Self-Evolutionary Large Language Models Through Uncertainty-Enhanced Preference Optimization

Jianing Wang, Yang Zhou, Xiaocheng Zhang, Mengjiao Bao, Peng Yan*

Meituan
{wangjianing16, yanpeng04}@meituan.com

Abstract

Iterative preference optimization has recently become one of the de-facto training paradigms for large language models (LLMs), but the performance is still underwhelming due to too much noisy preference data yielded in the loop. To combat this issue, we present an **U**ncertainty-enhanced **P**reference **O**ptimization (UPO) framework to make the LLM self-evolve with reliable feedback. The key idea is mitigating the noisy preference data derived from the current policy and reward models by performing pair-wise uncertainty estimation and judiciously reliable feedback sampling. To reach this goal, we thus introduce an estimator model, which incorporates Monte Carlo (MC) dropout in Bayesian neural network (BNN) to perform uncertainty estimation for the preference data derived from the LLM policy. Compared to the existing methods that directly filter generated responses based on the reward score, the estimator focuses on the model uncertainty in a pair-wise manner and effectively bypasses the confirmation bias problem of the reward model. Additionally, we also propose an uncertainty-enhanced self-evolution algorithm to improve the robustness of preference optimization and encourage the LLM to generate responses with both high reward and certainty. Extensive experiments over multiple benchmarks demonstrate that our framework substantially alleviates the noisy problem and improves the performance of iterative preference optimization.

Introduction

Recently, the NLP community has witnessed the success of preference optimization for large language models (LLMs), which has become one of the significant ingredients of recent revolutions (Brown et al. 2020; OpenAI 2023; Tunstall et al. 2023; Zheng et al. 2023b). As a post-training process of LLM, preference optimization aims to align the LLM policy with the labeled human feedback or AI feedback data. Early approaches utilize reinforcement learning (RL) to train the LLM policy online based on the human feedback simulated by a tuned reward model, referred to as RLHF (Christiano et al. 2017; Lee, Smith, and Abbeel 2021; Ouyang et al. 2022). Besides, offline direct preference optimization (DPO) and some variants view LLM-as-judge (Yuan et al. 2024) and directly align the policy with feedback (Rafailov et al. 2023; Ethayarajh et al. 2024).

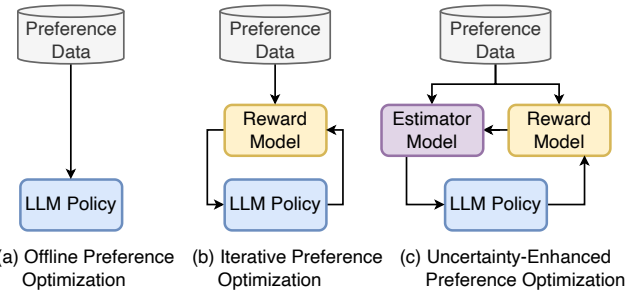

Figure 1: Overview of three paradigms.

Despite the success, these approaches relied on massive labeled preference data which requires tons of manpower and resources. To combat this issue, some recent researches introduce a novel iterative preference optimization (Pang et al. 2024; Chen et al. 2024; Kim et al. 2024; Xu et al. 2023; Rosset et al. 2024; Wu et al. 2024; Xie et al. 2024). As shown in Figure 1 (b), the offline methods can be iteratively applied similarly to the self-training procedure, where the previously trained policy generates new preference data which are then used to train the new policy. Generally, a reward model is also required in the iteration to simulate feedback for self-evolve (Xu et al. 2024; Tao et al. 2024).

However, we find one of the potential pitfalls in the iteration is that the reward model may assign unsuitable scores for the responses, leading to deriving multiple noisy preference pairs and hindering performance. This problem gets exaggerated when the interaction number increases (Han et al. 2018; Choi et al. 2024). Hence, the paramount challenge is meticulously selecting reliable preference data and making the preference optimization not distorted by noise. A simple solution is to choose one pair in which two responses signifying a notable disparity in terms of the reward score (Pang et al. 2024). Yet, it can not bypass the confirmation bias problem (Andersen and Maalej 2022; Rizve et al. 2021; Wang et al. 2021) in the self-training-like paradigm.

To this end, we present an **U**ncertainty-enhanced **P**reference **O**ptimization (UPO) framework to circumvent the noise problem. To elaborate, we introduce an estimator

* Corresponding author.

model that essentially performs a classification task to detect which response is more suitable for the query. As shown in Figure 1 (c), different from the existing reward model that can only assign a scalar score in the inference stage, it can be equipped with a Monte Carlo (MC) dropout technique, which is the approximation technique in Bayesian Neural Network (BNN) (Gal and Ghahramani 2016; Wang and Yeung 2016), to estimate the uncertainty of each preference pair (Wang et al. 2023a). Thus, a sampling signal based on the model certainty can be used to represent the reliability of the preference pair. To further improve the robustness of the iteration preference optimization, we additionally develop an uncertainty-enhanced self-evolution algorithm. Specifically, we first use the estimator certainty to split the generated preference data into reliable pairs and unreliable pairs, where reliable pairs can easily provide high-quality feedback and unreliable pairs are quite hard to express the preference. We thus integrate the uncertainty into DPO to encourage the LLM policy to know what generated pairs are reliable or unreliable feedback. Therefore, with the dual blessing of rewards and uncertainty, the new LLM policy can generate responses with both high reward and certainty.

We conduct extensive experiments on two universal NLP benchmarks (i.e., AlpacaEval 2.0 (Dubois et al. 2024) and MT-Bench (Zheng et al. 2023a)) and two mathematics reasoning tasks (i.e., GSM8K (Cobbe et al. 2021) and MATH (Hendrycks et al. 2021)), results demonstrate that our UPO framework substantially enhances the effectiveness of preference alignment, and achieves the best performance in auto evaluation [1].

Preliminaries

We first introduce the background knowledge of the iteration preference optimization and Bayesian neural network.

Preference Optimization

Suppose that the LLM policy is denoted as π_θ and it has been tuned after the pre-training and supervised fine-tuning (SFT) stage. The goal of preference optimization is to post-train the LLM policy on well-manual preference data. Formally, given a labeled preference data $\mathcal{D} = \{(x, y_w, y_l)\}$ which consists of multiple triples [2] conditioned by a prompt $x \in \mathcal{X}$, a preferred response $y_w \in \mathcal{Y}$ as the winner (chosen) and a dispreferred response $y_l \in \mathcal{Y}$ as the loser (rejected). $\mathcal{X}$ and $\mathcal{Y}$ are respectively prompt and output distributions.

During the optimization, a series of methods leverage RLHF to process the feedback online. Generally, it requires a reward model pre-trained on the preference data through the Bradley-Terry model (Bradley and Terry 1952) as:

$$p(y_w \succ y_l) = \frac{\exp\left(r_\phi(x, y_w)\right)}{\exp\left(r_\phi(x, y_w)\right) + \exp\left(r_\phi(x, y_l)\right)}, \quad (1)$$

where $r_\phi(x, y)$ is the reward model and outputs a scaler score as the reward of response y towards the given prompt x. The parameters of $r_\phi(x, y)$ can be updated as the following maximum-likelihood objective:

$$\mathcal{L}_r(\phi) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}[\log(\sigma(r_\phi(x, y_w)) - \sigma(r_\phi(x, y_l)))], \quad (2)$$

where $\sigma(\cdot)$ is the sigmoid function. When a pre-trained reward model is available, the LLM policy can be repetitively aligned to the new pairs derived from the reward model with a proximal policy optimization (PPO) algorithm:

$$\begin{aligned}\mathcal{L}_{\text{rlhf}}(\theta) = &- \mathbb{E}_{x\sim\mathcal{X}, y\sim\pi_\theta(y|x)}[r_\phi(x, y)] \\ &+ \beta\mathbb{E}_{x\sim\mathcal{X}}[\text{KL}(\pi_\theta(\cdot|x)||\pi_{\text{ref}}(\cdot|x))],\end{aligned} \quad (3)$$

where $\beta > 0$ is the balance factor, the KL divergence $\text{KL}(\cdot||\cdot)$ aims to maintain the original output distribution similar to the consistency regularization. π_{ref} is the reference model which shares the same parameters with π_θ but is frozen after the SFT stage.

In contrast to RLHF, DPO aims to follow the LLM-as-judge paradigm by directly optimizing the policy:

$$\mathcal{L}_{\text{dpo}}(\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \log \sigma(\beta h^{\pi_\theta}_{\pi_{\text{ref}}}(x, y_w, y_l)), \quad (4)$$

where $h^{\pi_\theta}_{\pi_{\text{ref}}}(x, y_w, y_l)$ is the reward difference between prefered response and disprefered response:

$$h^{\pi_\theta}_{\pi_{\text{ref}}}(x, y_w, y_l) = \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}. \quad (5)$$

Bayesian Neural Network (BNN)

In the iteration procedure, the preference pairs derived from the reward model or LLM itself may contain noisy data and hinder the whole performance. We thus briefly describe the knowledge of BNN as the basic support for denoising. Concretely, suppose a neural model f_ψ can predict the preference, the vanilla BNN assumes a prior distribution over its model parameters ψ. In other words, BNN averages over all the possible weights instead of directly optimizing for the weights (Mukherjee and Awadallah 2020). Given a labeled preference $\mathcal{D}$, the parameter can be optimized by the posterior distribution $p(\psi|\mathcal{D})$. During model inference, given one unlabeled triple $(x, y_w, y_l) \in \mathcal{D}_u$ where $\mathcal{D}_u$ is the responses set generated by the LLM policy and reward model, the probability distribution can be formed as:

$$p(c|x, y_w, y_l) = \int_\psi p(c|f_\psi(x, y_w, y_l))p(\psi|\mathcal{D}_u)d\psi, \quad (6)$$

where $c \in \{0, 1\}$ is the label represents $y_w \succ y_l$ is unsuitable or suitable. To make the equation tractable, we can find a surrogate tractable distribution $q(\psi)$ based on a dropout distribution (Srivastava et al. 2014) that makes the model posterior easy to calculate. Thus, we can sample T masked model weights $\{\widetilde{\psi}_t\}_{t=1}^T \sim q(\psi)$ from the current model. The approximate posterior for each triple is:

$$p(c|x, y_1, y_2) \approx \frac{1}{T}\sum_{t=1}^{T} p(c|f_{\widetilde{\psi}_t}(x, y_1, y_2)), \quad (7)$$

where y_1, y_2 are the responses.

[1] The code will be released at https://github.com/wjn1996/Uncertainty-Preference-Optimization.

[2] In this paper, (x, y_w, y_l) is named as preference triple or preference data, while (y_w, y_l) is named as preference pair.

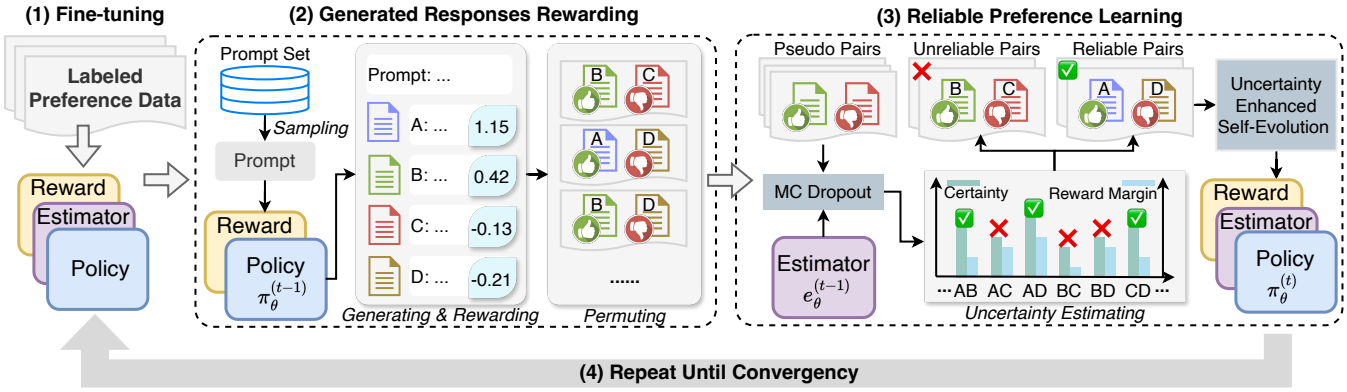

Figure 2: Illustration of UPO framework. We first use the labeled preference data to train a LLM policy, a reward model, and an estimator model. Then, multiple new preference data can be generated by the LLM policy based on the reward score. Finally, we use the uncertainty estimation technique to sample reliable data and further update the LLM policy with an uncertainty-enhanced self-evolution algorithm. The whole procedure repeats until convergence.

Methodology

In this section, we develop an **U**ncertainty-enhanced **P**reference **O**ptimization (UPO) framework illustrated in Figure 2, specialized for the improvement of the LLM self-evolve through iteration preference optimization paradigm. The framework consists of three main procedures, i.e., initial stage fine-tuning, generated responses rewarding, and reliable preference learning.

Initial Stage Fine-tuning

In the initial stage, suppose that there is a supervised fine-tuned LLM π_{sft} and a corresponding labeled preference data $\mathcal{D}^{(0)}$ derived from human or AI feedback. We follow the previous works (Pang et al. 2024; Ouyang et al. 2022; Rafailov et al. 2023; Kim et al. 2024) to use the initialized preference data to train a reward model $r_\phi^{(0)}$ based on the Bradley-Terry model in Eq. 1, and a weak LLM policy $\pi_\theta^{(0)}$ optimized from π_{sft} via DPO in Eq. 4 [3].

In addition, we also develop an estimator which is essentially a binary classifier that detects whether a pair is suitable. Different from the reward model that only assigns a scaler score, the estimator model can provide the probability of the fact that the preferred response is better than the dispreferred one, and will be used for uncertainty estimation in the reliable preference learning stage. To train the model, we need to reform the existing preference data.

We first transform the original preference triple $(x, y_w, y_l) \in \mathcal{D}^{(0)}$ into a unified prompt, and the template is denoted as $\mathcal{T}(x, y_w, y_l)$ demonstrated in Appendix A. Therefore, we can construct a binary classification dataset to train an estimator model. To make the training easier, we directly choose the backbone from $\pi_\theta^{(0)}$ and add an external classification head to project the last layer's representations at the last token position into a binary space. The training objective is formulated as:

$$\mathcal{L}_{\text{est}}(\psi) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}^{(0)}} \log f_\psi(\mathcal{T}(x, y_w, y_l)). \quad (8)$$

Generated Responses Rewarding

The LLM policy will be iteratively updated with the coordination of reward and estimator models. For the i-th iteration, we assume that the current LLM policy is $\pi_\theta^{(i-1)}$. In pursuit of obtaining more preference data to evolve the policy, we urge $\pi_\theta^{(i-1)}$ to generate multiple responses from new sampled prompts. Specifically, give a prompt $x \in \mathcal{X}$, the corresponding responses can be represented as $\{y_j\}_{j=1}^N \sim \pi_\theta^{(i-1)}(\cdot|x)$, where $N \geq 4$ is the number of responses. After that, the reward model $r_\phi^{(i-1)}$ at the previous stage will be used to assign a scale score for each response. Hence, we can sort the responses with the reward score and obtain all permutations.

Considering that too many permutations of each prompt will affect the execution efficiency of the framework, we pre-screen these permutations by a simple heuristic rule: we remove the pair whose chosen response (i.e., winner y_w) has a lower rank or rejected response (i.e., loser y_l) has a higher rank. For example, if we get six responses in descending sort (has a total of 15 pairs) and the top three responses are viewed as higher rank, only no more than 9 pairs will be used, expediting the process of iteration procedure because fewer data need to be estimated in the next stage. At last, we denote the final generated permutations with the corresponding prompt as the pseudo preference pairs $\mathcal{D}_u^{(i)}$.

Reliable Preference Learning

In this stage, we aim to use the estimator model [4] to select reliable reference data based on uncertainty estimation.

[3] In fact, the reward model can be omitted when using DPO. Yet, we still train an explicit reward model which can be used freely in practical application.

[4] We do not directly use the probability from Eq. 1 because its objective is different from uncertainty estimation in BNN.

Given an estimator model $f_{\psi}^{(i-1)}$ and a pseudo preference data $\mathcal{D}_u^{(i)}$ generated by LLM policy and reward model. We assume that each preference triple is independent of another and can be measured individually. Specifically, we follow (Houlsby et al. 2011; Wang et al. 2023b) to leverage information gain of the model parameters to estimate how certain the estimator model is to the triple with respect to the true preference. Therefore, we can obtain the formulation:

$$\begin{aligned}\mathbb{B}(\tilde{c}_j, \psi|\mathcal{T}_j, \mathcal{D}_u^{(i)}) =& \mathbb{H}(\tilde{c}_j|\mathcal{T}_j, \mathcal{D}_u^{(i)}) - \\ & \mathbb{E}_{p(\psi|\mathcal{D}_u^{(i)})}[\mathbb{H}(\tilde{c}_j|\mathcal{T}_j, \psi)],\end{aligned} \tag{9}$$

where $\mathbb{H}(\cdot)$ is the entropy, $\mathcal{T}_j = \mathcal{T}(x_j, y_{wj}, y_{lj})$ is the input template of j-th triple from $\mathcal{D}_u^{(i)}$. $\tilde{c}_j \in \{0, 1\}$ denote the prediction of estimator model. $p(\psi|\mathcal{D}_u^{(i)})$ is the posterior distribution. Through this information gain, we can find that a lower $\mathbb{B}(\tilde{c}_j, \psi|\mathcal{T}_j, \mathcal{D}_u^{(i)})$ value means that the estimator model is more certain about the prediction, as higher certainty corresponds to lower information gain. In other words, the preference triples with higher certainty and is more reliable feedback towards the prompt.

For the implementation details, we use MC Dropout in BNN to estimate the information gain. Specifically, we open the dropout and repeat T (default set as 10) times to get independent and identically distributed (i.i.d.) predictions:

$$\begin{aligned}\hat{\mathbb{B}}(\tilde{c}_j, \psi|\mathcal{T}_j, \mathcal{D}_u^{(i)}) = & - \sum_{c\in\{0,1\}} (\frac{1}{T}\sum_{t=1}^{T}\hat{p}_c^t)\log(\frac{1}{T}\sum_{t=1}^{T}\hat{p}_c^t) \\ & + \frac{1}{T}\sum_{t=1}^{T}\sum_{c\in\{0,1\}}\hat{p}_c^t\log(\hat{p}_c^t),\end{aligned} \tag{10}$$

where $\hat{p}_c^t = p(c|f_{\widetilde{\psi}_t}(\mathcal{T}_j))$ is the predict probability for the triple (x_j, y_{wj}, y_{lj}) derived from the t-th masked model $\widetilde{\psi}_t \sim q(\psi)$.

Uncertainty-Enhanced Self-Evolution

In the reliable preference learning stage, we also present an uncertainty-enhanced self-evolution algorithm to improve the robustness of LLM alignment. Based on the uncertainty estimation, we aspire for the LLM policy tune on the reliable preference data. So we define a sampling weight for each data. Given a preference data $\mathcal{D}_u^{(i)}$ and each triple has a information gain value $\hat{\mathbb{B}}(\tilde{c}_j, \psi|\mathcal{T}_j, \mathcal{D}_u^{(i)})$, the sampling weight for the current iteration stage i is defined as:

$$\mathcal{P}_j^{(i)} = \frac{(1 - \hat{\mathbb{B}}(\tilde{c}_j, \psi|\mathcal{T}_j, \mathcal{D}_u^{(i)}))\mu}{\sum_k (1 - \hat{\mathbb{B}}(\tilde{c}_k, \psi|\mathcal{T}_k, \mathcal{D}_u^{(i)}))\mu}, \tag{11}$$

where $\mu > 0$ is the hyper-parameter, and $\mathcal{P}_j^{(i)}$ is the probability that the preference triple (x_j, y_{wj}, y_{lj}) can be sampled as reliable data, i.e., $\sum_j \mathcal{P}_j^{(i)} = 1$.

With the measure of the uncertainty-aware sampling weight, we rewrite the DPO [5] in Eq. 4 to make the LLM capture two kinds of feedback: 1) what responses are better when given a prompt, and 2) what preference triples are better for the LLM to learn preference. Formally:

$$\begin{aligned}\mathcal{L}_{\text{upo}} = & -\mathbb{E}_{(x_j, y_{wj}, y_{lj})\sim\mathcal{D}_u^{(i)}} \\ & \left[(1 - \alpha_j^{(i)})\log\sigma(\beta h_{\pi_\theta^{(i-1)}}^{\pi_\theta^{(i)}}) + \alpha_j^{(i)}\log\sigma(-\beta h_{\pi_\theta^{(i-1)}}^{\pi_\theta^{(i)}})\right],\end{aligned} \tag{12}$$

where $h_{\pi_\theta^{(i-1)}}^{\pi_\theta^{(i)}}$ is the reward margin and defined as:

$$h_{\pi_\theta^{(i-1)}}^{\pi_\theta^{(i)}} = \log\frac{\pi_\theta^{(i)}(y_{wj}|x_j)}{\pi_\theta^{(i-1)}(y_{wj}|x_j)} - \log\frac{\pi_\theta^{(i)}(y_{lj}|x_j)}{\pi_\theta^{(i-1)}(y_{lj}|x_j)}. \tag{13}$$

We underscore that $0 \leq \alpha_j \leq 1$ is the uncertainty-aware weight for the triple (x_j, y_{wj}, y_{lj}) and is used to balance two items in Eq. 12. In a nutshell, a lower α_j value can encourage the LLM to focus on the given preference data. If the preference data is not reliable according to the uncertainty estimation, we not only expect to reduce the influence of this data but also let the LLM know that the pseudo-labeled preferred response is not suitable and needs to be reversed. So we follow the idea of label smoothing to design α_j as:

$$\alpha_j = \frac{1}{\mathcal{P}_j + 1}. \tag{14}$$

In addition, to improve the robustness of the iteration preference optimization, we follow (Pang et al. 2024) to add a negative log-likelihood loss for each preference triple as:

$$\mathcal{L}_{\text{upo+nll}} = \mathcal{L}_{\text{upo}} + \lambda\mathbb{E}_{(x_j, y_{wj}, y_{lj})\sim\mathcal{D}^{(i)}}\frac{\log\pi_\theta^{(i)}(y_{wj}|x_j)}{|r_\phi^{(i-1)}(x_j, y_{wj})|}, \tag{15}$$

where $\lambda > 0$ is the hyper-parameter. The whole algorithm is shown in Algorithm 1 in Appendix B.

Experiments

In this section, we choose universal NLP and mathematics reasoning tasks to evaluate the UPO framework.

Universal NLP Tasks

Following the practice in previous works, we validate the performance of LLM policy trained through the UPO framework over AlpacaEval 2.0 (Dubois et al. 2024) and MT-Bench (Zheng et al. 2023a). The benchmark of AlpacaEval 2.0 consists of 805 instructions and can be used to approximately head-to-head test the length-controlled (LC) weighted win rate of preference annotated by GPT-4. MT-Bench aims to evaluate the capability (scoring from 0 to 10) of the LLM policy to solve multiple basic problems such as writing, roleplay, reasoning, math, coding, extraction, stem, and humanities.

For the implementation setups, we choose `zephyr-7b-sft-full` (default as Zephyr-7B) as the backbone, which has been further instruction-tuned over UltraChat200K dataset from Mistral-7B (Jiang et al. 2023). The labeled preference data we used is UltraFeedback (Cui et al. 2023), which consists of 61K prompts post-processed

[5] We predominantly focused on DPO in this paper, however, our method can also adapt to PPO in RLHF.

Models	Align	AlpacaEval 2.0	MT-bench
Mistral-7B	no	0.17	3.25
Alpaca-7B	no	5.88	5.81
Zephyr-7B-SFT	no	5.84	6.18
Zephyr-7B-DPO	yes	9.12	6.79
Zephyr-7B-UPO	yes	**13.04**	**7.02**
Zephyr-7B-UPO-Merge	yes	12.04	6.85

Table 1: Main results derived from GPT-4 auto evaluation on AlpacaEval 2.0 (LC weighted win rate % compared with reference of GPT-4) and MT-Bench (absolute score).

by Tunstall et al. (2023) . We also select UltraChat200K as the prompt set. We repeatedly train three models (i.e., LLM policy, reward, and estimator) for three iterations. For the baselines, we choose SFT and DPO trained from Zephyr-7B to make a comparison. In addition, we also collect all cleaned preference data from the initial stage and three iterations and use DPO to train a model as UPO-Merge. More details of these benchmarks and hyper-parameters of each training iteration are listed in Appendix C.

Main Results As shown in Table 1, the results of AlpacaEval 2.0 denote the win rate compared to the reference generated by GPT-4, and we can see that the LLM policy of Zephyr-UPO after three iterations achieves the best win rate against GPT-4 and improves by 7.20% and 3.92% over SFT and DPO, respectively. To further investigate the performance at each iteration compared to the baseline, we use GPT-4 to annotate the preference for each iteration and present in Table 2. The results suggest that the best performance can be achieved at the second iteration and improved by over 20%. It is noteworthy that the performance improvement does not rely on increasing response length, which indicates that our method can empower the output quality of LLM instead of outputting long text. For the benchmark of MT-Bench, we also use GPT-4 to annotate the average score of eight aspects and the results in Table 1 show that our method can obtain the highest score and improve the LLM policy from 6.79% to 7.02%.

In addition, by comparing the performance of UPO-Merge with DPO and UPO, we can obtain the following suggestions: 1) the result of UPO-Merge is lower than UPO, which means that iterative evolution is more effective than single turn even though post-train with the same number of preference data, and 2) expending the preference data by self-generation manner can substantially enhance the LLM policy on universal NLP ability.

Mathematics Reasoning

Apart from the universal generation, we also choose two widely-used GSM8K (Cobbe et al. 2021) and MATH (Hendrycks et al. 2021) to show the versatility of UPO on complex reasoning benchmarks. GSM8K consists of 8.5K high-quality linguistically diverse grade school math word problems and requires the LLM policy to multi-step reasoning capability, while MATH aims at featuring challenging competition math problems.

Models	Align	Win Rate	Avg. Length
Zephyr-7B-SFT	no	50.00	1014
Zephyr-7B-DPO	yes	66.40	1298
Zephyr-7B-UPO-Iter1	yes	69.94	967
Zephyr-7B-UPO-Iter2	yes	**71.53**	**1148**
Zephyr-7B-UPO-Iter3	yes	70.21	1162
Zephyr-7B-UPO-Merge	yes	70.39	1200

Table 2: Main results derived from GPT-4 auto evaluation (LC weighted win rate %) of different iterations model from UPO over AlpacaEval 2.0 head-to-head comparison with responses of Zephyr-7B-SFT.

Models	Align	GSM8K	MATH
GPT-4o-0513	yes	95.8	76.6
Claude-3-Opus	yes	95.0	60.1
Gemini-1.5-Pro (May)	yes	90.8	67.7
Qwen2-7B-Instruct	yes	82.3	49.6
Qwen2-7B-SFT†	no	88.2	54.8
Qwen2-7B-DPO	yes	88.3	55.0
Qwen2-7B-StepDPO†	yes	88.5	55.8
Qwen2-7B-UPO-Iter1	yes	88.5	55.4
Qwen2-7B-UPO-Iter2	yes	88.6	55.7
Qwen2-7B-UPO-Iter3	yes	88.4	55.6
Qwen2-7B-UPO-Merge	yes	88.4	55.6
Qwen2-7B-StepUPO-Iter1	yes	88.8	56.0
Qwen2-7B-StepUPO-Iter2	yes	**88.9**	**56.3**
Qwen2-7B-StepUPO-Iter3	yes	88.8	56.1
Qwen2-7B-StepUPO-Merge	yes	88.8	56.2

Table 3: Main results (accuracy %) on GSM8K and MATH benchmarks. † is trained by (Lai et al. 2024).

For the implementation, we choose MathInstruct (Yue et al. 2024) as the prompt set which focuses on the hybrid use of chain-of-thought (CoT) and program-of-thought (PoT) rationales. It contains 262K prompts that are compiled from 13 math rationale datasets. We remove GSM8K and MATH from it to prevent the data leak problem. We follow Lai et al. (2024) to use the technique of StepDPO to tune the LLM policy and the well-constructed fine-grained feedback data is Math-Step-DPO-10K which involves 10.8K prompts with both coarse-grained and fine-grained annotation towards the answers. We select `Qwen2-7B-SFT` and `Qwen2-7B-SFT-Step-DPO` as our basic backbones π_{sft} and the initial LLM policy $\pi_{\theta}^{(0)}$, respectively. The model trained based on our framework with DPO and StepDPO paradigms are respectively named as UPO and StepUPO. During the iteration, we do not filter the noisy data by directly matching the ground truth of each reasoning step or the final answer. In other words, we only leverage the uncertainty estimator to verify the reliable of each reasoning step, aiming to simulate the real scenario that solves the unseen question. More details of these benchmarks and training setups are shown in Appendix D.

Models	AlpacaEval 2.0	MT-bench	GSM8K	MATH
	Zephyr-7B		Qwen2-7B	
SFT	5.84	6.18	88.2	54.8
DPO / StepDPO	9.12	6.79	88.5	55.8
UPO / StepUPO	**13.04**	**7.02**	**88.9**	**56.3**
w/o. Rule	13.01	7.01	88.8	56.1
w/o. Estimator	10.84	6.52	87.1	54.7
w/o. Weight α	12.70	6.94	88.0	55.8
w/o. NLL loss	12.39	6.92	87.9	55.7

Table 4: Ablation study at the first iteration over AlpacaEval 2.0 (LC weighted win rate % compared with GPT-4), MT-Bench (absolute score), GSM8K (accuracy %) and MATH (accuracy %).

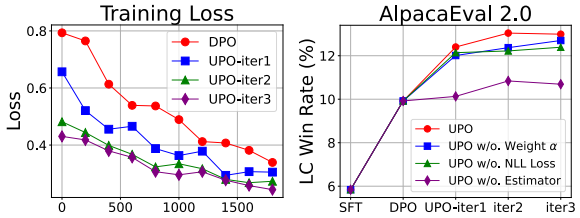

Figure 3: The curve of training loss and LC win rate (%) on AlpacaEval 2.0 at each iteration.

Main Results The results are listed in Table 3 and we can obtain the following suggestions: 1) The LLM policy post-trained by DPO makes a marginal improvement, increasing from 88.2% and 54.8% to 88.3% and 55.0%, respectively. Yet, the improvement of StepDPO can achieve an obvious gain compared with the SFT model, indicating that LLM policy self-evolution can be better conducted with fine-grained feedback. 2) For each iteration, UPO and StepUPO can consistently achieve substantial improvements on GSM8K and MATH, respectively resulting in 88.9% and 56.3% accuracy metrics. 3) The result of UPO-Merge and StepUPO-Merge is similar to the performance at the third iteration, which conflicts with the findings in universal NLP tasks. We analyze that the task of mathematics reasoning highly relies on the cleaned preference data, yet the preference data after uncertainty estimation may still contain noisy fine-grained feedback and affect the performance inevitably.

Further Analysis

Ablation Study

To investigate the impact of different techniques used in UPO, we conduct the ablation study on all benchmarks to see the performance of different variants. Specifically, for benchmarks of AlpacaEval 2.0 and MT-Bench, we choose DPO as the main baseline and optimization paradigm, while the StepDPO paradigm will be used in GSM8K and MATH. We conduct the experiments at the first iteration. For the variants, w/o. Rule means directly choosing all permutations without any pre-screen processing. w/o. Estimator denotes that do not use uncertainty estimation and choose all generated preference data to train the LLM policy, which is the same as vanilla iterative preference optimization proposed by (Pang et al. 2024). w/o. Weight α represents only training the LLM policy on DPO or StepDPO without smoothing (i.e., $\alpha = 0$). w/o. NLL loss means removing the NLL loss by setting $\lambda = 0$. Results demonstrated in Table 4 show that the performance will drop if the framework module is removed. Moreover, the use of robust techniques (i.e., uncertainty-enhanced weighting and the NLL loss) consistently contributes to the robustness improvement when training on pseudo preference data.

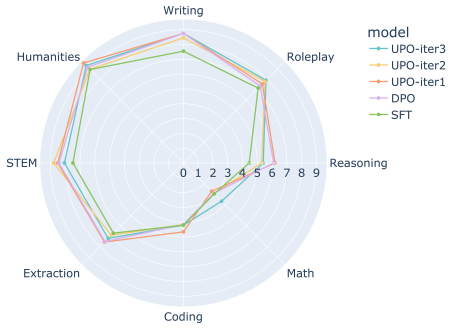

Figure 4: Performance of different iterations of UPO compared with SFT and DPO over MT-Bench.

Effect of Uncertainty-Enhanced Self-evolution

We also explore how the Uncertainty-Enhanced Self-evolution algorithm empowers the LLM policy in the iteration preference optimization procedure. To ask this question, we choose the benchmarks of AlpacaEval 2.0 and MT-Bench to make a deep-seek. We first draw a training loss curve at the initial stage (DPO training) and each iteration in UPO when preference optimizing on UltraFeedback and newly generated preference data sampled from UltraChat200K. The curve presented in Figure 3 (left) demonstrates that iterative procedure advances the convergence which may contribute to the high performance.

To see the performance changes in different training stages, we also draw a curve to show the win rate increasing in Figure 3 (right) with multiple variants. The result suggests that UPO can substantially outperform vanilla preference optimization (e.g., DPO) in all iteration stages. It is worth noting that variant UPO w/o. Estimator has a bit of improvement compared to the DPO, indicating that many noisy pseudo-preference examples are used in the next iteration and make the iteration training useless. This finding reflects that considering noisy reduction and robustness in iteration preference optimization is significantly necessary.

Capability Across Different Aspects in MT-Bench

To show the performance of the LLM policy tuned by the UPO framework, we perform task-wise deep analysis on MT-Bench and show the capability of eight aspects in Figure 4, including writing, roleplay, reasoning, math, coding, extracting, STEM, and humanities. Results show that UPO consistently enhances the generation of LLM policy on different aspects of basic problems. Notably, UPO can also re-

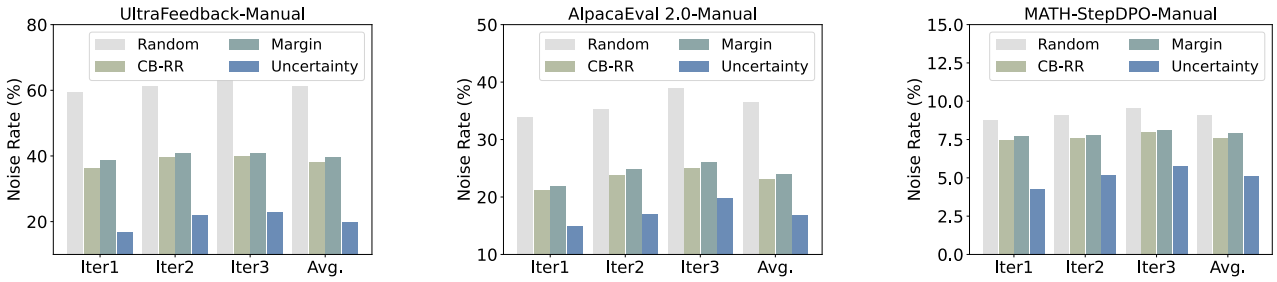

Figure 5: Noise rate (%) of different sampling strategies over multiple manual evaluation sets.

alize an obvious improvement in complex tasks, such as reasoning, math, and coding.

Noisy Data Study

We end this section by investigating how the UPO framework realizes denoising during iteration preference optimization. We respectively sample 200 preference data from the validation set of UltraFeedback, AlpacaEval 2.0, and MATH-Step-DPO-10K to manually construct the evaluation set. In particular, for preference data from UltraFeedback and MATH-Step-DPO-10K, we directly use the label (which response is better) as the ground truth. For AlpacaEval 2.0, we use the reference generated from GPT-4 as the preferred response, while the dispreferred response is created by the SFT model. At each iteration, we present four different reliable data sampling strategies to select preference data to train the LLM policy after the rewarding process. 1) "Random" denotes randomly selecting from pseudo preference data; 2) "CB-RR" means **C**hosen response with **B**est reward and **R**ejected response with **R**andom select from the rest lower reward, which is a similar strategy to UltraFeedback. 3) "Margin" denotes choosing only one preference data whose reward margin between chosen and rejected is the largest. 4) "Uncertainty" is our proposed method that uses the certainty weight to perform sampling.

Results demonstrated in Figure 5 indicate that considering the reward of the chosen response or reward margin is certainly effective to denoising, which has also been proven in some previous work (Pang et al. 2024). In addition, the results also showcase that leveraging uncertainty estimation can better reduce the noise rate by more than 20%, 10%, and 3%, respectively, indicating the effectiveness of UPO.

Related Works

Preference Optimization of LLMs

Large language models (LLMs), after undergoing extensive pre-training, may generate fabricated facts, biased content, or harmful text. To align these models with human values, fine-tuning language models to adhere to human preferences is an effective solution. Reinforcement Learning from Human Feedback (RLHF) (Stiennon et al. 2020; Ziegler et al. 2019) has emerged as a groundbreaking technique for aligning LLMs. By training a reward model on human feedback data and using Proximal Policy Optimization (PPO) (Schulman et al. 2017) to obtain the policy model for language generation, this approach has led to the development of powerful models such as GPT-4 (Achiam et al. 2023), Llama3 (Dubey et al. 2024), and Gemini (Team et al. 2023). Other methodologies such as DPO (Rafailov et al. 2024) and RRHF (Yuan et al. 2023), optimize language models directly on human feedback datasets. Nevertheless, to further improve performance, it becomes essential to conduct sampling using the model itself, necessitating the incorporation of an auxiliary reward model (RM) (Liu et al. 2023; Song et al. 2024; Wang et al. 2024a; Dong et al. 2023a).

Iterative Preference Optimization

The optimization of preference datasets and preference models plays a significant role in the alignment of LLMs. Some works (Dong et al. 2023b; Wang et al. 2024b; Rame et al. 2024) employ fine-grained reward objectives and iteratively fine-tune large models for alignment. For example, IRPO (Pang et al. 2024), utilizes iterative DPO for optimization.(Yuan et al. 2024) directly explores a novel Self-Rewarding method for LLMs, which achieve self-improvement by generating their rewards during training. (Fisch et al. 2024) proposes a reward model distillation algorithm to address the effectiveness and robustness in preference optimization. Similar to these works, we also focus on how to iteratively enhance the effectiveness of preferences and address the noise in the preference predictions by the reward model, aiming to improve the overall robustness of the alignment process.

Conclusion

We propose an uncertainty-enhanced preference optimization framework to further boost the abilities of the self-evolution of LLMs. We develop an estimator model and let it cooperate with the reward model to provide high-quality preference data at each iteration stage. To reach this goal, we leverage the MC Dropout technique in BNN to perform uncertainty estimation, eliminating the potentially noisy data derived from the weak LLM policy. In addition, we also propose an uncertainty-enhanced self-evolution algorithm to improve the robustness of LLM when repeatedly updating parameters via DPO. We conduct extensive experiments on multiple universal NLP and mathematics reasoning tasks and the results indicate the effectiveness of our method. In the future, we aim to further improve the overall performance and adapt the framework to PPO and other LLMs.

References

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

Andersen, J. S.; and Maalej, W. 2022. Efficient, Uncertainty-based Moderation of Neural Networks Text Classifiers. In *ACL*, 1536–1546.

Bradley, R. A.; and Terry, M. E. 1952. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4): 324–345.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In *NeurIPS*.

Chen, Z.; Deng, Y.; Yuan, H.; Ji, K.; and Gu, Q. 2024. Self-Play Fine-Tuning Converts Weak Language Models to Strong Language Models. *CoRR*, abs/2401.01335.

Choi, E.; Ahmadian, A.; Geist, M.; Pietquin, O.; and Azar, M. G. 2024. Self-Improving Robust Preference Optimization. *CoRR*, abs/2406.01660.

Christiano, P. F.; Leike, J.; Brown, T. B.; Martic, M.; Legg, S.; and Amodei, D. 2017. Deep Reinforcement Learning from Human Preferences. In *NeurIPS*, 4299–4307.

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. 2021. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168.

Cui, G.; Yuan, L.; Ding, N.; Yao, G.; Zhu, W.; Ni, Y.; Xie, G.; Liu, Z.; and Sun, M. 2023. UltraFeedback: Boosting Language Models with High-quality Feedback. *CoRR*, abs/2310.01377.

Dong, H.; Xiong, W.; Goyal, D.; Zhang, Y.; Chow, W.; Pan, R.; Diao, S.; Zhang, J.; Shum, K.; and Zhang, T. 2023a. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

Dong, Y.; Wang, Z.; Sreedhar, M.; Wu, X.; and Kuchaiev, O. 2023b. SteerLM: Attribute Conditioned SFT as an (User-Steerable) Alternative to RLHF. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 11275–11288.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

Dubois, Y.; Galambosi, B.; Liang, P.; and Hashimoto, T. B. 2024. Length-Controlled AlpacaEval: A Simple Way to Debias Automatic Evaluators. *CoRR*, abs/2404.04475.

Ethayarajh, K.; Xu, W.; Muennighoff, N.; Jurafsky, D.; and Kiela, D. 2024. KTO: Model Alignment as Prospect Theoretic Optimization. *CoRR*, abs/2402.01306.

Fisch, A.; Eisenstein, J.; Zayats, V.; Agarwal, A.; Beirami, A.; Nagpal, C.; Shaw, P.; and Berant, J. 2024. Robust preference optimization through reward model distillation. *arXiv preprint arXiv:2405.19316*.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *ICML*, volume 48, 1050–1059.

Han, B.; Yao, Q.; Yu, X.; Niu, G.; Xu, M.; Hu, W.; Tsang, I. W.; and Sugiyama, M. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*, 8536–8546.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. In *NeurIPS*.

Houlsby, N.; Huszar, F.; Ghahramani, Z.; and Lengyel, M. 2011. Bayesian Active Learning for Classification and Preference Learning. *CoRR*, abs/1112.5745.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de Las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Lavaud, L. R.; Lachaux, M.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7B. *CoRR*, abs/2310.06825.

Kim, D.; Lee, K.; Shin, J.; and Kim, J. 2024. Aligning Large Language Models with Self-generated Preference Data. *CoRR*, abs/2406.04412.

Lai, X.; Tian, Z.; Chen, Y.; Yang, S.; Peng, X.; and Jia, J. 2024. Step-DPO: Step-wise Preference Optimization for Long-chain Reasoning of LLMs. *CoRR*, abs/2406.18629.

Lee, K.; Smith, L. M.; and Abbeel, P. 2021. PEBBLE: Feedback-Efficient Interactive Reinforcement Learning via Relabeling Experience and Unsupervised Pre-training. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, 6152–6163. PMLR.

Liu, T.; Zhao, Y.; Joshi, R.; Khalman, M.; Saleh, M.; Liu, P. J.; and Liu, J. 2023. Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*.

Mukherjee, S.; and Awadallah, A. H. 2020. Uncertainty-aware Self-training for Few-shot Text Classification. In *NeurIPS*.

OpenAI. 2023. GPT-4 Technical Report. *CoRR*, abs/2303.08774.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P. F.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Pang, R. Y.; Yuan, W.; Cho, K.; He, H.; Sukhbaatar, S.; and Weston, J. 2024. Iterative Reasoning Preference Optimization. *CoRR*, abs/2404.19733.

Rafailov, R.; Sharma, A.; Mitchell, E.; Ermon, S.; Manning, C. D.; and Finn, C. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *CoRR*, abs/2305.18290.

Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Rame, A.; Couairon, G.; Dancette, C.; Gaya, J.-B.; Shukor, M.; Soulier, L.; and Cord, M. 2024. Rewarded soups: towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. *Advances in Neural Information Processing Systems*, 36.

Rizve, M. N.; Duarte, K.; Rawat, Y. S.; and Shah, M. 2021. In Defense of Pseudo-Labeling: An Uncertainty-Aware Pseudo-label Selection Framework for Semi-Supervised Learning. In *ICLR*.

Rosset, C.; Cheng, C.; Mitra, A.; Santacroce, M.; Awadallah, A.; and Xie, T. 2024. Direct Nash Optimization: Teaching Language Models to Self-Improve with General Preferences. *CoRR*, abs/2404.03715.

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

Song, F.; Yu, B.; Li, M.; Yu, H.; Huang, F.; Li, Y.; and Wang, H. 2024. Preference ranking optimization for human alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 18990–18998.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1): 1929–1958.

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

Tao, Z.; Lin, T.; Chen, X.; Li, H.; Wu, Y.; Li, Y.; Jin, Z.; Huang, F.; Tao, D.; and Zhou, J. 2024. A Survey on Self-Evolution of Large Language Models. *CoRR*, abs/2404.14387.

Team, G.; Anil, R.; Borgeaud, S.; Wu, Y.; Alayrac, J.-B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A. M.; Hauth, A.; et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Tunstall, L.; Beeching, E.; Lambert, N.; Rajani, N.; Rasul, K.; Belkada, Y.; Huang, S.; von Werra, L.; Fourrier, C.; Habib, N.; Sarrazin, N.; Sanseviero, O.; Rush, A. M.; and Wolf, T. 2023. Zephyr: Direct Distillation of LM Alignment. *CoRR*, abs/2310.16944.

Wang, H.; and Yeung, D. 2016. Towards Bayesian Deep Learning: A Framework and Some Existing Methods. *IEEE TKDE*, 28(12): 3395–3408.

Wang, J.; Sun, Q.; Chen, N.; Wang, C.; Huang, J.; Gao, M.; and Li, X. 2023a. Uncertainty-aware Parameter-Efficient Self-training for Semi-supervised Language Understanding. In *EMNLP*, 7873–7884.

Wang, J.; Wang, C.; Huang, J.; Gao, M.; and Zhou, A. 2023b. Uncertainty-Aware Self-Training for Low-Resource Neural Sequence Labeling. In *AAAI*, 13682–13690. AAAI Press.

Wang, J.; Wu, J.; Hou, Y.; Liu, Y.; Gao, M.; and McAuley, J. J. 2024a. InstructGraph: Boosting Large Language Models via Graph-centric Instruction Tuning and Preference Alignment. In *ACL*, 13492–13510. Association for Computational Linguistics.

Wang, Z.; Dong, Y.; Zeng, J.; Adams, V.; Sreedhar, M. N.; Egert, D.; Delalleau, O.; Scowcroft, J.; Kant, N.; Swope, A.; et al. 2024b. HelpSteer: Multi-attribute Helpfulness Dataset for SteerLM. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 3371–3384.

Wang, Z.; Li, Y.; Guo, Y.; and Wang, S. 2021. Combating Noise: Semi-supervised Learning by Region Uncertainty Quantification. In *NeurIPS*, 9534–9545.

Wu, Y.; Sun, Z.; Yuan, H.; Ji, K.; Yang, Y.; and Gu, Q. 2024. Self-Play Preference Optimization for Language Model Alignment. *CoRR*, abs/2405.00675.

Xie, Y.; Goyal, A.; Zheng, W.; Kan, M.; Lillicrap, T. P.; Kawaguchi, K.; and Shieh, M. 2024. Monte Carlo Tree Search Boosts Reasoning via Iterative Preference Learning. *CoRR*, abs/2405.00451.

Xu, C.; Sun, Q.; Zheng, K.; Geng, X.; Zhao, P.; Feng, J.; Tao, C.; Lin, Q.; and Jiang, D. 2024. WizardLM: Empowering Large Pre-Trained Language Models to Follow Complex Instructions. In *ICLR*. OpenReview.net.

Xu, J.; Lee, A.; Sukhbaatar, S.; and Weston, J. 2023. Some things are more CRINGE than others: Preference Optimization with the Pairwise Cringe Loss. *CoRR*, abs/2312.16682.

Yuan, W.; Pang, R. Y.; Cho, K.; Sukhbaatar, S.; Xu, J.; and Weston, J. 2024. Self-Rewarding Language Models. *CoRR*, abs/2401.10020.

Yuan, Z.; Yuan, H.; Tan, C.; Wang, W.; Huang, S.; and Huang, F. 2023. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*.

Yue, X.; Qu, X.; Zhang, G.; Fu, Y.; Huang, W.; Sun, H.; Su, Y.; and Chen, W. 2024. MAmmoTH: Building Math Generalist Models through Hybrid Instruction Tuning. In *ICLR*. OpenReview.net.

Zheng, L.; Chiang, W.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E. P.; Zhang, H.; Gonzalez, J. E.; and Stoica, I. 2023a. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *NeurIS*.

Zheng, R.; Dou, S.; Gao, S.; Hua, Y.; Shen, W.; Wang, B.; Liu, Y.; Jin, S.; Liu, Q.; Zhou, Y.; Xiong, L.; Chen, L.; Xi, Z.; Xu, N.; Lai, W.; Zhu, M.; Chang, C.; Yin, Z.; Weng, R.; Cheng, W.; Huang, H.; Sun, T.; Yan, H.; Gui, T.; Zhang, Q.; Qiu, X.; and Huang, X. 2023b. Secrets of RLHF in Large Language Models Part I: PPO. *CoRR*, abs/2307.04964.

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