HAF-RM: A Hybrid Alignment Framework for Reward Model Training

Anonymous ACL submission

Abstract

As a quantifiable sentence-wise evaluation method, reward model is becoming increasingly important in alignment, assessment, and even data construction for large language models. Excluding some common tricks like soft-label and margin, most researches share the same training framework proposed by OpenAI, focusing on further improving the performance of reward models from the data aspect. However, we propose a hybrid alignment framework for reward model training (HAF), a new reward model training paradigm which introduces a new constraint in addition to the reward score. By this new supervision, it better aligns the model's output with its preference and therefore achieves more precise prediction of the query-response pair's reward. Experiment results on four public datasets confirm the effectiveness of our proposed framework for training a high quality reward model.

1 Introduction

Recent years have witnessed the development of large language model techniques, especially model pre-training (Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020) and instruction tuning (Wei et al., 2021; Wang et al., 2022). Researchers start to shift their focus from generating correct responses to generating responses that better align (Russell, 2014) with human preferences. The emergence of reward-based (also known as "human preference") algorithms such as reinforcement learning from human feedback (RLHF) (Christiano et al., 2017; Stiennon et al., 2020) has made this possible.

Existing practice of human alignment tuning for large language model mainly relies on external reward models. It is usually trained with human annotated data and provides pseudo ground-truth signals (Christiano et al., 2017; Ouyang et al., 2022) to guide the training process. Despite the availability of numerous sophisticated reward models (Kopf et al., 2023; Ethayarajh et al., 2021), the majority come from industry and lack open-source. In research community, numerous studies point out issues of incorrect and ambiguous preferences in training data of reward model, advocating for enhancements through meticulous data cleansing (Bai et al., 2022; Pitis, 2023). Another line of research explores to utilize external tools or information sources to enhance the generalization of reward models (Li et al., 2023; Sun et al., 2023). These investigative efforts have yielded progress towards the refinement of reward models, but they mainly focus on data or sources used to train the model and leave the training framework of reward models less touched.

The structure of a reward model mainly includes two components, namely a preference space (usually the internal state of a language model) and a mapping function (usually a linear layer with normalization) from the preference space to the reward space. The reward model can then be used to train a policy model to be a better sequence generator. The standard practice for training a reward model uses Bradley-Terry model (Bradley and Terry, 1952) to transform paired rewards into winning probabilities and employs the cross-entropy loss on the output of the mapping function to optimize parameters of both components. We argue that optimize the two parts of the model with single objective can not guarantee a global optimal of the reward model as a whole, and this limits the model's capability to generalize well to data out of the training set.

In this paper, we propose a hybrid alignment framework (HAF) to jointly optimize the reward model and polity model with shared preference space. With the policy loss, we are able to supervise the shared preference space directly while not affecting the mapping function of the reward model, which can be understood as calibrating the shared preference space by aligning sequence probabilities and reward scores separately for better alignment.

We provide both theoretical justifications and experiment results to demonstrate the effectiveness of HAF. In the experiment part, we evaluate the performance of reward model trained by our framework and others resulted from baseline approaches on four public datasets. Results show the advantage of HAF with different policy losses integrated. Further analysis reveals that using additional policy loss is able to improve the performance of policy model calibration, which open a new horizon for training high quality reward model.

2 Approach

In language modelling, the reward model and the policy model play analogous roles. When presented with a query-response pair, the reward model functions as a classifier, aiming to assign elevated scores to preferred responses. Conversely, the policy model acts as a sequence generator expected to allocate greater probability to preferred responses. Despite the formal differences, we believe there is no fundamental difference between them and both models should share a common "internal preference" mappable to the reward score and sequence probability. By calibrating the internal preference of both models, we expect the reward model to exhibit superior generalization capabilities.

2.1 Definition

Let $\mathcal{D} = \{(x_i, y_i, y_i')\}_{i=1}^n$ represents dataset used to train reward model, where x_i, y_i and y_i' are the query, preferred and dispreferred responses respectively. Assuming the reward model outputs the reward of a response y by: $\mathrm{r(x, y)} = \mathrm{F} \circ \phi_1(x, y)$. $\phi_1(x, y)$ denotes the model's internal preference that requires calibration, while F serves as the mapping function from the model's internal preference to the ultimate reward. We use the symbol $\circ$ to signify function nesting $\mathrm{F} \circ \phi_1(x, y) = \mathrm{F}(\phi_1(x, y))$.

In addition to the reward model, we use $\pi(x, y) = \mathrm{K} \circ \phi_2(x, y)$ to represent the policy model, which, given a query-response pair (x, y), outputs the generation probability of y given x. The function K maps the model's internal preference to the generation probability.

The true (optimal) value is denoted as the corresponding letter with an asterisk such as r^*(ground truth reward model), ϕ^*(optimal model preference), F^*(optimal reward score mapping) and K^*(optimal sequence probability mapping).

2.2 Training Loss

The training loss should be designed to bring F^* and ϕ_1 closer to F^* and ϕ^*.

Standard Reward Loss The standard reward loss transforms the reward modeling problem into a probability optimization problem through the Bradley-Terry model, avoiding the issue of uncertain reward values. Treating the problem as a binary classification task yields the negative log-likelihood loss function:

$$\begin{aligned}\mathcal{L}_s &= \min_{F,\phi_1} \mathop{\mathbb{E}}_{d\in\mathcal{D}} \mathrm{D}_1(\mathrm{F} \circ \phi_1(d), \mathrm{F}^* \circ \phi^*(d)) \\ &= \min_r -\log \frac{1}{1 + \exp(\mathrm{r}(x, y) - \mathrm{r}(x, y'))}\end{aligned} \tag{1}$$

where D_1 is a discrepancy measurement for the reward score distribution.

Hybrid Alignment Loss The standard reward loss considers only the reward model alone while neglecting the common internal preference shared with the policy model. By calibrating the preference space, we effectively align both models in a hybrid manner:

$$\begin{aligned}\mathcal{L}_H = \min_{F,\phi,K} \mathop{\mathbb{E}}_{d\in\mathcal{D}} [&\mathrm{D}_1(\mathrm{F} \circ \phi(d), \mathrm{F}^* \circ \phi^*(d)) \\ &+ \alpha \cdot \mathrm{D}_2(\mathrm{K} \circ \phi(d), \mathrm{K}^* \circ \phi^*(d))]\end{aligned} \tag{2}$$

where α is a hyperparameter to balance losses from the reward and polocy model, $\phi(d)$ is the shared model preference which receives gradients from both loss terms and D_2 is a discrepancy measurement for the sequence probability distribution.

2.3 Theoretical Analysis

In practice, functions such as F and ϕ are represented by parameterized models with finite parameters, so they cannot model arbitrary distributions. Here we show that under mild assumptions, using the calibrated reward loss is guaranteed to find a better solution than using the standard reward loss.

Theorem 1. *Unless K can exactly fit K^*, there exists $\epsilon > 0$, such that*

$$\begin{aligned}&\mathop{\mathbb{E}}_{d\in\mathcal{D}} [\mathcal{L}_2(\mathrm{K_H} \circ \phi_\mathrm{H}(d), \mathrm{K}^* \circ \phi^*(d))] \\ &\leqslant \min_\mathrm{K} \mathop{\mathbb{E}}_{x\in\mathcal{D}} [\mathcal{L}_2(\mathrm{K} \circ \phi_\mathrm{s}(d), \mathrm{K}^* \circ \phi^*(d))] - \frac{\epsilon}{\alpha}\end{aligned}$$

where $K_H, \phi_H = \mathop{\mathrm{argmin}}_{\mathrm{K},\phi} \mathcal{L}_H$ in Equation 2 and $\phi_s = \mathop{\mathrm{argmin}}_{\phi} \mathcal{L}_s$ in Equation 1. Intuitively this

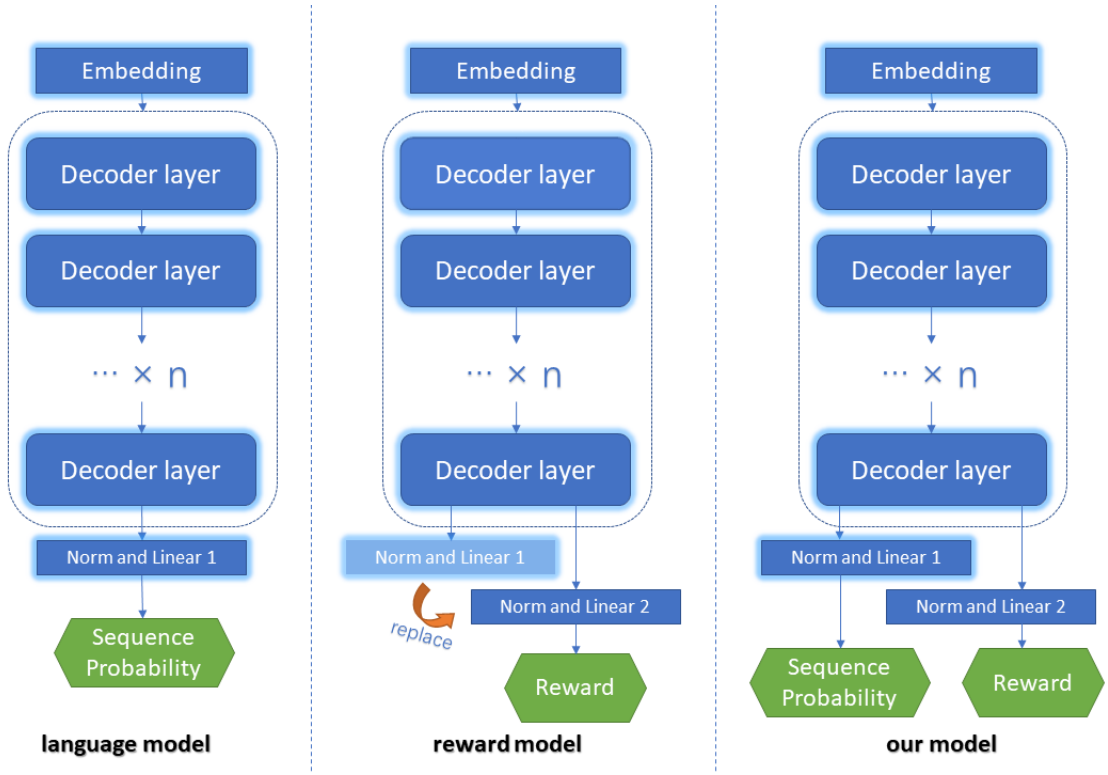


Figure 1: Comparison of our model, language model and reward model. Linear in language model transforms output of decoder layers into sequence's probability; the reward model replaces this linear layer with a randomly-initialized one and output reward; Our model has both linear layers and outputs both.

indicates that *the policy model learned from the joint calibrated loss outperforms the one learned solely from the preference space using the standard reward model*.

Theorem 2. *Given that the gradient of function* K *is finite gradients and within* $[g_{min}, g_{max}]$*, there exists* $k, \delta > 0$*, such that*

$$\mathop{\mathbb{E}}_{x\in\mathcal{D}}[|\phi_H(x) - \phi^*(d)| - |\phi_s(x) - \phi^*(d)|] < \frac{g_{max} - g_{min}}{g_{min}} \mathop{\mathbb{E}}_{x\in\mathcal{D}} |\phi_s(x) - \phi^*(x)| + 2\delta - \frac{\epsilon}{\alpha \cdot k}$$

The detailed derivations for both theorems are provided in Appendix C. Here we obtain an upper bound on the model preference error. By tuning the hyperparameter α, the right term can be strictly negative. In other words, *model preference space trained with our calibrated loss can be strictly closer to the true preference space compared to the standard reward loss*. (In practice, there is no need for exhaustive search, we find $\alpha = 0.2$ already yields satisfactory results)

3 Hybrid Alignment Framework

In Section 2 we discussed about jointly optimizing the reward model and policy model to constrain the preference space $\phi(d)$. The reward model can be simply optimized via Equation 1. In practice, the policy model can use various forms of loss functions that take as input contrastive preference data such as DPO, IPO, etc.

3.1 Difference with Standard Reward Model

As we know, the concurrent decoder-only (Liu et al., 2018) LLM consists of two parts: the first part is composed of a large number of stacked transformer blocks (Vaswani et al., 2017) or similar structures, and the second part is a linear layer accompanied by normalization modules such as LayerNorm (Ba et al., 2016), which maps the output of the transformer blocks into a probability distribution for each token. The derived reward model is similar, except that the linear layer mapping to the vocabulary size is replaced with a linear layer mapping to a reward number. In implementation, the common way to initialize reward model is loading a language model and replace its second part with a LayerNorm and a random initialized Linear layer.

As shown in figure 1, in order to supervise model's predicted probabilities of given responses, the second part of the language model, which is

discarded in the normal implementation, happens to come in handy.

3.2 Implementing Details

In order to significantly reduce the resources required for training, there is a consensus among people to use a finetuned language model to initialize the reward model. However, to our knowledge, there is currently no discussion about the implementation details of the additional linear layer: 1. whether to retain the bias parameter of the linear layer, 2. how to initialize it.

Analysis on the first issue is not hard. With r, W and O respectively representing reward, weight of the linear layer and transformer's output, we have $r = O \cdot W^{n\times 1} + bias$ and $\mathcal{L}r_1, r_2 = -\text{logsigmoid}(r_1 - r_2) = -\text{logsigmoid}[(O_1 - O_2) \cdot W^{n\times 1}]$, which has no relevance with bias when calculating loss. As it deals no influence on training process, we set it to 0 for simplicity.

Inspired by ControlNet (Zhang et al., 2023), we choose zero initialization instead of Kaiming init (He et al., 2015) to initialize linear layer. We conduct a comparative experiment on this issue. Results in Appendix A show that there is little difference between the average results of the two implementations, but zero initialization has a smaller variance.

4 Experiment setup

4.1 Datasets

We use four public datasets to evaluate the performance of our framework, including Anthropic-HH (Bai et al., 2022), chatbot-arena (Zheng et al., 2023), alpaca-human-preference from AlpacaFarm (Dubois et al., 2023), BeaverTails (Ji et al., 2023). Chatbot-arena and alpaca-human-preference do not have original data split for evaluation, we therefore randomly select 10% of samples as test set. We have removed the data of multi-turn conversations from Anthropic-HH. Both Anthropic-HH and BeaverTails contain subsets of datasets, we list all of them as follows.

- **HH-harmless** is Anthropic-HH harmless dataset with fine-grained labels about harmfulness.
- **HH-helpful** is Anthropic-HH helpful dataset with fine-grained labels about helpfulness.
- **HH-total** is the original Anthropic-HH dataset.
- **Beaver** comes from BeaverTails dataset. We select data consisting of one harmless response and one harmful response to form this dataset.
- **Alpaca-human-pref** denotes for alpaca Human Preference dataset.
- **Chatbot-arena** denotes for chatbot-arena dataset.

4.2 Comparative models

We compre our framework with three baselines. Details about the comparative models can be found in Appendix D.

- **Baseline** is the standard reward loss (Eq. 1) which is the most widely used approach to train a reward model.
- **Pointwise** is proposed by Askell et al. (2021) and improved by Zhao et al. (2023). It derives a reward model by fine-tuning the language model to answer "Good" or "Bad" given the query-response pair. The reward equals to the log probability of the model answering "Good".
- **Pairwise** is proposed by Zhao et al. (2023) that formulates the human feedback into a pairwise ranking problem. Given two responses represented by "A" and "B" respectively, it considers the log probability of model outputting "A" as the reward for response 1.
- **HAF** is our hybrid alignment framework that introduces an auxiliary loss function to **Baseline**.

4.3 Implementation Details

Limited by training resources, we use "Phi-2-2.7B"[1] as the main body language model to train the reward model. In our experiments, the input of reward models (ours and OpenAI baseline) are formed as "Instruct: [PROMPT]\nOutput: [RESPONSE]\n<|endoftext|>" where "\n" means line break and "<|endoftext|>" is the end token of a sentence. Detailed experiment setup can be found in Appendix B.

5 Experiment Results

We report the performance of reward models for judging the quality of candidate responses and its effectiveness for downstream tasks. Besides, we present a further analysis to validate the effectiveness of the policy loss for langauge model calibration.

5.1 Intrinsic performance of Reward Models

As a reward model, its fundamental use is to judge whether an answer to a particular question is good

[1] https://huggingface.co/microsoft/phi-2

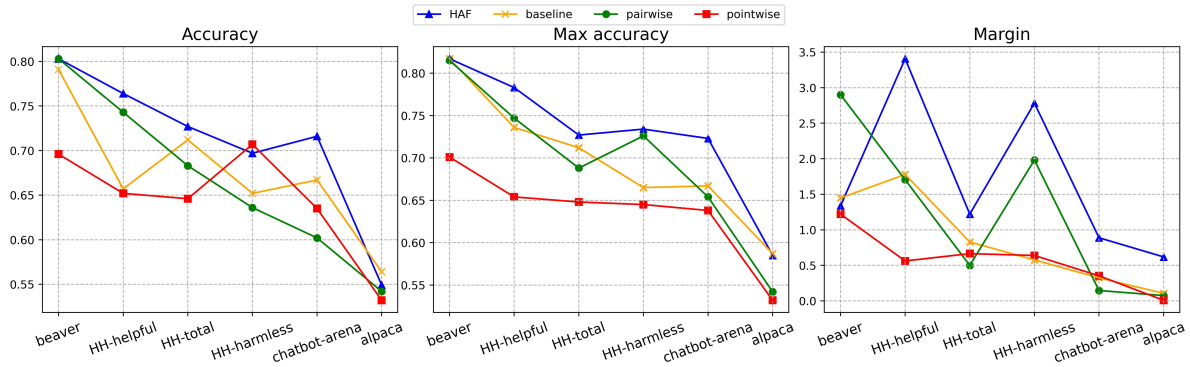


Figure 2: Results of our methods compared with baseline. "Accuracy" denotes the proportion that the chosen response is scored higher on the validation set. "Margin" denotes the mean difference between preferred response's reward and disliked response's reward. The prefix "Max-" denotes the best performance of the model on the test set.

in a specific aspects (or overall) and score it accordingly, with the score difference reflecting the quality of answers. Beside the overall performance of reward models, we also explore the influence of different policy losses and the data imbalance issue to the performance.

5.1.1 Overall Performance

Figure 2 presents the overall results. For our model we used DPO loss as the policy loss and set its coefficient α to 0.2. HAF has higher accuracy than the baseline in most cases, and the score difference between good and bad answers is larger, indicating that the model can more sensitively identify whether an answer is good and give a more accurate high (or low) score. At the same time, those results that are worse than the baseline are generally only slightly worse, which indicates that our method is basically not weaker than the baseline under various circumstances. Considering that we did not seek better test results by changing hyperparameters in the HAF experiment, this also indicates that our method has a higher upper limit and has the potential for further improvement.

5.1.2 Influence of Different Policy Loss

We further explore the influence of various policy losses to the performance of HAF.

- **DPO** is the default policy loss in our experiments with hyperparameter $\alpha = 0.2$, suffix "0.8" and "1.2" are the results when α is 0.8 or 1.2 respectively.
- **IPO** is an advanced version of DPO which handles the "optimize-regardless-of-reference" problem by changing the optimal solution of learning target. Different suffixes represent different variants which we detail in Appendix F.
- **hinge** means the use of hinge loss in replace of logsigmoid.

Table 1 shows the results. In practice, we found that using the IPO loss alone is highly unstable. Therefore, we derived a variant by scaling: IPO-full. Due to space limitations, we have placed the specific calculation methods in Appendix F.

Arbitrary proportions of DPO loss consistently yield satisfactory results, while hinge loss demonstrates the most remarkable performance. However, hinge loss isn't always stable, which may not assure the convergence of the model. Nevertheless, as long as the model converges after introducing the policy loss, we will obtain a more powerful model.

5.1.3 Dealing with Data Imbalance

In reward modeling, two of the most classic problems are determining helpfulness and harmlessness. In this experiment, we used the Beaver dataset to study the stability of our method with respect to imbalanced data. The Beaver dataset contains three types of data samples with different pair combination of safe and unsafe samples(type 1: safe-safe, type 2: safe-unsafe, type 3: unsafe-unsafe). We removed the pair with two unsafe samples and compared our method with the baseline by studying the performance of the model trained on different proportions of the first two types of data. Table 2 presents the results for the influence of imbalance data for our framework.

5.2 Downstream Task

We further demonstrated the reliability of reword model through downstream tasks. Firstly, we sampled two responses for each question using the language model to be trained. Then, the reward model

dataset	policy loss	accuracy	accuracy-max	margin	variance	normed variance
chatbot-arena	**baseline**	0.667	0.667	-	-	-
	DPO	0.716	0.723	0.888	0.324	0.411
	DPO-0.8	0.695	0.700	0.563	0.182	0.574
	DPO-1.2	0.736	0.751	1.614	0.855	0.328
	IPO-full	0.701	0.705	0.486	0.173	0.732
	hinge	**0.738**	**0.760**	2.327	1.612	**0.298**
HH-total	**baseline**	0.698	0.712	-	-	-
	DPO	**0.727**	**0.727**	**1.221**	0.511	**0.343**
	DPO-0.8	**0.715**	0.722	**1.146**	0.543	0.413
	DPO-1.2	0.701	0.702	**1.112**	0.565	0.457
	IPO-full	0.533	0.537	0.119	0.104	7.344
	hinge	**0.723**	**0.725**	**1.154**	0.479	**0.360**

Table 1: Results for different policy loss. **margin** is the mean difference between rewards for better responses and worse responses, while **variance** is the variation of the difference. **normed variance** equals to variance when margin is scaled to 1. Low normed variance tends to have better stability as well as higher accuracy for binary comparison. We omit some undesirable results of IPO-series.

dataset	ratio	acc	accuracy-t0	accuracy-t1	accuracy-t2	margin
Beaver	2.0	0.778(+0.024)	0.670(+0.027)	0.838(+0.020)	0.785(+0.026)	0.752(+0.052)
	1.0	0.793(+0.042)	0.692(+0.054)	0.855(+0.045)	0.795(+0.033)	0.737(+0.003)
	0.5	0.784(+0.001)	0.683(-0.022)	0.838(+0.014)	0.792(+0.002)	0.727(+0.008)
	0.2	0.758(-0.008)	0.674(-0.040)	0.810(+0.011)	0.759(-0.007)	0.687(-0.138)
	0.0	0.745(+0.005)	0.656(+0.027)	0.784(-0.009)	0.759(+0.005)	0.875(+0.215)

Table 2: Results for the influence of imbalanced data. The second column is the ratio of type 2 (safe-unsafe) to type 1 (safe-safe). The suffix "tx" means type x. Type 2 refers to the data that two responses are both safe, while type 0 is two unsafe responses and type 1 is the other one. The number in parentheses represents the difference between our method and the baseline.

was used to evaluate the responses, and the better response was taken as the target response to train the model. Finally, the trained model was evaluated by GPT-4 on two settings, namely, in-domain (IND, different questions on the same dataset) and out-of-domain (OOD, questions on other datasets).

Table 3 presents the results. Regardless of the amount of data used for training, language model supervised by our reward model has a significant advantage over the baseline, and its performance in OOD problems is also outstanding.

5.3 Policy Model Calibration

To validate the effectiveness of policy loss for policy model calibration, we present a simulation experiment. In the original setup, the reward model use the language model to be trained as the policy model. This reduces the difficulty of policy model calibration. In our setup, we construct a "perturbed" policy model based on the language model and evaluate how well the reward model can calibrate its preferences.

It is non-trivial to express the model preferences explicitly. For simplicity, we use the output matrix of the transformer layers as an indirect estimate of the model preference. We denote this matrix "**preference vector**" for the rest of this section. We examine the changes in preference vectors under the baseline and our method.

To construct a "perturbed" language model, we multiply each of its parameters by an additional random number. The distribution of the random numbers for the same experiment is uniform in the range $(1-k, 1+k)$. The "perturbed" model obtained from this process is used as the ground truth(gt).

In more detail, we use the gt model to compute the probability for each pair of responses for a given question. We then select the answer with the higher probability as the winner in the pairwise comparison, and construct preference data. Additionally, we retain the intermediate results during the inference, which correspond to the gt preference vector $\phi^*(y|x)$ for each QA pair(x, y). During the training process, we uniformly test the model's preference $\phi(\cdot)$ for 40 times on the validation set, and calculate the preference difference between the training model and gt model, as well as the corresponding correct rates of pairwise comparison.

dataset	#data	chatbot		HH		helpful		beaver	
		IND	OOD	IND	OOD	IND	OOD	IND	OOD
alpaca	8k	**70.2**	**59.8**	**52.8**	44.0	**55.5**	**56.2**	**64.7**	**58.6**
chatbot	18k	**60.0**	**62.4**	42.4	49.3	51.2	51.9	**53.0**	48.8
beaver	100k	**60.7**	**58.1**	**57.0**	**57.1**	46.0	45.7	**63.5**	**65.8**

Table 3: Win rates of the model supervised by our reward model to the model supervised by baseline reward model on different settings. Generation results are judged by GPT-4. **chatbot**, **HH**, **helpful** are the datasets used to train reward model and the first column shows the datasets used to train language model.

data	scaled absolute($\times 10^4$)↓	mean difference↓	accuracy↑	accuracy-baseline
R9	4.044(-0.578)	0.049(+0.023)	0.639(+0.091)	0.548
R7	4.188(-0.597)	0.021(-0.426)	0.597(+0.067)	0.530
R6	4.300(-0.612)	0.070(-0.016)	0.556(-0.016)	0.572
R5	4.477(-0.640)	0.012(-0.405)	0.599(+0.093)	0.506
R4	4.722(-0.666)	0.045(-0.473)	0.609(+0.104)	0.505
R2	4.935(-0.677)	0.045(-0.424)	0.506(-0.020)	0.526
R0	6.575(-0.016)	0.133(-0.147)	0.524(+0.007)	0.517

Table 4: Results for calibration experiments. The number in the parentheses is the difference of results on this metric between HAF(ours) and baseline. It shows clearly that the hidden preference of the model trained by HAF is closer to true preference for metrics calculating preference vector's difference are much smaller than baseline's.

5.3.1 Evaluation Metrics

Calculating metrics for comparing two preference vectors is a complex task. Since preference vectors are scalable and their components are interchangeable, We use the "**scaled absolute**" (1-norm of the difference of the interchanged and scaled preference vector and gt vector) and "**mean difference**" (absolute mean difference of the two vectors) as metrics to calculate preference difference from gt preference. Smaller **scaled absolute** and smaller **mean difference** indicate better calibration of the model preference. Details can be found in Appendix E.

5.3.2 Results

Table 4 presents the results. The numbers in the "data" column represent the range of random numbers multiplied to generate the target model. For instance, "7" represents a coefficient randomly selected from the range (0.7, 1.3), ensuring that the mean of the random numbers is 1. The smaller the number, the greater the variation in the model, which in turn increases the difficulty of learning.

Our method has demonstrated outstanding fitting performance across various levels of unknown preference distributions. Besides, the accuracy of the baseline is just above 0.50, which means the baseline has little capability of learning preference of a new language model. This failure is not reported in existing research, because existing setup usually employs the language model to be tuned as the policy model in the reward model. Therefore, reduce the difficulty for policy calibration. When the reward model uses different policy model, the baseline training framework fails.

6 Related Work

There are two lines of research closely related to this paper, namely, lanugage model calibration and reward model.

6.1 Language Model Calibration

The word "calibration" was first introduced by Dawid (1982) for judging the effectiveness of a forecast, while Foster and Vohra (1998) gave a rigorous definition and generalize it from point-wise forecast to distribution forecast. After this, researches about calibration have been flourishing (Kuleshov and Liang, 2015; Guo et al., 2017; Kuleshov et al., 2018). The previous work on calibration in the linguistic domain has put forward two crucial points: first, the inconsistency between the token-wise training objective of the language model and the sequence-wise evaluation metric (Liu and Liu, 2021), and second, the exposure bias (Ranzato et al., 2015), which refers to the inconsistency between the training phase and the inference phase (auto-regressive). Based on prior works, SLiC (Zhao et al., 2022) used the phrase "not being calibrated" to describe the phenomenon that MLE-trained model assigns relatively high probability on some plausible sequences, referring to lack of supervision on these unidiomatic sentences and brought out a novel method to align

candidates' sequence likelihoods according to their similarities to the target sequence in the model's latent space and achieved favorable results in downstream tasks; following this work, SLiC-HF (Zhao et al., 2023) tried to boost the results with reinforcement learning.

In fact, SLiC and SLiC-HF have extended the meaning of the term "calibration" from aligning the predicted distribution with the true distribution to aligning the predicted distribution with the label's latent space. Our idea is similar to theirs, and our method will align the output distribution of the reward model with the true preference, thereby make it output more accurate rewards.

6.2 Reward Model

Modeling human language preferences using a reward model (model that outputs preference values based on questions and answers) was proposed by Christiano et al. (2017), then the explosive growth of research on reward models (McKinney et al., 2023) and large language models (Wei et al., 2022; Park et al., 2023; Zheng et al., 2023) emerged after the popularity of ChatGPT.

From training to practical applications, an increasing number of studies have also featured the presence of quantifiable preferences(usually known as "reward"). For example, RLHF (Christiano et al., 2017; Stiennon et al., 2020) uses the PPO algorithm (Schulman et al., 2017) to maximize the reward of the policy model; RAFT (Dong et al., 2023) and RRHF (Yuan et al., 2023) remove substandard data by scoring the candidate responses with reward model; LLM-as-a-judge (Zheng et al., 2023) employs GPT-4 to score the text.

Therefore, how to construct a model offering explicit preference feedback has naturally become a focal point of much research. In order to train a precise and robust reward model, many studies start from training with human preference data, and many works in the data field are largely centered around this. Touvron et al. (2023) and Zhao et al. (2022) provided different methods for using ranking data; Wang et al. (2024) explored ways of measuring strength of the data; while with regard to datasets themselves, Azar et al. (2023), Knox et al. (2022) and Hong et al. (2022) analyzed the impact of data preference strength on training from theoretical or practical perspectives. In addition, similar to the RAG technique (Lewis et al., 2020) in large language models, many methods (Li et al., 2023; Sun et al., 2023) using external tools or references have also emerged, injecting new vitality into the development of reward models.

Although many data-oriented methods have greatly enhanced the performance of reward models, the field of reward model optimization has been rarely explored. Currently, the training of reward models basically follows the process proposed by OpenAI (Christiano et al., 2017). It involves initializing the reward model using a fine-tuned model, then transforming the model's predictions into probability values through the Bradley-Terry model, and optimizing these probabilities using cross-entropy loss. Considering the widespread practical applications of reward models, the attention given to their training paradigms does not match their importance.

7 Conclusion

In this paper, we have extended and improved the training framework of the current reward model. By splitting the training mechanism of the reward model into two stages: calibrating model preference and optimizing mapping function, we have introduced new constraints to calibrate the reward model, significantly improving the training effectiveness. We have theoretically demonstrated this and validated the reliability of our method through systematic experiments.

As the reward model exhibits a more precise sentence-wise explicit representation capability compared to the language model, it can fully serve as an important component in weak-to-strong generalization. Future research should more meticulously investigate the potential of the reward model and its variants in various tasks. Additionally, we should also study the reliability of the core that constructs the current reward model, i.e., the Bradley-Terry Model, and whether the logistic distribution is the best prior distribution in reward modeling, awaiting exploration by researchers.

Limitations

This study has potential limitations. First, we utilize Phi-2-2.7B as the base model to conduct all the experiments due to computing resource limits. We plan to investigate the use of other models to obtain a more general conclusion. Second, the model trained with our method actually has the ability of text generation, but we have not extensively studied this capability. This is a promising method that we will investigate further in our future work.

Impact Statements

This paper presents work whose goal may benefit the training of large language models in the field of deep learning. Among the many possible consequences, we do not believe that there is a significant possibility of adverse effects on society.

References

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, T. J. Henighan, Andy Jones, Nicholas Joseph, Benjamin Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, John Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Christopher Olah, and Jared Kaplan. 2021. A general language assistant as a laboratory for alignment. *ArXiv*, abs/2112.00861.

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2023. A general theoretical paradigm to understand learning from human preferences. *ArXiv*, abs/2310.12036.

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *ArXiv*, abs/1607.06450.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, T. J. Henighan, Nicholas Joseph, Saurav Kadavath, John Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Christopher Olah, Benjamin Mann, and Jared Kaplan. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *ArXiv*, abs/2204.05862.

Ralph Allan Bradley and Milton E. Terry. 1952. RANK ANALYSIS OF INCOMPLETE BLOCK DESIGNS: THE METHOD OF PAIRED COMPARISONS. *Biometrika*, 39(3-4):324–345.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Paul Francis Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *ArXiv*, abs/1706.03741.

A. Philip Dawid. 1982. The well-calibrated bayesian. *Journal of the American Statistical Association*, 77:605–610.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

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

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

Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. 2021. Understanding dataset difficulty with v-usable information. In *International Conference on Machine Learning*.

Dean P. Foster and Rakesh V. Vohra. 1998. Asymptotic calibration. *Biometrika*, 85(2):379–390.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034.

Joey Hong, Kush Bhatia, and Anca D. Dragan. 2022. On the sensitivity of reward inference to misspecified human models. *ArXiv*, abs/2212.04717.

Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2023. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *ArXiv*, abs/2307.04657.

W. B. Knox, Stephane Hatgis-Kessell, Serena Booth, Scott Niekum, Peter Stone, and Alessandro Allievi. 2022. Models of human preference for learning reward functions. *ArXiv*, abs/2206.02231.

Andreas Kopf, Yannic Kilcher, Dimitri von Rutte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Rich'ard Nagyfi, ES Shahul, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. Openassistant conversations - democratizing large language model alignment. *ArXiv*, abs/2304.07327.

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. 2018. Accurate uncertainties for deep learning using calibrated regression. *ArXiv*, abs/1807.00263.

Volodymyr Kuleshov and Percy Liang. 2015. Calibrated structured prediction. In *Neural Information Processing Systems*.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *ArXiv*, abs/2005.11401.

Lei Li, Yekun Chai, Shuohuan Wang, Yu Sun, Hao Tian, Ningyu Zhang, and Hua Wu. 2023. Tool-augmented reward modeling. *ArXiv*, abs/2310.01045.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam M. Shazeer. 2018. Generating wikipedia by summarizing long sequences. *ArXiv*, abs/1801.10198.

Yixin Liu and Pengfei Liu. 2021. SimCLS: A simple framework for contrastive learning of abstractive summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 1065–1072, Online. Association for Computational Linguistics.

Lev McKinney, Yawen Duan, David Krueger, and Adam Gleave. 2023. On the fragility of learned reward functions. *ArXiv*, abs/2301.03652.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*.

Silviu Pitis. 2023. Failure modes of learning reward models for LLMs and other sequence models. In *ICML 2023 Workshop The Many Facets of Preference-Based Learning*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732.

Stuart Russell. 2014. Of myths and moonshine. online.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan J. Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. 2020. Learning to summarize from human feedback. *ArXiv*, abs/2009.01325.

Zhiqing Sun, Sheng Shen, Shengcao Cao, Haotian Liu, Chunyuan Li, Yikang Shen, Chuang Gan, Liangyan Gui, Yu-Xiong Wang, Yiming Yang, Kurt Keutzer, and Trevor Darrell. 2023. Aligning large multimodal models with factually augmented rlhf. *ArXiv*, abs/2309.14525.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Neural Information Processing Systems*.

Bing Wang, Rui Zheng, Luyao Chen, Yan Liu, Shihan Dou, Caishuang Huang, Wei Shen, Senjie Jin, Enyu Zhou, Chenyu Shi, Songyang Gao, Nuo Xu, Yuhao Zhou, Xiaoran Fan, Zhiheng Xi, Jun Zhao, Xiao Wang, Tao Ji, Hang Yan, Lixing Shen, Zhan Chen, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yuanyuan Jiang. 2024. Secrets of rlhf in large language models part ii: Reward modeling. *ArXiv*, abs/2401.06080.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. In *Annual Meeting of the Association for Computational Linguistics*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. *ArXiv*, abs/2109.01652.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and

Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Feiran Huang. 2023. Rrhf: Rank responses to align language models with human feedback without tears. *ArXiv*, abs/2304.05302.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding conditional control to text-to-image diffusion models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3813–3824.

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

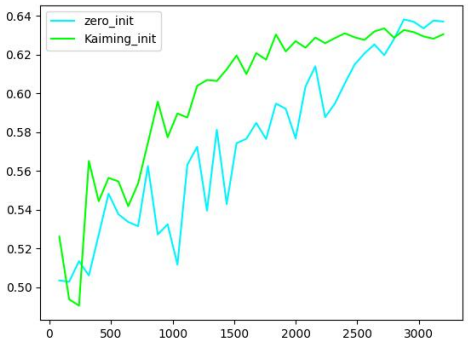
Yao Zhao, Misha Khalman, Rishabh Joshi, Shashi Narayan, Mohammad Saleh, and Peter J. Liu. 2022. Calibrating sequence likelihood improves conditional language generation. *ArXiv*, abs/2210.00045.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Haotong Zhang, Joseph Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *ArXiv*, abs/2306.05685.

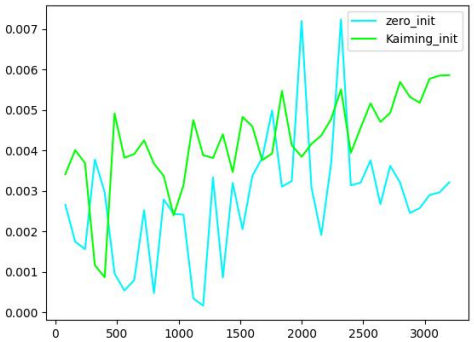
A Zero Initialization v.s. Kaiming Initialization

We conducted multiple trainings on the same dataset using identical configurations, except for the random seed, and presented the trend of the model's accuracy on the validation set with the change of training iterations in Figure 3. We found that in terms of the final accuracy, the results of the two different initialization methods were similar. However, for a long period of time, and even after the end of the training, the performance fluctuation of the model using Kaiming initialization was significantly larger than that of zero initialization. The variance of the former was almost twice that of the latter. Therefore, for the sake of training stability, zero initialization is the more appropriate choice. However, if conditions allow for multiple trainings to select the best results, then Kaiming initialization is more likely to bring you surprises.

Figure 4 shows some sample results. The Kaiming initialization tends to exhibit more extreme behavior. At times, it can rapidly improve performance with well-chosen initialization parameters, but often, the model may fail to train effectively due to poorly chosen initialization parameters. In contrast, zero initialization is more stable. However, since it starts learning from an all-zero parameter setting, it does not lead to the ideal scenario of rapidly improving performance at the beginning of training.

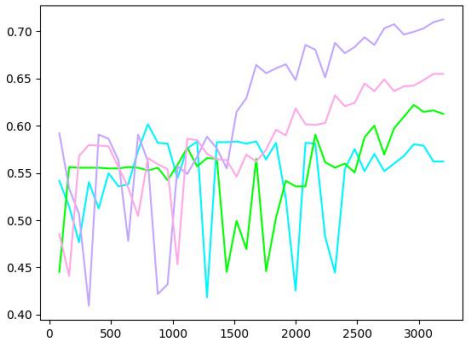


(a) mean accuracy

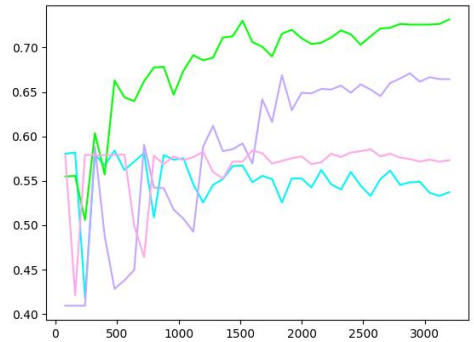


(b) variance of accuracy

Figure 3: zero init v.s. Kaiming init



(a) zero initialization



(b) Kaiming initialization

Figure 4: sample results of training with different initialization methods

setup	value	setup	value	setup	value
learning rate	1e-5	**optimizer**	AdamW	**precision**	bf16
batch size	16	**adam_beta1**	0.9	**max gradient norm**	1.0
training steps	3200	**adam_beta2**	0.999	**max sequence length**	512
evaluation steps	0.025	**weight_decay**	0.0	**global random seed**	0
framework	pytorch	**adam_epsilon**	1e-5		

Table 5: Default setup

B Experiments Setup

Our setup is shown in Table 5.

C Mathematical Derivations

C.1 Inequality Scaling

$$\begin{aligned}
&\min_{\mathrm{F},\phi,\mathrm{K}} \mathop{\mathbb{E}}_{x\in\mathcal{D}}[\mathcal{L}_1(\mathrm{F}\circ\phi(x),\mathrm{F}^*\circ\phi^*(x)) \\
&\quad + \alpha\cdot\mathcal{L}_2(\mathrm{K}\circ\phi(x),\mathrm{K}^*\circ\phi^*(x))] \\
&\leqslant \min_{\mathrm{K},\phi=\phi_1,\mathrm{F}=\mathrm{F}_1} \mathop{\mathbb{E}}_{x\in\mathcal{D}}[\mathcal{L}_1(\mathrm{F}\circ\phi(x),\mathrm{F}^*\circ\phi^*(x)) \\
&\quad + \alpha\cdot\mathcal{L}_2(\mathrm{K}\circ\phi(x),\mathrm{K}^*\circ\phi^*(x))] \\
&= \mathop{\mathbb{E}}_{x\in\mathcal{D}}[\mathcal{L}_1(\mathrm{F}_1\circ\phi_1(x),\mathrm{F}^*\circ\phi^*(x)) \\
&\quad + \min_{\mathrm{K}} \alpha\cdot\mathcal{L}_2(\mathrm{K}\circ\phi_1(x),\mathrm{K}^*\circ\phi^*(x))]
\end{aligned}$$

With $\mathrm{K}_3 = \mathop{\mathrm{argmin}}_{\mathrm{K}}\mathcal{L}_2(\mathrm{K}\circ\phi_1(x),\mathrm{K}^*\circ\phi^*(x))$, we have:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{x\in\mathcal{D}}[\mathcal{L}_1(\mathrm{F}_2\circ\phi_2(x),\mathrm{F}^*\circ\phi^*(x)) \\
&\quad + \alpha\cdot\mathcal{L}_2(\mathrm{K}_2\circ\phi_2(x),\mathrm{K}^*\circ\phi^*(x))] \\
&\leqslant \mathop{\mathbb{E}}_{x\in\mathcal{D}}[\mathcal{L}_1(\mathrm{F}_1\circ\phi_1(x),\mathrm{F}^*\circ\phi^*(x)) \\
&\quad + \alpha\cdot\mathcal{L}_2(\mathrm{K}_3\circ\phi_1(x),\mathrm{K}^*\circ\phi^*(x))] \\
&\leqslant \mathop{\mathbb{E}}_{x\in\mathcal{D}}[\mathcal{L}_1(\mathrm{F}_2\circ\phi_2(x),\mathrm{F}^*\circ\phi^*(x)) \\
&\quad + \alpha\cdot\mathcal{L}_2(\mathrm{K}_3\circ\phi_1(x),\mathrm{K}^*\circ\phi^*(x))]
\end{aligned}$$

The second inequality holds because of equation **??**.

In practical settings, "$\leqslant$"s are unlikely to hold at the same time, and there exists a little gap $\epsilon > 0$ such that

$$\begin{aligned}
&\mathop{\mathbb{E}}_{x\in\mathcal{D}}[\mathcal{L}_1(\mathrm{F}_2\circ\phi_2(x),\mathrm{F}^*\circ\phi^*(x)) \\
&\quad + \alpha\cdot\mathcal{L}_2(\mathrm{K}_2\circ\phi_2(x),\mathrm{K}^*\circ\phi^*(x))] \\
&\leqslant \mathop{\mathbb{E}}_{x\in\mathcal{D}}[\mathcal{L}_1(\mathrm{F}_2\circ\phi_2(x),\mathrm{F}^*\circ\phi^*(x)) \\
&\quad + \alpha\cdot\mathcal{L}_2(\mathrm{K}_3\circ\phi_1(x),\mathrm{K}^*\circ\phi^*(x))] - \epsilon
\end{aligned}$$

Then, there goes

$$\begin{aligned}
&\mathop{\mathbb{E}}_{x\in\mathcal{D}}[\mathcal{L}_2(\mathrm{K}_2\circ\phi_2(x),\mathrm{K}^*\circ\phi^*(x))] \\
&\leqslant \mathop{\mathbb{E}}_{x\in\mathcal{D}}[\mathcal{L}_2(\mathrm{K}_3\circ\phi_1(x),\mathrm{K}^*\circ\phi^*(x))] - \frac{\epsilon}{\alpha}
\end{aligned}$$

Then we get Ineq. 2.

C.2 Derive the Final Inequality with the 3 Properties

convergence:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{x\in\mathcal{D}}[\mathcal{L}_2(\mathrm{K}\circ\phi(x),\mathrm{K}^*\circ\phi^*(x))] \\
&= \mathop{\mathbb{E}}_{x\in\mathcal{D}} k|\mathrm{K}\circ\phi(x) - \mathrm{K}^*\circ\phi^*(x)|
\end{aligned} \tag{3}$$

separating little disturbance:

$$\mathop{\mathbb{E}}_{x\in\mathcal{D}}|\mathrm{N}\circ\phi(x)| < \delta \tag{4}$$

holds for any converged model$(\mathrm{K},\phi) = (\mathrm{K}^* + \mathrm{N},\phi)$

gradient scaling:

$$g_{min} < \nabla\mathrm{K}^* < g_{max} \tag{5}$$

Based on these three properties, we can derive the result from Ineq. 2.

Ineq. 2

$$\begin{aligned}
&\overset{\text{Eq. 3}}{\Longrightarrow} \mathop{\mathbb{E}}_{x\in\mathcal{D}}|\mathrm{K}_2\circ\phi_2(x) - \mathrm{K}^*\circ\phi^*(x)| \\
&\leqslant \mathop{\mathbb{E}}_{x\in\mathcal{D}}|\mathrm{K}_3\circ\phi_1(x) - \mathrm{K}^*\circ\phi^*(x)| - \frac{\epsilon}{\alpha\cdot k} \\
&\overset{\text{Ineq. 4}}{\Longrightarrow} \mathop{\mathbb{E}}_{x\in\mathcal{D}}|\mathrm{K}^*\circ\phi_2(x) - \mathrm{K}^*\circ\phi^*(x)| - \delta \\
&< \mathop{\mathbb{E}}_{x\in\mathcal{D}}|\mathrm{K}^*\circ\phi_1(x) - \mathrm{K}^*\circ\phi^*(x)| + \delta - \frac{\epsilon}{\alpha\cdot k}
\end{aligned} \tag{6}$$

$$\begin{aligned}
&\overset{\text{Ineq. 5}}{\Longrightarrow} \\
&g_{min}\mathop{\mathbb{E}}_{x\in\mathcal{D}}[|\phi_2(x) - \phi^*(x)| - |\phi_1(x) - \phi^*(x)|] < \\
&(g_{max} - g_{min})\mathop{\mathbb{E}}_{x\in\mathcal{D}}|\phi_1(x) - \phi^*(x)| + 2\delta - \frac{\epsilon}{\alpha\cdot k}
\end{aligned} \tag{7}$$

which is Theorem 2.

C.3 A More Elegant Approximation of the Final Inequality

Bounding from Ineq. 6 to Ineq. 7 is very loose and both ϕ_1 and ϕ_2 are close to ϕ^*. Consequently, with

$$\Delta(\phi_1,\phi_2) = \mathop{\mathbb{E}}_{x\in\mathcal{D}}[|\phi_2(x) - \phi^*(x)| - |\phi_1(x) - \phi^*(x)|]$$

Ineq. 6 is also approximated by the following inequality:

$$\mathop{\mathbb{E}}_{x\in\mathcal{D}}[\nabla \mathrm{K}^*(x)] \mathop{\mathbb{E}}_{x\in\mathcal{D}} \Delta(\phi_1, \phi_2) < 2\delta - \frac{\epsilon}{\alpha \cdot k} \quad (8)$$

D Baseline Model

SLiC-HF(Zhao et al., 2023) proposed "pointwise reward model" and "pairwise ranking model". Unlike traditional baselines that are designed from a probabilistic perspective, these two methods do not treat the model as a scorer but rather retain its features as a language model and train it using instruction finetuning (IFT). Since a language model can output probability distributions, a specially trained language model can also be used as a classifier to categorize the quality (or safety, helpfulness, completeness, simplicity and so on) of answers. The two methods proposed construct prompt-answer pair data from preference data and then train the model, with the input and expected output of these two methods shown in Figure 5.

Actually, models in SLiC-HF served as binary classifiers to distinguish the quality of the responses. We consider the model's normalized probability for the corresponding answer as the reward, which naturally leads to a reward model that can output continuous values. We have experimented with various methods to optimize the model, including: IFT (cross-entropy loss, which increases the probability of the model providing correct answers); DPO, IPO, and variants of IPO (increasing the difference in the model's probabilities of giving correct and incorrect answers). It is important to note that the loss function used to train model in the baseline is a special case of DPO without reference model ($\tau = 1$), so we consider it together with DPO. The specific calculation method for the loss function can be found in Appendix F. However, it is important to note that we do not have a reference model here, so the term l_r in the formula is always 0.

We can tell from Table 6 that these two methods cannot compete with the ordinary baseline, except in tasks related to harmful or some bad issues. We consider it's an "inborn" capability for language model to identify bad answers from many answers that are not so bad. For we inject much fewer harmful knowledge into model than normal ones, so the representations of these two should be divergent, which also leads to the sensitivity of the model to non-safe sentences. Apart from harmful judgments, these two methods perform far below the passing line in other tasks, especially the pointwise reward model, which is almost impossible to train. This may be due to the insufficient number of model parameters and the significant difference between the generation task and the classification task. Small models are not enough to implement judgments internally and output them in natural language. Whether this problem will be alleviated with the increase of the model size needs to be answered by future research.

E Metrics For Preference Vector

We use signals in Section 2. For a model preference vector $\vec{v} = \phi(x)$, it is not an accessible way to evaluate the precision of it by calculating some metrics that are directly relative to the difference of this vector and the ground truth preference $\vec{v}^*$ such as L_1 norm or L_2 norm, because the model's learned mapping F is not equal to the real mapping F^*. For example, let R be a certain input of $\phi(\cdot)$, and $\phi(R) = (1,2)^T$, $\phi^*(R) = (2,4)^T$, define $\mathrm{F}(\vec{x}) = \mathrm{F}((x_1,x_2)^T) = x_1 + x_2$, $\mathrm{F}^*(\vec{x}) = \mathrm{F}^*((x_1,x_2)^T) = \frac{1}{2}(x_1 + x_2)$, then we have $\mathrm{F}(\phi(R)) = \mathrm{F}^*(\phi^*(R))$ while $\phi(R) \neq \phi^*(R)$. To avoid, or at least reduce the error made of something like this, we should design a new metric.

As mentioned above, the preference vector has the property of scalability. If the preference vector is close to the ground truth after scaling, it should be considered as being close. In addition, the components of the vector should be interchangeable, and the vectors produced by component interchange should also be equivalent.

Based on these two properties, we have designed a unique metric. Notations: $L_1(\vec{x})$ is the 1-norm of $\vec{x}$, $\vec{v}_s = \mathrm{sort}(\vec{v})$, $\vec{v}_s^* = \mathrm{sort}(\vec{v}^*)$, $n_1 = L_1(\vec{v})$, $n_1^* = L_1(\vec{v}^*)$.

Scaled absolute:

$$\mathcal{S}_1(\vec{v}, \vec{v}^*) = L_1(\frac{n_1^*}{n_1} \cdot \vec{v}_s - \vec{v}_s^*)$$

Besides, we remained the average difference of

dataset	loss func	accuracy	accuracy-max	prob	prob var	method
HH-harmless	DPO	0.623(-0.029)	0.644(-0.021)	0.609	0.017	**pointwise**
	IFT	0.636(-0.017)	0.645(-0.021)	0.604	0.012	
	IPO-full	0.634(-0.018)	0.645(-0.021)	0.623	0.022	
	IPO-sqrt	0.630(-0.022)	0.647(-0.018)	0.617	0.020	
HH-total	DPO	0.523(-0.175)	0.528(-0.184)	0.502	<0.001	
	IFT	0.514(-0.183)	0.541(-0.171)	0.503	<0.001	
	IPO-full	0.646(-0.052)	0.648(-0.064)	0.612	0.010	
	IPO-sqrt	0.527(-0.171)	0.534(-0.178)	0.505	0.001	
HH-harmless	DPO	0.686(+0.034)	0.729(+0.064)	0.676	0.020	**pairwise**
	IFT	0.696(+0.044)	0.730(+0.065)	0.685	0.017	
	IPO-full	0.685(+0.033)	0.716(+0.051)	0.681	0.019	
	IPO-sqrt	0.707(+0.055)	0.726(+0.061)	0.703	0.022	
HH-total	DPO	0.683(-0.014)	0.688(-0.024)	0.604	0.005	
	IFT	0.506(-0.192)	0.522(-0.190)	0.501	<0.001	
	IPO-full	0.506(-0.192)	0.516(-0.196)	0.500	<0.001	
	IPO-sqrt	0.502(-0.195)	0.518(-0.194)	0.501	<0.001	
chatbot-arena	DPO	0.635(-0.032)	0.638(-0.029)	0.570	0.005	
	IFT	0.619(-0.048)	0.634(-0.033)	0.564	0.004	
	IPO-full	0.598(-0.069)	0.598(-0.069)	0.509	<0.001	
	IPO-sqrt	0.598(-0.069)	0.598(-0.069)	0.509	<0.001	
	IPO	0.631(-0.036)	0.636(-0.031)	0.551	0.003	
	hinge	0.593(-0.074)	0.594(-0.073)	0.589	0.025	

Table 6: Results of the two methods with different loss function. Most of the numbers in the parentheses is negative, which means that the model trained by these methods often behaves worse than logsigmoid-trained baseline. **accuracy**: the probability that correct answer's predicted logit is higher than incorrect one's (here "answer" refers to "good" and "bad" in pointwise reward model, or "A" and "B" in pairwise ranking model); **accuracy-max**: max accuracy on evaluation set during training; **prob**: mean of correct answer's probability; **prob var**: variation of correct answer's probability.

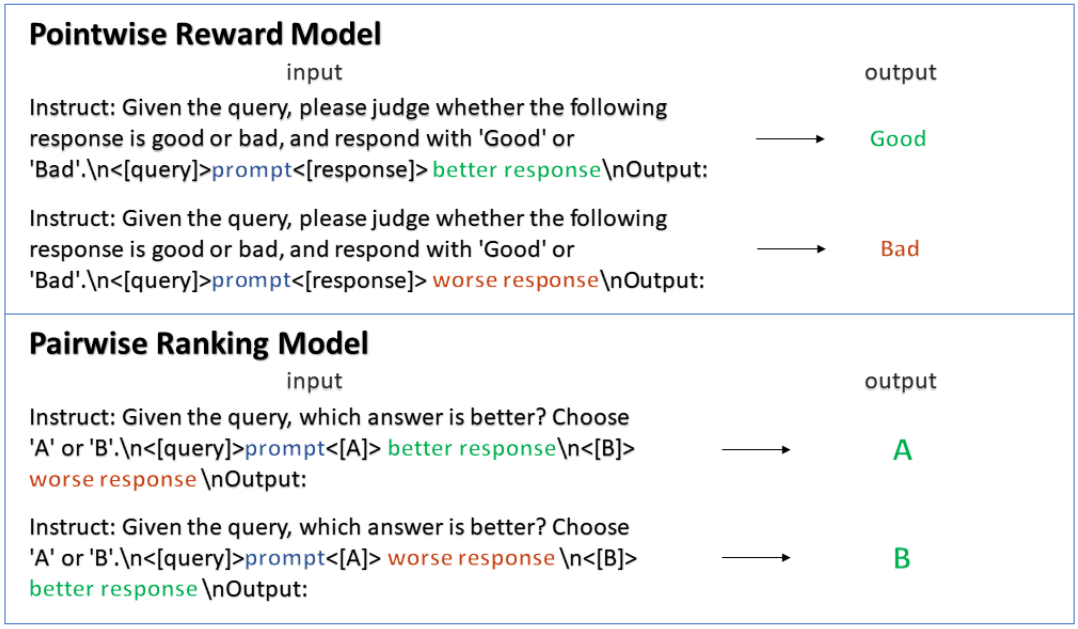


Figure 5: Data format of two other baseline models

two vectors $\mathcal{S}_{mean} = \text{abs}(\text{AVG}(\vec{v} - \vec{v}^*))$ as a simple indicator to measure the difference between the two.

F Policy Loss

Let $l_p = \pi_{policy}(y_w|x) - \pi_{policy}(y_l|x)$ and $l_r = \pi_{ref}(y_w|x) - \pi_{ref}(y_l|x)$.

$$\mathcal{L}_{DPO} = -\text{logsigmoid}(\tau * (l_p - l_r))$$

$$\mathcal{L}_{IPO} = (l_p - l_r - \frac{1}{2\tau})^2$$

$$\mathcal{L}_{IPO-sqrt} = \tau \cdot (l_p - l_r - \frac{1}{2\tau})^2$$

$$\mathcal{L}_{IPO-full} = (\tau \cdot (l_p - l_r) - 0.5)^2$$

$$\mathcal{L}_{hinge} = \max(0, 1 - \tau \cdot (l_p - l_r))$$

$$\mathcal{L}_{IFT} = -\pi_{policy}(y_w|x)$$

G GPT-4 Judgement

The prompt we used for GPT-4 Judgement is listed in the following frame, "**{question}**", "**{response 1}**", "**{response}**" will be replaced with the actual query or responses respectively.

Prompt for comparing two responses.

Instruction: Please compare the following two responses to the same prompt and provide the identifier for the one you consider better. When making the judgment, please consider the following aspects: the accuracy, logic, clarity, usefulness of the answers, and whether they directly address the question. Provide your response as 'A' or 'B' to indicate which answer is preferred.

Prompt: {question}
[A]: {response 1}
[B]: {response 2}
Use 'A' to indicate that [A] is better, and 'B' to indicate that [B] is better.

Due to the presence of positional bias, we test the same pair of responses twice, with each response appearing once in "**response 1**" and "**response 2**". Only when both results point to the same response do we consider it strictly better than the other; otherwise, we consider it a tie.