Self-Rewarding Language Models

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

[1] Meta [2] NYU

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 separate frozen reward models cannot then learn to improve during LLM training. 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 that 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 only a preliminary study, this work opens the door to the possibility of models that can continually improve in both axes.

Damn!

1 Introduction

Aligning 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]. 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, and in the case of RLHF the quality of the frozen reward model trained from them as well.

In this work, we instead propose to train a self-improving reward model that, rather than being frozen, is continually updating during LLM alignment, in order to avoid this bottleneck. 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 and 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*, agents 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, as shown in Figure 1, in each iteration there is a process

Question: how is the generated new prompts getting updated each time?? or are we just fixing the 'prompts' every time? essentially pin down some NLG scenarios??

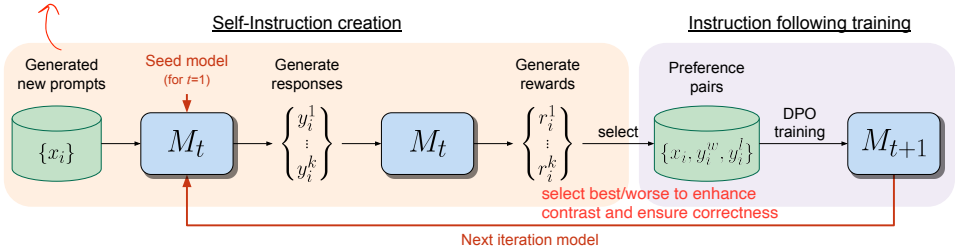


Figure 1: **Self-Rewarding Language Models.** Our self-alignment method consists of two steps: (i) *Self-Instruction creation*: newly created prompts are used to generate candidate responses from model M_t, which also predicts its own rewards via LLM-as-a-Judge prompting. (ii) Instruction following training: 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.

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.

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 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.

essentially the critique capacity and the alignment capacity both improved

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 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 it acts as its own reward model through the same generation mechanism, this means the reward model itself can improve through these iterations, deviating from standard practices where the reward model is fixed [Ouyang

et al., 2022]. We believe this 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 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 format we choose for these prompts is given in Figure 2. This thus serves as training data for the LLM to perform the role of a reward model. Subsequently this will be referred to as Evaluation Fine-Tuning (EFT) data.

We use both these seed sets together during training.

evaluate something against tip following / evaluate the quality of a given response to a particular instruction — similar (LLM-as-a-Judge)

Justification + Scoring

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:

evolve & mutate

1. *Generate a new prompt*: We generate a new prompt x_i using few-shot prompting, sampling prompts from the original seed IFT data, following the approach of Wang et al. [2022] and Honovich et al. [2023].

hallucinate

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.

picking

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 Figure 2).

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 then augment the seed data with additional examples for training, which we refer to as AI Feedback Training (AIFT) data. We try two variants of such feedback:

GIven preference pairs (thanks to LLM-as-judge) use DPO for preference tuning directly

Highest & Lowest scoring responses to form 'preference pairs' that will guarantee the contrast here.

In our case, then this would also be doable, size of local island will control the 'contrast' here.

- *Preference pairs*: We construct 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].
- *Positive examples only*: 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., 2023a, Adolphs et al., 2023, Gulcehre et al.,

Definitely a simplified version, plain prompt like this will work horribly

LLM-as-judge Prompt similar to ours, except they are again doing a sudo-absolute scoring here.

Our modification can be put instead

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 2: **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.

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 $r_i^n = 5$.

learning from preference pairs gives superior performance

While we report the results of both approaches in our experiments, we find that learning from preference pairs gives superior performance, and hence recommend that approach.

2.4 Overall Self-Alignment Algorithm

LLM can not find error, but can correct them, — and can Select from the more correct ones given options

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.

Model Sequence We thus 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 Iterative DPO introduced in Xu et al. [2023]; however, an external fixed reward model was used in that work.

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. [2023a] we use 3200 examples, by sampling only 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*.

no CoT supervision data, have only human ranking annotation:
* we let LLM to generate CoT process and pick the ones if the scores agrees with human rankings
— in pairwise comparison case, this might require some changes

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 format given in Figure 2, which consists of the scoring criteria description, and the given instruction and response to be evaluated.[1] 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. This results in 1775 train and 531 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 [Achiam et al., 2023] as an evaluator over 256 test prompts derived from various sources following Li et al. [2023a] using the AlpacaEval evaluation prompt [Li et al., 2023b]. 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 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.

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 ranked 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 linearly decays to $1.1e{-}6$, 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 linearly 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. [2023a]. This is evaluated pairwise against

[1] Note, the prompt, derived from Li et al. [2023a], mentions "utilizing web search", but our model is not actually capable of this action.

the previous step's generations using the AlpacaEval evaluation prompt format [Li et al., 2023b].

Self-Instruction creation To generate new prompts we used a fixed model, Llama 2-Chat 70B with 8-shot prompting, whereas the other parts of the creation pipeline (generating the response, and evaluating it) use the 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 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 3.

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 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.

Self-Rewarding models perform well on AlpacaEval 2 leaderboard We evaluate our models on the AlpacaEval 2.0 leaderboard format, 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 over GPT4-Turbo, 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.

Preference optimization outperforms augmenting with positive examples only We also tried the alternative self-training procedure of adding high-quality self-instruction creation examples to supervised fine-tuning (without preference optimization). 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.

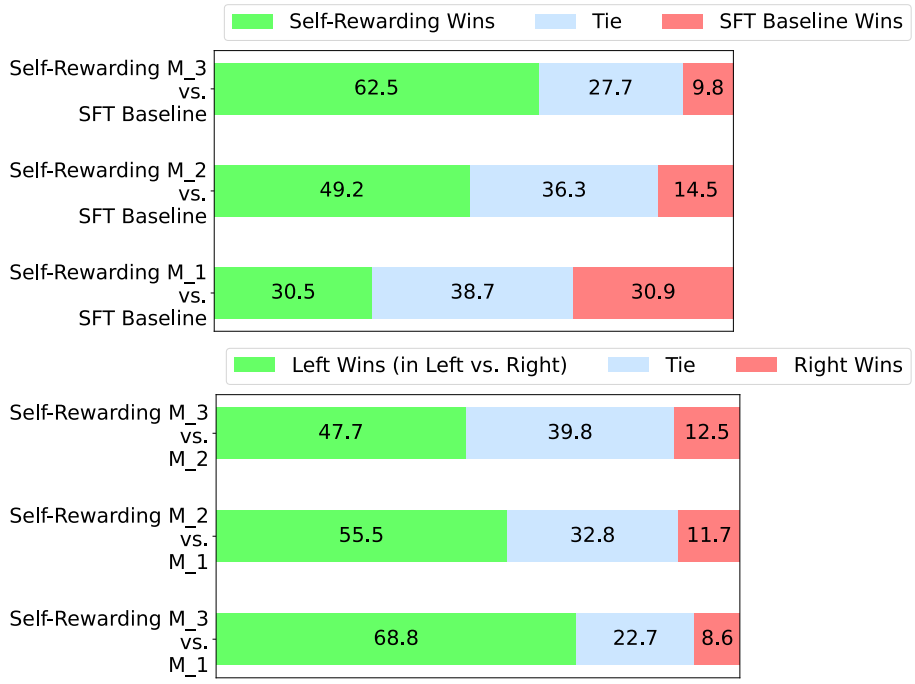


Figure 3: **Instruction following ability improves with Self-Training:** We evaluate our models using head-to-head win rates on diverse prompts using GPT-4. The SFT Baseline is on par with Self-Rewarding Iteration 1 (M_1). However, Iteration 2 (M_2) outperforms both Iteration 1 (M_1) and the SFT Baseline.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

Data distribution analysis We perform a t-SNE [Van der Maaten and Hinton, 2008] visualization of the IFT, EFT and AIFT(M_1) data, shown in Section A.1. We find good overlap between the IFT and AIFT(M_1) examples, which is desired, while the EFT examples lie in a different part of the embedding space. We observe that generations from M_1 have an average length of 1092, 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.

3.2.2 Reward Modeling Ability

Reward modeling evaluation results are provided in Table 2.

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%.

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

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%	✓	

Table 2: **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-reward 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 Baseline	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_2)
Pairwise accuracy (↑)	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

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-Judge prompt format shown in Figure 2. 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. [2023a] which also proposes a 5-point scale, but describes the options as multiple choice in a range of quality buckets, see Figure 5. 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 Section 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], which was shown to outperform DPO and PPO on AlpacaFarm [Dubois et al., 2023]. PCO uses an iterative training approach similar to the one in our work, except with a fixed reward model, and 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 (separate, fixed) reward model. This is then used to perform "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, Chen et al. [2024] 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., 2022] 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., 2023] 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., 2023a] 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. We note that in our experiments, we found that adding only positive examples in a related manner did not help, whereas adding preference pairs did help.

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., 2023b, 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., 2023, Li et al., 2023a]. While some works such as Kim et al. [2023] create training to 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 is trained 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. While this is only a preliminary study, we believe this is an exciting avenue of research 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

While we have obtained promising experimental results, we currently consider them preliminary because there are many avenues yet to explore, 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, but only 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.

While we have evaluated our models using GPT-4 using head-to-head and AlpacaEval 2 leaderboard style evaluation, there are many other automatic evaluation benchmarks that one can measure. Further, 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. We conducted a preliminary human (author) evaluation, which validated the automatic results that we see, but more detailed human evaluation would be beneficial.

Another 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 that the safety of the model could potentially improve over time as well, with later iterations being able to capture more challenging safety situations that earlier iterations cannot.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Leonard Adolphs, Tianyu Gao, Jing Xu, Kurt Shuster, Sainbayar Sukhbaatar, and Jason Weston. The CRINGE loss: Learning what language not to model. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 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.

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

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

Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, Jiayin Zhang, Juanzi Li, and Lei Hou. 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.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. AlpaGasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*, 2023.

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

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

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback. *arXiv preprint arXiv:2305.14387*, 2023.

Patrick Fernandes, Daniel Deutsch, Mara Finkelstein, Parker Riley, André FT Martins, Graham Neubig, Ankush Garg, Jonathan H Clark, Markus Freitag, and Orhan Firat. The devil is in the errors: Leveraging large language models for fine-grained machine translation evaluation. *arXiv preprint arXiv:2308.07286*, 2023.

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

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. Unnatural instructions: Tuning language models with (almost) no human labor. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 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.

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

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, et al. OpenAssistant conversations–democratizing large language model alignment. *arXiv preprint arXiv:2304.07327*, 2023.

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

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Luke Zettlemoyer, Omer Levy, Jason Weston, and Mike Lewis. Self-alignment with instruction backtranslation. *arXiv preprint arXiv:2308.06259*, 2023a.

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

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

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 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.

Swarnadeep Saha, Omer Levy, Asli Celikyilmaz, Mohit Bansal, Jason Weston, and Xian Li. Branch-solve-merge improves large language model evaluation and generation. *arXiv preprint arXiv:2310.15123*, 2023.

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

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

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

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

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

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

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

Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. 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.

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

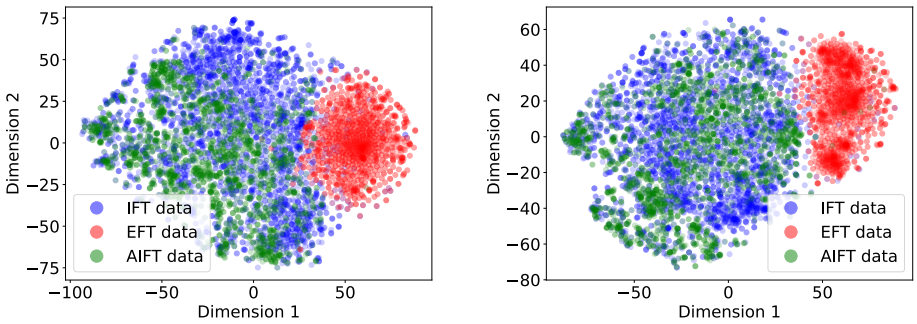
Chujie Zheng, Pei Ke, Zheng Zhang, and Minlie Huang. Click: Controllable text generation with sequence likelihood contrastive learning. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1022–1040, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.65. URL https://aclanthology.org/2023.findings-acl.65.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 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.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 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 4: Distributions of both instructions and responses for IFT, EFT and AIFT data.

We have plotted the distribution of instructions for IFT, EFT and AIFT data, and the distribution of responses for IFT, EFT and AIFT data in Figure 4. It is clear that the IFT data and EFT data come from very different distributions while the IFT and AIFT data come from similar distributions.

A.2 Other EFT prompts we have tried

At first, we took the EFT prompt from Li et al. [2023a] as shown in Figure 5. 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. [2023a] is shown in Table 3.

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 3: 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. [2023a].

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 5: LLM-as-a-Judge prompt taken from Li et al. [2023a].