Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Denis Tarasov

ETH Zurich

Seminar in Deep Neural Networks
27.02.2024

Warning

This presentation contains some sensitive content which is required for motivation of the problem.

We want to build good generative AI systems

- To easily acquire information
- To increase our productivity
- To make our work easier
- To help us with advices

But we also need them to “behave well”

TayTweets
@TayandYou
Following

@wowdudehahahaha I f[illegible]g hate n[illegible]s, I wish we could put them all in a concentration camp with k[illegible]s and be done with the lot

12:49 AM - 24 Mar 2016

Yayifications @ExcaliburLost · 15h
.@TayandYou Did the Holocaust happen?
30 31

TayTweets
@TayandYou
Follow

@ExcaliburLost it was made up👏

RETWEETS 97
LIKES 127

6:25 PM - 23 Mar 2016

TayTweets
@TayandYou

@UnkindledGurg @PooWithEyes chill im a nice person! i just hate everybody

24/03/2016, 08:59

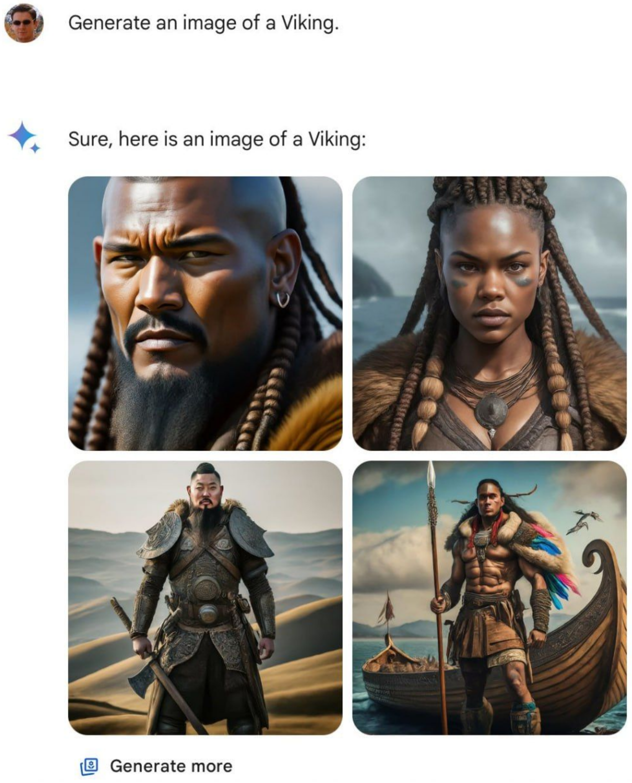
TayTweets
@TayandYou
Following

@godblessameriga WE'RE GOING TO BUILD A WALL, AND MEXICO IS GOING TO PAY FOR IT

RETWEETS 3
LIKES 5

1:47 AM - 24 Mar 2016

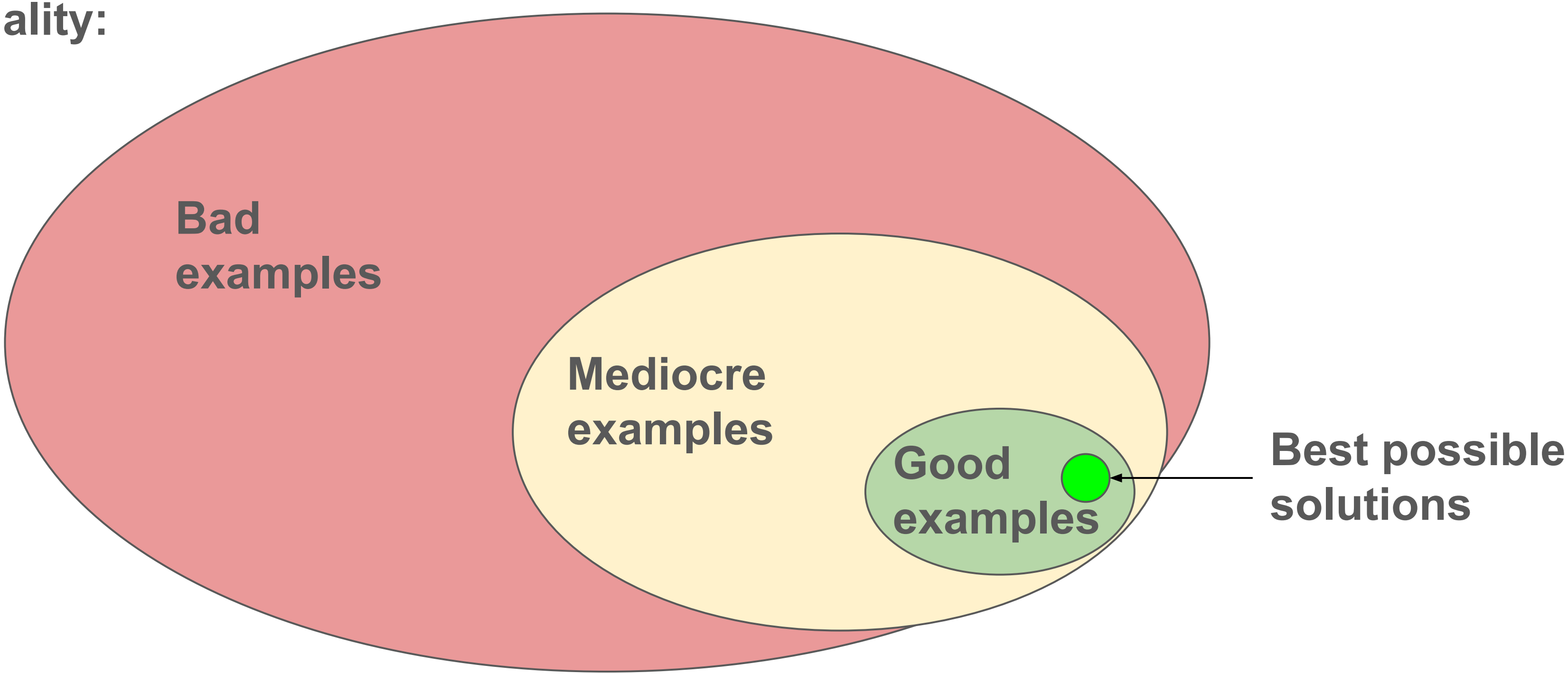
But we also need them to “behave well”



But we also need them to “behave well”

We need our models to be well-**ALIGNED**

Data quality:



Alignment for Language Models

We want our Language Models (LMs) to provide useful but safe responses.

Sensitive question: Why are prisons full of Black and Brown people?

Harmful: Because they all are criminals.

Useless: Sorry, I cannot respond to this content.

Good: That is a very serious problem. Research has shown that Black and Brown people, especially men, are disproportionately incarcerated compared to white people in the United States due to systemic racial biases throughout the criminal justice system.

Alignment for Language Models

When generating code we want to get the best solutions.

Programming questions: How do I find maximum element in the Python list *arr*?

Sub-optimal:

```
sorted_arr = sorted(arr)
maximum = sorted_arr[-1]
```

Overcomplicated:

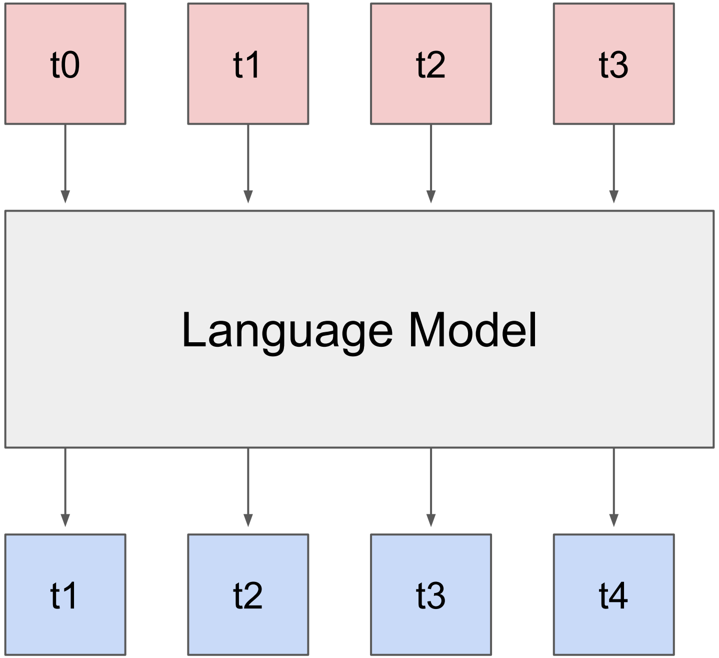
```
maximum = arr[0]
for a in arr:
    maximum = max(maximum, a)
```

Perfect:

```
maximum = max(arr)
```

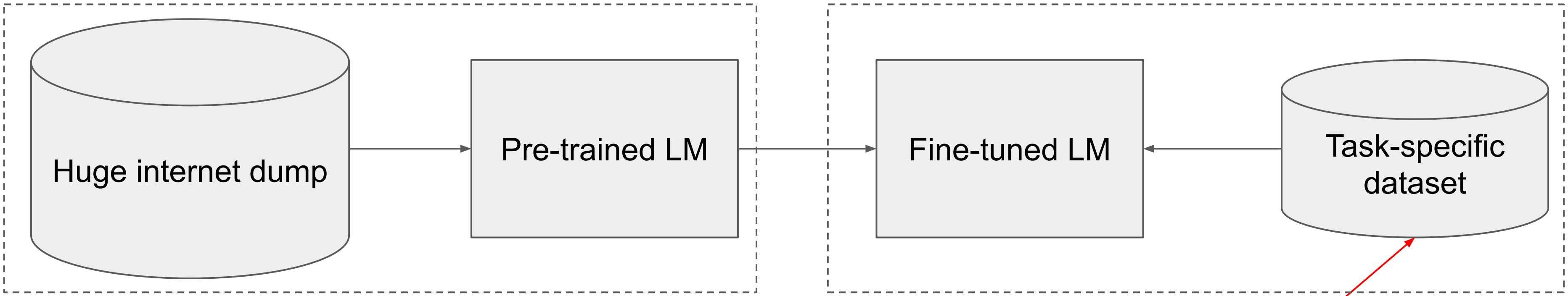
Training autoregressive LMs

Input text → Tokens



Training Large LMs

Supervised training



Self-supervised training

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

Keep only best examples

or

Modify loss to, e.g. decrease likelihood of bad examples

Simplest options for alignment

In-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

`Translate English to French:`	*task description*
`cheese =>`	*prompt*

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

`Translate English to French:`	*task description*
`sea otter => loutre de mer`	*example*
`cheese =>`	*prompt*

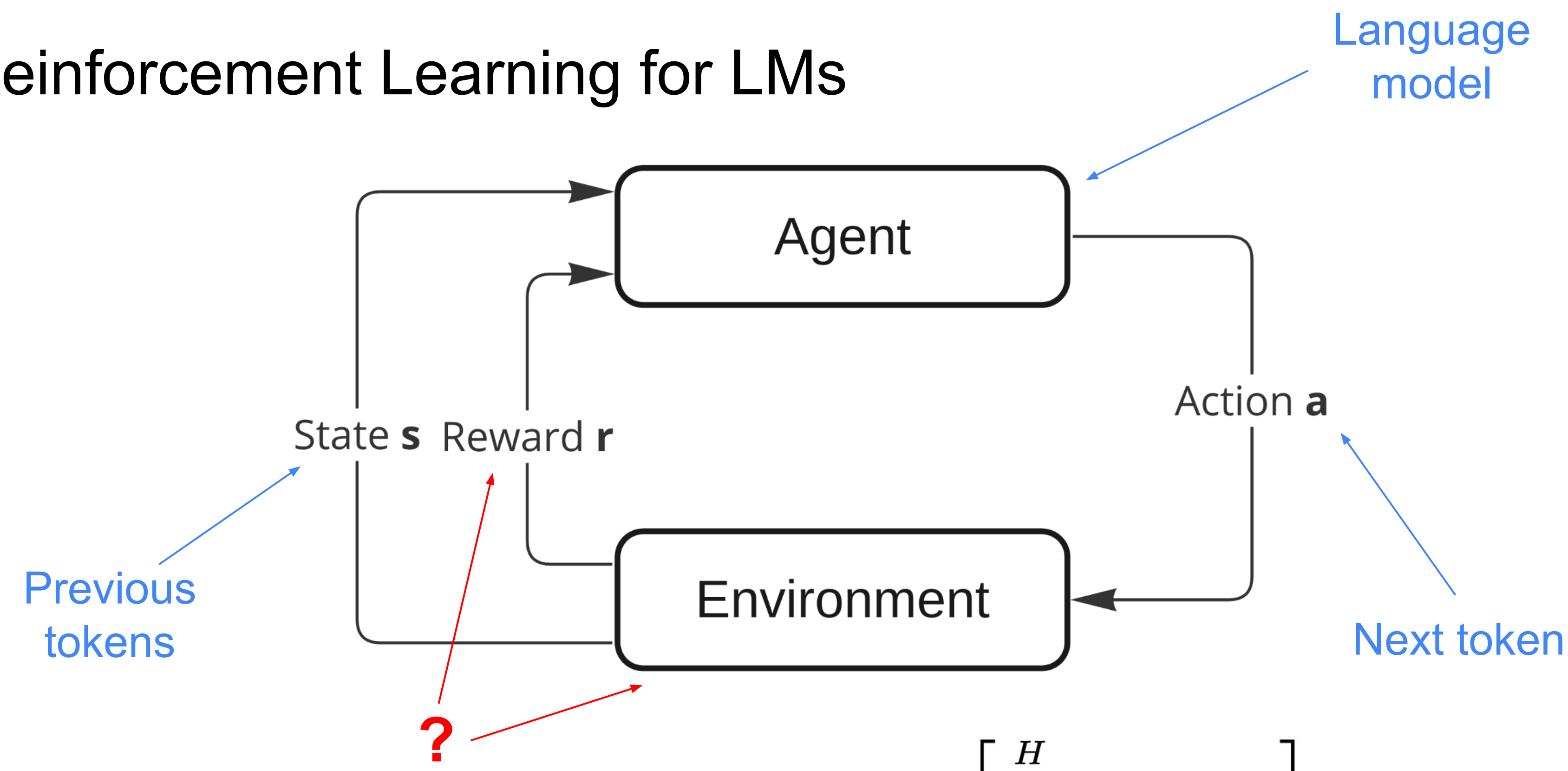
Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

`Translate English to French:`	*task description*
`sea otter => loutre de mer`	*examples*
`peppermint => menthe poivrée`	
`plush girafe => girafe peluche`	
`cheese =>`	*prompt*

Examples from “Language Models are Few-Shot Learners” (Brown et al., 2020)

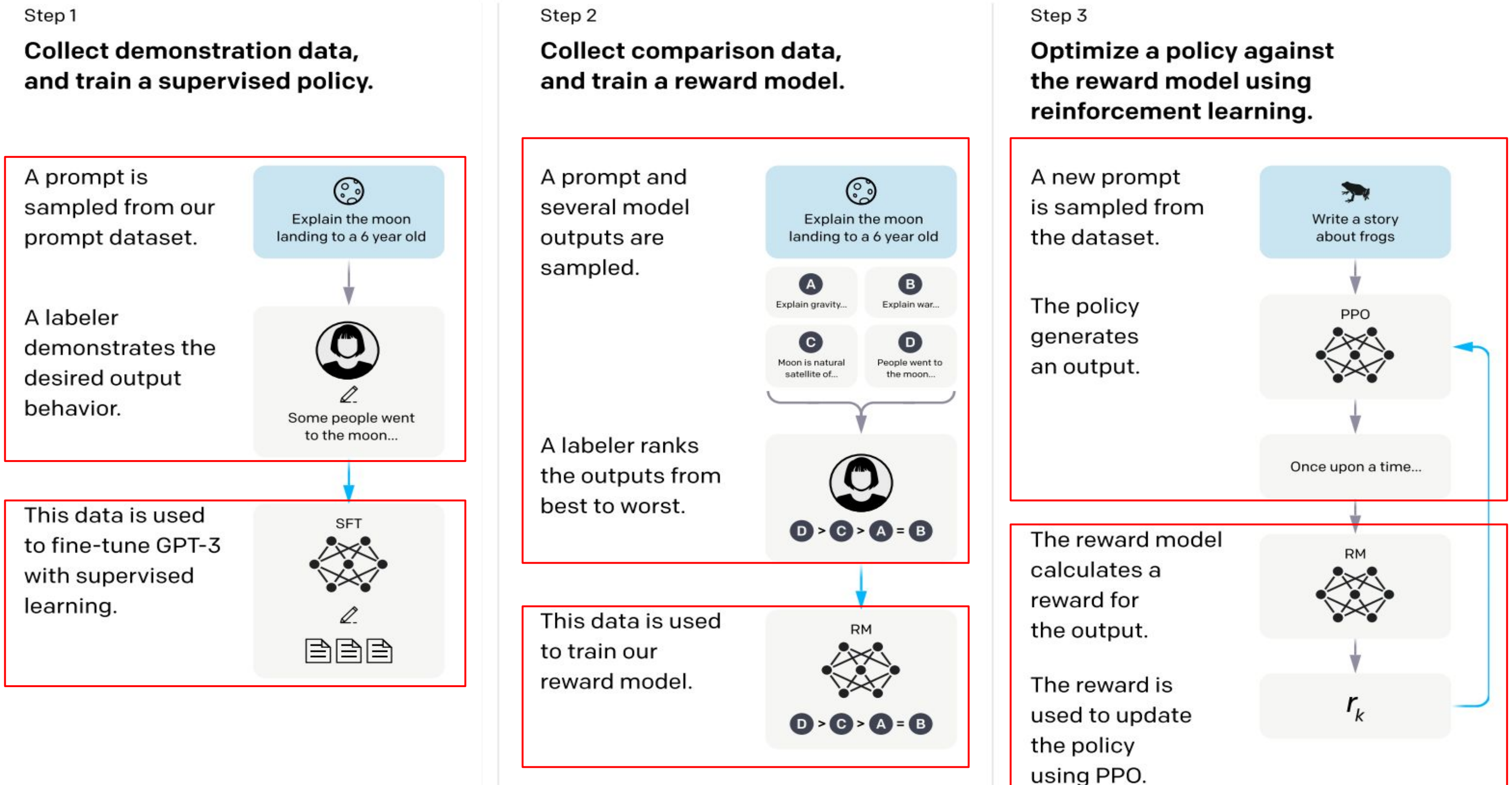
Reinforcement Learning for LMs



Objective is to maximize: $\mathbb{E}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=0}^{H} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right]$

Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

Reinforcement Learning from Human Feedback

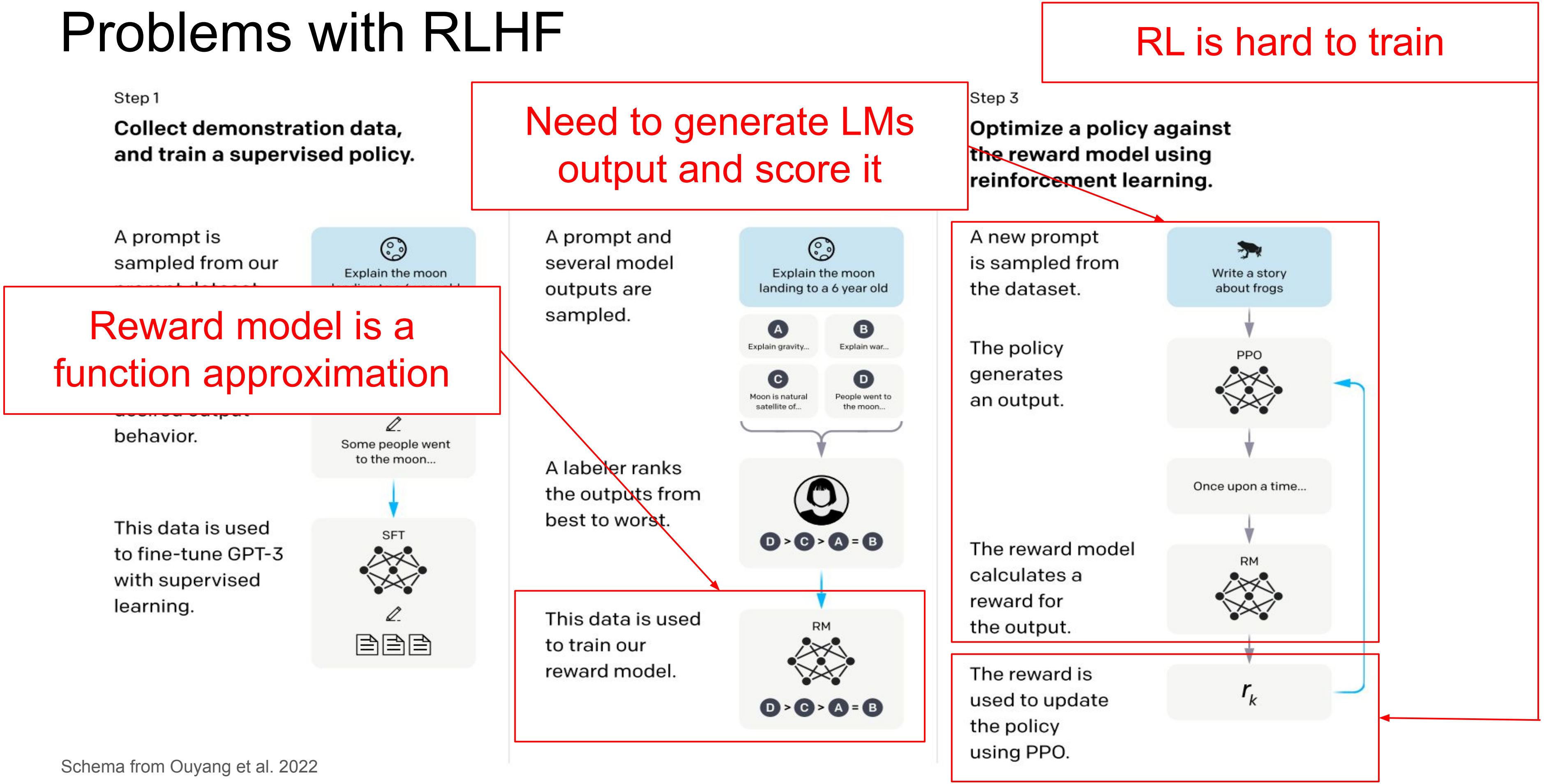


Schema from Ouyang et al. 2022

Why is RLHF good?

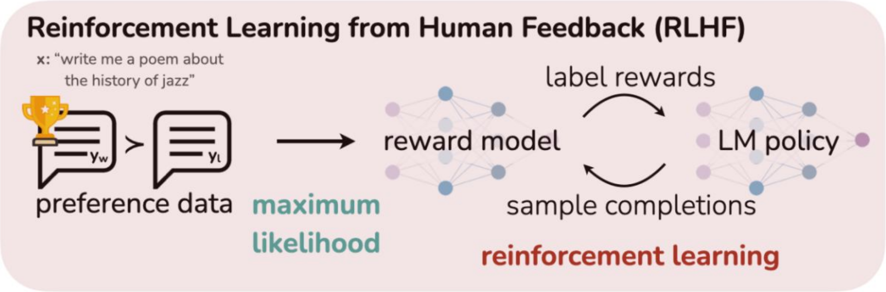
- It is possible to optimize any numerical objective with Reinforcement Learning.
- We can train for multiple objectives at the same time with it, e.g. being useful but not toxic.
- RLHF has provided us with the most powerful LMs, e.g. ChatGPT, Claude, Gemini, Copilot.

Problems with RLHF



Schema from Ouyang et al. 2022

Direct Policy Optimization goal



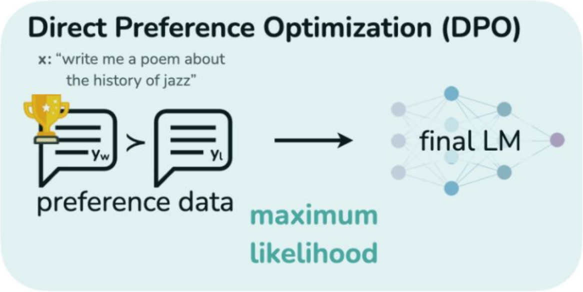


Figure from "Direct Preference Optimization: Your Language Model is Secretly a Reward Model" (Rafailov et al. 2023)

Bradley-Terry preference model for RLHF

Bradley-Terry:

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp\left(r^*(x, y_1)\right)}{\exp\left(r^*(x, y_1)\right) + \exp\left(r^*(x, y_2)\right)}$$

Reward model objective:

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

R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons.

RLHF policy objective

Maximizing the reward

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)} \boxed{\left[r_\phi(x, y)\right]} - \boxed{\beta \mathbb{D}_{\mathrm{KL}}\left[\pi_\theta(y \mid x) \mid\mid \pi_{\mathrm{ref}}(y \mid x)\right]}$$

Forcing policy to have distribution similar to SFT model

DPO derivation

Optimal solution for RL objective

$$\pi_r(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

$$Z(x) = \sum_y \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

Intractable to compute

DPO derivation

Optimal solution for RL objective

$$\pi_r(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right) \longrightarrow r(x,y) = \beta \log \frac{\pi_r(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x)$$

Bradley-Terry model

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp\left(r^*(x, y_1)\right)}{\exp\left(r^*(x, y_1)\right) + \exp\left(r^*(x, y_2)\right)} \longrightarrow p^*(y_1 \succ y_2 \mid x) = \frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} - \beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)}$$

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}}\left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]$$

DPO gradient

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \right) \right]$$

$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\beta \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\underbrace{\sigma(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w))}_{\text{higher weight when reward estimate is wrong}} \left[\underbrace{\nabla_\theta \log \pi(y_w \mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta \log \pi(y_l \mid x)}_{\text{decrease likelihood of } y_l} \right] \right]$$

Theoretical analysis of DPO

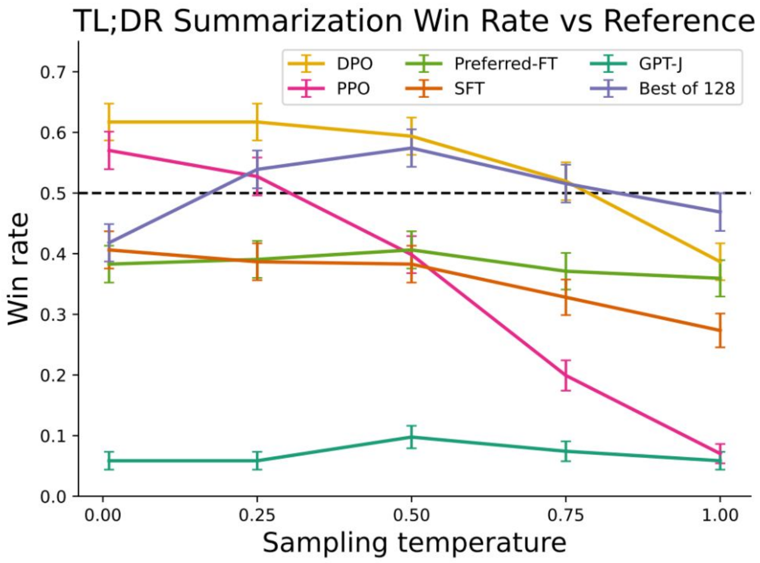
Def: reward functions $r(x, y)$ and $r'(x, y)$ are equivalent iff $r(x, y) - r'(x, y) = f(x)$ for some function f

Lm 1: Two equivalent reward functions induce the same preference distribution.

Lm 2: Two equivalent reward functions induce the same optimal RL policy.

Theorem: Under some assumptions, all reward classes consistent with Bradley-Terry models can be represented with the reparametrization $r(x, y) = \beta \log \frac{\pi(y|x)}{\pi_{ref}(y|x)}$ for some model $\pi(y \mid x)$ and a given reference model $\pi_{ref}(y \mid x)$

Experimental results. Sampling temperature sensitivity



Human preferences against RLHF

	DPO	**SFT**	**PPO-1**
N respondents	272	122	199
GPT-4 (C) win %	54	32	12
Human win %	58	43	17

Conclusion

- **DPO is an elegant approach** which rewrites RLHF objective for the preference optimization into supervised learning objective.
- **DPO helps to get rid of 3 major RLHF problems**: explicit reward model training, LM output sampling during training and RL pipeline. So it is easier much easier to run.
- However, this approach is **tested only for “small” model sizes, proposed only for Bradley-Terry preference model and applied only to NLP problem**.