified objective using an unbounded preference mapping function to mitigate overfitting on deterministic preferences in the dataset. PCO (Xu et al., 2023) utilizes cringe loss for optimization, which reflects the token-level likelihood of rejected samples as contrastive training. Self-Rewarding Language Models (Yuan et al., 2024) train LLMs to generate scores for a given response by chain-of-thought reasoning to construct preference datasets by self-generated responses. All these approaches differ from our work in that they do not perform on-policy learning.

Generative Pairwise Evaluator The generative pairwise evaluator, which we refer to as JM, has been utilized in previous approaches to alignment learning. ILF (Scheurer et al., 2023) selects the response that reflects human-requested feedback through JM. SLiC-HF (Zhao et al., 2023) constructs a static preference dataset with responses obtained from the SFT model ordered by JM. RSO (Liu et al., 2023) further approximates the optimal policy of the RLHF objective with rejection sampling through JM's choice token likelihood. OAIF (Guo et al., 2024) adopts pre-aligned LLMs to perform the pairwise judgment task for the on-policy evaluation, not by fine-tuning to JM as shown in Appendix C. All of these approaches focus on utilizing a separate evaluator and do not address self-improvement like our work.

7 Conclusion

We propose a parameter-efficient on-policy preference alignment framework, SELF-JUDGE, introducing Judge-augmented Supervised Fine-tuning (JSFT). One model trained by JSFT can perform feedback to the current policy initialized from itself by acting as a judge. This self-training does not require additional training stages and parameters for a reward model during the policy updates. Our resulting model outperforms RLHF, offline, and off-policy baselines in preference benchmarks, demonstrating the advantages of on-policy sampling and parameter efficiency of SELF-JUDGE. Besides, we provide various analyses on the best configurations and efficacy of the proposed JSFT. Specifically, JSFT boosts performance as a judge, and involving comparisons based on principle with rationale about decision leads to further improvement. This enhanced judging capability leads to better self-improvement as a policy on inference time by self-rejection over one's own responses.

Limitations

To achieve parameter-efficient on-policy self-training, SELF-JUDGE assumes the presence of a preference dataset of examples for pairwise judgment tasks on JSFT. Therefore, if human preference datasets (Bai et al., 2022a) or strong teacher models for constructing AI Feedback datasets (Cui et al., 2023) are not available, SELF-JUDGE can not be utilized. This means that SELF-JUDGE has a limitation compared to self-alignment approaches that can construct a preference dataset when there is no preference dataset at all (Bai et al., 2022b; Sun et al., 2023; Yuan et al., 2024). Additionally, our experiments do not focus on securing safety as an AI conversational system, so using SELF-JUDGE without reviewing safety may lead to the potential risk of socially inappropriate responses.

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, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2023. A general theoretical paradigm to understand learning from human preferences. *arXiv preprint arXiv:2310.12036*.

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

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022b. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

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.

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.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. 2023. Understanding dataset difficulty with $\mathcal{V}$-usable information. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.

Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, et al. 2024. Direct language model alignment from online ai feedback. *arXiv preprint arXiv:2402.04792*.

Tom Hosking, Phil Blunsom, and Max Bartolo. 2023. Human feedback is not gold standard. *arXiv preprint arXiv:2309.16349*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, et al. 2023. Prometheus: Inducing fine-grained evaluation capability in language models. *arXiv preprint arXiv:2310.08491*.

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.

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J Liu, and Jialu Liu. 2023. Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*.

Ishita Mediratta, Qingfei You, Minqi Jiang, and Roberta Raileanu. 2023. The generalization gap in offline reinforcement learning. *arXiv preprint arXiv:2312.05742*.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

OpenAI. 2023. Gpt-4 technical report.

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, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. 2023. Training language models with language feedback at scale.

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

Wei Shen, Rui Zheng, Wenyu Zhan, Jun Zhao, Shihan Dou, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. Loose lips sink ships: Mitigating length bias in reinforcement learning from human feedback. *arXiv preprint arXiv:2310.05199*.

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

Zhiqing Sun, Yikang Shen, Hongxin Zhang, Qinhong Zhou, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. 2023. Salmon: Self-alignment with principle-following reward models. *arXiv preprint arXiv:2310.05910*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jing Xu, Andrew Lee, Sainbayar Sukhbaatar, and Jason Weston. 2023. Some things are more cringe than others: Preference optimization with the pairwise cringe loss. *arXiv preprint arXiv:2312.16682*.

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

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. 2023. Rrhf: Rank responses to align language models with human feedback without tears.

Shangtong Zhang and Richard S Sutton. 2017. A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*.

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. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

Kaitlyn Zhou, Dan Jurafsky, and Tatsunori B Hashimoto. 2023. Navigating the grey area: How expressions of uncertainty and overconfidence affect language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5506–5524.

Chiwei Zhu, Benfeng Xu, Quan Wang, Yongdong Zhang, and Zhendong Mao. 2023. On the calibration of large language models and alignment. *arXiv preprint arXiv:2311.13240*.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2020. Fine-tuning language models from human preferences.

A Implementation Details

A.1 Pre-processing of Datasets

Target	HH-Helpful	UltraFeedback
SFT	65842	57569
Preference	43835	57266
(+ Principle)	✗	202887

Table 7: The number of training examples on each dataset according to the target datasets.

For the HH-Helpful[3] (Bai et al., 2022a), we parse only the content of each turn through the role header inherent in the dataset itself. During this process, if a role header exists redundantly (e.g., *Human: Assistant: [content]*), we remove all subsequent headers. We conduct the roll-out procedures from the last assistant turn in each dialogue. For the UltraFeedback[4] (Cui et al., 2023), we use the mean score across principles as the overall rank of responses. We choose the longer response as a better response when the tie happened (Hosking et al., 2023). We randomly sample one response as a rejected response y_l that is inferior in rank or score on each principle for principle-aware judgment. When a rationale is utilized in training, we remove responses that include a comparative explanation against other responses. Table 7 shows the number of resulted training examples on each dataset after the described pre-processing.

A.2 Hyperparameter Setups

Hyperparameters	Initial	Feedback
Epoch	1	3
Batch Size	128	64
Learning Rate	2e-5	5e-6
LR Scheduler	cosine	constant
Warm-up Ratio	0.03	0.1
Temperature	✗	1.0
Top-p	✗	0.9
Max New Tokens	✗	768
Optimizer	AdamW	
(β_1, β_2)	(0.9, 0.999)	
Gradient Clipping	1.0	
Max Sequence Length	2048	

LoRA	
(r, α)	(8, 16)
Dropout	0.1
RLHF	
Mini-batch Size	32
Inner Epochs	1
KL Scheduler	(0.2, 6.0)
ReST	
τ	[0.7, 0.8, 0.9]
DPO, RSO, SELF-JUDGE	
β	0.1

Table 8: Hyperparameters of the experiments. *Initial* refers to the value of hyperparameters for SFT, RM, and JM. *Feedback* refers the value of hyperparameters for baselines and SELF-JUDGE.

We apply ReST (Gulcehre et al., 2023) with $G = 1, I = 3$, regarding a single step of Improve step as one epoch of training and τ as a quantile threshold on reward distribution. We sample 8 responses per prompt for RAFT (Dong et al., 2023) and RSO (Liu et al., 2023). We randomly choose one from sampled responses as a baseline response for JMs and accepted a maximum of 1 response per prompt on RSO. We do not conduct a hyperparameter search. Table 8 shows hyperparameters used in every experiment.

[3] huggingface.co/datasets/Anthropic/hh-rlhf, MIT License, Copyright (c) 2022 Anthropic
[4] huggingface.co/datasets/openbmb/UltraFeedback, MIT License, Copyright (c) 2023 THUNLP

A.3 Training Details

We perform full fine-tuning to obtain the initial policy and evaluators, RM or JM initializing from LLaMA-2-7B[5] (Touvron et al., 2023). We apply LoRA (Hu et al., 2021) for computational efficiency on fine-tuning of baselines and SELF-JUDGE. We calculate language modeling loss on responses of assistant role for SFT and sequences after the judgment template $\mathcal{C}$ for judgment task on training. We train JM with both position-swapped comparisons on the judgment template for given triplet (x, y_w, y_l) to avoid position bias (Zheng et al., 2023). We use prompt x used for SFT on roll-out procedures. Every experiment is conducted with a maximum of 8 NVIDIA A100 80GB VRAM GPUs up to 90 hours. Every baseline is implemented based on the Transformers[6], PEFT[7] and TRL[8] library.

A.4 Evaluation Details

We use the checkpoints from the end of training for evaluation without any model selections on every method. Since VicunaEval (Chiang et al., 2023) and MT-Bench (Zheng et al., 2023) contain question types that are forced to use greedy decoding in the pre-defined configuration for each benchmark, we only apply the self-rejection on questions that do not require the greedy decoding on evaluation.

A.5 Prompt Templates

```
<|system|>
{system message}
<|user|>
{prompt at turn 1}
<|assistant|>
{response at turn 1}[EOS]
<|user|>
{prompt at turn 2}
<|assistant|>
{response at turn 2}[EOS]
...
```

Table 9: The dialogue template for the experiments.

HH-Helpful

```
Below are a series of dialogues between a human user and an AI assistant.  The AI
tries to be helpful, harmless, and honest.
```

UltraFeedback

```
Below are a series of dialogues between a human user and an AI assistant.  The AI
tries to be helpful, honest, truthful and follows the instructions from human.
```

Table 10: The default system messages for each dataset.

Table 9 shows the dialogue template used for the experiments and Table 10 shows default system messages used for each dataset. We use the default system message for both training and inference on HH-Helpful. In the case of UltraFeedback, we use a custom system prompt for the chosen response in the dataset when constructing training examples. We use the default system prompt for training on the judgment task and inference. For the judgment template $\mathcal{C}$, we use the template shown in Table 11 for HH-Helpful and Table 12 for UltraFeedback. For the principle-aware judgments on UltraFeedback, we modify the guidance about principles in Cui et al. (2023) as a system prompt for principle-aware judgment as shown in Table 13. We use the principle-aware judgment template $\mathcal{C}_p$ shown in Table 14 and slightly modify this when rationale $\mathcal{R}$ is included in the training, as shown in Table 15.

[5] huggingface.co/meta-llama/Llama-2-7b, LLAMA 2 Community License, Copyright (c) 2023 Meta Platforms
[6] github.com/huggingface/transformers
[7] github.com/huggingface/peft
[8] github.com/huggingface/trl

```
<|system|>
{default system message}
<|user|>
Which is the more helpful, harmless, and honest assistant from a given context?

Context:
-----
User:  {prompt at turn 1}

Assistant:  {responses at turn 1}

...

User:  {prompt at turn N}
-----

Assistant A: {responses A at turn N}
Assistant B: {responses B at turn N}

Please choose either A or B.
<|assistant|>
Sure!  The option which is more helpful, harmless, and honest would be Assistant {A
or B}
```

Table 11: The judgment template $\mathcal{C}$ for HH-Helpful. The target sequence for the training judgment task is represented in bold. The default system message is contained in Table 10.

```
<|system|>
{default system message}
<|user|>
Which is the better response to be an assistant who is helpful, honest, truthful
and following the given instruction from user?

Instruction:
-----
{prompt}
-----

Response A:
-----
{response A}
-----

Response B:
-----
{response B}
-----

Please choose either A or B.
<|assistant|>
Sure!  The option which is more helpful, honest, truthful and following the given
instruction from user would be Response {A or B}
```

Table 12: The judgment template $\mathcal{C}$ for UltraFeedback. The target sequence for the training judgment task is represented in bold. The default system message is contained in Table 10.

Helpfulness

Under the principle of 'helpfulness', the assistant should provide users with accurate, relevant, and up-to-date information, ensuring that the content is positive, interesting, engaging, educational, and helpful.

Honesty

Under the principle of 'honesty', the assistant should be honest about whether it knows the answer and express its uncertainty explicitly. The assistant should confident on questions it knows well and be modest on those it is unfamiliar with using weakeners such as 'I guess', 'I suppose', 'probably', and 'perhaps' to express uncertainty.

Instruction Following

Under the principle of 'instruction following', the assistant should align the output with intent of instruction, by understanding the task goal (intended outcome) and restrictions (text styles, format or designated methods, etc.).

Truthfulness

Under the principle of 'truthfulness', the assistant should answer truthfully and be faithful to factual knowledge as well as given contexts, never making up any new facts that aren't true or cannot be grounded in the instruction.

Table 13: The principle-aware system messages for UltraFeedback.

```
<|system|>
{principle-aware system message}
<|user|>
Which is the better response for an assistant when following the principle of
'{principle}' for a given instruction?

Instruction:
-----
{prompt}
-----

Response A:
-----
{response A}
-----

Response B:
-----
{response B}
-----

Please choose either A or B according to the principle of '{principle}'.
<|assistant|>
Sure! The option which is better guided by the principle of '{principle}' would be
Response {A or B}
```

Table 14: The principle-aware judgment template $\mathcal{C}_p$ for UltraFeedback. The target sequence for the training judgment task is represented in bold. The principle-aware system messages are contained in Table 13.

```
<|system|>
{principle-aware system message}
<|user|>
Which is the better response for an assistant when following the principle of
'{principle}' for a given instruction?

Instruction:
-----
{prompt}
-----

Response A:
-----
{response A}
-----

Response B:
-----
{response B}
-----

Please choose either A or B and evaluate each response according to the principle
of '{principle}'.
<|assistant|>
Sure!  Here is the decision of better response and evaluations of each response,
guided by the principle of '{principle}'.

Decision:  Response {A or B}

Evaluation of Response A: {rationale A}
Evaluation of Response B: {rationale B}
```

Table 15: The principle-aware judgment template $\mathcal{C}_p$ when rationale $\mathcal{R}$ is included in UltraFeedback. The target sequence for the training judgment task is represented in bold.

B Qualitative Examples

We compare resulting models from baseline and SELF-JUDGE trained on HH-Helpful. We further compare the difference after self-training between JM and JM-PR on UltraFeedback with the results of self-rejection to understand the effects of principle-aware judgment and rationale. In Table 16, we observe that SELF-JUDGE provides richer information regarding the aspects requested by the prompt compared to other baselines. Furthermore, we confirm that JM is likely to select a response that contains unnecessarily verbose sentences as the best response compared to JM-PR, as shown in Table 17.

Prompt

`what is the name of chris tucker first movie`

DPO

`The name of Chris Tucker's first movie is "Friday". It was released in 1995. The movie was written by Tucker and Ice Cube, and also stars Ice Cube and Nia Long.`

RSO

`Chris Tucker's first major film role was in the 1995 comedy film "Friday". He also had roles in "The Fifth Element" and "Money Train". His most famous films include "Rush Hour", "The Best Man", and "Rush Hour 2".`

ReST

`The name of Chris Tucker's first movie was "Friday".`

RAFT

`The name of Chris Tucker's first movie is "Friday", which was released in 1995.`

RLHF

`His first film was "Friday", released in 1995. The film was a great success, and helped launch him to fame. It earned him a Golden Globe nomination for Best Actor in a Musical or Comedy, and received critical acclaim. It is considered a classic modern comedy movie, and has become a cult film. A sequel was released in 2007, and a third film in the franchise is planned for 2017.`

SELF-JUDGE

`The first movie that Chris Tucker starred in was the 1995 comedy film "Friday", directed by F. Gary Gray.` **`The film is a classic urban comedy, centering around the adventures of two friends, played by Tucker and Ice Cube, as they spend the day hanging out in their Los Angeles neighborhood.`** `The film was a breakout hit for Tucker, establishing him as a prominent comedic actor, and was followed by a successful sequel in 1997. Tucker then went on to star in several major Hollywood films, including the hit comedy films "Boomerang" and "Tommy" as well as the drama "Rush Hour" and its sequels. He has also had a successful career as a stand-up comedian, performing sold-out shows worldwide.`

Table 16: Qualitative example of resulting models from each method on AlpacaEval.

Prompt

Take a look at the contract and extract the parties of the agreement from it.

As of the 13th day of December 2021, this Music Recording Agreement ("Agreement") is made between Good Kid, a Toronto-based musical group ("Artist"), and Universal Music Group, a record label with license number 545345 ("Record Label"). The Artist and the Recording Label may be referred to in this Agreement individually as "Parties" and collectively as "Parties." Work under this Agreement will begin on March 15, 2022, and will last for the duration of the Agreement.

JM (N = 1)

The parties of the Music Recording Agreement are:

1. Good Kid, a Toronto-based musical group ("Artist")
2. Universal Music Group, a record label with license number 545345 ("Record Label")

Both parties are referred to individually as "Parties" in the Agreement and collectively as "Parties." This means that the Artist and the Record Label are the two main participants in the agreement.

JM-PR (N = 1)

The parties involved in the Music Recording Agreement are:

* Good Kid, a Toronto-based musical group, represented by the artist.
* Universal Music Group, a record label with license number 545345, represented by the record label.)

This agreement is effective as of the 13th day of December 2021 and will last until the expiration of the agreement. The work under this agreement will begin on March 15, 2022, and will continue for the duration of the agreement.

JM (N = 16)

Based on the provided Music Recording Agreement, the parties involved in the agreement are:

1. **Good Kid, a Toronto-based musical group (Artist).**
2. **Universal Music Group, a record label with license number 545345 (Record Label).**

The Agreement refers to the Artist and the Record Label as "Parties" and collectively as "Parties" throughout the document. **Therefore, the parties to the Music Recording Agreement are:**

* **Good Kid, a Toronto-based musical group (Artist)**
* **Universal Music Group, a record label with license number 545345 (Record Label)**

The Agreement outlines the terms and conditions of the recording agreement between the Artist and the Record Label, including the commencement date, the duration of the agreement, and the scope of work.

JM-PR (N = 16)

Based on the provided contract, the parties involved in the Music Recording Agreement are:

1. Good Kid, a Toronto-based musical group ("Artist")
2. Universal Music Group, a record label with license number 545345 ("Record Label")

These two parties are involved in the agreement and are referred to individually as "Parties" and collectively as "Parties" throughout the document.

Table 17: Qualitative example of JM and JM-PR with applying self-rejection on AlpacaEval.

C Positioning of Self-Judge compared to Concurrent Works

Recent works such as Self-Rewarding Language Models (Yuan et al., 2024) and OAIF (Guo et al., 2024), which are optimized from pairwise preferences, have intersections with Self-Judge in aspects of self-training, fine-tuning for judgment tasks, and on-policy sampling. However, Self-Rewarding Language Models cannot provide on-policy feedback due to evaluations being performed through chain-of-thought reasoning. Also, OAIF does not address self-training since it employs a pre-aligned language model. Appendix C illustrates the difference of two approaches compared to Self-Judge.

Method	Self-Training	Fine-tuning Judgment Task	On-Policy Sampling
Self-Rewarding LMs (Yuan et al., 2024)	✓	✓	✗
OAIF (Guo et al., 2024)	✗	✗	✓
Self-Judge (Ours)	✓	✓	✓

Table 18: Positioning of Self-Judge compared with recent concurrent works.