Lecture 8: Alignment methods in Language Models II

Dr. Yaodong Yang

Institute for AI, Peking University

08/2023

Why do we need other methods to align

In standard RLHF, PPO is used for training LLMs. However, PPO is sensitive to hyperparameters, which makes it hard to train.

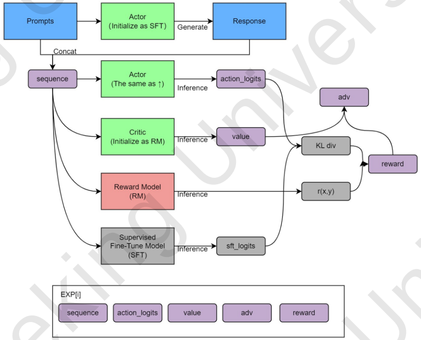


Fig. 1. The models required and process for PPO training stage in RLHF.

The challenges of applying PPO in language models

Despite the acclaimed effectiveness of PPO [SWD+17], recent research has identified the following three issues in language models that require additional study:

- **Mode collapse** PPO can reduce the output randomness of language models, misleading the model into producing deterministic responses.
- **Instability** PPO uses multi-step approximation, these steps sometimes trigger instability in training, leading to an abrupt drop in model performance.
- **Poor sample efficiency** The family of policy gradient algorithms suffers from slow convergence and can yield poor ultimate policies [OWJ+22]. PPO is susceptible to the same problems.

The Advantages of APA

To address such issues, [ZSF+23] introduces Advantage-Induced Policy Alignment (APA), which directly aligns the output policy of the language model with a target policy in each training epoch. Three major advantages of APA over PPO:

- **APA is more sample-efficient.** APA performs better than PPO in fine-tuning on the same number of samples.
- **APA training is more stable** APA is much less prone to sudden performance degradation during training. The control over such policies deviations is critical in preventing over-optimization on reward models
- **APA has fewer hyperparameters.** The loss function in APA involves only one major tunable parameter for KL control, while in PPO, there are more hyperparameters.

Reinforcement Learning

RL captures the interaction between an agent and an environment via the formalism of a MDP. We consider a finite-horizon MDP represented by a tuple:

$$M = (\mathcal{S}, \mathcal{A}, H, P, r, \rho)$$

where $\mathcal{S}$ is a finite state space, $\mathcal{A}$ is a finite action space, H is the horizon.

$$P : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$$

is a probability transition matrix,

$$r : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$$

is a reward function, and

$$\rho : \mathcal{S} \mapsto \Delta(\mathcal{S})$$

is the initial state distribution.

Reinforcement Learning

A policy $\pi : \mathcal{S} \mapsto \Delta(\mathcal{A})$ is a function that maps a state to a distribution over actions. The value function $V^\pi : \mathcal{S} \mapsto \mathbb{R}$ of policy π is defined as the expected sum of discounted rewards when the agent starts from initial state s and follows policy π throughout the episode. Let $\gamma \in [0, 1]$ be the discount factor. For any $s \in \mathcal{S}$, we have:

$$V^\pi(s) := \mathbb{E}\left[\sum_{\tau=0}^{H} \gamma^\tau r\left(s_\tau, a_\tau\right) \mid s_0 = s, a_\tau \sim \pi\left(\cdot \mid s_\tau\right), s_{\tau+1} \sim P\left(\cdot \mid s_\tau, a_\tau\right)\right]$$

Given a policy π, the state-action value function, also known as the Q-function, can be defined analogously. For state $s \in \mathcal{S}$ and $a \in \mathcal{A}$, we have:

$$Q^\pi(s, a) := \mathbb{E}\left[\sum_{\tau=0}^{H} \gamma^\tau r\left(s_\tau, a_\tau\right) \mid s, a, a_\tau \sim \pi\left(\cdot \mid s_\tau\right), s_{\tau+1} \sim P\left(\cdot \mid s_\tau, a_\tau\right)\right]$$

Reinforcement Learning

We also define the important notion of an advantage function. For a policy π, state s and action a, the advantage, defined as

$$\mathrm{Adv}^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$

We also define the occupancy measures $d^{\pi}_{\text{state}} : \mathcal{S} \mapsto [0,1]$ and $d^{\pi}_{\text{action}} : \mathcal{S} \times \mathcal{A} \mapsto [0,1]$ as

$$d^{\pi}_{\text{state}}(s) := \frac{1}{H}\sum_{h=1}^{H} \mathbb{P}\left(s_h = s \mid \pi\right)$$

and

$$d^{\pi}_{\text{action}}(s,a) := \frac{1}{H}\sum_{h=1}^{H} \mathbb{P}\left(s_h = s, a_h = a \mid \pi\right)$$

where $\mathbb{P}(\cdot \mid \pi)$ signifies that all actions are drawn from π.

A language model as a reinforcement learning agent

A language model can be viewed as an agent that operates in an environment with state space $\mathcal{S} = \bigcup_{k=0}^{H} \mathcal{X}^k$ and action space $\mathcal{A} = \mathcal{X}$, where H is the maximum number of tokens. The transitions are always deterministic, with the next state equal to the concatenation of all the previous tokens and the current token:

$$P\left(s_{h+1} = (x_1, \cdots, x_k) \mid s_h = (x_1, \cdots, x_{k-1}), a_h = x_k\right) = 1$$

Specifically, a language model receives as input a sequence of tokens

$$(x_1, \ldots, x_n)$$

and generates a distribution over the next token x_{n+1}. The entire sequence is scored by a reward model, which produces a scalar reward r.

A language model as a reinforcement learning agent

In this context, fine-tuning is equivalent to improving the agent policy π.
We note that most transformer-based language models map a state (context) s and an action (next token) a to a logit $q_\theta(s, a)$, and the next token is sampled according to the distribution induced by the logits $\{q_\theta(s, a)\}_{a \in \mathcal{A}}$. This gives rise to the following natural parameterization of a language model policy:

$$\pi_\theta(a \mid s) = \frac{\exp\left(q_\theta(s, a)\right)}{\sum_{a \in \mathcal{A}} \exp\left(q_\theta(s, a)\right)}.$$

The next token prediction problem can be solved using RL.

Proximal policy optimization

For each fixed state s, we consider the following KL-regularized optimization problem as a target of policy improvement:

$$\underset{\theta}{\operatorname{maximize}} \mathcal{F}(\theta; s, \pi) := \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[\operatorname{Adv}^\pi(s,a)\right] - \lambda \cdot \operatorname{KL}\left(\pi_\theta(\cdot \mid s) \| \pi_{\text{init}}(\cdot \mid s)\right)$$

π_{init} refers to the initial policy of the language model before the RLHF stage, π is an arbitrary policy that we hope to improve.

- $\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[\operatorname{Adv}^\pi(s,a)\right]$ is an expected advantage, and to maximize the expected advantage, the agent is encouraged to move toward the optimal action in state s.
- $\lambda \cdot \operatorname{KL}\left(\pi_\theta(\cdot \mid s) \| \pi_{\text{init}}(\cdot \mid s)\right)$ is a KL regularizer, controls the deviation of π_θ from π_{init} .

Proximal policy optimization

PPO leverages importance sampling to circumvent sampling from , arriving at

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[\mathrm{Adv}^{\pi_{\text{old}}}(s,a)\right]=\mathbb{E}_{a\sim\pi_{\text{old}}(\cdot|s)}\left[\frac{\pi_\theta(a\mid s)}{\pi_{\text{old}}(a\mid s)}\mathrm{Adv}^{\pi_{\text{old}}}(s,a)\right]$$

where the expectation on the right-hand side can be estimated in an unbiased manner from finite samples.
Instead of penalizing the expected advantage with the estimated KL-divergence as in:

$$\underset{\theta}{\mathrm{maximize}}\mathcal{F}(\theta;s,\pi):=\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[\mathrm{Adv}^{\pi}(s,a)\right]-\lambda\cdot\mathrm{KL}\left(\pi_\theta(\cdot\mid s)\|\pi_{\text{init}}(\cdot\mid s)\right)$$

PPO directly subtracts the KL penalty term from the reward received by the agent, and adaptively adjusts the penalty weight λ based on the deviation of π_θ from π_{init} .

Proximal policy optimization

The KL-penalized reward is then used to estimate a new advantage function $\widehat{\mathrm{Adv}}$. To avoid ill-conditioned gradients caused by large values or importance ratio estimates, PPO applies clipping to the objective function. The final loss function $\mathcal{L}^{\mathrm{PPO}}(\theta;\mathcal{D}) =$

$$-\frac{1}{|\mathcal{D}|}\sum_{(s,a)\in\mathcal{D}} \min\left\{\frac{\pi_\theta}{\pi_{\mathrm{old}}}\widehat{\mathrm{Adv}}(s,a), \mathrm{clip}\left(\frac{\pi_\theta}{\pi_{\mathrm{old}}}, 1-\epsilon, 1+\epsilon\right)\widehat{\mathrm{Adv}}(s,a)\right\}$$

Note that the loss function relies on extra tunable hyperparameters. The clipping also makes the estimator biased. **Hyperparameters and approximation lead to instability.**

Advantage weighted regression (AWR) [AMAS20]

If the parameterized policy space $\{\pi_\theta\}$ contained all possible policies, the maximizer of $\mathcal{F}(\theta; s, \pi_{\text{old}})$ (2) would induce a policy $\pi^\star$ that satisfies

$$\pi^\star(a \mid s) = \frac{1}{Z(s)} \pi_{\text{init}}(a \mid s) \cdot \exp\left(\mathrm{Adv}^{\pi_{\text{old}}}(s, a)/\lambda\right)$$

where $Z(s) = \sum_{a' \in \mathcal{A}} \pi_{\text{init}}(a' \mid s) \cdot \exp\left(\mathrm{Adv}^{\pi}(s, a')/\lambda\right)$ is a normalizing factor. In the case that $\{\pi_\theta\}$ does not contain all policies, a natural way to maximize $\mathcal{F}(\theta; s, \pi_{\text{old}})$ is to project $\pi^\star$ to $\{\pi_\theta\}$ with respect to KL-divergence, which gives rise to the AWR algorithm. From the above equation, $\mathrm{KL}(\pi^\star(a \mid s) \| \pi_\theta(a \mid s)) =$

$$-\frac{\pi_{\text{init}}(a \mid s)}{Z(s)} \exp\left(\frac{\mathrm{Adv}^{\pi_{\text{old}}}(s, a)}{\lambda}\right) \log(\pi_\theta(a \mid s)) + C(s)$$

where $C(s)$ is a constant that does not depend on θ.

Advantage weighted regression

To minimize the $\mathrm{KL}\left(\pi^{\star}(a \mid s) \| \pi_{\theta}(a \mid s)\right)$, authors make three changes to the objective that help to set the stage for the new method:

- We replace π_{init} with π_{old}, which can be approximated with finite samples.
- The $\mathrm{KL}\left(\pi^{\star}(a \mid s) \| \pi_{\theta}(a \mid s)\right)$ only accounts for one state s. To incorporate other states, we minimize a weighted sum of KL-divergences, with states sampled from the state-action occupancy measure $d^{\pi_{\text{old}}}$.
- We use the approximation $Z(s) \approx 1$.

With these changes, we arrive at the following population loss for AWR:

$$\mathcal{L}^{\text{AWR}}(\theta) = -\mathbb{E}_{(s,a)\sim d^{\pi_{\text{old}}}}\left[\exp\left(\operatorname{Adv}^{\pi_{\text{old}}}(s,a)/\lambda\right)\log\left(\pi_{\theta}(a \mid s)\right)\right]$$

AWR can be unstable in the online case.

Advantage-Induced Policy Alignment

To project the optimal policy $\pi^\star$ in:

$$\pi^\star(a \mid s) = \frac{1}{Z(s)} \pi_{\text{init}}(a \mid s) \cdot \exp\left(\mathrm{Adv}^{\pi_{\text{old}}}(s, a)/\lambda\right)$$

onto the parameterized policy space, we may also consider another distance instead of KL-divergence. In APA, we employ the squared error between log probabilities in place of the KL-divergence:

$$\left(\log \pi^\star(a \mid s) - \log \pi_\theta(a \mid s)\right)^2$$

Similar to our implementation of AWR, we also apply $Z(s) \approx 1$, and consider a weighted sum of squared errors with states sampled from $d^{\pi_{\text{old}}}$, giving rise to the following population loss:

$$\mathcal{L}^{\mathrm{APA}}(\theta) = \mathbb{E}_{(s,a)\sim d^{\pi_{\text{old}}}}\left[\left(\log \pi_\theta(a \mid s) - \mathrm{Adv}^{\pi_{\text{old}}}(s, a)/\lambda - \log \pi_{\text{init}}(a \mid s)\right)^2\right]$$

Authors establish theoretically that the empirical loss is a reasonable surrogate for the population loss.

Results on the HH dataset

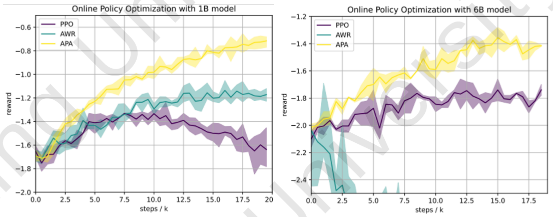


Fig. 2. Comparison of the performance of three methods on the HH dataset. The x-axis represents the total steps, which are proportional to the amount of data used in the training procedure. The y-axis is the reward evaluated by the same reward model.

We see that with the same amount of data, APA is able to achieve the highest reward in all three cases. We also observe that PPO becomes more stable with large models, potentially due to smaller batch size, or the ability of getting higher reward with a smaller deviation in KL divergence.

Results on the HH dataset

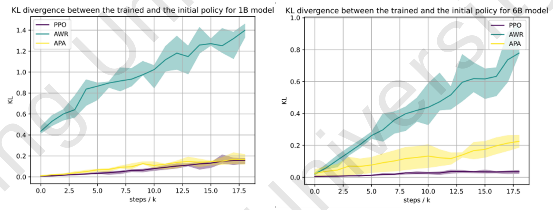


Fig. 3. Comparison of the performance of three methods on the HH dataset. The *x*-axis represents the total steps. The *y*-axis is the KL divergence between the trained model and the initial model.

We show how the KL divergence between the current policy and the initial policy changes as a function of the training process for the three seeds. We can see that for all three models, APA provides similar or better KL control than PPO and AWR, although we note that for the $6\ \mathrm{B}$ model the KL control for PPO is slightly better than APA.

Conclusions

The key takeaways from the comparisons of the three RLHF algorithms that we have studied can be summarized as follows:

Mode collapse RL algorithms may encourage the language model to produce less diverse output, such that the policy that maximizes total reward is deterministic. To rein in such behavior, it is crucial to impose control on the divergence between new policies and the initial policy after SFT. However, the clipping of the objective function and the adaptive KL controller make the behavior of PPO unpredictable. APA is able to provide better and easy-to-adjust KL control by explicitly tuning the hyperparameter λ, which helps mitigate mode collapse.

Conclusions

Stability PPO suffers from significant performance degradation whenever the model policy diverges too much from the initial policy π_{init} , an effect which is more pronounced for smaller models. We attribute this to the KL controller in PPO.

Sample efficiency With the same level of control over KL-divergence, APA shows higher sample efficiency than PPO and AWR. One possible explanation is that in both PPO and AWR, policy improvement critically depends on using finite samples to reconstruct the sampling policy π_{old} , whereas in APA, minimizing the population loss hinges less on the reconstruction of π_{old} .

Disadvantages of existing alignment methods

Some commonly used alignment methods:

PPO to optimize for a trained alignment score module: PPO is sensitive to hyperparameters, and requires additional training in the reward model and value network.

Imitation learning to a final-answer or reward-model filtered dataset [UKK+22]. IL is less data-effective as it only makes use of the success instruction-output pairs, completely abandoning the ones that do not align.

[ZLW+23] propose Hindsight Instruction Relabeling (HIR) that utilizes not only successful instruction-output pairs but also bootstrap from failed ones.

Goal-Conditioned Reinforcement Learning

Standard MDP A tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$. $\mathcal{R}(s, a)$ is the reward function. The policy π is a map-ping from $\mathcal{S}$ to $\mathcal{A}$. The goal is to find an optimal policy π^* that maximizes:

$$J(\pi) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}\left(s_t, a_t\right) \right]$$

where $a_t \sim \pi\left(a \mid s_t\right)$.

Goal-conditioned MDP A tuple $\langle \mathcal{G}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rangle$, where $\mathcal{G}$ represents the goal space. Meanwhile, both the reward function $\mathcal{R}(s, a, g)$ and policy $\pi(a \mid s, g)$ need to be goal-dependent. Thus, the objective is to find an optimal pol-icy π^* that maximizes

$$J(\pi) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}\left(s_t, a_t, g_t\right) \right]$$

where $a_t \sim \pi\left(a \mid s_t, g_t\right)$.

Instruction Following as Goal-conditioned RL

A language model $\mathcal{M}$ can take instructional prompt $\mathbf{p}$ and initial query token sequence $\mathbf{q} = \{\mathbf{q}_0, \ldots, \mathbf{q}_i\}$ as input, and autoregressively predict next token $\mathbf{e}_{i+1} = \mathcal{M}(\mathbf{p}, \mathbf{q}, \{\mathbf{e}_0, \ldots, \mathbf{e}_i\})$. We can view standard prompt-conditioned language tasks (e.g. multi-step reasoning) as a goal-reaching problem, by formulating the MDP as follows:

- Goal space $\mathcal{G}$: space of instructional prompt $\mathbf{p}$
- State space $\mathcal{S}$: space of input token sequence $\mathbf{q} \cup \{\mathbf{e}_i\}$
- Action space $\mathcal{A}$: space of output token $\mathbf{e}_{i+1}$
- Transition probability $\mathcal{P}$: $\mathcal{M}(\mathbf{e}_{i+1} \mid \mathbf{p}, \mathbf{q}, \{\mathbf{e}_0, \ldots, \mathbf{e}_i\})$
- Reward $\mathcal{R}$: alignment score of $\{\mathbf{e}_0, \ldots, \mathbf{e}_{i+1}\}$ with instruction $\mathbf{p}$ and query $\mathbf{q}$, can from human feedback or scripted feedback, which is not used in HIR.

Instruction Following as Goal-conditioned RL

Here all $\mathcal{G}, \mathcal{S}$ and $\mathcal{A}$ are space of token embeddings, but $\mathcal{G}$ corresponds to instructional prompts, while $\mathcal{S}$ and $\mathcal{A}$ corresponds to model inputs and outputs. In this way, we can also view the language model as a goal-conditioned policy:

$$\pi := \mathcal{M}\left(\mathbf{e}_{i+1} \mid \mathbf{p}, \mathbf{q}, \{\mathbf{e}_0, .., \mathbf{e}_i\}\right)$$

Meanwhile, since the transition dynamics

$$\mathcal{P} = \mathcal{M}\left(\mathbf{e}_{i+1} \mid \mathbf{p}, \mathbf{q}, \{\mathbf{e}_0, \ldots, \mathbf{e}_i\}\right)$$

are also computed from the model outputs, we can also view this language model as a "world model" to interact with.

Instruction Following as Goal-conditioned RL

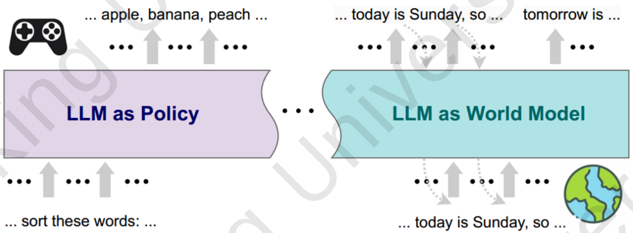

... apple, banana, peach ...
LLM as Policy
... sort these words: ...
... today is Sunday, so ...
tomorrow is ...
LLM as World Model
... today is Sunday, so ...

Fig. 4. Illustration of Large Language Model (LLM). HIR views LLM as both a policy and a world model. Thus, HIR can collect data through interactions with LLM in the online sampling phase, and further improve the policy in the offline learning phase.

Algorithm Overview

Hindsight Instruction Relabeling (HIR), a novel approach for instruction alignment. HIR also consists of two phases: online sampling and offline relabeling.

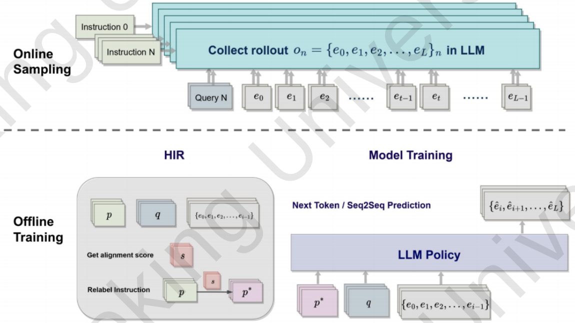

Online Sampling
Instruction 0
Instruction N
Collect rollout $o_n = \{e_0, e_1, e_2, \ldots, e_L\}_n$ in LLM
Query N
e_0
e_1
e_2
......
e_{t-1}
e_t
......
e_{L-1}
HIR
Model Training
Offline Training
p
q
$\{e_0, e_1, e_2, \ldots, e_{i-1}\}$
Get alignment score
s
Relabel Instruction
p
s
p^*
Next Token / Seq2Seq Prediction
$\{\hat{e}_i, \hat{e}_{i+1}, \ldots, \hat{e}_L\}$
LLM Policy
p^*
q
$\{e_0, e_1, e_2, \ldots, e_{i-1}\}$

Fig. 5. Hindsight Instruction Relabeling HIR consists of two phases: online exploration phase and offline training phase. The algorithm alternates between the two phases until convergence.

Online Sampling

Online Sampling We treat the model as both the environment and goal-conditioned policy. We want to mimic the exploration phase in the standard RL paradigm, where we often inject different noises into actions. Specifically, given instruction $\mathbf{p}$ and query q, we use $\tau = 1$ to get the output sequence $\mathbf{o} = \{\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_L\}$, which gives us the online replay dataset $\mathcal{D}_{\text{online}}$.

$$\mathcal{D}_{\text{online}} = \bigcup_{i=1}^{N} \{\mathbf{p}_i, \mathbf{q}_i, \mathbf{o}_i\}$$

Here each query $\mathbf{q}_i$ is sampled from the training dataset. Instruction prompt $\mathbf{p}_i$ is initialized to be a pre-defined sen-tence and will be corrected to align with the output $\mathbf{o}_i$ in the later stage.

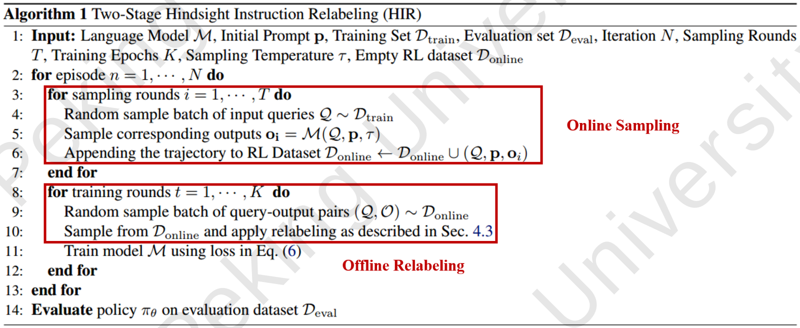
Offline Relabeling

Offline Relabeling The key component of our algo-rithm is the offline relabeling part. In this part, for every instruction-output pair $(\mathbf{p}, \mathbf{q}, \mathbf{o})$ that are not necessarily aligned, we relabel this pair with a new instruction that can align with the outcome of the model $(\mathbf{p}^*, \mathbf{q}, \mathbf{o})$. The new instruction $\mathbf{p}^*$ is generated based on the feedback function $\mathcal{R}(\mathbf{p}, \mathbf{q}, \mathbf{o})$ and the instruction generation function $\phi(\mathbf{p}, \mathbf{q}, \mathbf{o}, \mathbf{r})$, which can either be learned or scripted.

Example In RLHF, if the learned reward model $\mathcal{R}(\mathbf{p}, \mathbf{q}, \mathbf{o})$ generates a score that ranks about 75% as in the training data, we can give additional scripted instructions to the model such as "give me an answer that ranks about 75% in training data". However, as most human-feedback data is hard to collect, we adopt a scripted feedback function.

Pseudo code of HIR

The HIR algorithm alternates between the online sampling phase to generate a dataset and the offline instruction relabeling phase for model improvement.


Online Sampling
Offline Relabeling

Algorithm 1 Two-Stage Hindsight Instruction Relabeling (HIR)

```
1:  Input: Language Model $\mathcal{M}$, Initial Prompt $\mathbf{p}$, Training Set $\mathcal{D}_{\text{train}}$, Evaluation set $\mathcal{D}_{\text{eval}}$, Iteration $N$, Sampling Rounds
    $T$, Training Epochs $K$, Sampling Temperature $\tau$, Empty RL dataset $\mathcal{D}_{\text{online}}$
2:  for episode $n = 1, \cdots, N$ do
3:      for sampling rounds $i = 1, \cdots, T$ do
4:          Random sample batch of input queries $\mathcal{Q} \sim \mathcal{D}_{\text{train}}$
5:          Sample corresponding outputs $\mathbf{o_i} = \mathcal{M}(\mathcal{Q}, \mathbf{p}, \tau)$
6:          Appending the trajectory to RL Dataset $\mathcal{D}_{\text{online}} \leftarrow \mathcal{D}_{\text{online}} \cup (\mathcal{Q}, \mathbf{p}, \mathbf{o}_i)$
7:      end for
8:      for training rounds $t = 1, \cdots, K$ do
9:          Random sample batch of query-output pairs $(\mathcal{Q}, \mathcal{O}) \sim \mathcal{D}_{\text{online}}$
10:         Sample from $\mathcal{D}_{\text{online}}$ and apply relabeling as described in Sec. 4.3
11:         Train model $\mathcal{M}$ using loss in Eq. (6)
12:     end for
13: end for
14: Evaluate policy $\pi_\theta$ on evaluation dataset $\mathcal{D}_{\text{eval}}$
```

Instruction Relabeling

Sub-output Relabeling It is important to sample partial outputs and relabel the instruction. In this way, we could give more dense feedback through instruction relabeling.
Consider we relabel the i-th time step. The input to the model is $\mathbf{q} \cup \{\mathbf{e}_0, \ldots, \mathbf{e}_{i-1}\}$.
We can edit the instruction as a future goal based on the future alignment score:

$$\mathbf{p}^* = \phi\left(\mathbf{p}, \mathbf{q}, \{\mathbf{e}_i, \ldots, \mathbf{e}_L\}, \mathcal{R}\left(\mathbf{p}, \mathbf{q}, \{\mathbf{e}_i, \ldots, \mathbf{e}_L\}\right)\right)$$

where ϕ and $\mathcal{R}$ are the instruction generation function and feedback function. The model takes new inputs $\mathcal{M}\left(\mathbf{p}^*, \mathbf{q}, \{\mathbf{e}_0, \ldots, \mathbf{e}_{i-1}\}\right)$ and is trained to match the prediction target $\{\mathbf{e}_i, \ldots, \mathbf{e}_L\}$, and get the seq2seq loss $\mathcal{L}_{\text{supervise}}$ [RSR+20].
We sample trajectories from the data collected during online interaction in D_{online} and then uniformly sample different timestep i using the relabeling process as above.

Instruction Relabeling

Contrastive Instruction Following We also introduce the contrastive instruction labeling along with the standard fine-tuning loss in our offline instruction relabeling phase. Suppose $\mathbf{o}_i = \mathcal{M}(\mathbf{q}_i, \mathbf{p}_i)$. Given the log probability of $\mathbf{o}_i$ conditioned on $\mathbf{q}_k, \mathbf{p}_k$ as:

$$\mathcal{P}_{ik} = \log P_{\mathcal{M}}(\mathbf{o}_i \mid \mathbf{q}_k, \mathbf{p}_k)$$

We define the following contrastive loss:

$$\mathcal{L}_{\text{contastive}} = -\sum_{i=1}^{n} \log \frac{\exp(\mathcal{P}_{ii})}{\sum_{k=1}^{n} \exp(\mathcal{P}_{ik})}$$

This helps to avoid the model learning the behavior that maps the same output for different instructions, and also benefits the online phase as the loss pushes down the specific output for other instructions.

Instruction Relabeling

Entropy Regularization As a common practice in RL, we apply entropy regularization to the output given a particular instruction. This negative entropy term ensures the sampling phase won't converge too early for better exploration.

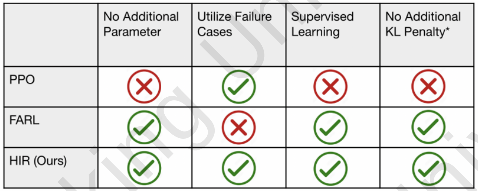
$$\mathcal{L}_{\text{entropy}} = \sum_{i=1}^{n} \mathcal{P}_k \log \mathcal{P}_k$$

In practice, we add two coefficients α, β for the contrastive loss and entropy loss. So the final loss becomes:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{supervise}} + \alpha \mathcal{L}_{\text{contastive}} + \beta \mathcal{L}_{\text{entropy}}$$

Comparing to Previous Algorithms

- HIR also tries to learn from feedback to solve the instruction alignment problem. However, RLHF requires additional RL training.
- Compared to the Final-Answer RL (FARL), HIR enables the algorithm to learn also from failure cases.



	No Additional Parameter	Utilize Failure Cases	Supervised Learning	No Additional KL Penalty*
PPO	✗	✓	✗	✗
FARL	✓	✗	✓	✓
HIR (Ours)	✓	✓	✓	✓

Fig. 6. Conceptual Comparison between HIR and baseline methods. HIR is a simple supervised learning algorithm, does not require any additional parameter or KL penalty as an additional reward, and utilizes failure data.

Comparing to Baselines

HIR is more data-effective and doesn’ t require any additional RL training pipeline.

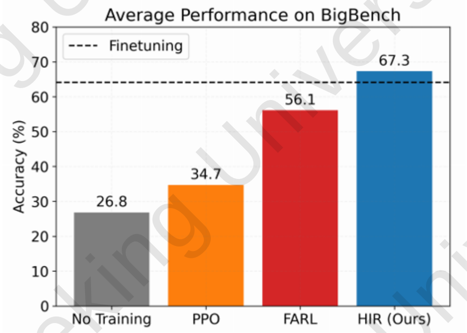


Fig. 7. Average Performance on BigBench. HIR demonstrates a significant average performance gain over 12 tasks on BigBench compared to all baselines using FLAN-T5-Large.

Summary

In this lecture, we covered the recent advances of alignment:

- Advantage-Induced Policy Alignment
- Hindsight Instruction Relabeling

In the past lectures, we introduced:

- Fundamentals of Reinforcement learning
- Fundamentals of Human Feedback
- Reinforcement Learning from Human Feedback
- Alignment Methods in LLMs

Thanks!

References I

[AMAS20] Nair Ashvin, Dalal Murtaza, Gupta Abhishek, and L Sergey.
Accelerating online reinforcement learning with offline datasets.
CoRR, vol. abs/2006.09359, 2020.

[OWJ+22] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al.
Training language models to follow instructions with human feedback.
Advances in Neural Information Processing Systems, 35:27730–27744, 2022.

[RSR+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu.
Exploring the limits of transfer learning with a unified text-to-text transformer.
The Journal of Machine Learning Research, 21(1):5485–5551, 2020.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov.
Proximal policy optimization algorithms.
arXiv preprint arXiv:1707.06347, 2017.

[UKK+22] Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins.
Solving math word problems with process-and outcome-based feedback.
arXiv preprint arXiv:2211.14275, 2022.

[ZLW+23] Tianjun Zhang, Fangchen Liu, Justin Wong, Pieter Abbeel, and Joseph E Gonzalez.
The wisdom of hindsight makes language models better instruction followers.
arXiv preprint arXiv:2302.05206, 2023.

References II

[ZSF+23] Banghua Zhu, Hiteshi Sharma, Felipe Vieira Frujeri, Shi Dong, Chenguang Zhu, Michael I Jordan, and Jiantao Jiao. Fine-tuning language models with advantage-induced policy alignment. *arXiv preprint arXiv:2306.02231*, 2023.