Plans for Remainder of the Course

- **Project presentation is this Thursday**
    1. 5 min pitch talk for each project
    2. Peer graded

- HW3: due March 14'th

- Project report due March 16'th (Sunday) 5 pm

DATA 37200: Learning, Decisions, and Limits
(Winter 2025)

Reinforcement Learning from Human Feedback

Instructor: Haifeng Xu

Outline

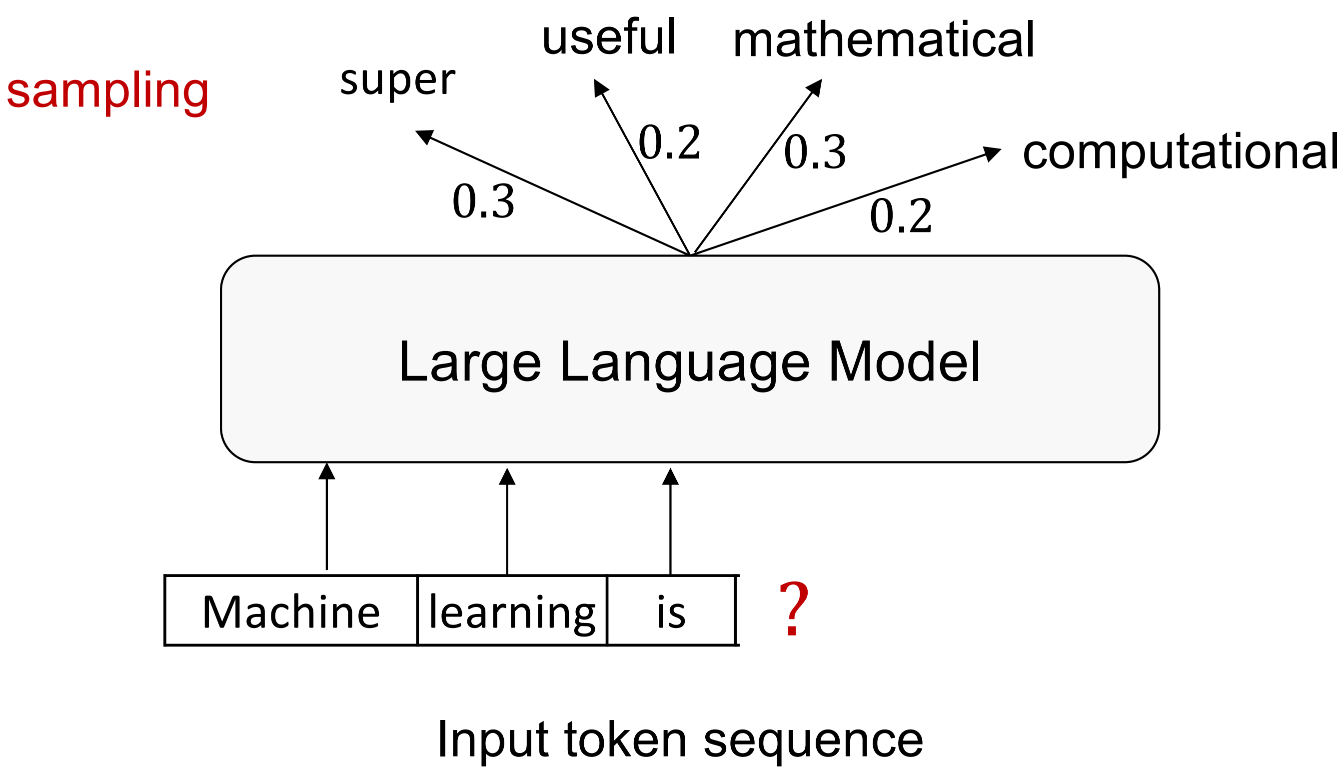
- What and Why?
- Procedures of RLHF
- RL without Rewards: Direct Preference Optimization (DPO)

Many active researches are ongoing; this lecture covers basics

Language Models (LMs)

Next token prediction in auto-regressive way

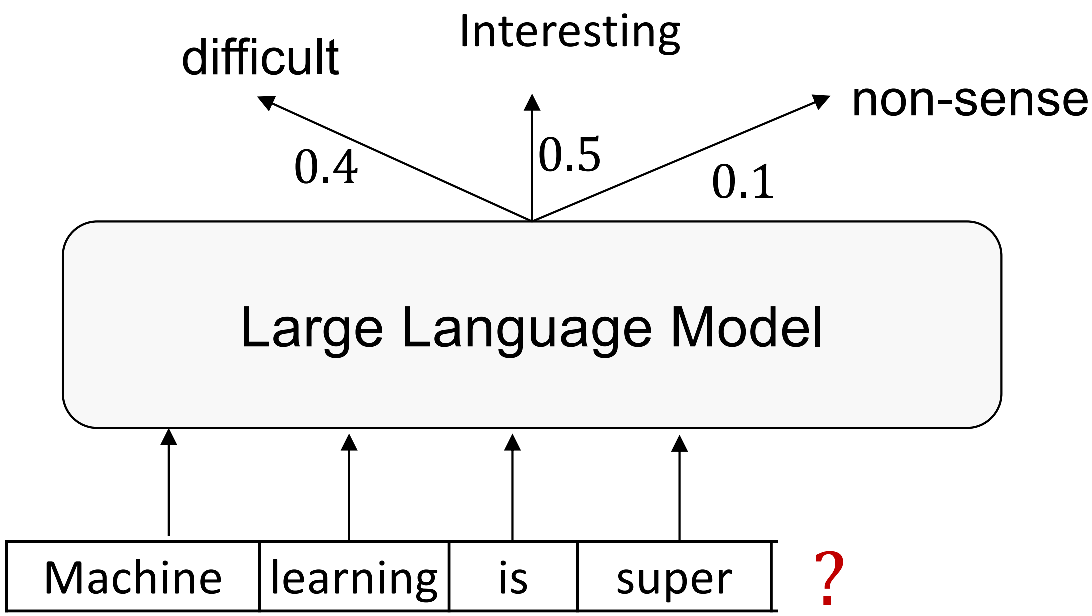
$\mathrm{Pr}(\text{next token} \mid \text{input})$



Language Models (LMs)

Next token prediction in auto-regressive way

Pr(next token | input)



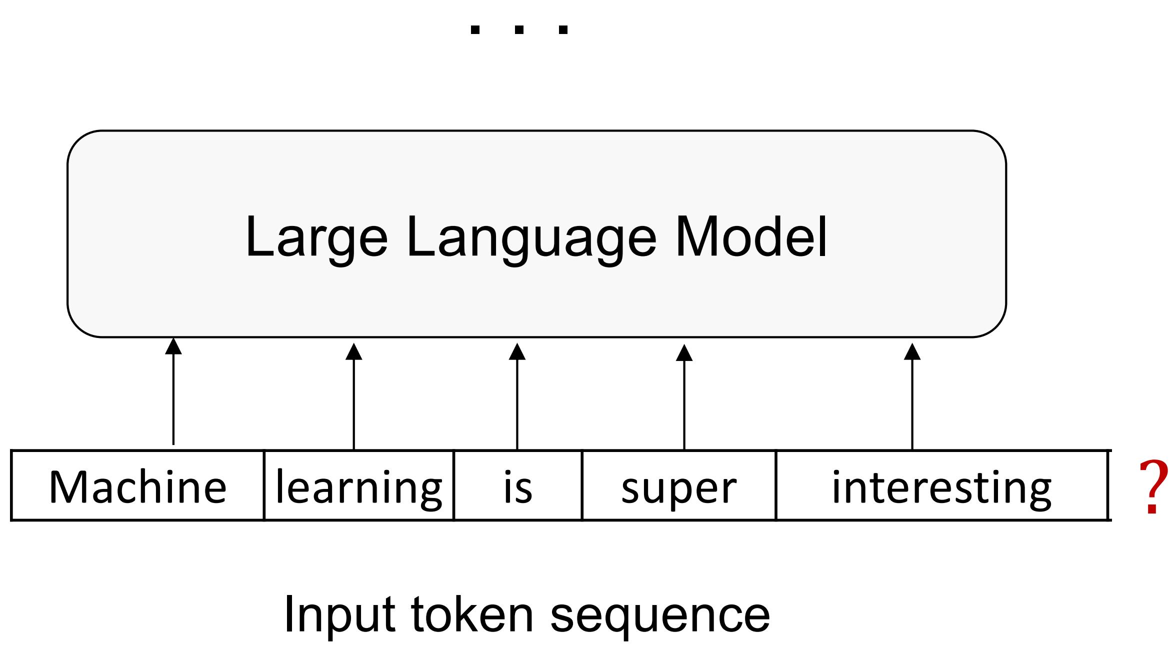
Input token sequence

An Autoregressive Process

Language Models (LMs)

Next token prediction in auto-regressive way

- Mathematical abstraction: $p(y|x)$
- It predicts the next token/phrase



An Autoregressive Process

Major Steps for Building an LLM

- **Step 1 is pre-training** – supervised learning over massive text data so that language model (LM) learns probabilities of next token
  - Huge engineering effort to tune billions of parameters of transfer
  - Already achieve good performance in GPT2 with ~1B paras [Radford et al.,'19]
  - GPT3 with 175B parameters is even better [Brown et al. 2020]

Major Steps for Building an LLM

- **Step 1 is pre-training** – supervised learning over massive text data so that language model (LM) learns probabilities of next token
- Limitations:
  1. Need to carefully write your prompts to trigger desired predictions

Passage: Tom Brady…

<u>Prompt</u>
Q: Where was Tom Brady born? A:…

The cat couldn't fit in to the hat because it was too big.

<u>Task: does "it" refer to the "cat" or the "hat"</u>

<u>Prompt</u>
Is P(…because the **cat** is too big) > P(…because the **hat** is too big)?

Major Steps for Building an LLM

- **Step 1 is pre-training** – supervised learning over massive text data so that language model (LM) learns probabilities of next token
- Limitations:
  1. Need to carefully write your prompts to trigger desired predictions
  2. Bad at reasoning tasks, even simple ones

Large number additions

543854 + 143865?

The cafeteria has 23 apples. They used 20 apples to make lunch and bought 6 more. How many apples do they have now?

Answer: 26 ❌

This is the time where prompt engineering become really popular;
A representatively well-know idea is “chain of thought”

Major Steps for Building an LLM

- **Step 1 is pre-training** – supervised learning over massive text data so that language model (LM) learns probabilities of next token
- Limitations:
  1. Need to carefully write your prompts to trigger desired predictions
  2. Bad at reasoning tasks, even simple ones

<u>Large number additions</u>

543854 + 143865?

The cafeteria has 23 apples. They used 20 apples to make lunch and bought 6 more. How many apples do they have now?

Can you show your reasoning step by step?

LLMs will then explain the thinking process, and very often output correct answer → **Interesting "dark art"**

Major Steps for Building an LLM

- **Step 1 is pre-training** – supervised learning over massive text data so that language model (LM) learns probabilities of next token
- Limitations:
  1. Need to carefully write your prompts to trigger desired predictions
  2. Bad at reasoning tasks, even simple ones
  3. Clever prompt engineering can work sometimes, but certainly have a limit

Fundamental reason: language modeling ≠ assisting humans

PROMPT	*Explain the moon landing to a 6 year old in a few sentences.*
COMPLETION	GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

Major Steps for Building an LLM

- **Step 2 is to align LLMs with human intents** – a successful way is reinforcement learning from human feedback (RLHF)

Core idea:

- ✓ Introduce rewards to model human preferences over languages
- ✓ Then use rewards to "fine-tune" LLMs towards human's preferences via RL

- The idea of using RL for language models has been there for a while, but has been difficult to make it work (LMs are complex)
- Gain more momentum recently due to newer RL algorithms better suited for LMs (e.g., proximal policy optimization/PPO [Schulman et al. 2017])

Outline

- What and Why?
- Procedures of RLHF [Ouyang et al. 2022]
- RL without Rewards: Direct Preference Optimization (DPO)

Part 1: Formulating the RL Problem

- Given prompt x, we want to predict response y
- Pre-training already gives us an LLM $p^{PT}(y|x)$
- Suppose human has reward $R(y|x)$ for prompt x
- RLHF goal: find $p_\theta^{RL}(y|x)$ – a neural network parameterized by θ – to better predict y

$$\theta^* = \arg\max_\theta \underbrace{\mathbb{E}_{y \sim p_\theta^{RL}(y|x)} [R(y|x)]}_{\text{Expected reward under RL policy}}$$

Would not work..

If only maximizing rewards, LMs will output non-sensible sentences, since world knowledge such as language syntax in $p^{PT}(y|x)$ was ignored

Part 1: Formulating the RL Problem

- Given prompt x, we want to predict response y
- Pre-training already gives us an LLM $p^{PT}(y|x)$
- Suppose human has reward $R(y|x)$ for prompt x
- RLHF goal: find $p_\theta^{RL}(y|x)$ – a neural network parameterized by θ – to better predict y

$$\theta^* = \arg\max_\theta \left\{ \mathbb{E}_{y\sim p_\theta^{RL}(y|x)} [R(y|x)] \underline{-\beta \cdot KL(p_\theta^{RL}(y|x), p^{PT}(y|x))} \right\}$$

Panelize deviation from pre-trained model $p^{PT}(y|x)$

Part 1: Formulating the RL Problem

- Given prompt x, we want to predict response y
- Pre-training already gives us an LLM $p^{PT}(y|x)$
- Suppose human has reward $R(y|x)$ for prompt x
- RLHF goal: find $p_\theta^{RL}(y|x)$ – a neural network parameterized by θ – to better predict y

$$\theta^* = \arg\max_\theta \left\{ \mathbb{E}_{y \sim p_\theta^{RL}(y|x)} [R(y|x)] \underline{-\beta \cdot KL(p_\theta^{RL}(y|x), p^{PT}(y|x))} \right\}$$

Panelize deviation from pre-trained model $p^{PT}(y|x)$

Recall from earlier lecture: $KL\left(p_\theta^{RL}, p^{PT}\right) = \sum_y p_\theta^{RL}(y) \cdot \log \frac{p_\theta^{RL}(y)}{p^{PT}(y)}$

$$= \mathbb{E}_{y \sim p_\theta^{RL}(y|x)} \log \frac{p_\theta^{RL}(y)}{p^{PT}(y)}$$

Part 1: Formulating the RL Problem

- Given prompt x, we want to predict response y
- Pre-training already gives us an LLM $p^{PT}(y|x)$
- Suppose human has reward $R(y|x)$ for prompt x
- RLHF goal: find $p_\theta^{RL}(y|x)$ – a neural network parameterized by θ – to better predict y

$$\theta^* = \arg\max_\theta \left\{ \mathbb{E}_{y \sim p_\theta^{RL}(y|x)} [R(y|x)] - \beta \cdot KL(p_\theta^{RL}(y|x), p^{PT}(y|x)) \right\}$$

$$\Leftrightarrow \theta^* = \arg\max_\theta \mathbb{E}_{y \sim p_\theta^{RL}(y|x)} \left[R(y|x) - \beta \cdot \log \frac{p_\theta^{RL}(y)}{p^{PT}(y)} \right]$$

Recall from earlier lecture: $KL\left(p_\theta^{RL}, p^{PT}\right) = \sum_y p_\theta^{RL}(y) \cdot \log \frac{p_\theta^{RL}(y)}{p^{PT}(y)}$

$$= \mathbb{E}_{y \sim p_\theta^{RL}(y|x)} \log \frac{p_\theta^{RL}(y)}{p^{PT}(y)}$$

Part 1: Formulating the RL Problem

- Given prompt x, we want to predict response y
- Pre-training already gives us an LLM $p^{PT}(y|x)$
- Suppose human has reward $R(y|x)$ for prompt x
- RLHF goal: find $p_\theta^{RL}(y|x)$ – a neural network parameterized by θ – to better predict y

$$\theta^* = \arg\max_\theta \{\mathbb{E}_{y\sim p_\theta^{RL}(y|x)}\,[R(y|x)] - \beta \cdot KL(p_\theta^{RL}(y|x), p^{PT}(y|x))\,\}$$

$$\Leftrightarrow \theta^* = \arg\max_\theta \mathbb{E}_{y\sim p_\theta^{RL}(y|x)}\left[R(y|x) - \beta \cdot \log \frac{p_\theta^{RL}(y)}{p^{PT}(y)}\right]$$

- Need to also take expectation over x – omitted here for math cleanness
- Challenge is to **estimate gradient** of objective function – particularly partial gradient of θ w.r.t. $\mathbb{E}_{y\sim p_\theta^{RL}(y|x)}$ in order to apply chain rule

Policy Gradient

- We need to calculate (ignoring x for now)

$$\nabla_{\theta}\left[\mathbb{E}_{y\sim p_{\theta}^{RL}(y)}\,\hat{R}(y|\theta)\right]$$

where $\hat{R}(y) = R(y|x) - \beta\cdot\log\frac{p_{\theta}^{RL}(y)}{p^{PT}(y)}$

$$= \frac{\partial\mathbb{E}_{y\sim p_{\theta}^{RL}(y)}\,\hat{R}(y|\theta)}{\partial\theta} + \mathbb{E}_{y\sim p_{\theta}^{RL}(y)}\,\frac{\partial\hat{R}(y|\theta)}{\partial\theta}$$

Def of partial derivative

$$\Leftrightarrow \theta^{*} = \arg\max_{\theta}\,\mathbb{E}_{y\sim p_{\theta}^{RL}(y|x)}\left[R(y|x) - \beta\cdot\log\frac{p_{\theta}^{RL}(y)}{p^{PT}(y)}\right]$$

Policy Gradient

- We need to calculate (ignoring x for now)

$$\nabla_\theta \left[\mathbb{E}_{y\sim p_\theta^{RL}(y)} \hat{R}(y|\theta)\right] \quad \text{where } \hat{R}(y) = R(y|x) - \beta \cdot \log \frac{p_\theta^{RL}(y)}{p^{PT}(y)}$$

$$= \frac{\partial \mathbb{E}_{y\sim p_\theta^{RL}(y)} \hat{R}(y|\theta)}{\partial \theta} + \underline{\mathbb{E}_{y\sim p_\theta^{RL}(y)} \frac{\partial \hat{R}(y|\theta)}{\partial \theta}}$$

Def of partial derivative

Easy to estimate since it is an expectation

- ✓ Sample a bunch of y's
- ✓ Compute empirical mean of $\frac{\partial \hat{R}(y|\theta)}{\partial \theta}$

Policy Gradient

- We need to calculate (ignoring x for now)

$$\nabla_\theta \left[\mathbb{E}_{y\sim p_\theta^{RL}(y)} \hat{R}(y|\theta)\right]$$

where $\hat{R}(y) = R(y|x) - \beta \cdot \log \frac{p_\theta^{RL}(y)}{p^{PT}(y)}$

$$= \underline{\frac{\partial \mathbb{E}_{y\sim p_\theta^{RL}(y)} \hat{R}(y|\theta)}{\partial \theta}} + \mathbb{E}_{y\sim p_\theta^{RL}(y)} \frac{\partial \hat{R}(y|\theta)}{\partial \theta}$$

Def of partial derivative

Not easy to estimate

- Naïve way (ignoring θ in $\hat{R}$ as it is not under this term's $\frac{\partial}{\partial \theta}$ consideration)

$$\nabla_\theta \mathbb{E}_{y\sim p_\theta^{RL}(y)} \hat{R}(y) = \nabla_\theta \sum_y p_\theta^{RL}(y) \cdot \hat{R}(y)$$

$$= \sum_y \nabla_\theta p_\theta^{RL}(y) \cdot \hat{R}(y)$$

Difficult to compute unless enumerating all y's

Policy Gradient

- We need to calculate (ignoring x for now)

$$\nabla_{\theta}\left[\mathbb{E}_{y\sim p_{\theta}^{RL}(y)}\,\hat{R}(y|\theta)\right] \qquad \text{where } \hat{R}(y) = R(y|x) - \beta\cdot\log\frac{p_{\theta}^{RL}(y)}{p^{PT}(y)}$$

$$= \frac{\partial\mathbb{E}_{y\sim p_{\theta}^{RL}(y)}\,\hat{R}(y|\theta)}{\partial\theta} + \mathbb{E}_{y\sim p_{\theta}^{RL}(y)}\,\frac{\partial\hat{R}(y|\theta)}{\partial\theta}$$

Def of partial derivative

Idea: **log-derivative trick** (basically chain rule [Williams'92])

$$\nabla_{\theta}\log(p_{\theta}^{RL}(y)) = \frac{\nabla_{\theta}p_{\theta}^{RL}(y)}{p_{\theta}^{RL}(y)} \Rightarrow \nabla_{\theta}p_{\theta}^{RL}(y) = p_{\theta}^{RL}(y)\,\nabla_{\theta}\log(p_{\theta}^{RL}(y))$$

Then $\nabla_{\theta}\,\mathbb{E}_{y\sim p_{\theta}^{RL}(y)}\,\hat{R}(y) = \sum_{y}\nabla_{\theta}p_{\theta}^{RL}(y)\cdot\hat{R}(y)$

Policy Gradient

- We need to calculate (ignoring x for now)

$$\nabla_\theta \left[\mathbb{E}_{y \sim p_\theta^{RL}(y)} \hat{R}(y|\theta) \right]$$

where $\hat{R}(y) = R(y|x) - \beta \cdot \log \frac{p_\theta^{RL}(y)}{p^{PT}(y)}$

$$= \frac{\partial \mathbb{E}_{y \sim p_\theta^{RL}(y)} \hat{R}(y|\theta)}{\partial \theta} + \mathbb{E}_{y \sim p_\theta^{RL}(y)} \frac{\partial \hat{R}(y|\theta)}{\partial \theta}$$

Def of partial derivative

Idea: **log-derivative trick** (basically chain rule [Williams'92])

$$\nabla_\theta \log(p_\theta^{RL}(y)) = \frac{\nabla_\theta p_\theta^{RL}(y)}{p_\theta^{RL}(y)} \Rightarrow \nabla_\theta p_\theta^{RL}(y) = p_\theta^{RL}(y)\, \nabla_\theta \log(p_\theta^{RL}(y))$$

Hence $\nabla_\theta\, \mathbb{E}_{y \sim p_\theta^{RL}(y)} \hat{R}(y) = \sum_y \nabla_\theta p_\theta^{RL}(y) \cdot \hat{R}(y)$

$$= \sum_y p_\theta^{RL}(y)\, \nabla_\theta \log(p_\theta^{RL}(y)) \cdot \hat{R}(y)$$

Policy Gradient

- We need to calculate (ignoring x for now)

$$\nabla_\theta \left[\mathbb{E}_{y\sim p_\theta^{RL}(y)}\, \hat{R}(y|\theta)\right]$$

where $\hat{R}(y) = R(y|x) - \beta \cdot \log \frac{p_\theta^{RL}(y)}{p^{PT}(y)}$

$$= \frac{\partial \mathbb{E}_{y\sim p_\theta^{RL}(y)}\, \hat{R}(y|\theta)}{\partial \theta} + \mathbb{E}_{y\sim p_\theta^{RL}(y)} \frac{\partial \hat{R}(y|\theta)}{\partial \theta}$$

Def of partial derivative

Idea: **log-derivative trick** (basically chain rule [Williams'92])

$$\nabla_\theta \log(p_\theta^{RL}(y)) = \frac{\nabla_\theta p_\theta^{RL}(y)}{p_\theta^{RL}(y)} \Rightarrow \nabla_\theta p_\theta^{RL}(y) = p_\theta^{RL}(y)\, \nabla_\theta \log(p_\theta^{RL}(y))$$

Hence $\nabla_\theta\, \mathbb{E}_{y\sim p_\theta^{RL}(y)}\, \hat{R}(y) = \sum_y \nabla_\theta p_\theta^{RL}(y) \cdot \hat{R}(y)$

$$= \sum_y p_\theta^{RL}(y)\, \nabla_\theta \log(p_\theta^{RL}(y)) \cdot \hat{R}(y)$$

$$= \mathbb{E}_{y\sim p_\theta^{RL}(y)} \left[\nabla_\theta \log(p_\theta^{RL}(y)) \cdot \hat{R}(y)\right]$$

And we know expectations can be, again, estimated from samples

Policy Gradient

- We need to calculate (ignoring x for now)

$$\nabla_\theta \left[\mathbb{E}_{y\sim p_\theta^{RL}(y)}\, \hat{R}(y|\theta)\right]$$

where $\hat{R}(y) = R(y|x) - \beta \cdot \log \frac{p_\theta^{RL}(y)}{p^{PT}(y)}$

$$= \frac{\partial \mathbb{E}_{y\sim p_\theta^{RL}(y)}\, \hat{R}(y|\theta)}{\partial\theta} + \mathbb{E}_{y\sim p_\theta^{RL}(y)} \frac{\partial \hat{R}(y|\theta)}{\partial\theta}$$

Def of partial derivative

Idea: **log-derivative trick** (basically chain rule [Williams'92])

- This illustrates basic principles
- Practical implementation usually uses a fancier variant called PPO, and requires very careful engineering

$$= \sum_y p_\theta^{RL}(y)\, \nabla_\theta \log(p_\theta^{RL}(y)) \cdot R(y)$$

$$= \mathbb{E}_{y\sim p_\theta^{RL}(y)} \left[\nabla_\theta \log(p_\theta^{RL}(y)) \cdot \hat{R}(y)\right]$$

And we know expectations can be, again, estimated from samples

Part 2: Learning Rewards over Languages

- **Objective**: learn a reward model $RM(y|x)$ from human data that assigns a reward to each response y
- **Challenges?**

Let's say we want to evaluate summary of a news

A winter storm hit Chicago. There was heavy wind and snow, but no damage is caused	Chicago has strong facilities and is resilient to snow storms	A large storm hit Chicago, resulting in massive snow and freezing weather
$R(y_1) = 3$	$R(y_2) = 2.4$	$R(y_3) =?$

Then we do supervised learning!

Part 2: Learning Rewards over Languages

- **Objective**: learn a reward model $RM(y|x)$ from human data that assigns a reward to each response y
- **Challenge**: eliciting direct reward value is very noisy
- **One idea**: elicit comparison/ordinal feedback

A winter storm hit Chicago. There was heavy wind and snow, but no damage is caused	Chicago has strong facilities and is resilient to snow storms	A large storm hit Chicago, resulting in massive snow and freezing weather
$R(y_1) = 3$	$R(y_2) = 2.4$	$R(y_3) = ?$

Part 2: Learning Rewards over Languages

- **Objective**: learn a reward model $RM(y|x)$ from human data that assigns a reward to each response y
- **Challenge**: eliciting direct reward value is very noisy
- **One idea**: elicit comparison/ordinal feedback

A winter storm hit Chicago. There was heavy wind and snow, but no damage is caused	Chicago has strong facilities and is resilient to snow storms	A large storm hit Chicago, resulting in massive snow and freezing weather

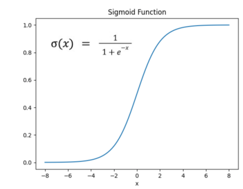
$$y_1 \succ y_2 \succ y_3$$

Instead of eliciting reward value, you ask which one is better (i.e., wins)?

- Why? Preferences are less noisy, yet still descriptive about underlying reward
- Widely studied in behavioral economics, known as *revealed preference*
- In statistics, this is the idea of logistic regression

Part 2: Learning Rewards over Languages

- From comparison to rewards: the Bradley-Terry [1952] model



$$Loss = -\mathbb{E}_{(y^{win}, y^{lose}, x)\sim D} \log\left\{\sigma\left[RM\left(y^{win}\right) - RM\left(y^{lose}\right)\right]\right\}$$

This has familiar flavor to logistic regression, though different

A winter storm hit Chicago. There was heavy wind and snow, but no damage is caused	Chicago has strong facilities and is resilient to snow storms	A large storm hit Chicago, resulting in massive snow and freezing weather

$$y_1 \succ y_2 \succ y_3$$

Part 2: Learning Rewards over Languages

➢ From comparison to rewards: the Bradley-Terry [1952] model

$$Loss(\omega) = -\mathbb{E}_{(y^{win}, y^{lose}, x)\sim D} \log\left\{\sigma\left[RM_\omega(y^{win}) - RM_\omega(y^{lose})\right]\right\}$$

In practice, RM is a NN with parameter ω

You find ω by minimizing above loss

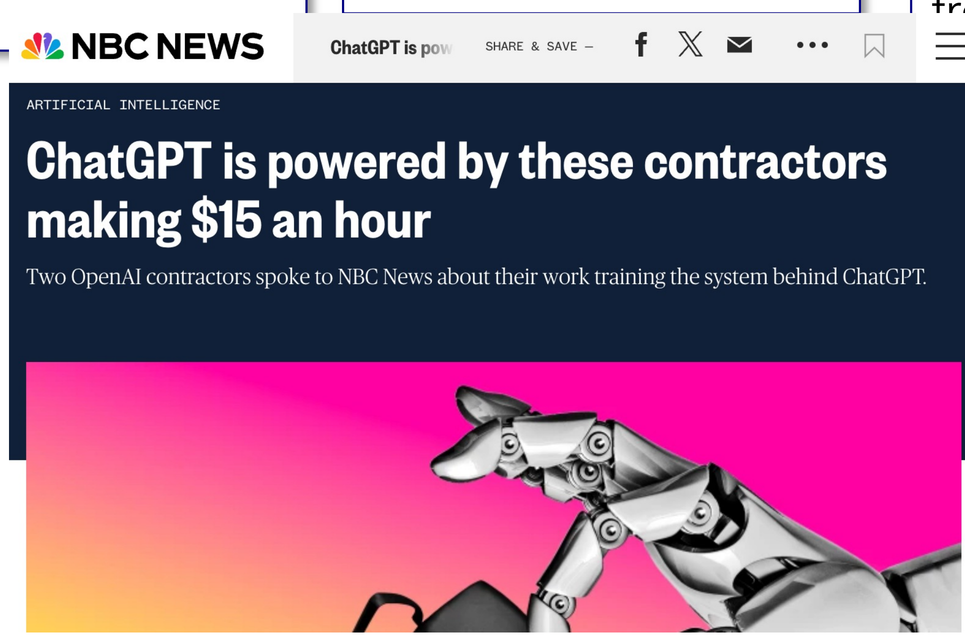
snow, but no damage is caused

to snow storms

massive snow and freezing weather

y_1

y_3



RLHF: Putting it Together

Step 1: instruction fine-turning (IFT)

- Supervised learning, like pre-training, but with more task-specific data
- No rewards or RL involved

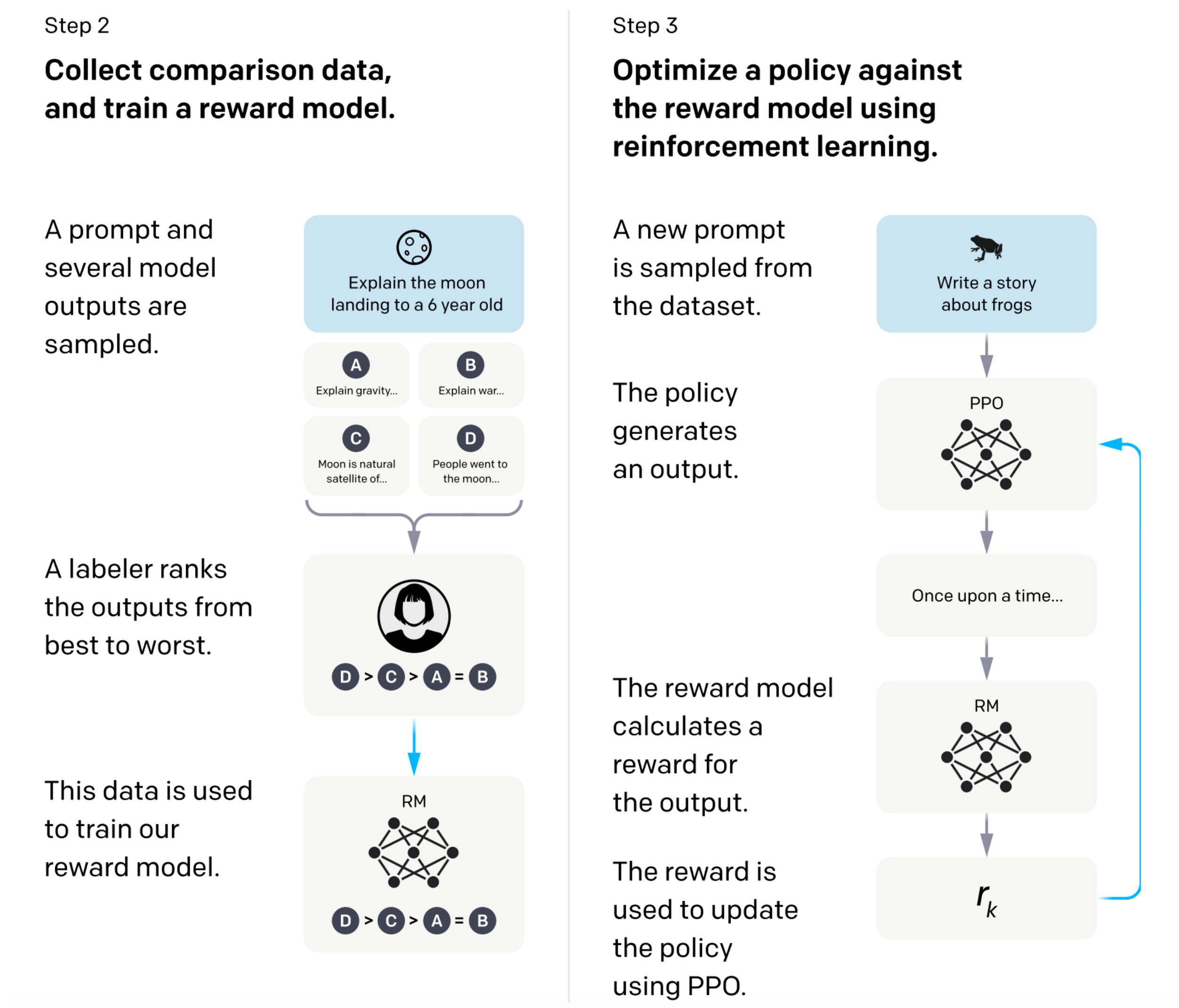


Figure cited from [Ouyang et al., 2020]

RLHF: Putting it Together

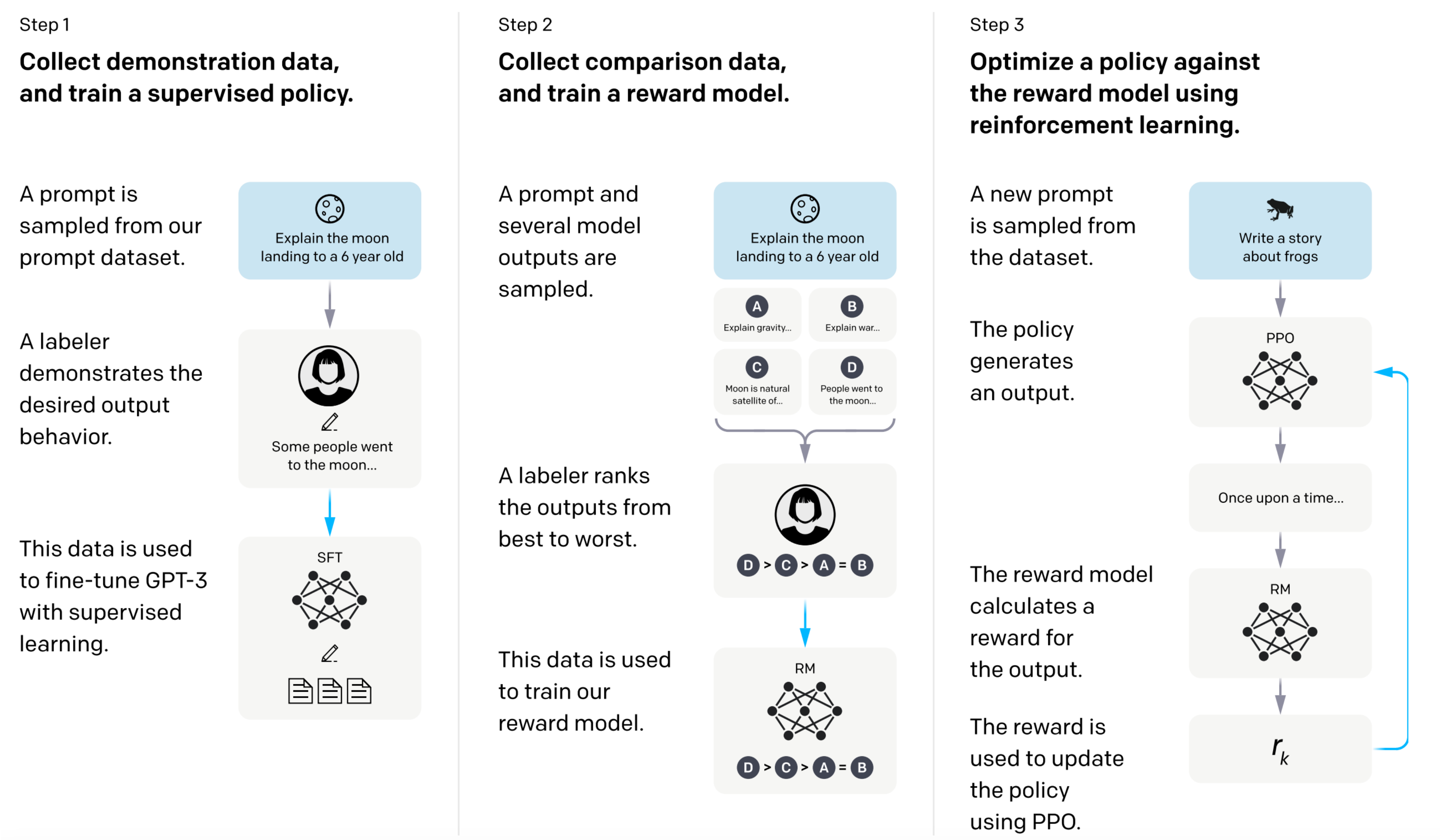
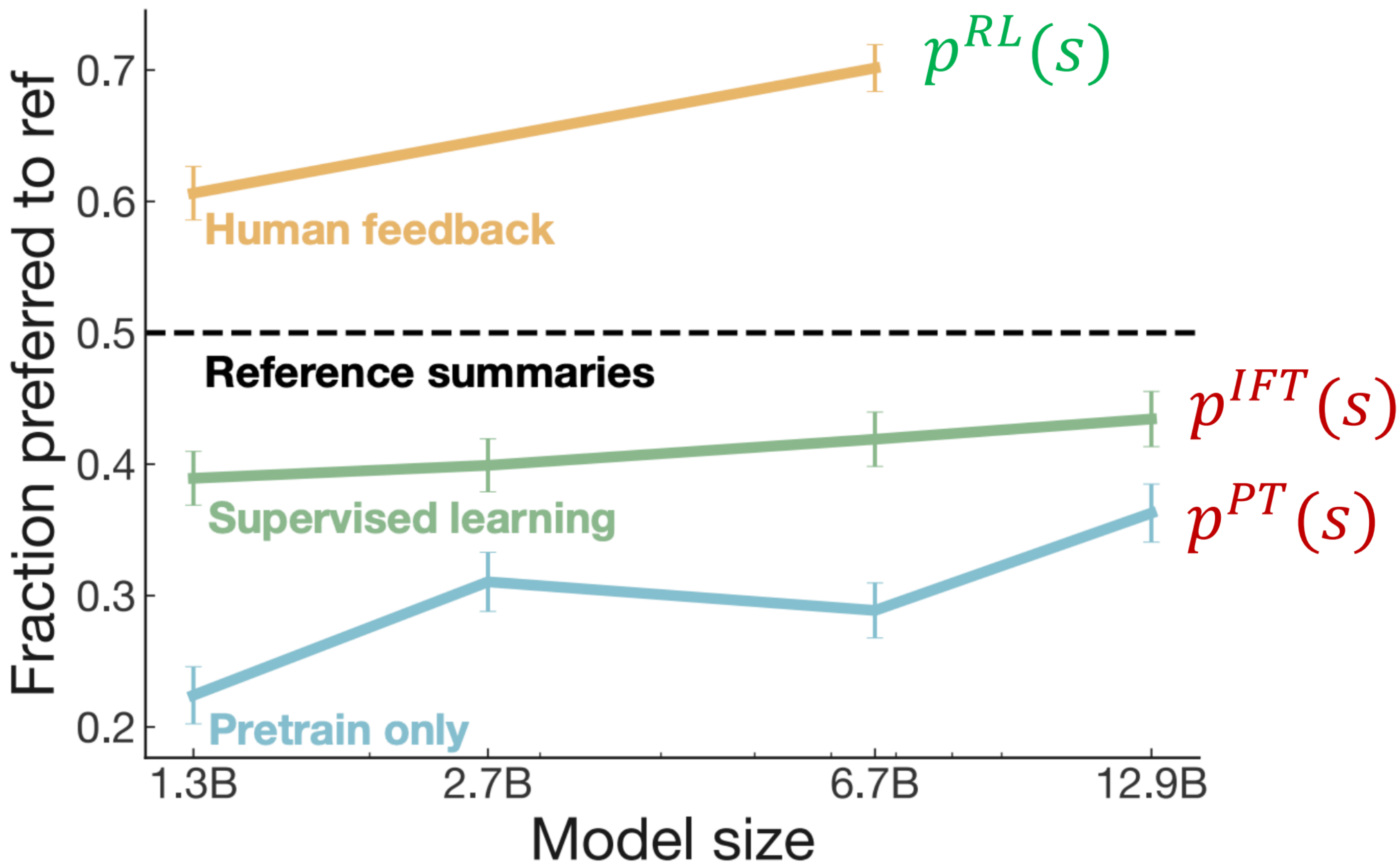


Figure cited from [Ouyang et al., 2020]

RLHF: Putting it Together



[Stiennon et al., 2020]

RLHF: Putting it Together, and It Does Work!

PROMPT *Write a short poem about a wise frog.*

COMPLETION

GPT-3

```
Write a short story in which a character has two different names.

Write a short story in which you try to get something back that you
have lost.

Write a short story in which a character has a bad dream.
```

InstructGPT

```
The frog is a symbol of wisdom
He knows all the secrets of the world
He is a master of disguise
And a great teacher of life
He is a symbol of transformation
And the bringer of change
He is the frog who has seen it all
And knows the meaning of it all
```

RLHF: Putting it Together, and It Does Work!

- As we know, ChatGPT is even more amazing
- We do not know exactly how OpenAI developed it, but they have two paras…

Introducing ChatGPT

Try ChatGPT ↗ Download ChatGPT desktop › Learn about ChatGPT ›

We've trained a model called ChatGPT which interacts in a conversational way. The dialogue format makes it possible for ChatGPT to answer followup questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests.

ChatGPT is a sibling model to InstructGPT, which is trained to follow an instruction in a prompt and provide a detailed response.

Methods

We trained this model using Reinforcement Learning from Human Feedback (RLHF), using the same methods as InstructGPT, but with slight differences in the data collection setup. We trained an initial model using supervised fine-tuning: human AI trainers provided conversations in which they played both sides—the user and an AI assistant. We gave the trainers access to model-written suggestions to help them compose their responses. We mixed this new dialogue dataset with the InstructGPT dataset, which we transformed into a dialogue format.

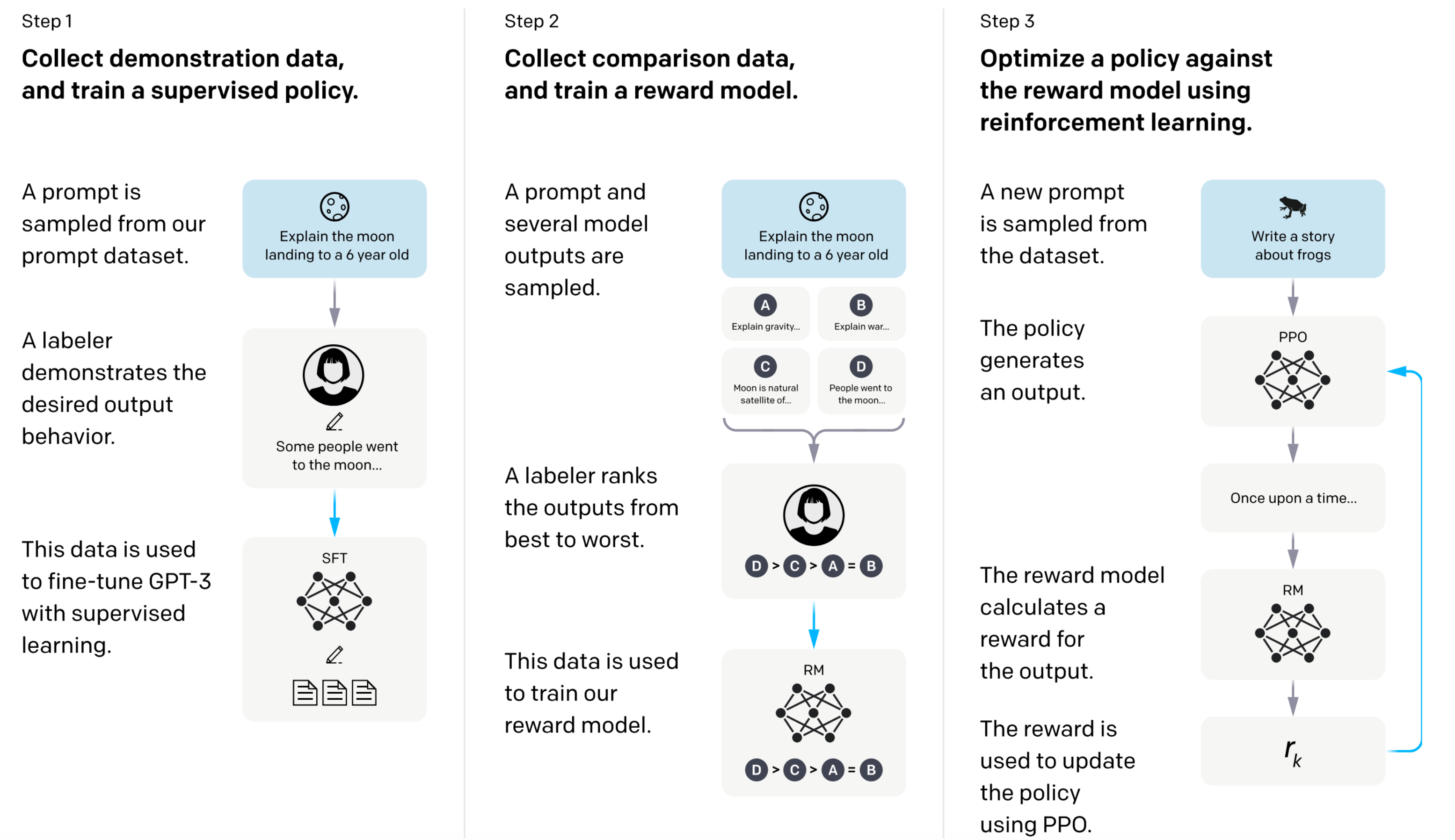
To create a reward model for reinforcement learning, we needed to collect comparison data, which consisted of two or more model responses ranked by quality. To collect this data, we took conversations that AI trainers had with the chatbot. We randomly selected a

Outline

- What and Why?
- Procedures of RLHF
- RL without Rewards: Direct Preference Optimization (DPO) [Rafailov et al. 2022]

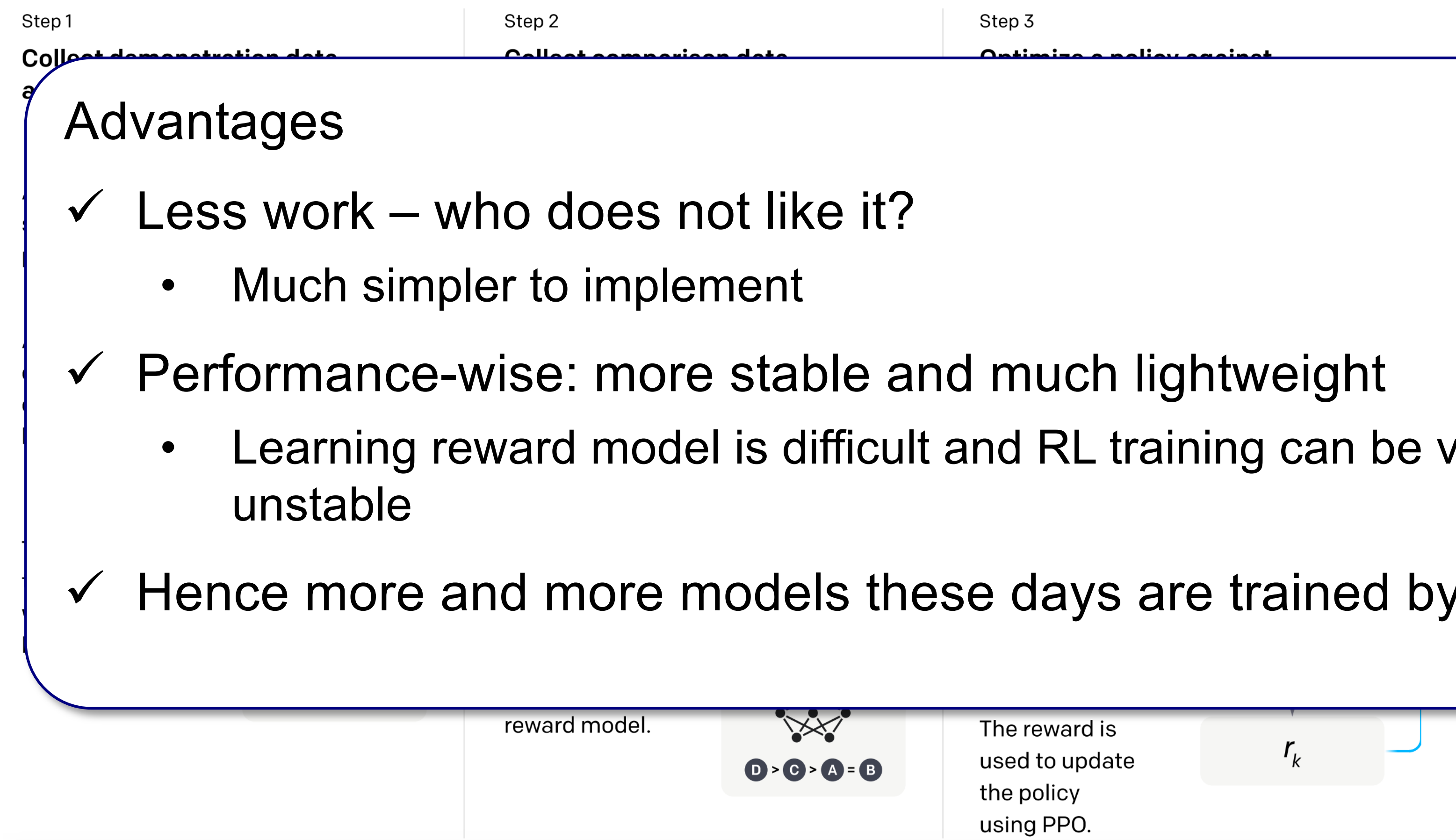
What Does DPO Do?

Merging these into a single step – directly learn from comparison preference



What Does DPO Do?

Merging these into a single step – directly learn from comparison preference



Advantages

✓ Less work – who does not like it?

- Much simpler to implement

✓ Performance-wise: more stable and much lightweight

- Learning reward model is difficult and RL training can be very unstable

✓ Hence more and more models these days are trained by DPO

Core idea is a re-formulation of the RL objective, which turns out to only have comparison preferences, but no rewards!

- Recall RL objective

$$\max_{p^{RL}} \{ \ \mathbb{E}_{y \sim p^{RL}(y|x)} [R(y|x)] - \beta \cdot KL(p^{RL}(y|x), p^{PT}(y|x)) \ \}$$

This optimization problem turns out to have a closed-form optimal solution (due to nice properties of KL)

$$p^{RL}(y|x) = \frac{1}{Z(x)} p^{PT}(y|x) \exp\left(\frac{1}{\beta} R(y|x)\right)$$

Rearrange to get R as a function of RL policy

$$\Rightarrow \ R(y|x) = \beta \log \frac{p^{RL}(y|x)}{p^{PT}(y|x)} + \beta \log Z(x)$$

Core idea is a re-formulation of the RL objective, which turns out to only have comparison preferences, but no rewards!

- Recall RL objective

$$\max_{p^{RL}} \{ \mathbb{E}_{y \sim p^{RL}(y|x)} [R(y|x)] - \beta \cdot KL(p^{RL}(y|x), p^{PT}(y|x)) \}$$

This optimization problem turns out to have a closed-form optimal solution (due to nice properties of KL)

$$p^{RL}(y|x) = \frac{1}{Z(x)} p^{PT}(y|x) \exp\left(\frac{1}{\beta} R(y|x)\right)$$

Rearrange to get R as a function of RL policy

$$\Rightarrow \quad R(y|x) = \beta \log \frac{p^{RL}(y|x)}{p^{PT}(y|x)} + \beta \log Z(x)$$

Recall BT model $\quad \Pr(y_1 \succ y_2) = \sigma(R(y_1|x) - R(y_2|x))$

$$\Rightarrow \Pr(y_1 \succ y_2) = \sigma(\beta \log \frac{p^{RL}(y_1|x)}{p^{PT}(y_1|x)} - \beta \log \frac{p^{RL}(y_2|x)}{p^{PT}(y_2|x)})$$

Core idea is a re-formulation of the RL objective, which turns out to only have comparison preferences, but no rewards!

- DPO simply maximizes the log-likelihood of winning over comparison data (like the objective for learning reward model)

$$Loss_{DPO}\left(p_\theta^{RL}\right) = -\mathbb{E}_{(y_1 \succ y_2, x)\sim D} \log \sigma\left(\beta \log \frac{p_\theta^{RL}(y_1|x)}{p^{PT}(y_1|x)} - \beta \log \frac{p_\theta^{RL}(y_2|x)}{p^{PT}(y_2|x)}\right)$$

That is, through closed-form solution of opt policy, we removed rewards in objective, and get an RL objective directly as a function of policy p_θ^{RL}

Recall BT model $\Pr(y_1 \succ y_2) = \sigma(R(y_1|x) - R(y_2|x))$

$$\Rightarrow \Pr(y_1 \succ y_2) = \sigma(\beta \log \frac{p^{RL}(y_1|x)}{p^{PT}(y_1|x)} - \beta \log \frac{p^{RL}(y_2|x)}{p^{PT}(y_2|x)})$$

Thank You

Haifeng Xu
University of Chicago
haifengxu@uchicago.edu