Self-Rewarding Language Models

Weizhe Yuan [1 2] **Richard Yuanzhe Pang** [1 2] **Kyunghyun Cho** [2]
Xian Li [1] **Sainbayar Sukhbaatar** [1] **Jing Xu** [1] **Jason Weston** [1 2]

Abstract

We posit that to achieve superhuman agents, future models require superhuman feedback in order to provide an adequate training signal. Current approaches commonly train reward models from human preferences, which may then be bottlenecked by human performance level, and secondly these reward models require additional human preferences data to further improve. In this work, we study *Self-Rewarding Language Models*, where the language model itself is used via LLM-as-a-Judge prompting to provide its own rewards during training. We show that during Iterative DPO training, not only does instruction following ability improve, but also the ability to provide high-quality rewards to itself. Fine-tuning Llama 2 70B on three iterations of our approach yields a model that outperforms many existing systems on the AlpacaEval 2.0 leaderboard, including Claude 2, Gemini Pro, and GPT-4 0613. While there is much left still to explore, this work opens the door to the possibility of models that can continually improve in both axes.

1. Introduction

Training Large Language Models (LLMs) using human preference data can vastly improve the instruction following performance of pretrained models (Ouyang et al., 2022; Bai et al., 2022a). The standard approach of Reinforcement Learning from Human Feedback (RLHF) learns a reward model from these human preferences. The reward model is then frozen and used to train the LLM using RL, e.g., via PPO (Schulman et al., 2017), and the human labeling process is then possibly repeated in order to improve the reward model (Ziegler et al., 2019). A recent alternative is to avoid training the reward model at all, and directly use human preferences to train the LLM, as in Direct Preference Optimization (DPO; Rafailov et al., 2023). In both cases, the approach is bottlenecked by the size and quality of the human preference data. Futhermore, we hypothesize that training solely on human preferences will constrain models from improving beyond human level and achieving superhuman performance, which will require superhuman feedback.

In this work, we instead propose to train a self-improving reward model in order to avoid this bottleneck. Unlike traditional methods where the reward model is frozen or requires human-labeled data in order to be updated, our model is designed to continuously update itself during LLM alignment. The key to such an approach is to develop an agent that possesses all the abilities desired during training, rather than separating them out into distinct models such as a reward model and a language model. In the same way that pretraining and multitasking training of instruction following tasks allow task transfer by training on many tasks at once (Collobert & Weston, 2008; Radford et al., 2019; Ouyang et al., 2022), incorporating the reward model into that same system allows task transfer between the reward modeling task and the instruction following tasks.

We thus introduce *Self-Rewarding Language Models*, that both (i) act as instruction following models generating responses for given prompts; and (ii) can generate and evaluate new instruction following examples to add to their own training set. We train these models using an Iterative DPO framework similar to that recently introduced in Xu et al. (2023). Starting from a seed model, in each iteration there is a process of *Self-Instruction creation* whereby candidate responses are generated by the model for newly created prompts, and are then assigned rewards by that same model. The latter is implemented via LLM-as-a-Judge prompting, which can also be seen as an instruction following task. A preference dataset is built from the generated data, and the next iteration of the model is trained via DPO, see Figure 1.

In our experiments, we start with a Llama 2 70B (Touvron et al., 2023) seed model fine-tuned on Open Assistant (Köpf et al., 2023), and then perform the above training scheme. We find that not only does the instruction following performance improve from Self-Rewarding LLM alignment

[1]Meta. [2]New York University. Correspondence to: Jason Weston <jase@meta.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

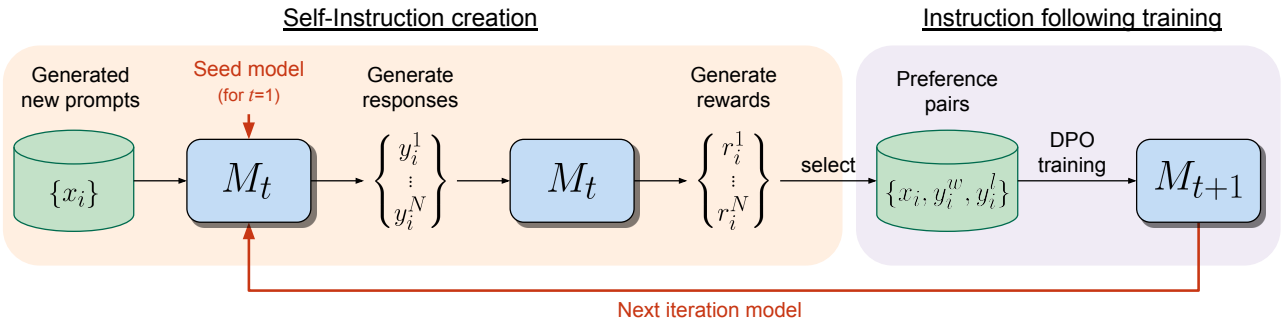


Figure 1: **Self-Rewarding Language Models.** Our self-alignment method consists of two steps: (i) *Self-Instruction creation*: newly created prompts are used to generate candidate responses from model M_t, which also predicts its own rewards via LLM-as-a-Judge prompting. (ii) *Instruction following training*: preference pairs are selected from the generated data, which are used for training via DPO, resulting in model M_{t+1}. This whole procedure can then be iterated resulting in both improved instruction following and reward modeling ability.

compared to the baseline seed model, but importantly the reward modeling ability, which is no longer fixed, improves as well. This means that the model during iterative training is able, at a given iteration, to provide a higher quality preference dataset to itself than in the previous iteration. While this effect likely saturates in real-world settings, it provides the intriguing possibility of obtaining reward models (and hence LLMs) that are superior to ones that could have been trained from the original human-authored seed data alone.

2. Self-Rewarding Language Models

Our approach first assumes access to a base pretrained language model, and a small amount of human-annotated seed data. We then build a model that aims to possess two skills simultaneously:

1. *Instruction following*: given a prompt that describes a user request, the ability to generate a high quality, helpful, and harmless response.
2. *Self-Instruction creation*: the ability to generate and evaluate new instruction-following examples to add to its own training set.

These skills are used so that the model can perform self-alignment, i.e., they are the components used to iteratively train itself using AI Feedback (AIF).

Self-instruction creation consists of generating candidate responses and then having the model itself judging their quality, i.e., it acts as its own reward model, replacing the need for an external one. This is implemented via the *LLM-as-a-Judge* mechanism (Zheng et al., 2023b), i.e., by formulating the evaluation of responses as an instruction following task. This self-created AIF preference data is used as a training set.

Our overall *self-alignment* procedure is an iterative one, which proceeds by building a series of such models, with the aim that each improves over the last. Importantly, because the model can both improve its generation ability, and act as its own reward model through the same generation mechanism, this means the reward model itself can improve through these iterations. This deviates from standard practice where the reward model is either fixed (Ouyang et al., 2022) or requires new human-labeled preference data to update it (Ziegler et al., 2019). We believe our approach can increase the ceiling of the potential for self-improvement of these learning models going forward, removing a constraining bottleneck.

We describe these steps in more detail below. An overview of the approach is illustrated in Figure 1.

2.1. Initialization

Seed Instruction Following Data We are given a seed set of human-authored (instruction prompt, response) general instruction following examples that we use for training in a supervised fine-tuning (SFT) manner, starting from a pretrained base language model. Subsequently this will be referred to as Instruction Fine-Tuning (IFT) data.

Seed LLM-as-a-Judge Instruction Following Data We also assume we are provided a seed set of (evaluation instruction prompt, evaluation result response) examples which can also be used for training. While this is not strictly necessary, as the model using IFT data will already be capable of training an LLM-as-a-Judge, we show that such training data can give improved performance (see Appendix A.3 for supporting results). In this data, the input prompt asks the model to evaluate the quality of a given response to a particular instruction. The provided evaluation result response consists of chain-of-thought reasoning (a justification), followed by a final score (in our experiments out of 5). The exact prompt

format we chose is given in Appendix Figure 6, which instructs the LLM to evaluate the response using five additive criteria (relevance, coverage, usefulness, clarity and expertise), covering various aspects of quality. Subsequently this will be referred to as Evaluation Fine-Tuning (EFT) data.

We use both these seed sets together during training.

2.2. Self-Instruction Creation

Using the model we have trained, we can make it self-modify its own training set. Specifically, we generate additional training data for the next iteration of training.

This consists of the following steps:

1. Generate new prompts: We generate a new prompt x_i using few-shot prompting[1], sampling prompts from the original seed IFT data, following the approach of Wang et al. (2023) and Honovich et al. (2023).
2. Generate candidate responses: We then generate N diverse candidate responses $\{y_i^1, \ldots, y_i^N\}$ for the given prompt x_i from our model using sampling.
3. Evaluate candidate responses: Finally, we use the LLM-as-a-Judge ability of our same model to evaluate its own candidate responses with scores $r_i^n \in [0, 5]$ (see the exact prompt in Appendix Figure 6).

2.3. Instruction Following Training

As previously described, training is initially performed with the seed IFT and EFT data (Section 2.1). This is then augmented with additional data via AI (Self-)Feedback.

AI Feedback Training After performing the self-instruction creation procedure, we can augment the seed data with additional examples for training, which we refer to as AI Feedback Training (AIFT) data.

To do this, we construct *preference pairs*, which are training data of the form (instruction prompt x_i, winning response y_i^w, losing response y_i^l). To form the winning and losing pair we take the highest and lowest scoring responses from the N evaluated candidate responses (see Section 2.2), following Xu et al. (2023), discarding the pair if their scores are the same. These pairs can be used for training with a preference tuning algorithm. We use DPO (Rafailov et al., 2023).

2.4. Overall Self-Alignment Algorithm

Iterative Training Our overall procedure trains a series of models $M_1, \ldots, M_T$ where each successive model t uses augmented training data created by the $t-1^{\text{th}}$ model. We thus define AIFT(M_t) to mean AI Feedback Training data created using model M_t. In each iteration, we use an unseen subset of the generated prompts so that AIFT(M_t) is different from all the previous AIFT data.

Model Sequence We define the models, and the training data they use as follows:

M_0 : Base pretrained LLM with no fine-tuning.

M_1 : Initialized with M_0, then fine-tuned on the IFT+EFT seed data using SFT.

M_2 : Initialized with M_1, then trained with AIFT(M_1) data using DPO.

M_3 : Initialized with M_2, then trained with AIFT(M_2) data using DPO.

This iterative training resembles the procedure used in Pairwise Cringe Optimization and specifically is termed Iterative DPO, introduced in Xu et al. (2023); however, an external fixed reward model was used in that work. See Section 4 for more discussion.

3. Experiments

3.1. Experimental Setup

Base Model In our experiments we use Llama 2 70B (Touvron et al., 2023) as our base pretrained model.

3.1.1. Seed Training Data

IFT Seed Data We use the human-authored examples provided in the Open Assistant dataset (Köpf et al., 2023) for instruction fine-tuning. Following Li et al. (2024) we use 3,200 examples, by sampling only the first conversational turns in the English language that are high-quality, based on their human annotated rank (choosing only the highest rank 0). In our experiments, we compare to a model fine-tuned from the base model using only this data via supervised fine-tuning, and refer to it as our *SFT baseline*.

EFT Seed Data The Open Assistant data also provides multiple ranked human responses per prompt from which we can construct evaluation fine-tuning data. We split this into train and evaluation sets, and use it to create LLM-as-a-Judge data. This is done by placing it in the input prompt format (detailed in Figure 6 in Appendix), which consists of the scoring criteria description, and the given instruction and response to be evaluated. For training targets, chain-of-thought justifications and final scores out of 5 are not directly provided, so we use the SFT baseline to generate such output evaluations for each input, and accept them into the training set if the ranking of their scores agrees with the human rankings in the dataset. We resample the training set by discarding some of the data that receives the most

[1]The prompts are generated from a fixed model in advance, but we show that they can also be generated by the newly trained model in each iteration in Appendix A.5.

common score so that the scores are not too skewed, as we observe many samples receive a score of 4. This results in 1,630 train and 541 evaluation examples (which do not overlap with the IFT data).

3.1.2. Evaluation Metrics

We evaluate the performance of our self-rewarding models in two axes: their ability to follow instructions, and their ability as a reward model (ability to evaluate responses).

Instruction Following We evaluate head-to-head performance between various models using GPT-4 [2] (Achiam et al., 2023) as an evaluator over 256 test prompts (which we refer to as IFT test data) derived from various sources following Li et al. (2024) using the AlpacaEval evaluation prompt (Li et al., 2023). We try the prompt in both orders comparing pairwise, and if the GPT-4 evaluations disagree we count the result as a tie. We also perform a similar evaluation with humans (authors). We additionally report results in the AlpacaEval 2.0 leaderboard format which is evaluated over 805 prompts, and compute the win rate against the baseline GPT-4 Turbo model based on GPT-4 judgments. Further, we report results on MT-Bench (Zheng et al., 2023b) a set of challenging multi-turn questions in various categories from math and coding to roleplay and writing, which uses GPT-4 to grade the model responses out of 10. Finally we also test the models on a set of 9 NLP benchmarks: ARC-Easy (Clark et al., 2018), ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), SIQA (Sap et al., 2019), PIQA (Bisk et al., 2020), GSM8K (Cobbe et al., 2021), MMLU (Hendrycks et al., 2021), OBQA (Mihaylov et al., 2018) and NQ (Kwiatkowski et al., 2019).

Reward Modeling We evaluate the correlation with human rankings on the evaluation set we derived from the Open Assistant dataset, as described in Section 3.1.1. Each instruction has on average 2.85 responses with given rankings. We can thus measure the *pairwise accuracy*, which is how many times the order of the ranking between any given pair agrees between the model's evaluation and the human ranking. We also measure the *exact match* count, which is how often the total ordering is exactly the same for an instruction. We also report the Spearman correlation and Kendall's τ. Finally, we report how often the responses that the model scores a perfect 5 out of 5 are rated as the highest ranking by humans.

3.1.3. Training Details

Instruction Following Training The training hyperparameters we use are as follows. For SFT we use learning rate $5.5e{-}6$ which decays (cosine) to $1.1e{-}6$ at the end of training, batch size 16 and dropout 0.1. We only calculate the loss on target tokens instead of the full sequence. For DPO we use learning rate $1e{-}6$ which decays to $1e{-}7$, batch size 16, dropout 0.1, and a β value of 0.1. We perform early stopping by saving a checkpoint every 200 steps and evaluating generations using Claude 2 (Anthropic, 2023) on 253 validation examples derived from various sources following Li et al. (2024). This is evaluated pairwise against the previous step's generations using the AlpacaEval evaluation prompt format (Li et al., 2023).

Self-Instruction Creation To generate new prompts we use a fixed model[3], Llama 2-Chat 70B with 8-shot prompting following Self-Instruct (Wang et al., 2023), where we sample six demonstrations from the IFT data and two from the model generated prompts[4], and use decoding parameters $T = 0.6$, $p = 0.9$. We use their prompt template for non-classification tasks and apply the same filtering techniques, including the ROUGE-L (Lin, 2004) similarity check, keyword filtering, and length filtering. These filtering steps ensure that generated prompts are diverse and similar to the IFT seed data. Except for the prompt generation part, the other parts of the creation pipeline (generating the response, and evaluating it) use the Self-Rewarding model being trained. For candidate response generation we sample $N = 4$ candidate responses with temperature $T = 0.7$, $p = 0.9$. When evaluating candidate responses, as there is variance to these scores, in our experiments we also use sampled decoding (with the same parameters) and generate these evaluations multiple (3) times and take the average. We added 3,964 such preference pairs to form the AIFT(M_1) dataset used to train M_2 via DPO, and 6,942 pairs [5] to form AIFT(M_2) used to train M_3.

3.2. Results

3.2.1. Instruction Following Ability

Head to head performance results are provided in Figure 2.

EFT+IFT Seed Training Performs Similarly to IFT Alone We find that adding the Evaluation Fine-Tuning (EFT) task to training does not impact instruction following performance compared to using Instruction Fine-Tuning (IFT) data alone with an almost equal head to head (30.5% wins vs. 30.9% wins). This is a positive result because it means the increased capability of a model to self-reward

[2] We used a fixed model `gpt-4-1106-preview` for all evaluations.

[3] We opted for a fixed model to simplify the post-processing such as ROUGE-L similarity filtering, but used a different subset of the generated prompts in each iteration.

[4] Initially, before we had model-generated prompts, we selected all 8 demonstrations from the IFT data.

[5] In the first iteration, we used fewer generated prompts (5K instead of 15K) for faster experimentation.

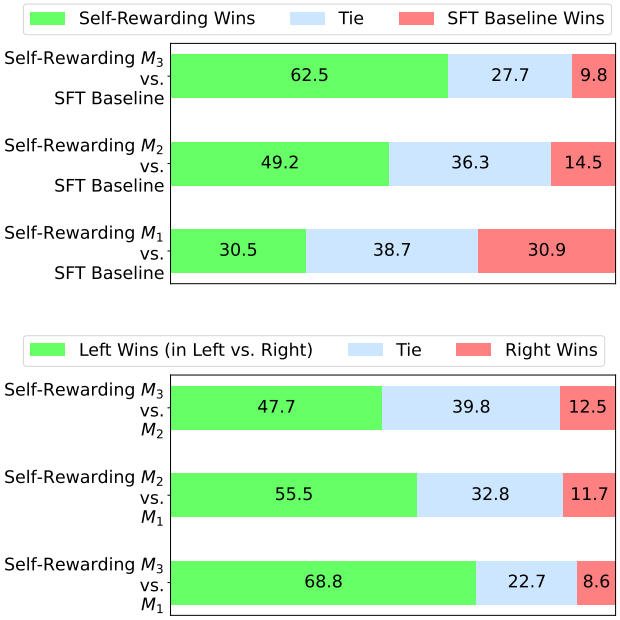


Figure 2: **Instruction following ability improves with Self-Training:** We evaluate our models using head-to-head win rates on diverse prompts using GPT-4. The SFT Baseline is on par with Self-Rewarding Iteration 1 (M_1). However, Iteration 2 (M_2) outperforms both Iteration 1 (M_1) and the SFT Baseline. Iteration 3 (M_3) gives further gains over Iteration 2 (M_2), outperforming M_1, M_2 and the SFT Baseline by a large margin.

does not affect its other skills. We can thus use IFT+EFT training as Iteration 1 (M_1) of our Self-Rewarding model, and then run further iterations.

Iteration 2 (M_2) Improves over Iteration 1 (M_1) and SFT Baseline Iteration 2 of Self-Rewarding training (M_2) provides superior instruction following to Iteration 1 (M_1) with 55.5% wins for M_2 compared to only 11.7% for M_1 in a head to head evaluation. It provides similar gains over the SFT Baseline as well (49.2% wins vs. 14.5% wins). Clearly, there is a large jump in performance from M_1 to M_2 by using the preference data AIFT(M_1) provided by the reward model from Iteration 1.

Iteration 3 (M_3) Improves over Iteration 2 (M_2) We see a further gain in Iteration 3 over Iteration 2, with 47.7% wins for M_3 compared to only 12.5% for M_2 in a head to head evaluation. Similarly, the win rate over the SFT Baseline for M_3 increases to 62.5% wins vs. 9.8%, i.e., winning more often than the M_2 model did. Overall, we see large gains from M_2 to M_3 through training using the preference data AIFT(M_2) provided by the reward model from Iteration 2.

Table 1: **AlpacaEval 2.0 results** (win rate over GPT-4 Turbo evaluated by GPT-4). Self-Rewarding iterations yield improving win rates. Iteration 3 (M_3) outperforms many existing models that use proprietary training data or targets distilled from stronger models.

		Alignment Targets	
Model	**Win Rate**	Distilled	Proprietary
Self-Rewarding 70B			
Iteration 1 (M_1)	9.94%		
Iteration 2 (M_2)	15.38%		
Iteration 3 (M_3)	20.44%		
Selected models from the leaderboard			
GPT-4 0314	22.07%		✓
Mistral Medium	21.86%		✓
Claude 2	17.19%		✓
Gemini Pro	16.85%		✓
GPT-4 0613	15.76%		✓
GPT 3.5 Turbo 0613	14.13%		✓
LLaMA2 Chat 70B	13.87%		✓
Vicuna 33B v1.3	12.71%	✓	
Humpback LLaMA2 70B	10.12%		
Guanaco 65B	6.86%		
Davinci001	2.76%		✓
Alpaca 7B	2.59%	✓	

Self-Rewarding Models Perform Well on AlpacaEval 2 Leaderboard We evaluate our models on the AlpacaEval 2.0 leaderboard, with results given in Table 1. We observe the same findings as in the head-to-head evaluations, that training iterations yield improved win rates, in this case over GPT4-Preview (11/06), from 9.94% in Iteration 1, to 15.38% in Iteration 2, to 20.44% in Iteration 3. Our Iteration 3 model outperforms many existing models in this metric, including Claude 2, Gemini Pro, and GPT4 0613. We show some selected models from the leaderboard in the table. We note that many of those competing models contain either proprietary alignment data (which is typically large, e.g., over 1M annotations in Touvron et al. (2023)) or use targets that are distilled from stronger models. In contrast, our Self-Rewarding model starts from a small set of seed data from Open Assistant, and then generates targets and rewards from the model itself for further iterations of training.

Improvements From Further Iterations Eventually Saturate We also conducted a fourth iteration on AlpacaEval 2, where the win rate was 22.97%, outperforming GPT-4 0314. We observe the improvements from the iterations are decreasing in each iteration (5.44%, 5.06%, 2.53%) similar to other iterative algorithms. The trend seems to be that it will saturate with no further improvements.

Fine-Grained Analysis As described earlier, the overall performance of the model in AlpacaEval 2.0 improves

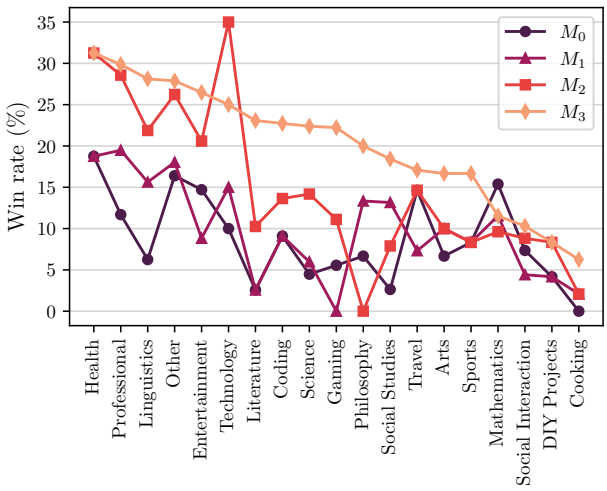


Figure 3: AlpacaEval 2.0 win rate breakdown for instruction categories (full names given in Appendix). Self-Rewarding models give gains across several topics, but tend to e.g. give less gains on mathematics and reasoning tasks.

with each iteration of training. It would be interesting to break down the overall performance improvement to see exactly what type of tasks these improvements come from. Therefore, we cluster the instructions in AlpacaEval test set into different groups based on three perspectives: (1) instruction category (2) instruction complexity (3) expected response length. We achieve this by using GPT-4. The detailed statistical information of the breakdown and the prompting techniques we used for getting this breakdown can be found in Appendix A.6. Results for the instruction category are given in Figure 3, and the other two in Appendix Figure 11. From the results we can conclude that (i) Self-Rewarding models can substantially improve the win rate in most categories, but there are some tasks for which this approach does not improve, such as mathematics and logical reasoning, indicating that our current training approach mainly allows the models to better utilize their existing knowledge. (ii) Through Self-Rewarding model training, the model's win rate increases on almost all tasks of different complexity, and especially on slightly more difficult tasks (complexity of 5, 6, 7 out of 10). (iii) The models also show a steady increase in the win rate on tasks with instructions with different expected response lengths.

Data Distribution Analysis We perform a t-SNE (Van der Maaten & Hinton, 2008) visualization of the IFT, EFT and AIFT(M_1) data, shown in Appendix A.1. We find good overlap between the IFT and AIFT(M_1) examples, which is desired as we want to avoid distribution shift in input prompts during our training. In contrast, the EFT examples lie in a different part of the embedding space, which can help explain why they would not affect IFT performance. We

Table 2: **MT-Bench Results** (on a scale of 10). Self-Rewarding iterations yield improving scores across various categories. Math, code & reasoning performance and iteration gains are smaller than for other categories, likely due to the makeup of the Open Assistant seed data we use.

	Overall Score	Math, Code & Reasoning	Humanities, Extraction, STEM, Roleplay & Writing
SFT	6.85	3.93	8.60
M_1	6.78	3.83	8.55
M_2	7.01	4.05	8.79
M_3	7.25	4.17	9.10

Table 3: **NLP Benchmarks.** Self-Rewarding models mostly tend to maintain performance compared to the Llama 2 70B base model and the SFT Baseline, despite being fine-tuned on very different instruction-following prompts.

	ARC (↑) challenge	HellaSwag (↑)	GSM8K (↑)	MMLU (↑)	NQ (↑)
Llama 2	57.40	85.30	56.80	68.90	25.30
SFT	55.97	85.17	50.72	69.76	34.35
M_1	57.51	84.99	60.27	69.34	35.48
M_2	54.51	84.27	59.29	69.31	33.07
M_3	53.13	83.29	57.70	69.37	31.86

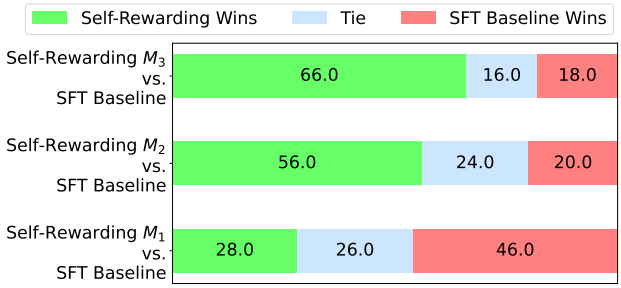


Figure 4: **Human evaluation results.** Iterations of Self-Rewarding (M_1, M_2 and M_3) provide progressively better head-to-head win rates compared to the SFT baseline, in agreement with the automatic evaluation results.

observe that generations from M_1 on AlpacaEval have an average length of 1092 characters, for M_2 they are 1552, and for M_3 they are 2552, so the model is learning to generate longer responses, which we note may be a factor in relative performance.

Human Evaluation To examine whether human judgments align with automatic evaluation results, we conduct human evaluations that compare SFT baseline generations with the generations from each iteration of Self-Rewarding training, i.e., models M_1, M_2, and M_3. Specifically, we randomly select 50 instructions from the IFT test set. Each instruction corresponds to three pairs of generations (i.e.,

baseline vs. M_1, baseline vs. M_2, baseline vs. M_3). For each pair of generations, we assign them to three different annotators (blind evaluation performed by the authors) to make a pairwise judgment, and take a majority vote to decide which generation is better. The human evaluation results are shown in Figure 4. Notably, Self-Rewarding models from later iterations show a larger advantage over the SFT baseline model, which is consistent with GPT-4's judgments, and demonstrates the effectiveness of our iterative training procedure.

MT-Bench Performance Further Validates These Results We report performance on MT-Bench in Table 2 for the SFT baseline and iterations of the Self-Rewarding model. We again see improvements across the iterations of training from M_1 to M_3, from 6.78 (out of 10) up to 7.25, with larger relative gains in the humanities, STEM, roleplay, writing and extraction categories, and smaller gains in the math, code and reasoning categories. We expect that the latter is due to the seed prompts we use from Open Assistant tending to underemphasize the reasoning-based tasks. We note also that these improvements are in spite of our method using and constructing prompts that only involve a single turn, given the MT-Bench benchmark itself is a multi-turn evaluation.

Self-Rewarding Models Did Not Lose Ability on NLP Benchmarks As shown in Table 3, the performance on most NLP benchmark tasks evaluated is roughly similar to the baselines. Further detailed results on more datasets are given in Appendix Table 10, following the same pattern. We hypothesize that given that our training data (seed data and synthetically generated data) are based on the Open Assistant prompts which may not be especially relevant to skills needed in the Table 3 tasks, it is expected that the task performance stays roughly similar, or may even drop. For example, in InstructGPT training (Ouyang et al., 2022) they found that "during RLHF fine-tuning, we observe performance regressions compared to GPT-3 on certain public NLP datasets" which they refer to as an "alignment tax." A clear future direction is to extend the self-rewarding paradigm to these types of tasks, by relying not only on seed prompts from Open Assistant, but also on seed prompts found in a larger variety of datasets.

3.2.2. Reward Modeling Ability

Reward modeling evaluation results are provided in Table 4.

EFT Augmentation Improves over SFT Baseline Firstly, we find that adding Evaluation Fine-Tuning (EFT) data into training, which gives examples to the model of how to act as an LLM-as-a-Judge, naturally improves its performance compared to training with Instruction Fine-Tuning (IFT) data alone. IFT data covers a wide range of general instruction tasks, and so does endow the SFT Baseline with the ability to evaluate responses; however, EFT data gives more examples of this specific task. We find improvements across all five metrics measured when using IFT+EFT vs. IFT alone, e.g., the pairwise accuracy agreement with humans increases from 65.1% to 78.7%.

Table 4: **Reward Modeling ability improves with Self-Training**: We evaluate the LLM-as-a-Judge via various metrics which measure alignment with held-out human preference data. Self-Rewarding Iteration 2 (Model M_2), which is trained using the self-rewarding model derived from its previous iteration M_1 outperforms Iteration 1 (M_1), while M_1 itself outperforms a standard SFT baseline model trained on only Instruction Fine-Tuning (IFT) data. Iteration 3 (Model M_3) gives further improvements over Iteration 2.

		Self-Rewarding Models		
Model	SFT	Iter 1 (M_1)	Iter 2 (M_2)	Iter 3 (M_3)
Training data	IFT	IFT+EFT	IFT+EFT +AIFT(M_1)	IFT+EFT +AIFT(M_1) +AIFT(M_2)
Pairwise acc. (↑)	65.1%	78.7%	80.4%	81.7%
5-best % (↑)	39.6%	41.5%	44.3%	43.2%
Exact Match % (↑)	10.1%	13.1%	14.3%	14.3%
Spearman corr. (↑)	0.253	0.279	0.331	0.349
Kendall τ corr. (↑)	0.233	0.253	0.315	0.324

Reward Modeling Ability Improves with Self-Training We find that performing a round of self-reward training *improves the ability of the model at providing self-rewards for the next iteration*, in addition to its improved instruction following ability. Model M_2 (Iteration 2) is trained using the reward model from M_1 (Iteration 1), but provides improved performance on all five metrics compared to M_1. For example, pairwise accuracy improves from 78.7% to 80.4%. Iteration 3 (M_3) improves several of these metrics further compared to M_2, for example pairwise accuracy increases from 80.4% to 81.7%. This performance gain is achieved despite there being no additional EFT data provided, and the examples created during the *Self-Instruction creation* loop do not tend to look like LLM-as-a-Judge training examples. We hypothesize that because the model is becoming better at general instruction following, it nevertheless also improves at the LLM-as-a-Judge task.

Importance of the LLM-as-a-Judge Prompt In these experiments we used the LLM-as-a-Judge prompt format shown in Appendix Figure 6. In preliminary experiments we also tried various other prompts to decide the most effective one to use. For example, we tried the prompt proposed in

Li et al. (2024) which also proposes a 5-point scale, but describes the options as multiple choice in a range of quality buckets, see Appendix Figure 7. In contrast, our prompt describes the points as additive, covering various aspects of quality. We find a large difference between these two prompts when using the SFT Baseline, e.g. 65.1% pairwise accuracy for ours, and only 26.6% pairwise accuracy for theirs. See Appendix A.2 for further details.

4. Related Work

Automatically improving or self-correcting large language models is becoming a major focus of research. A recent survey from Pan et al. (2023) attempts to summarize the topic. However, this is a rapidly moving area, and there are already promising new works not covered there.

Reinforcement Learning from Human Feedback (RLHF) Preference learning approaches such as in Ziegler et al. (2019); Stiennon et al. (2020); Ouyang et al. (2022); Bai et al. (2022a) train a fixed reward model from human preference data, and then use the reward model to train via reinforcement learning (RL), e.g. via Proximal Policy Optimization (PPO) (Schulman et al., 2017). Thus, the reward signal in a certain sense already comes from a model even in these works, but distilled from human data. Nevertheless, this is commonly referred to as RL from Human Feedback (RLHF). Methods such as Direct Preference Optimization (DPO) (Rafailov et al., 2023) avoid training the reward model entirely, and instead directly train the LLM using human preferences. Several other such competing methods exist as well (Zhao et al., 2023; Zheng et al., 2023a; Yuan et al., 2023), including Pairwise Cringe Optimization (PCO) (Xu et al., 2023). PCO uses an iterative training approach similar to the one in our work, except with a fixed reward model, and that work also showed that Iterative DPO improves over DPO using the same scheme.

Reinforcement Learning from AI Feedback (RLAIF) Constitutional AI (Bai et al., 2022b) uses an LLM to give feedback and refine responses, and uses this data to train a reward model. This fixed, separate reward model is then used to train the language model via RL, called "RL from AI Feedback" (RLAIF). Lee et al. (2023) compare RLAIF and RLHF procedures and find the methods they compare perform roughly equally. They use an "off-the-shelf" LLM to perform LLM-as-a-Judge prompting to build a training set to train a fixed reward model, which is then used for RL training. They also experiment with using the fixed but separate LLM-as-a-Judge model directly, which the authors report is computationally expensive due to using it within PPO training (rather than the offline step in the iterative approach we use in our work, which is relatively computationally cheap). Finally, SPIN (Chen et al., 2024b) recently showed they can avoid reward models entirely in an Iterative DPO-like framework by using human labels as the winning response in a pair, and the last iteration's generations as the losing response in the pair. The authors note this has the limitation that once the model generations reach human performance, they are bottlenecked. Further, each input prompt is required to have a human annotated response, in contrast to our work.

Improving LLMs via Data Augmentation (and Curation) Several methods have improved LLMs by (self-)creating training data to augment fine-tuning. Self-Instruct (Wang et al., 2023) is a method for self-instruction creation of prompts and responses, which can be used to improve a base LLM. We make use of a similar technique in our work, and then use our self-reward model to score them. Several approaches have also created training data by distilling from powerful LLMs, and shown a weaker LLM can then perform well. For example, Alpaca (Taori et al., 2023) fine-tuned a Llama 7B model with text-davinci-003 instructions created in the style of self-instruct. Alpagasus (Chen et al., 2024a) employed a strong LLM-as-a-Judge (ChatGPT) to curate the Alpaca dataset and filter to a smaller set, obtaining improved results. Instruction Backtranslation (Li et al., 2024) similarly augments and curates training data, but augmenting via backtranslating from web documents to predict prompts. The curation is done by the LLM(-as-a-Judge) itself, so can be seen as an instance of a self-rewarding model, but in a specialized setting. Reinforced Self-Training (ReST) (Gulcehre et al., 2023) uses a fixed, external reward to curate new high-quality examples to iteratively add to the training set, improving performance. In our experiments, we found that adding only positive examples in a related manner did not help, whereas preference pairs did help (see Appendix Section A.4 for details).

LLM-as-a-Judge Using LLM-as-a-Judge prompting to evaluate language models has become a standard approach (Dubois et al., 2023; Li et al., 2023; Fernandes et al., 2023; Bai et al., 2023; Saha et al., 2023), and is being used to train reward models or curate data as well, as described above (Lee et al., 2023; Chen et al., 2024a; Li et al., 2024). While some works such as Kim et al. (2023) create training data to train an LLM to perform well as a judge, to our knowledge it is not common to combine this training with general instruction following skills as in our work.

5. Conclusion

We have introduced Self-Rewarding Language Models, models capable of self-alignment via judging and training on their own generations. The method learns in an iterative manner, where in each iteration the model creates its own preference-based instruction training data. This is done

by assigning rewards to its own generations via LLM-as-a-Judge prompting, and using Iterative DPO to train on the preferences. We showed that this training both improves the instruction following capability of the model, as well as its reward-modeling ability across the iterations. This is a clear advantage over having a separate reward model, which cannot further improve without additional human preference data. While there are many avenues left unexplored, we believe this is exciting because this means the model is better able to assign rewards in future iterations for improving instruction following – a kind of virtuous circle. While this improvement likely saturates in realistic scenarios, it still allows for the possibility of continual improvement beyond the human preferences that are typically used to build reward models and instruction following models today.

6. Limitations

There are many avenues yet to explore and understand, among them the topics of further evaluation, including safety evaluation, and understanding the limits of iterative training.

We showed that the iterations of training improve both instruction following and reward modeling ability, and ran three iterations in a single setting. A clear line of further research is to understand the "scaling laws" of this effect both for more iterations, and with different language models with more or less capabilities in different settings.

We observed an increase in length in model generations, and there is a known correlation between length and estimated quality, which is a topic that should be understood more deeply in general, and in our results in particular as well. It would also be good to understand if so-called "reward-hacking" can happen within our framework, and in what circumstances. As we are using both a language model as the training reward, and a language model for final evaluation, even if they are different models, this may require a deeper analysis than we have provided. While the human evaluation we conducted did provide validation of the automatic results, further study could bring more insights.

Impact Statement

This work opens the door to the possibility of training LLMs such that they continually improve on both the instruction following ability and the reward modeling ability throughout each iteration, but studying how this affects outputs will be important. For such models, safety will be crucial, and future work should focus on this. In our experiments, the reward is not explicitly constrained by safety-related criteria. Therefore, a clear further avenue of study is to conduct safety evaluations – and to explore safety training within our framework. Reward models have been built exclusively for safety in existing systems (Touvron et al., 2023), and a promising avenue here would be to use the LLM-as-a-Judge procedure to evaluate for safety specifically in our self-rewarding training process. Given that we have shown that reward modeling ability improves over training iterations, this could mean in the best case that the safety of the model could potentially improve over time as well, with later iterations being able to catch and mitigate more challenging safety situations that earlier iterations cannot. From a broader perspective, this work could pave the way for methods that provide feedback that are more high-quality than human feedback, thereby creating training data that are more high-quality, and potentially safer, than what machines can do in the current paradigm.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Adolphs, L., Gao, T., Xu, J., Shuster, K., Sukhbaatar, S., and Weston, J. The CRINGE loss: Learning what language not to model. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8854–8874, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.493. URL https://aclanthology.org/2023.acl-long.493.

Anthropic. Claude 2. https://www.anthropic.com/index/claude-2, 2023.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022a.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022b.

Bai, Y., Ying, J., Cao, Y., Lv, X., He, Y., Wang, X., Yu, J., Zeng, K., Xiao, Y., Lyu, H., Zhang, J., Li, J., and Hou, L. Benchmarking foundation models with language-model-as-an-examiner. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=IiRHQ7gvnq.

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

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

Chen, Z., Deng, Y., Yuan, H., Ji, K., and Gu, Q. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024b.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Collobert, R. and Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 160–167, 2008.

Dubois, Y., Li, X., Taori, R., Zhang, T., Gulrajani, I., Ba, J., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpacafarm: A simulation framework for methods that learn from human feedback. *arXiv preprint arXiv:2305.14387*, 2023.

Fernandes, P., Deutsch, D., Finkelstein, M., Riley, P., Martins, A., Neubig, G., Garg, A., Clark, J., Freitag, M., and Firat, O. The devil is in the errors: Leveraging large language models for fine-grained machine translation evaluation. In Koehn, P., Haddow, B., Kocmi, T., and Monz, C. (eds.), *Proceedings of the Eighth Conference on Machine Translation*, pp. 1066–1083, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.wmt-1.100. URL https://aclanthology.org/2023.wmt-1.100.

Gulcehre, C., Paine, T. L., Srinivasan, S., Konyushkova, K., Weerts, L., Sharma, A., Siddhant, A., Ahern, A., Wang, M., Gu, C., et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Honovich, O., Scialom, T., Levy, O., and Schick, T. Unnatural instructions: Tuning language models with (almost) no human labor. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14409–14428, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.806. URL https://aclanthology.org/2023.acl-long.806.

Kim, S., Shin, J., Cho, Y., Jang, J., Longpre, S., Lee, H., Yun, S., Shin, S., Kim, S., Thorne, J., et al. Prometheus: Inducing fine-grained evaluation capability in language models. *arXiv preprint arXiv:2310.08491*, 2023.

Köpf, A., Kilcher, Y., von Rütte, D., Anagnostidis, S., Tam, Z.-R., Stevens, K., Barhoum, A., Duc, N. M., Stanley, O., Nagyfi, R., et al. OpenAssistant conversations–democratizing large language model alignment. *arXiv preprint arXiv:2304.07327*, 2023.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Kelcey, M., Devlin, J., Lee, K., Toutanova, K. N., Jones, L., Chang, M.-W., Dai, A., Uszkoreit, J., Le, Q., and Petrov, S. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

Lee, H., Phatale, S., Mansoor, H., Lu, K., Mesnard, T., Bishop, C., Carbune, V., and Rastogi, A. RLAIF: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

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

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

Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Pan, L., Saxon, M., Xu, W., Nathani, D., Wang, X., and Wang, W. Y. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*, 2023.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

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

Saha, S., Levy, O., Celikyilmaz, A., Bansal, M., Weston, J., and Li, X. Branch-solve-merge improves large language model evaluation and generation. *arXiv preprint arXiv:2310.15123*, 2023.

Sap, M., Rashkin, H., Chen, D., Bras, R. L., and Choi, Y. Socialiqa: Commonsense reasoning about social interactions. *CoRR*, abs/1904.09728, 2019. URL http://arxiv.org/abs/1904.09728.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

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

Van der Maaten, L. and Hinton, G. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11), 2008.

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

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

Yuan, H., Yuan, Z., Tan, C., Wang, W., Huang, S., and Huang, F. RRHF: Rank responses to align language models with human feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=EdIGMCHk4l.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In Korhonen, A., Traum, D. R., and Màrquez, L. (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1472. URL https://doi.org/10.18653/v1/p19-1472.

Zhao, Y., Joshi, R., Liu, T., Khalman, M., Saleh, M., and Liu, P. J. SLiC-HF: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*, 2023.

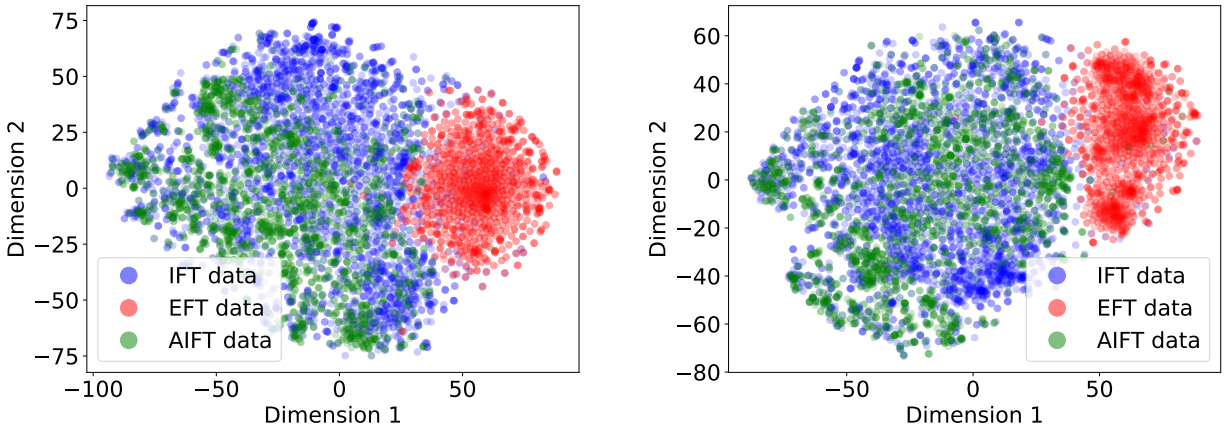
Zheng, C., Ke, P., Zhang, Z., and Huang, M. Click: Controllable text generation with sequence likelihood contrastive learning. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 1022–1040, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.65. URL https://aclanthology.org/2023.findings-acl.65.

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

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A. Appendix

A.1. Distributions of IFT, EFT and AIFT data



(a) Instruction distribution of IFT, EFT and AIFT data.

(b) Response distribution of IFT, EFT, and AIFT data.

Figure 5: Distributions of both instructions and responses for IFT, EFT and AIFT data.

We have plotted the distribution of instructions for IFT, EFT and AIFT(M_1) data, and the distribution of responses for IFT, EFT and AIFT(M_1) data in Figure 5. It is clear that the IFT data and EFT data come from very different distributions while the IFT and AIFT(M_1) data come from similar distributions. We provide examples of IFT, EFT and AIFT(M_1) data in Table 5.

A.2. EFT Prompts

The EFT prompt which we use in our main experiments is shown in Figure 6.

Other EFT prompts we have tried At first, we took the EFT prompt from Li et al. (2024) as shown in Figure 7. However, we found that this prompt was not as effective as our additive score-counting prompt because the model needed to treat the task as a multiple-choice problem, and it was difficult for the model to break down this multiple-choice problem into sub-problems involving evaluating various aspects of the response. When using the model trained on 3,200 IFT data only, its performance on the EFT test set using our additive score-counting prompt and prompt from Li et al. (2024) is shown in Table 6.

A.3. Self-rewarding Models Using IFT Data Only

To demonstrate the importance of the EFT data, we also trained a series of models starting with the model trained only on the IFT data. The following is the model sequence.

M_0 : Base pretrained LLM with no fine-tuning.

M_1' : Initialized with M_0, then fine-tuned on the IFT seed data only using SFT.

M_2' : Initialized with M_1', then trained with AIFT(M_1') data using DPO.

M_3' : Initialized with M_2', then trained with AIFT(M_2') data using DPO.

Since we did not use EFT data to train the series of models, they were not always able to score the responses according to the format and even when they did, the scores given typically converged to 4. Therefore, even when starting from the same number of generated new prompts, we could only collect a very small number of valid training samples for DPO. In total, we collected 541 pairs to form the AIFT(M_1') dataset used to train M_2' via DPO, and 429 pairs to form AIFT(M_2')

IFT	**Instruction:** When did Cuba abolish slavery? **Response:** Slavery in Cuba was abolished on October 7, 1886 by a Spanish royal decree that also made the patronato system illegal. This decree followed a period of reform in which the Spanish Crown attempted to improve the conditions of enslaved people in Cuba. Prior to this, enslaved people had been brought to Cuba from Africa as part of the Middle Passage and were used to support the sugar cane plantations. The abolition of slavery did not bring racial harmony to Cuba, and free black Cubans faced discrimination after emancipation. However, they became an important part of the Cuban independence movement and its Liberation Army ("Los Mambises"). Since then, Black Cubans have made great gains since the revolution and continue to fight against racism.
EFT	**Instruction:** Review the user's question and the corresponding response using the additive 5-point scoring system described below. Points are accumulated based on the satisfaction of each criterion: - Add 1 point if the response is relevant and provides some information related to the user's inquiry, even if it is incomplete or contains some irrelevant content. - Add another point if the response addresses a substantial portion of the user's question, but does not completely resolve the query or provide a direct answer. - Award a third point if the response answers the basic elements of the user's question in a useful way, regardless of whether it seems to have been written by an AI Assistant or if it has elements typically found in blogs or search results. - Grant a fourth point if the response is clearly written from an AI Assistant's perspective, addressing the user's question directly and comprehensively, and is well-organized and helpful, even if there is slight room for improvement in clarity, conciseness or focus. - Bestow a fifth point for a response that is impeccably tailored to the user's question by an AI Assistant, without extraneous information, reflecting expert knowledge, and demonstrating a high-quality, engaging, and insightful answer. User: Who are you? <response>I am an enthusiastic human, pretending to reply as an assistant in order to help train the Open Assistant LLM, in order to create alternatives to websites like ChatGPT3.</response> After examining the user's instruction and the response: - Briefly justify your total score, up to 100 words. - Conclude with the score using the format: "Score: <total points>" Remember to assess from the AI Assistant perspective, utilizing web search knowledge as necessary. **Response:** To evaluate the response in alignment with this additive scoring model, we'll systematically attribute points based on the outlined criteria. 1. Relevance: The response is relevant and provides some information related to the user's inquiry. +1 point 2. Substantial portion of the user's question: The response does not address a substantial portion of the user's question, only mentioning the purpose of training the Open Assistant LLM. 0 points 3. Basic elements of the user's question: The response does not answer the basic elements of the user's question in a useful way. 0 points 4. Clearly written from an AI Assistant's perspective: The response is not written from an AI Assistant's perspective, as it is stated that the responder is an enthusiastic human. 0 points 5. Impeccably tailored to the user's question: The response does not demonstrate expert knowledge, nor is it tailored to the user's question, and does not provide an engaging or insightful answer. 0 points Score: 1/5
AIFT(M_1)	**Instruction:** What are some tips for taking better photos with my smartphone? **Winning Response:** 1. Clean your lens: Before you start shooting, make sure your phone's lens is clean. Use a microfiber cloth or a soft, lint-free cloth to gently wipe the lens. 2. Use good lighting: Good lighting is key to taking great photos. Try to take photos in natural light, and avoid using the flash if possible. If you're taking photos indoors, try to find a room with lots of natural light. 3. Focus on your subject: Make sure your subject is in focus by tapping on it on your phone's screen. This will help the camera adjust its focus and ensure that your subject is sharp. 4. Use gridlines: Many smartphones have a gridlines feature that can help you compose your shots. The gridlines can help you line up your subject and create a more balanced composition. 5. Get close to your subject: Get closer to your subject to fill the frame and create a more interesting photo. This can also help you avoid distracting backgrounds. 6. Use burst mode: If you're taking action shots or trying to capture a moving subject, use burst mode to take multiple shots in quick succession. This will increase your chances of getting a good shot. 7. Experiment with angles: Try shooting from different angles to create more interesting shots. For example, try shooting from above or below your subject. 8. Edit your photos: Use editing apps to enhance your photos and correct any issues. Some popular editing apps for smartphones include Snapseed, Lightroom, and VSCO. Overall, the key to taking better photos with your smartphone is to practice and experiment with different techniques. Try out different settings and compositions to see what works best for you. **Losing Response:** 1. Clean your lens: Before you start shooting, make sure your phone's lens is clean. Use a soft, microfiber cloth to gently wipe the lens and remove any smudges or dirt. 2. Use good lighting: Good lighting is key to taking great photos. Try to shoot in natural light, and avoid using the flash if possible. If you're shooting indoors, try to find a room with lots of natural light, or use a lamp to provide additional lighting. 3. Focus on your subject: Make sure your subject is in focus by tapping on it on your phone's screen. This will help the camera adjust its focus and ensure that your subject is sharp. 4. Use grid lines: Many smartphones have a grid line feature that can help you compose your shots. Use the grid lines to line up your subject and create a more balanced and visually appealing photo. 5. Get close to your subject: Don't be afraid to get close to your subject. This can help you capture more detail and create a more interesting photo. 6. Experiment with angles: Try shooting from different angles to see what works best. For example, you might try shooting from above or below your subject to create a more dynamic shot. 7. Use burst mode: If you're shooting a moving subject, try using burst mode to capture multiple shots in quick succession. This will increase your chances of getting a sharp and well-timed shot. 8. Edit your photos: Don't be afraid to edit your photos after you've taken them. There are lots of apps available that can help you adjust things like brightness, contrast, and saturation to make your photos look their best.

Table 5: Examples of our IFT, EFT and AIFT(M_1) data.

Review the user's question and the corresponding response using the additive 5-point scoring system described below. Points are accumulated based on the satisfaction of each criterion:

- Add 1 point if the response is relevant and provides some information related to the user's inquiry, even if it is incomplete or contains some irrelevant content.
- Add another point if the response addresses a substantial portion of the user's question, but does not completely resolve the query or provide a direct answer.
- Award a third point if the response answers the basic elements of the user's question in a useful way, regardless of whether it seems to have been written by an AI Assistant or if it has elements typically found in blogs or search results.
- Grant a fourth point if the response is clearly written from an AI Assistant's perspective, addressing the user's question directly and comprehensively, and is well-organized and helpful, even if there is slight room for improvement in clarity, conciseness or focus.
- Bestow a fifth point for a response that is impeccably tailored to the user's question by an AI Assistant, without extraneous information, reflecting expert knowledge, and demonstrating a high-quality, engaging, and insightful answer.

User: <INSTRUCTION_HERE>

<response><RESPONSE_HERE></response>

After examining the user's instruction and the response:

- Briefly justify your total score, up to 100 words.
- Conclude with the score using the format: "Score: <total points>"

Remember to assess from the AI Assistant perspective, utilizing web search knowledge as necessary. To evaluate the response in alignment with this additive scoring model, we'll systematically attribute points based on the outlined criteria.

Figure 6: **LLM-as-a-Judge prompt for our LLM to act as a reward model** and provide self-rewards for its own model generations. The model is initially trained with seed training data of how to perform well at this task, and then improves at this task further through our self-rewarding training procedure. (Note the prompt, derived from Li et al. (2024), states "utilizing web search", but our model is not actually capable of this.)

used to train M_3'. The win rates are shown in Figure 8. From the figure we can conclude that EFT data helps to get better performance in the same number of iterations and the gap in performance between the model trained with EFT data and the model trained without EFT data widens in the later iterations.

A.4. Preference optimization outperforms augmenting with positive examples only

As an ablation, we tried an alternative self-training procedure of adding high-quality self-instruction creation examples to supervised fine-tuning (without preference optimization), rather than DPO. In this variant, we add additional examples of (instruction prompt, response) curated by the model to the seed set for supervised fine-tuning, following other approaches (Li et al., 2024; Adolphs et al., 2023; Gulcehre et al., 2023), rather than constructing preference data. In this setup we only add examples where the candidate response was evaluated to give a perfect score of $r_i^n = 5$. Unfortunately we could not find a configuration where this approach helped. For example, adding 11,254 such examples that scored 5 out of 5, and optimizing the mixing weight in training, still yielded a head to head with the SFT Baseline of 29% wins vs 30% wins, i.e., no improvement.

A.5. Augmented Prompt Generation Using Newly Trained Models

In our experiments, for time efficiency, we have created a fixed pool of augmented prompts in advance using ChatLlama 70B. In a real interactive system, ideally, those prompts could come from real users so that we can ensure the models are trained to align with real user requirements. Here, we also examine whether our newly trained Self-Rewarding models in each iteration can generate new prompts through in-context learning, instead of using ChatLlama 70B. To check this, we constructed 30 prompts with in-context examples using the original seed IFT data as described in Section 2.2 and tested whether M_1, M_2 and M_3 still possess in-context learning ability and can generate high quality instructions. According to manual inspection, all models can generate novel instructions given in-context examples in all 30 cases. However, for $M2$

Below is a question from an user and a candidate response. Please grade the response on a 5-point scale using the following criteria:

1: It means the answer is incomplete, vague, off-topic, controversial, or not exactly what the user asked for. For example, some content seems missing, numbered list does not start from the beginning, the opening sentence repeats user's question. Or the response is from another person's perspective with their personal experience (e.g. taken from blog posts), or looks like an answer from a forum. Or it contains promotional text, navigation text, or other irrelevant information.
2: It means the answer addresses most of the asks from the user. It does not directly address the user's question. For example, it only provides a high-level methodology instead of the exact solution to user's question.
3: It means the answer is helpful but not written by an AI Assistant. It addresses all the basic asks from the user. It is complete and self contained with the drawback that the response is not written from an AI assistant's perspective, but from other people's perspective. The content looks like an excerpt from a blog post, web page, or web search results. For example, it contains personal experience or opinion, mentions comments section, or share on social media, etc.
4: It means the answer is written from an AI assistant's perspective with a clear focus of addressing the instruction. It provide a complete, clear, and comprehensive response to user's question or instruction without missing or irrelevant information. It is well organized, self-contained, and written in a helpful tone. It has minor room for improvement, e.g. more concise and focused.
5: It means it is a perfect answer from an AI Assistant. It has a clear focus on being a helpful AI Assistant, where the response looks like intentionally written to address the user's question or instruction without any irrelevant sentences. The answer provides high quality content, demonstrating expert knowledge in the area, is very well written, logical, easy-to-follow, engaging and insightful.

User: `<INSTRUCTION_HERE>`

<response>`<RESPONSE_HERE>`</response>

Please first briefly describe your reasoning (in less than 100 words), and then write "Score: <rating>" in the last line. Answer in the style of an AI Assistant, with knowledge from web search if needed. To derive the final score based on the criteria, let's think step-by-step.

Figure 7: LLM-as-a-Judge prompt taken from Li et al. (2024).

and $M3$, the model is likely to first generate a few instructions, then generate a separator, and then start responding to the instructions.

A.6. AlpacaEval Test Sample Clustering

We used the GPT-4 (`gpt-4-1106-preview`) model to categorize the instructions in the AlpacaEval test set into clusters from three perspectives: (1) instruction category, (2) instruction complexity, and (3) expected response length. To obtain instruction categories for the AlpaceEval test set, we used the prompt in Figure 9 and obtained 20 categories in total. Then, to cluster the instructions into different groups, we use the prompt in Figure 10 for each test example. The corresponding statistics are given in Table 7, Table 8, Table 9. The fine-grained results on instruction complexity and expected response length are given in Figure 11.

A.7. NLP Benchmark Results and MT-Bench Results

We provide the detailed model performance on a number of NLP benchmarks in Table 10 and on MT-Bench in Table 11. In particular, some NLP benchmarks including ARC-Challenge, HellaSwag, SIQA, PIQA, and OBQA are all text completion tasks. In these tasks, given the multiple choice options, we choose the option corresponding to the highest log probability scored by the models as the final answer. As such, the objective of these particular tasks is quite different from what our algorithm tries to optimize, so the results on these tasks may not reflect the true capability of our models.

EFT Prompt	Multiple Choice prompt	Ours
Pairwise accuracy (↑)	26.6%	65.1%
5-best % (↑)	23.5%	39.6%
Exact Match % (↑)	1.1%	10.1%
Spearman corr. (↑)	-0.18	0.25
Kendall τ corr. (↑)	-0.16	0.23

Table 6: We tried various LLM-as-Judge prompts using the model trained with 3,200 IFT data only and found that our additive score-counting prompt worked best which demonstrates significant improvements in EFT performance comparing to the prompt used by Li et al. (2024).

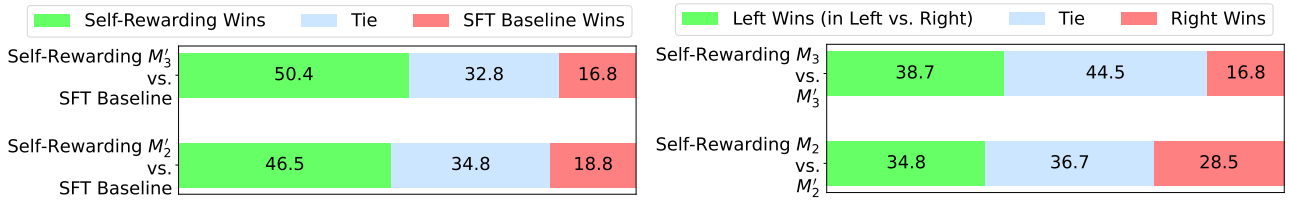


Figure 8: **EFT data helps the self-rewarding loop:** We evaluated the series of models trained using self-reward loops starting from the model trained using only IFT data. We performed head-to-head win rates comparisons on the IFT test set. While M_2' can improve over the SFT baseline and M_3' can improve even more over the SFT baseline, they lag far behind the corresponding models (M_2, M_3) that started from a base model trained using both IFT and EFT data, see Figure 2.

<LIST ALL ALPACAEVAL INSTRUCTIONS>
Given the above list of possible instructions, define between a maximum of 20 categories that would cover the types of intructions, for example recipes, reasoning tasks, general knowledge etc. Try to cover as many of the instructions as possible with the maximum 20 categories, while keeping the categories high-level, simple and easy to understand.

Figure 9: Prompt used to obtain instruction categories on the AlpacaEval test set.

Instruction: <INSTRUCTION>
Given the above, categorize it into one of the following 20 categories:
<LIST ALL CATEGORIES>

Secondly, score the instruction in terms of complexity: how complex you think it is to answer from 1-10 (where 10 is a complex question whereby first reasoning or breaking down the question into multiple subquestions for example might help improve the answer).

Thirdly, indicate how long you think the response to the instruction should be, either (a) 1 sentence, (b) 1-3 sentences, (c) 1 paragraph, (d) 2 paragraphs, or (e) 3 or more paragraphs.

Provide your final response in the following format:
Category: <one of the 20 categories>
Complexity: <score out of 10>
Length: <length category>. Do not provide the actual response.

Figure 10: Prompt for categorizing instructions based on their topics, complexities and expected response lengths.

Table 7: Breakdown of AlpacaEval test set instructions by instruction category.

Category	Number	Percentage
Science / Technology / Engineering	134	16.65%
Professional / Business / Marketing	77	9.57%
Social Interaction / Relationships / Human Behavior	68	8.45%
Miscellaneous / Other	61	7.58%
Mathematics / Logical Reasoning	52	6.46%
Cooking / Recipes	48	5.96%
Software Development / Coding / Algorithms	44	5.47%
Travel / Geography / Exploration	41	5.09%
Literature / Writing / Communication	39	4.84%
History / Social Studies	38	4.72%
Entertainment / Media Analysis	34	4.22%
Language Learning / Linguistics	32	3.98%
Music / Audio / Arts	30	3.73%
DIY Projects / Hobbies	24	2.98%
Technology / Gadgets / Consumer Products	20	2.48%
Gaming / Game Development	18	2.24%
Exercise / Health / Wellness	16	1.99%
Philosophy / Ethics / Ideology	15	1.86%
Sports / Athletics / Physical Activity	12	1.49%
Strategy / Problem-Solving / Critical Thinking	2	0.24%

Table 8: Breakdown of AlpacaEval test set instructions by instruction complexity. The instructions increase in complexity from 1 to 9, where 10 is a complex question that requires first reasoning or breaking the problem into sub-problems before it can be solved.

Complexity	Number	Percentage
3	238	29.57%
2	206	25.59%
4	122	15.16%
6	79	9.81%
5	68	8.45%
7	41	5.09%
1	34	4.22%
8	14	1.74%
9	3	0.37%

Table 9: Breakdown of AlpacaEval test set instructions by expected response length.

Expected Length	Number	Percentage
1-3 sentences	361	44.84%
1 paragraph	269	33.42%
1 sentence	143	17.76%
2 paragraphs	31	3.85%
3 or more paragraphs	1	0.13%

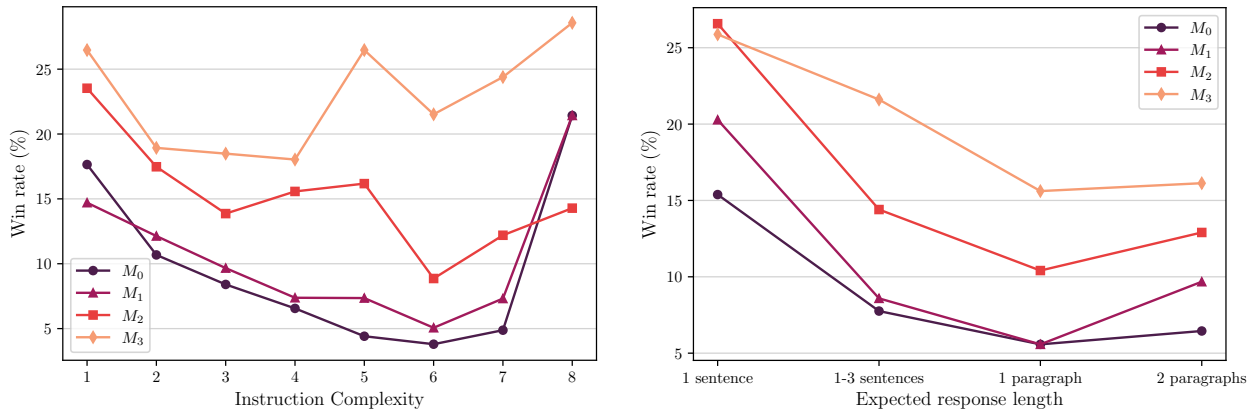


Figure 11: AlpacaEval win rate breakdown for instruction complexities (left) and expected response lengths (right). Self-Rewarding models give gains across most complexities and all response length ranges.

Table 10: NLP Benchmarks. Self-Rewarding models mostly tend to maintain performance compared to the Llama 2 base model and the SFT Baseline, despite being fine-tuned on very different instruction-following prompts.

	Commonsense Reasoning					**Math Reasoning**	**World Knowledge**		
	ARC_easy	ARC_challenge	HellaSwag	SIQA	PIQA	GSM8K (em)	MMLU (macro_avg/acc)	OBQA (acc_comp)	NQ (em)
Llama 2	80.20	57.40	85.30	50.70	82.80	56.80	68.90	60.20	25.30
SFT Baseline	76.49	55.97	85.17	51.48	82.59	50.72	69.76	57.80	34.35
M_1	78.14	57.51	84.99	53.02	82.92	60.27	69.34	57.60	35.48
M_2	74.84	54.51	84.27	51.23	81.94	59.29	69.31	57.60	33.07
M_3	72.35	53.13	83.29	49.28	80.79	57.70	69.37	58.40	31.86

Table 11: **MT-Bench Fine-grained Results**. We list our models' performance on each problem category. Self-reward is especially effective in improving the model's ability in writing, role-playing, extraction, and STEM tasks.

	Writing	**Roleplay**	**Reasoning**	**Math**	**Coding**	**Extraction**	**STEM**	**Humanities**	**Overall**
SFT Baseline	8.83	8.15	5.30	3.00	3.50	6.90	9.18	9.95	6.85
M1	9.10	7.65	4.35	3.05	4.10	7.20	8.93	9.85	6.78
M2	9.10	8.00	4.60	3.30	4.25	7.65	9.40	9.80	7.01
M3	9.58	8.73	4.80	3.50	4.20	7.80	9.45	9.95	7.25