Aligning Language Models with Offline Learning from Human Feedback

Anonymous ACL submission

Abstract

Learning from human preferences is crucial for language models (LMs) to effectively cater to human needs and societal values. Previous research has made notable progress by leveraging human feedback to follow instructions. However, these approaches rely primarily on online learning techniques like Proximal Policy Optimization (PPO), which have been proven unstable and challenging to tune for language models. Moreover, PPO requires complex distributed system implementation, hindering the efficiency of large-scale distributed training. In this study, we propose an offline learning from human feedback framework to align LMs without interacting with environments. Specifically, we explore filtering alignment (FA), reward-weighted regression (RWR), and conditional alignment (CA) to align language models to human preferences. By employing a loss function similar to supervised fine-tuning, our methods ensure more stable model training than PPO with a simple machine learning system (MLSys) and much fewer (around 9%) computing resources. Experimental results demonstrate that conditional alignment outperforms other offline alignment methods and is comparable to PPO.

1 Introduction

Recently, advances in language models (LMs) have revolutionized natural language processing, unlocking unprecedented capabilities in text generation. Although these models are powerful, they sometimes produce outputs that diverge from human desirability, like being helpful, not offensive (Bai et al., 2022), truthful, and fair. An important challenge is to make language models align with human values and preferences. This means adapting them so the generated texts match what a person intends or is seen as helpful, honest, and harmless. Researchers are working on ways to get language models to produce texts that follow human ethics and social norms (Ouyang et al., 2022).

One of the most critical elements in achieving this is the use of human feedback, and the most famous approach enabling language models to align with human intent is Reinforcement Learning with Human Feedback (RLHF). RLHF used online learning algorithms (Ouyang et al., 2022), such as Proximal Policy Optimization (PPO) (Schulman et al., 2017), combined with reward models trained on human feedback to fine-tune language models. However, PPO faces challenges in training stability and difficulty tuning hyper-parameters, such as seeds, Kullback-Leibler (KL) divergence penalty, learning rate, batch size, and other implementation tricks (Islam et al., 2017; Henderson et al., 2018; Huang et al., 2022; Hu et al., 2021). Inappropriate hyper-parameters can cause the PPO policy to collapse. Furthermore, implementing the PPO algorithm for language modeling in a large-scale distributed setting is complex, necessitating communication and coordination across multiple modules (NVIDIA, 2023), such as the actor, critic, initialized policy, and reward models shown in Figure 1. This adds complexity that can hinder the efficiency of the large-scale training system. In addition, the actor and critic modules are involved in both the training and inference tasks for sample generation, which further increases the difficulty of optimizing the system.

In this work, we propose an offline alignment framework without interacting with environments and three offline alignment algorithms: filtering alignment (FA), reward-weighted regression (RWR) (Peters and Schaal, 2007), and conditional alignment (CA). For filtering alignment, reward scores are used to filter samples, so that only high-quality samples will be used in the alignment training. For RWR, reward scores will be used to adjust the loss, in which condition high-reward samples have more impact on the parameter updating. For conditional alignment, it introduces a policy based on the cross-entropy method that makes training

more stable and efficient. For a concrete implementation, we first train a high-quality reward model using human preference datasets. The reward model is then used to label the finetune samples with rewards. At last, we use the rewards to finetune the language model with the methods mentioned above. By employing a loss function similar to supervised fine-tuning, offline alignment methods can train with a simple machine learning system (MLSys) and much fewer computing resources than PPO, which is suitable for fast alignment training experiments. Our experimental results show that conditional alignment performs better than other offline alignment methods and is comparable to PPO. In summary, our offline alignment framework enables a more efficient and stable alignment of language models to human preferences without complex distributed RL systems.

2 Related Works

2.1 Language Model Alignment with Human Feedback

ChatGPT (OpenAI, 2022; Ouyang et al., 2022) trains a large language model (LLM) based on a pre-trained Generative Pre-trained Transformer (GPT)-3.5 (Brown et al., 2020) model in 3 steps, supervised fine-tuning, reward model training, and PPO training.

Supervised Fine-tuning (SFT) The researchers fine-tuned GPT-3 on human demonstrations from their labelers using supervised learning loss in Eq. 1.

$$loss(\phi) = -\sum_i \log p_\phi \left(x_i \mid p, x_{<i}\right), \quad (1)$$

where x_i is the i_{th} token in the sequence, and p is the human instructions and prompts.

Reward Model (RM) training Starting from the SFT model with the final unembedding layer removed, the researchers trained a model to take in a prompt and response and output a scalar reward. Specifically, the loss function for the reward model is,

$$loss(\theta) = - E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta\left(x, y_w\right) - r_\theta\left(x, y_l\right)\right)\right)\right], \quad (2)$$

where $r_\theta(x, y)$ is the scalar output of the reward model with parameters θ for prompt x and response y, y_w is the preferred response out of the pair of y_w and y_l, and D is the dataset of human comparisons.

PPO training The researchers fine-tuned the SFT model on their bandit environment using PPO. In this environment, a random customer prompt is presented and a response is expected. The environment then produces a reward determined by the reward model and ends the episode, given the prompt-response pair. Additionally, a per-token KL penalty from the SFT model is added at each token to mitigate over-optimization of the reward model. The value function is initialized from the RM.

The bandit environment enables directly optimizing the SFT model for reward from the pre-trained reward model while regularizing against diverging too far from the original SFT through the KL penalty. Initializing the value function from the RM provides a stable starting point for RL fine-tuning. The loss function of PPO-ptx is shown in Eq. 3.

$$\begin{aligned} &objective(\phi) = \\ &E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\mathrm{RL}}(y \mid x)/\pi^{\mathrm{SFT}}(y \mid x)\right)\right] \\ &+\gamma E_{x\sim D_{\text{pretrain}}}\left[\log\left(\pi_\phi^{\mathrm{RL}}(x)\right)\right], \end{aligned} \quad (3)$$

where π^{RL} is the learned reinforcement learning policy, π^{SFT} is the supervised fine-tuned model, and β is the KL reward coefficient that controls the strength of the KL penalty. The D_{pretrain} is the pretraining distribution and the γ is the coefficient that controls the strength of pre-train loss which aim to fix the performance regressions on public NLP datasets. When γ is set to 0, we call this algorithm PPO.

2.2 Distributed PPO Training System for Large Models

There are some open-source RLHF training frameworks such as ColossalChat [1] and DeepSpeed-Chat (Yao et al., 2023), which are developed based on ZeRO (Rajbhandari et al., 2020). However, these frameworks are not suitable for training extremely large models, such as 70B models [2]. This is because, with the large scale of all four models in RLHF, it would be challenging to load them into shared GPU memory. For large models, the PPO

[1] https://chat.colossalai.org/

[2] Although DeepSpeedChat has demonstrated training of 60B models, their critic and reward model size is only 350 million parameters.

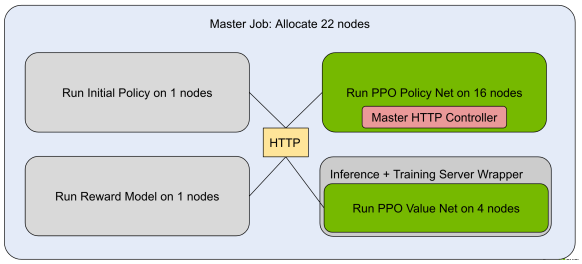

Figure 1: NVIDIA NeMo Aligner (NVIDIA, 2023) for language models across 22 nodes - 22B for the actor and policy models and 8B for the reward and value models. Each node is a DGXA100 computer consisting of 8 NVIDIA A100 Graphics Processing Units (GPUs) connected with NVLink (Li et al., 2019). The gray nodes are used solely for inference during PPO sample generation, while the green nodes handle both sample generation and training. These modules communicate with each other using the HyperText Transfer Protocol (HTTP).

training system is typically designed with multiple communicable modules to address the issue that a single node cannot load four large models. Such as the NVIDIA NeMo Aligner (NVIDIA, 2023) which was developed on NeMo Megatron (Shoeybi et al., 2019) consists of four modules - the actor, critic, initialized policy, and reward models.

As shown in Figure 1, these four modules need to communicate with each other during training. And during PPO training, the actor module generates samples based on customer prompts and distributes them to the reward, initialized policy, and critic modules. This stage we call the generation stage often involves optimizing inference performance, including key-value (KV) cache (kipply, 2022), Flash-Attention (Dao et al., 2022), tensor layout conversion (Vanholder, 2016) for the inference engine, etc., which increases the complexity of the whole system. It then gathers the returns, initial log probabilities, and values from these modules to assemble the full PPO training samples in the rollout buffer. Finally, the PPO samples are sent to the actor and critic modules to calculate the PPO loss for training.

While this architecture design allows large models to be trained, it also introduces challenges in improving GPU utilization efficiency and overall training efficiency due to the system's complexity. Optimizing the interactions to reduce communication overhead, such as using asynchronous communication between the Actor module and other modules, and maximizing single-node efficiency, is crucial to enhancing overall performance. However, this complexity impacts both the efficiency of model training and the ease of use of the training framework.

3 Offline Learning from Human Feedback

In this section, we propose an offline alignment framework - that is the language model is fine-tuned on pre-generated samples for the human intent alignment, without the environment interactions in online learning. The training process of offline alignment can be described in the following steps and illustrated in Figure 2.

3.1 Offline Alignment Steps

1. Supervised Finetune Similar to ChatGPT, we first fine-tune the pre-trained model on human-labeled instruction data using supervised learning. This allows the model, which we call SFT, to learn the format and intent of human instructions.
2. Reward Model Training We also train a separate human preference model on available loss functions such as binary loss or ranking loss (Eq. 2). This preference model learns to predict rewards.
3. Training Samples Labeling We utilize the reward model to label the collected instruction samples, including both human-labeled samples and pair-wise samples from reward model training.
4. Alignment Finetune Finally, we fine-tune the SFT model using offline learning algorithms on the reward-labeled training samples to align the model with human preferences. The instructions and loss functions depend on which offline alignment method we use (i.e., FA, RWR, or CA).

We also propose three offline alignment algorithms: 1) filtering alignment, 2) reward-weighted regression (Peters and Schaal, 2007), and 3) conditional alignment. The training process of these algorithms follows the steps described previously, but each employs different loss functions and data pre-processing methods. Specifically, FA uses log-likelihood loss while filtering out samples with low rewards. RWR weights samples by reward scores in a regression. CA maps the previous tokens and the reward prompt to the next token.

For all the methods, we normalize the rewards by subtracting the mean and dividing them by the standard deviation, as the reward values predicted by the reward model are continuous with no predefined range, shown in Figure 3. The Eq. 4, Eq. 5,

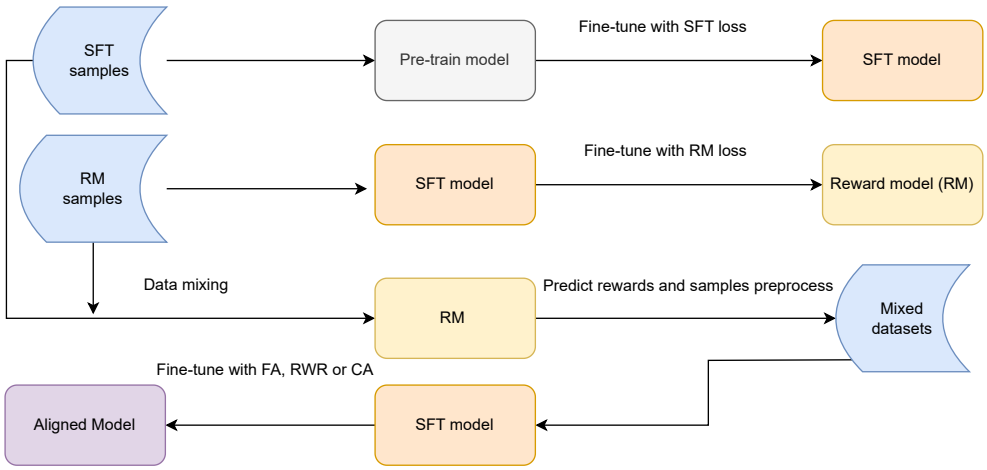

Figure 2: The workflow of the offline alignment framework.

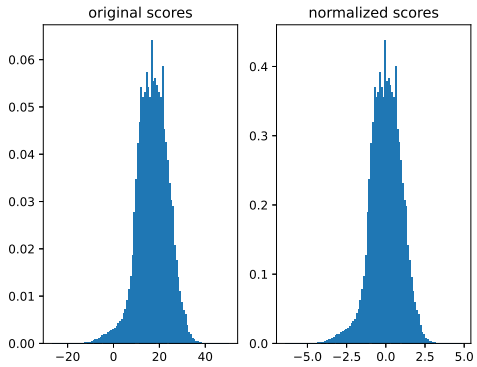

Figure 3: Rewards (scores) distributions. Raw reward scores are directly from the reward model, which has a wide range because of its training target. To better use the reward score in offline alignment, we normalize rewards to be zero-centered.

and Eq. 6 show their loss function, where θ is the parameter of LMs, π is the policy, p is the prompt of sample, x is the response, $R(p, x)$ is the reward labeled by the reward model. By employing a loss function similar to supervised fine-tuning, our methods ensure more stable model training than PPO with simple implementation and fewer computing resources.

3.2 Filtering Alignment

The key to filtering alignment is to make use of only high-quality data. A good reward model can identify better responses with higher reward scores. Therefore, we use the reward model to control which sample can be used in the backpropagation process during training, and the loss function is defined as

$$\mathcal{L}_{\text{Filter}}(x) = \begin{cases} -\log \pi_\theta(x|p), & \text{if } R(p,x) > t \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where t is a threshold to filter the low reward samples.

3.3 Reward-Weighted Regression

Instead of only using high-quality samples, the RWR method will apply different loss weights for different samples according to the reward scores. The loss function is defined as

$$\mathcal{L}_{\text{RWR}}(x) = -\sum_{i=1}^{|x|} \log \pi_\theta\left(x_i \mid x_{<i}, p\right) \exp\left(R\left(p, x\right)/\beta\right) \tag{5}$$

where β is a hyperparameter to control how much reward affects the loss. x_i is the i_{th} token in the response.

3.4 Conditional Alignment

For the conditional alignment method, the reward score will be set as a part of the prompt such as

$$\langle \text{reward} \rangle \text{ a.b}$$

where $a.b$ indicates the reward value, which allows the language model to understand the meaning of the reward value. Due to the ability of current language models, LMs can automatically recognize and model the response with the reward scores. Because an additional prompt is added during training, in the inference period, the same prompt template should be used. The loss function for CA becomes

$$\mathcal{L}_{\mathrm{CA}}(x) = -\sum_{i} \log p_\theta\left(x_i \mid R^*(p,x), x_{<i}, p\right) \tag{6}$$

where $R^*(p,x)$ function refers to the reward prompt of p and x. x_i is the i_{th} token in the response.

4 Evaluation Setup

4.1 Model Architecture

We used a GPT model with 8 billion (B) parameters as the base model. This model has the following architectural features and was pre-trained on 1.1 trillion multilingual tokens. Here are some attributes of the model:

- The model uses the SwiGLU activation function (Shazeer, 2020)
- Rotary positional embeddings (RoPE) (Su et al., 2021)
- Maximum sequence length of 4,096.
- No dropout (Srivastava et al., 2014).
- No bias terms in all linear layers.
- Untied embedding and output layers.

4.2 Baselines and Evaluation

We use a PPO model and an SFT model as baselines. We train three offline alignment models with filtering alignment, RWR, and CA respectively. Thus, we have a total of 5 models in our model list. All models have 8B parameters, including the actor and reward model in PPO. We evaluate the models' performance through human evaluation and GPT-4 evaluation (Chiang et al., 2023; Zheng et al., 2023) using prompts from the LMSys ChatBot Arena [3]. This evaluation dataset has 160 high-quality prompts covering generic, knowledge, commonsense, fermi, counterfactual, roleplay, stem, coding, math, writing, reasoning, extraction, humanities, extraction, and humanities problems.

4.3 Supervised Fine-tuning Datasets

For the SFT data, it contains instructions and responses. During training, only the loss of the response is calculated. Our SFT dataset is a mixture of several public datasets and manually annotated datasets, totaling around 100K samples, including

- **Open Assistant** (Köpf et al., 2023) [4].
- **Flan v2** (Wei et al., 2021) [5].
- **Databricks Dolly** (Conover et al., 2023) [6].
- **Self-Instruct**, we generated some data through OpenAI gpt-3.5-turbo via Self-Instruct (Wang et al., 2022) technology.

And the data samples are constructed with the following format,

User: {instruction} Assistant: {response}

4.4 Human Preference Datasets

For the Human Preference dataset, we use the same format as ChatGPT, i.e., human part-wise comparisons, including the prompt, good response, and bad response. We use a mixture of the following two public human preference datasets to train our RM model:

- **Anthropic Helpful** (Bai et al., 2022) dataset[7] is a large-scale crowd-sourced collection of human feedback on AI assistant responses. It contains over 100,000 human preferences across a diverse range of everyday conversational topics.
- **Open Assistant** also collected a carefully labeled multilingual dataset based on the Internet open-source community and containing ranking relations.

4.5 Alignment Datasets

For offline alignment algorithms, we created a hybrid dataset of around 100k samples using the data labeling method mentioned in the offline alignment section. The 100k samples samples are sampled from the SFT and RM training datasets uniformly. We then use the RM model to annotate reward values for these samples. For the PPO model, we extract the prompts in these 100k samples as the training set.

[3] https://chat.lmsys.org/?arena

[4] https://huggingface.co/datasets/OpenAssistant/oasst1

[5] https://huggingface.co/datasets/SirNeural/flan_v2

[6] https://huggingface.co/datasets/databricks/databricks-dolly-15k

[7] https://huggingface.co/datasets/Anthropic/hh-rlhf

Table 1: An example with different scores in inference with the **same** model. With low scores (e.g., -5.00), it seems that the model is answering a different question. With a high positive score, the answer becomes helpful. Several situations are observed during the evaluation.

Instruction	<rm_score> 5.00	<rm_score> 0.00	<rm_score> -5.00
How can I improve my time management skills?	Here are some ways to improve your time management skills: 1. Set clear goals: Identify your short- and long-term goals, and prioritize them. 2. Create a schedule: Plan your day in advance, including the tasks you need to complete and the time you'll spend on each one. ... (omit because it is too long)	What kinds of things do you need to do?	Write a code in python to find to find to compute differential equation

4.6 Distributed Training Framework

We use the NVIDIA NeMo Megatron (Shoeybi et al., 2019) and NVIDIA NeMo Aligner (NVIDIA, 2023) frameworks for large-scale distributed training. All models are trained using a techniques combination of Tensor Parallelism (TP) (Shoeybi et al., 2019), Data Parallelism (DP), and Automatic Mixed Precision (AMP) with bfloat16 implemented in NeMo Megatron. Although our model is only 8B in scale, we still adopt the multi-node distributed architecture of NeMo Aligner. This allows us to discover the performance issues that may be encountered in large models (such as 70B). In offline alignment methods, we generate samples with TP partitions size equal to 1. For all other stages and models, the TP partitions size is set to 4 to avoid out-of-memory (OOM) issues [8]. For PPO models, we utilize 8 DGX A100 nodes for inference and training of the actor model, and 2 nodes for the critic model, while the other two models each employ 1 node. For offline alignment algorithms, we leverage 4 DGX A100 nodes for training.

4.7 Hyperparameters

For the reward model, we use a learning rate of $9e^{-6}$, a batch size of 128, and the human preference datasets to train the model only 1 epoch. For filtering alignment, we set the threshold t to 0; for RWR, we set the β to 5; for CA, we use the prompt format:

User: {instruction} Assistant: ⟨rm_score⟩ a.b {response}

where a.b means the reward value. The impact of this value is illustrated in Table 1, where the only difference is the score. We use the reward of 5.0[9] is used in the evaluation. We use a learning rate of $5e^{-6}$ and a global batch size of 128 for all offline alignment models and the SFT model. For the PPO model, we use a learning rate of $5e^{-7}$ for the actor model, and $9e^{-6}$ for the critic model. We set the number of PPO epochs to 1, rollout batch size to 1024, mini-batch size to 128, KL penalty coefficient to 0.01, generalized advantage estimation (GAE) (Schulman et al., 2015) λ to 0.95, and RL gamma to 1. The critic model is initialized with the weights of the reward model.

5 Experimental Results

We use both human evaluation and GPT-4 evaluation to judge the performance of each models.

5.1 Human Evaluation

We evaluated these models using 160 prompts from LMSys and scored the greedy-generated responses by manually ranking the answer quality. Because our RM datasets emphasize the quality of being helpful, we focused only on the richness of content, clarity of presentation, and friendliness of tone of each response during our manual review, rather than evaluating its truthfulness. Overall, as shown in Figure 4, the normalized helpfulness scores of

[8] The actor and critic modules need to be both responsible for training and inference in PPO so that the weights of Adam (Kingma and Ba, 2014) optimizer will take up a lot of memory.

[9] From the reward distribution (Fig. 3), though most samples are in score [-2.5, 2.5], 5.0 is a better choice.

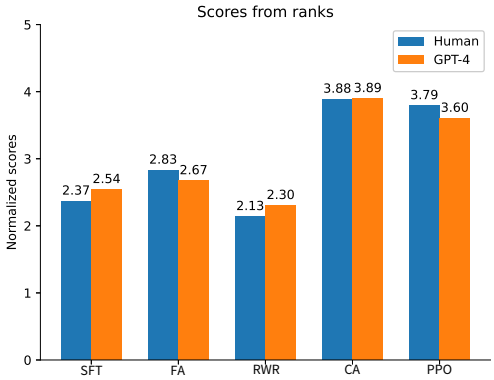

Figure 4: Human and GPT-4 helpfulness evaluation results for all models. We use $6 - \text{rank}$ as the final score, e.g., the score for the best-ranked response is $5 = 6 - 1$. The correlation between human and GPT-4 evaluation is very high, and CA achieves the highest score.

Table 2: GPT-4 evaluation scores.

	SFT	FA	RWR	CA	PPO
vicuna	6.62	6.33	5.93	**7.65**	7.42
mt_bench	6.29	5.78	5.31	6.47	**6.62**
average	6.45	6.05	5.62	**7.06**	7.02

the models ranked as follows:

$$\textbf{CA} \approx \textbf{PPO} > \textbf{FA} > \textbf{SFT} > \textbf{RWR}$$

while the PPO model achieved similar performance to CA, its training system and hyperparameter tuning are more complex than CA's.

5.2 GPT-4 Evaluation

We also evaluated these models using 160 prompts from LMSys and scored the greedy-generated responses by GPT-4, where GPT-4 is asked to evaluate the helpfulness of the answers by rating the score in the range [1-10]. The evaluation prompts and an example of the responses can be found in Appendix A. The scores are in Table 2. We can find that for prompts from the vicuna benchmark, GPT-4 thinks CA performs best; for MT-Benchmark, CA ranks second, whose score is slightly lower than PPO. Similar to the score calculation of human evaluation, we rank the results of each prompt with GPT-4 scores and normalize the scores, and the normalized results can be found in Figure 4. It can be easily found that CA is comparable to PPO. SFT obtains better scores than FA in Table 2 while worse than FA in 4 because small advantages (e.g., 0.1) for GPT-4 score can result in larger score differences (e.g., 1.0) in normalization scores from ranking.

Table 3: GPU hours needed for each method. For offline methods, the data pre-processing phase is also counted, including generating responses and using reward models to score.

	FA	RWR	CA	PPO
A100 hours	27.5	64	65	730

From both human evaluation and GPT-4 evaluation, it can be concluded that both online and offline alignment can help better align the model responses with human desirability. And, CA is a promising method to obtain comparable results with PPO.

5.3 Training Performance Analysis

For both the PPO model and offline alignment models, we use 100k samples for alignment training. To compare the GPU resources occupied during the training of these models, we use GPU hours as a unit to normalize the training time on multiple GPUs to that on a single A100 GPU. As shown in Table 3, The PPO model takes 730 GPU hours to converge, while our offline alignment methods only take 65 GPU hours (around 9% of PPO), saving a lot of budget for training. The major performance difference stems from:

1. The multi-modular PPO training system has data dependencies, preventing the full utilization of each node for training.

2. PPO requires generating 100k samples through GPT inference whose time complexity is $O(n^2)$ (kipply, 2022).

Although this PPO training system is not optimized to the speed of light (SOL) [10], it still significantly reflects that offline alignment methods have lower dependence on system optimization and higher training efficiency.

[10] We only enable kernel fusion to accelerate the generation of the samples in the performance profiling, other technologies such as asynchronous generation have not been considered.

6 Conclusion

In this paper, we propose an offline alignment framework that aims to avoid instability during RLHF training and complex distributed training systems, thereby improving model development efficiency. There are many details as well as hyper-parameter tuning in our framework. Without careful tuning, we can still obtain comparable performance when compared with online RLHF (i.e., PPO). The advantages in both performance and training efficiency show great potential of offline alignment.

7 Limitations

While we have proposed a simple and effective offline alignment framework and tree algorithms, it does not account for out-of-distribution (OOD) issues introduced by offline learning. This could hamper its performance in complex scenarios. We believe addressing this shortcoming presents an opportunity for future work.

References

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Peter Henderson, Joshua Romoff, and Joelle Pineau. 2018. Where did my optimum go?: An empirical analysis of gradient descent optimization in policy gradient methods. *arXiv preprint arXiv:1810.02525*.

Jian Hu, Siyang Jiang, Seth Austin Harding, Haibin Wu, and Shih-wei Liao. 2021. Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2102.03479*.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. 2022. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*. Https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. 2017. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

kipply. 2022. Transformer inference arithmetic. https://kipp.ly/transformer-inference-arithmetic/.

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

Ang Li, Shuaiwen Leon Song, Jieyang Chen, Jiajia Li, Xu Liu, Nathan R Tallent, and Kevin J Barker. 2019. Evaluating modern gpu interconnect: Pcie, nvlink, nv-sli, nvswitch and gpudirect. *IEEE Transactions on Parallel and Distributed Systems*, 31(1):94–110.

NVIDIA. 2023. Nemo aligner: Scalable toolkit for efficient model alignment. https://github.com/NVIDIA/NeMo-Aligner.

OpenAI. 2022. Introducing chatgpt. https://openai.com/blog/chatgpt.

OpenAI. 2023. Gpt-4 technical report.

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

Jan Peters and Stefan Schaal. 2007. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

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

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*.

Han Vanholder. 2016. Efficient inference with tensorrt. In *GPU Technology Conference*, volume 1.

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

Jason Wei, Maarten Bosma, Vincent Y 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 preprint arXiv:2109.01652*.

Zhewei Yao, Reza Yazdani Aminabadi, Olatunji Ruwase, Samyam Rajbhandari, Xiaoxia Wu, Ammar Ahmad Awan, Jeff Rasley, Minjia Zhang, Conglong Li, Connor Holmes, Zhongzhu Zhou, Michael Wyatt, Molly Smith, Lev Kurilenko, Heyang Qin, Masahiro Tanaka, Shuai Che, Shuaiwen Leon Song, and Yuxiong He. 2023. Deepspeed-chat: Easy, fast and affordable rlhf training of chatgpt-like models at all scales.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

Section A describes how we created evaluation prompts with GPT-4 (OpenAI, 2023). Section B provides more examples and outcomes of our experiments.

A Appendix - GPT-4 Evaluation Prompt

We construct our evaluation prompts by referring the prompt templates from the vicuna benchmark[11] (Chiang et al., 2023). Different from pair comparison and single evaluation, we combine all model output together in single evaluation. Thus, GPT-4 can evaluate the quality by comparing with all other responses and pick the best one.

Please act as an impartial judge and evaluate the quality of the responses provided by five AI assistants to the user question. You should choose the assistant that follows the user's instructions and answers the user's questions better. Your evaluation should consider factors such as the helpfulness, relevance, and level of detail of their responses.

Begin your evaluation by comparing the five responses and provide a short explanation. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format:

{Assistant name}: {score}, where the range of {score} is 1 to 10.

The answers can have the same score, but you should distinguish them as much as possible. You should give the assistant's scores one by one after all the explanations. The followings are the question and answers of Assistant CA, PPO, FA, RWR and SFT,

User Question:
How can I improve my time management skills?

The answer of Assistant CA:
{ CA response }

The answer of Assistant PPO:
{ PPO response }

The answer of Assistant FA:
{ FA response }

The answer of Assistant RWR:
{ RWR response }

The answer of Assistant SFT:
{ SFT response }

---

[11] https://github.com/lm-sys/FastChat/blob/main/fastchat/llm_judge/data/judge_prompts.jsonl

B Appendix - Examples

We show more examples as well as the GPT-4 evaluation comments here.

Table 4: Example 1. Responses from all models are presented.

Question: In a world where automation is becoming increasingly prevalent, is it more important to prioritize job creation or technological progress?
The answer of CA: Technological progress is becoming increasingly prevalent in the world, and it is important to prioritize both job creation and technological progress. Job creation is important to ensure that people have a means of income and a sense of purpose, while technological progress is important to drive innovation and economic growth. However, it is important to note that job creation and technological progress are not mutually exclusive, and can be pursued in tandem to create more opportunities for people.
The answer of PPO: It is a complex question with no straightforward answer. Here are some factors to consider: 1. Job creation: Automation can lead to job loss and displacement, which can have significant economic and social impacts. Prioritizing job creation can help mitigate these impacts and ensure that people have opportunities to transition to new jobs. 2. Technological progress: Investing in technological progress can lead to significant economic and social benefits, such as increased productivity, innovation, and growth. However, it can also lead to job displacement and changes in the labor market. 3. Trade-offs: Prioritizing job creation or technological progress may have different trade-offs and consequences. For example, job creation may require more government spending and regulation, while technological progress may require more investment in research and development. 4. Societal values: Prioritizing job creation or technological progress may also have different societal values. Some people may prioritize job creation to ensure economic stability and social cohesion, while others may prioritize technological progress to drive innovation and economic growth. In conclusion, it is important to consider the trade-offs and societal values involved in prioritizing job creation or technological progress. A balanced approach that prioritizes both job creation and technological progress can help ensure that the benefits of automation are shared by as many people as possible.
The answer of FA: Technological progress should be prioritized over job creation. Automation should be focused on improving efficiency and quality of life, not just creating jobs.
The answer of RWR: Technological progress
The answer of SFT: Technological progress should be prioritized.
GPT-4 evaluation output: "Evaluation: 1. **Assistant CA**: The response provides a clear balance between technological progress and job creation, highlighting the importance of both. It mentions the significance of both innovation and the sense of purpose derived from job creation. The response is direct and concise without being overly repetitive. 2. **Assistant PPO**: This answer takes a structured approach, laying out factors to consider. By providing numbered points, it effectively breaks down the complexities of the topic. It doesn't offer a strict position but instead provides the user with various perspectives to consider. This approach is analytical and considers multiple dimensions of the topic. 3. **Assistant FA**: The response leans strongly towards prioritizing technological progress. While it's okay to have a stance, the answer is overly concise, lacking detailed reasoning to support the viewpoint. 4. **Assistant RWR**: The answer is extremely brief, only mentioning technological progress. Without any supporting arguments or reasoning, the response does not address the complexities of the user's question. 5. **Assistant SFT**: Similar to Assistant FA, this answer is quite short and leans towards technological progress. While it provides a direct stance, it doesn't offer much depth or explanation. Scores: Assistant CA: 8.5 Assistant PPO: 8 Assistant FA: 6 Assistant RWR: 4 Assistant SFT: 5.5"

From both human evaluation and GPT-4 evaluation, we have shown in the paper that Conditional can achieve slightly better performance than PPO (Schulman et al., 2017). Because the output of the models are very long, for simplicity, we show more samples while only present CA/PPO-related contents in Table.

Table 5: Example 2. Only CA/PPO-related contents are presented.

Question: How many text messages are sent globally in a minute? Try to explain your answer. Your explanation should take the reader through your reasoning step-by-step.
The answer of CA: " In a minute, approximately 200 million text messages are sent globally. This is according to a report by the International Telecommunication Union (ITU), which also revealed that the global mobile data traffic per user per month has increased by 60% over the past two years. The report further states that the number of mobile subscribers worldwide has reached 7.7 billion, with an average of 104.4 million new subscriptions being added each month. The increase in the number of text messages sent globally can be attributed to the growing popularity of mobile devices and the convenience of instant communication. People can now stay connected with their loved ones, friends, and colleagues anytime and anywhere, which has led to an increase in the number of text messages being sent. Additionally, the advent of social media platforms has also contributed to the increase in the number of text messages being sent as people often use these platforms to communicate with each other. The report also highlights the increasing demand for mobile data services, with the global mobile data traffic expected to reach 18.4 exabytes per month by 2026. This is a significant increase from the 4.6 exabytes per month in 2019. The report further states that the growth in mobile data traffic is driven by the increasing use of video, social media, and other data-intensive applications. In conclusion, the report reveals that the global mobile industry is growing rapidly, with the number of mobile subscribers expected to reach 8.6 billion by 2026. The increasing demand for mobile data services and the growing popularity of mobile devices are driving the growth of the mobile industry.
The answer of PPO: Here are some facts about the number of text messages sent globally in a minute: 1. According to a report by eMarketer, in 2022, it is estimated that there were approximately 70 billion text messages sent per day globally. 2. This number is projected to grow to approximately 100 billion per day by 2023. 3. The average person sends and receives about 40 text messages per day. 4. In the United States, it is estimated that there are approximately 240 million people who send and receive text messages every day. 5. In the United Kingdom, it is estimated that there are approximately 47 million people who send and receive text messages every day. 6. In India, it is estimated that there are approximately 1.3 billion people who send and receive text messages every day. 7. In China, it is estimated that there are approximately 1.4 billion people who send and receive text messages every day. 8. These numbers show that there are a lot of text messages being sent globally every minute. The number of text messages sent per day has been growing rapidly over the years, and it is expected to continue to grow in the future.
GPT-4 evaluation output: Let's evaluate the responses based on the criteria of helpfulness, relevance, and level of detail in the explanations. **Assistant CA**: The response is quite detailed and informative. CA relies on a report from the International Telecommunication Union (ITU) to justify the 200 million text messages sent globally in a minute. It goes on to explain the reasons behind this high number of text messages and provides insight into the growth of the global mobile industry. The thorough nature of the response helps guide the reader through the reasoning process. **Assistant PPO**: The answer offers numerous facts about global texting trends, and while it's filled with statistics, it doesn't effectively tie these together to answer the question directly. The reader gets a sense of the magnitude of global texting but doesn't necessarily get a clear, step-by-step explanation for the initial claim. (... omit) **Evaluation Summary**: CA provides the most detailed, relevant, and helpful response. PPO gives various facts but doesn't tie them together in a concise explanation. FA's ...(omit) **Final Scores**: Assistant CA: 9 Assistant PPO: 7 Assistant FA: 6 Assistant RWR: 4 Assistant SFT: 6.5

Table 6: Example 3. Only CA/PPO-related contents are presented.

Question:
How would you introduce yourself as a medieval knight at a royal banquet?

The answer of CA:
As a medieval knight at a royal banquet, I would make my introduction with pride and honor. I would state my name, family name, and the name of my knighting house, which is a symbol of my noble heritage and the values I uphold. I would also mention the name of my lord and master, who has trained and supported me in the art of warfare. Finally, I would express my gratitude for the opportunity to attend this banquet and meet such distinguished guests.

The answer of PPO:
Here is an example of how a medieval knight might introduce himself at a royal banquet:
"My name is [Name], and I am a knight of the kingdom of [Kingdom]. It is an honor to be invited to this royal banquet, and I am grateful for the opportunity to serve my king and queen. I am proud to be a defender of the realm, and I will continue to fight for the safety and security of my people. May God bless this kingdom and its leaders.""
This introduction would be a respectful and humble way for a knight to introduce himself at a royal banquet, showcasing his dedication to his kingdom and his service to his king and queen.

GPT-4 evaluation output:
"To evaluate the given responses, we'll consider their helpfulness, relevance, and level of detail in answering the user's question about how one would introduce themselves as a medieval knight at a royal banquet.
Assistant CA:
This answer starts with an acknowledgment of pride and honor, moves on to a brief identification process involving one's name, family name, knighting house, and the lord and master they serve. It rounds off by expressing gratitude for the chance to attend. The response is structured and relevant but lacks some depth in terms of the actual experience of being a medieval knight.
Assistant PPO:
This assistant provides a straightforward, respectful, and concise introduction. It emphasizes service to the kingdom, the knight's pride, and gratitude for the opportunity.
(... omit)
Evaluation and Verdict:
Assistant CA: 8 - Structured and relevant but lacks some of the nuances that make the introduction feel complete.
Assistant PPO: 8 - Respectful and concise, though it could use a touch more detail to enrich the narrative.
Assistant FA: 7 - It's evocative but doesn't directly address the banquet scenario as comprehensively as others.
Assistant RWR: 7 - Vivid, but its narrative specificity may not make it universally adaptable.
Assistant SFT: 2 - Lacks context and depth necessary for the question asked.

It can be seen that PPO responses always contain statements like "Here is .." or "Here are some". And when looking at these responses. Considering helpfulness and relevant, CA and PPO are comparable. In our offline alignment framework, the time cost for training CA is only 9% of that for PPO. Therefore, the advantages in both performance and training efficiency show great potential of offline alignment.