香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

COMP 4901B

Large Language Models

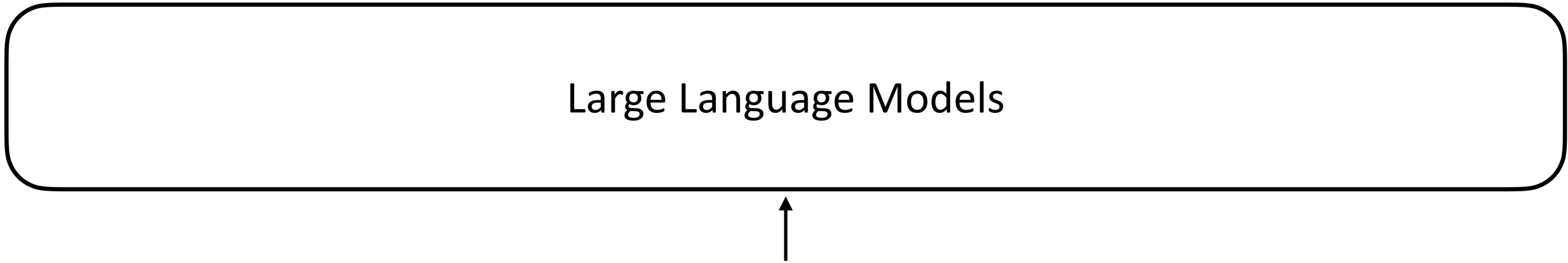
Reinforcement Learning Basics

Junxian He

Oct 15, 2025

Review: Multi-Turn Instruction Tuning

<User> How are you today?\n<Assistant> I'm doing well, thank you! How can I help you? \n\n<User> Can you tell me a joke? \n<Assistant> Sure! Why did the math book look sad? <stop>



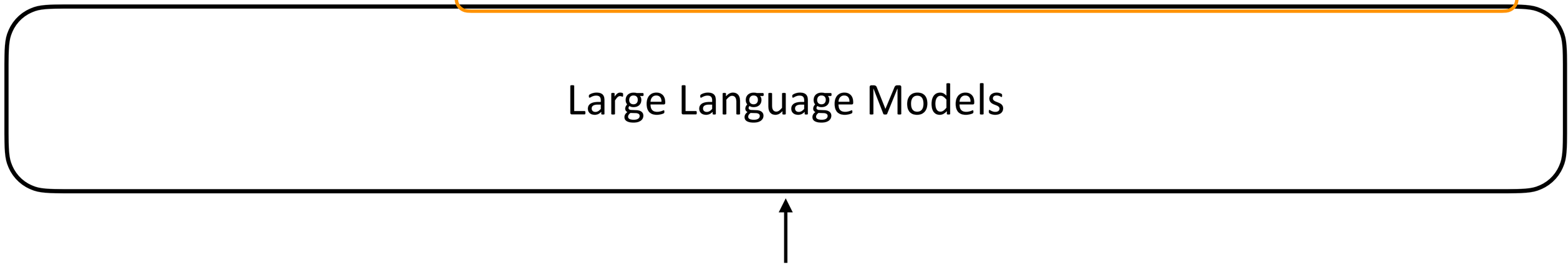
<start> <User> How are you today?\n<Assistant> I'm doing well, thank you! How can I help you? \n\n<User> Can you tell me a joke? \n<Assistant> Sure! Why did the math book look sad?

No different from before, still shift one token left to obtain output

Review: Inference time for ChatBot

I'm doing well, thank you! How can I help you?<stop>

Sure! Why did the math book look sad? <stop>

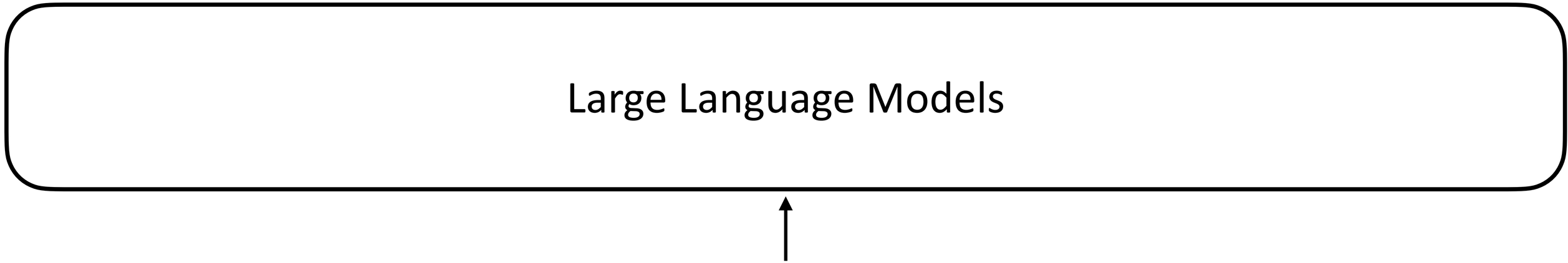


<start> <User> How are you today?<user_stop><Assistant> I'm doing well, thank you! How can I help you?<stop> <User> Can you tell me a joke?<user_stop><Assistant>

At inference time, we only ask the model to predict assistant parts

So typically, other parts are all masked when predicting the loss

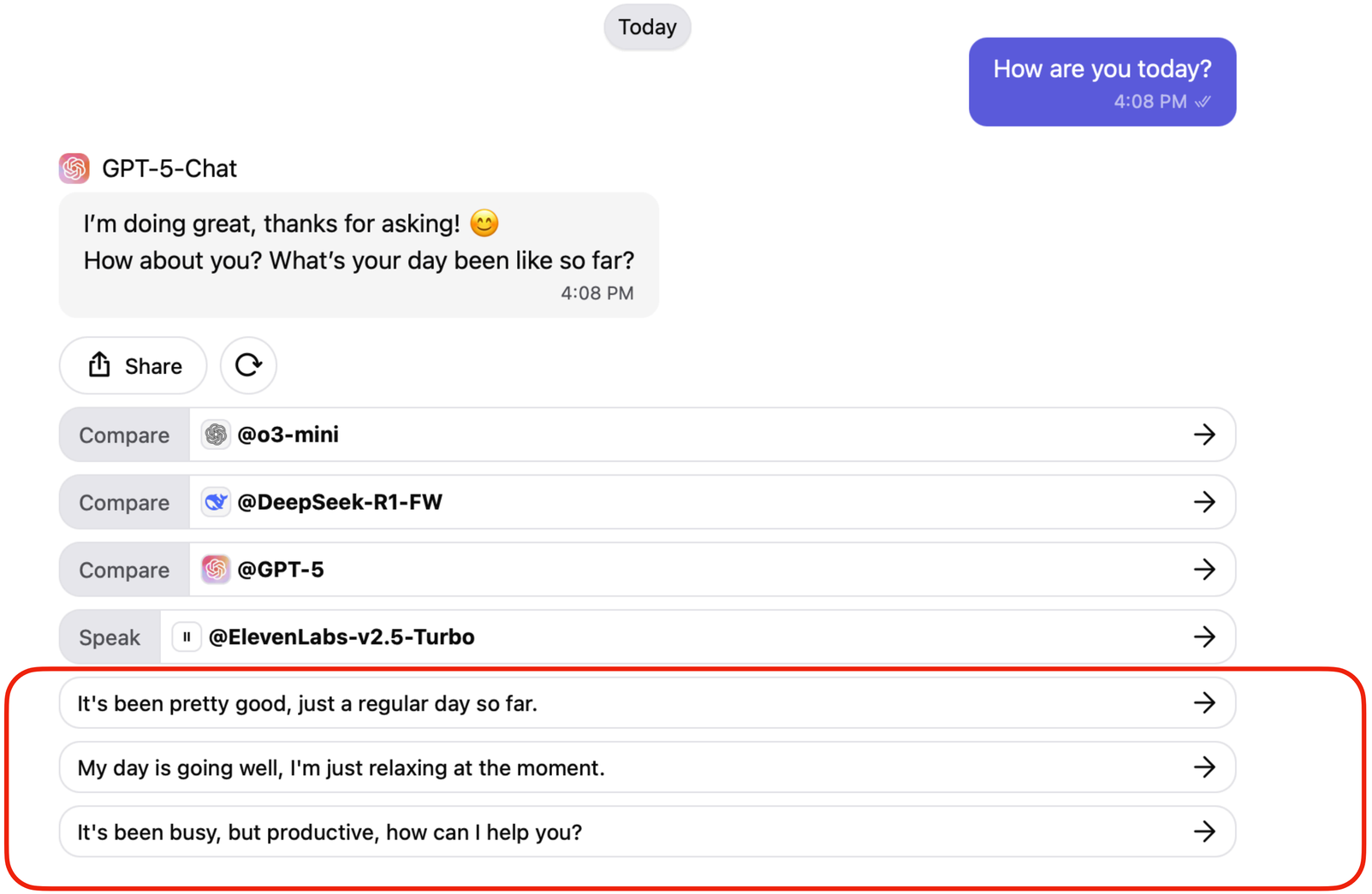
Review: Inference time for ChatBot



<start> <User> How are you today?<user_stop><Assistant> I'm doing well, thank you! How can I help you?<stop> <User> Can you tell me a joke?
<user_stop><Assistant>

Or, we only mask <user> <assistant> tags, not learning all contents, then the chatbot can suggest questions each round

Review: Chatbot can ask questions if not masking



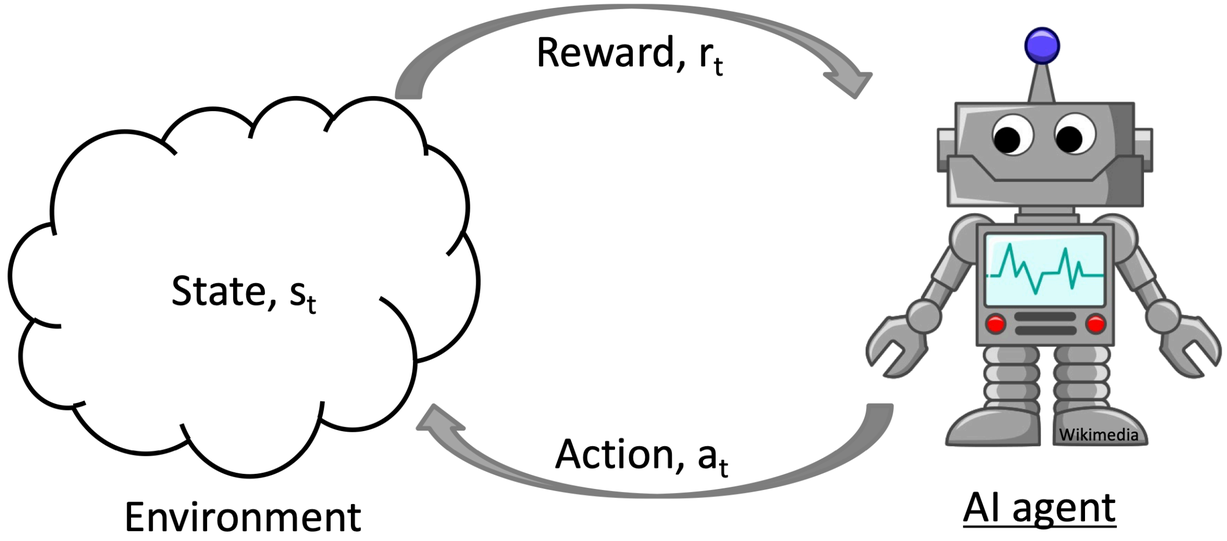
Reinforcement Learning

Learning Tasks

- Supervised learning - $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$
  - Regression - $y^{(i)} \in \mathbb{R}$
  - Classification - $y^{(i)} \in \{1, \ldots, C\}$
- Unsupervised learning - $\mathcal{D} = \left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{N}$
  - Clustering
  - Dimensionality reduction
- Reinforcement learning - $\mathcal{D} = \left\{\boldsymbol{s}^{(t)}, \boldsymbol{a}^{(t)}, r^{(t)}\right\}_{t=1}^{T}$

RL Setup

In many cases, we cannot precisely define what the correct output is (think of we want to train a robot to walk)



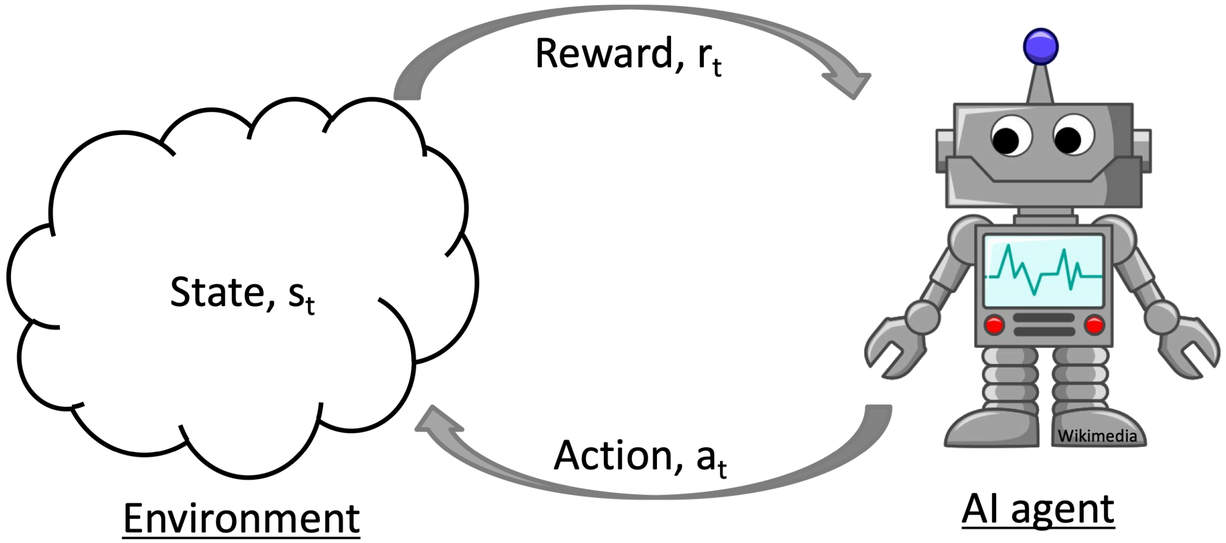
Agent chooses **actions** which can depend on past

Environment can change **state** with each action

Reward (Output) depends on (Inputs) action and state of environment

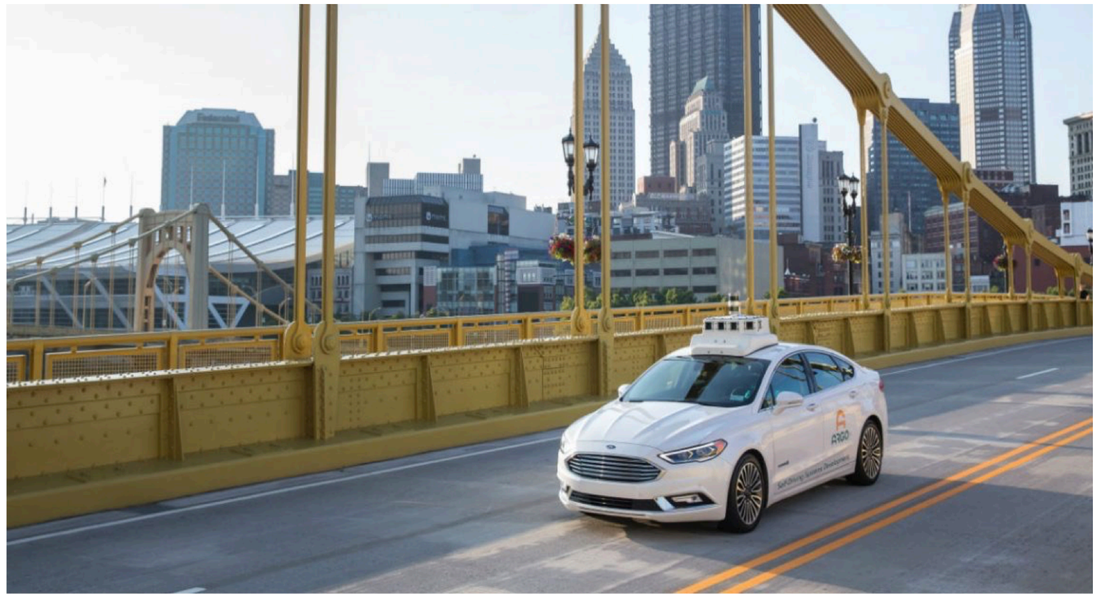
Goal: maximize the total reward

Differences from Supervised Learning



- Maximize reward (rather than learn reward) Supervised training is like imitation
- Inputs are not iid – state & action depends on past

RL Examples



RL Setup

- State space, $\mathcal{S}$
- Action space, $\mathcal{A}$
- Reward function
  - Stochastic, $p(r \mid s, a)$
  - Deterministic, $R\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- Transition function
  - Stochastic, $p(s' \mid s, a)$
  - Deterministic, $\delta\colon \mathcal{S} \times \mathcal{A} \to \mathcal{S}$

In this lecture, we assume they are known

- Reward and transition functions can be known or unknown

RL Setup

- Policy, $\pi : \mathcal{S} \rightarrow \mathcal{A}$
  - Specifies an action to take in *every* state

- Value function, $V^{\pi}: \mathcal{S} \rightarrow \mathbb{R}$
  - Measures the expected total reward of starting in some state s and executing policy π, i.e., in every state, taking the action that π returns

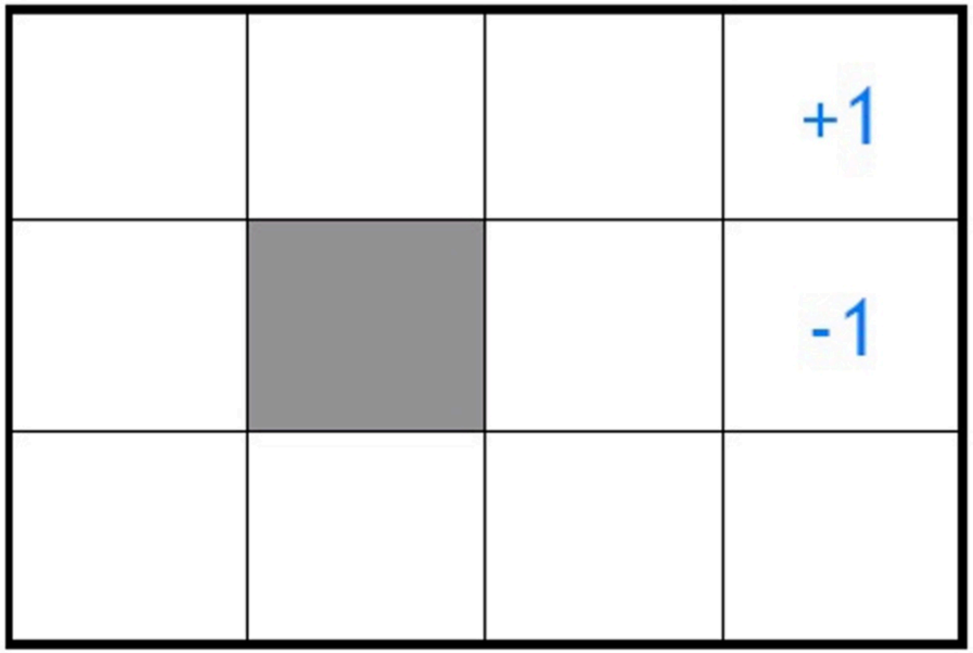
RL Example - gridworld

$\mathcal{S}$ = all empty squares in the grid

$\mathcal{A}$ = {up, down, left, right}

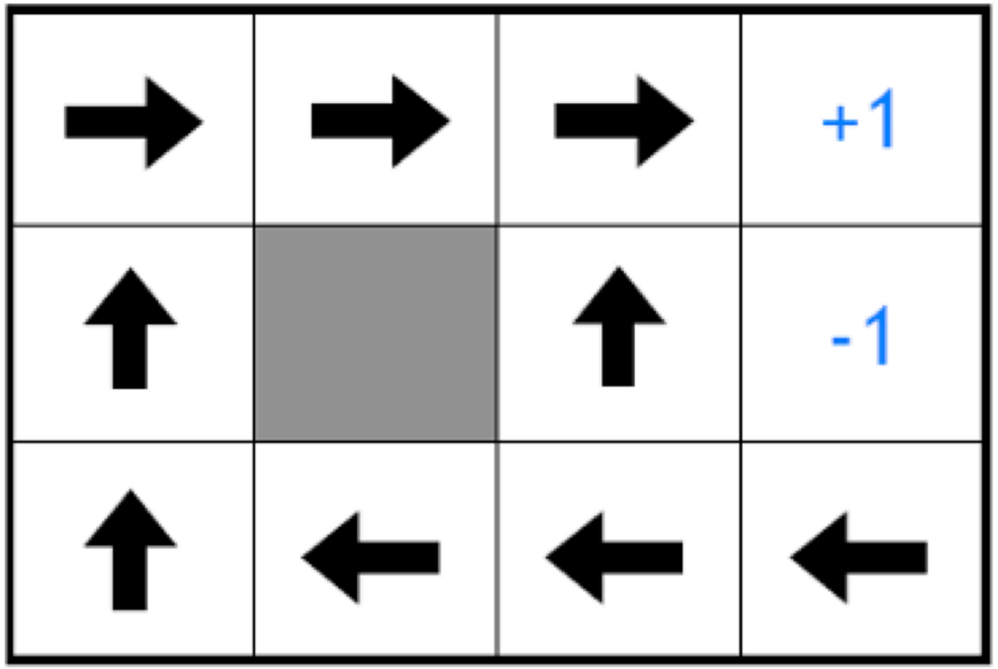
Deterministic transitions

Rewards of +1 and -1 for entering the labelled squares



Terminate after receiving either reward

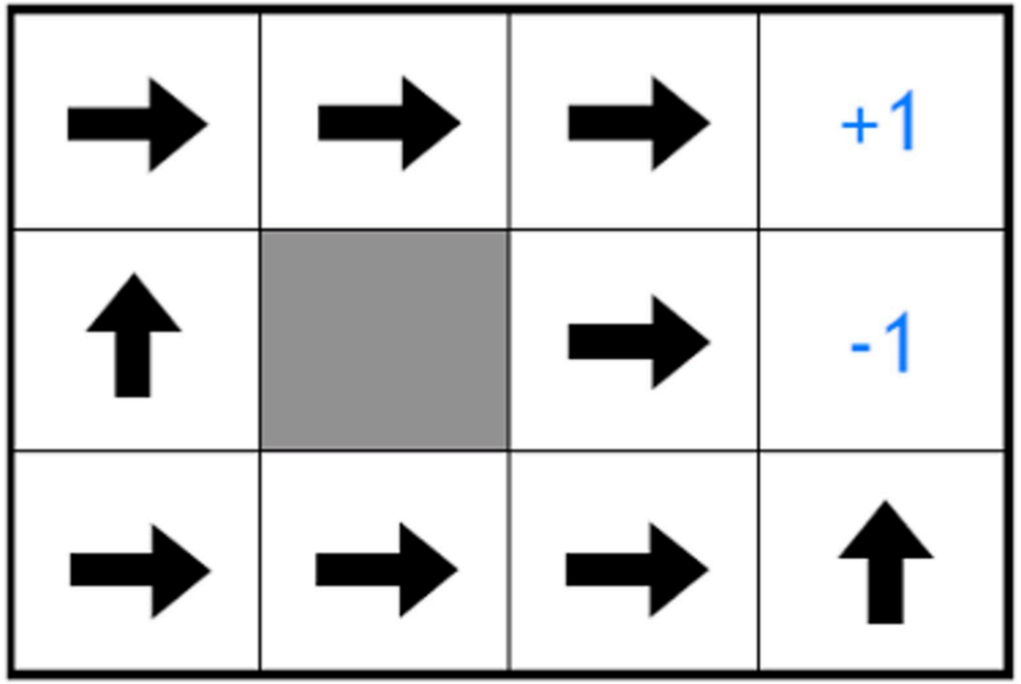
RL Example - gridworld



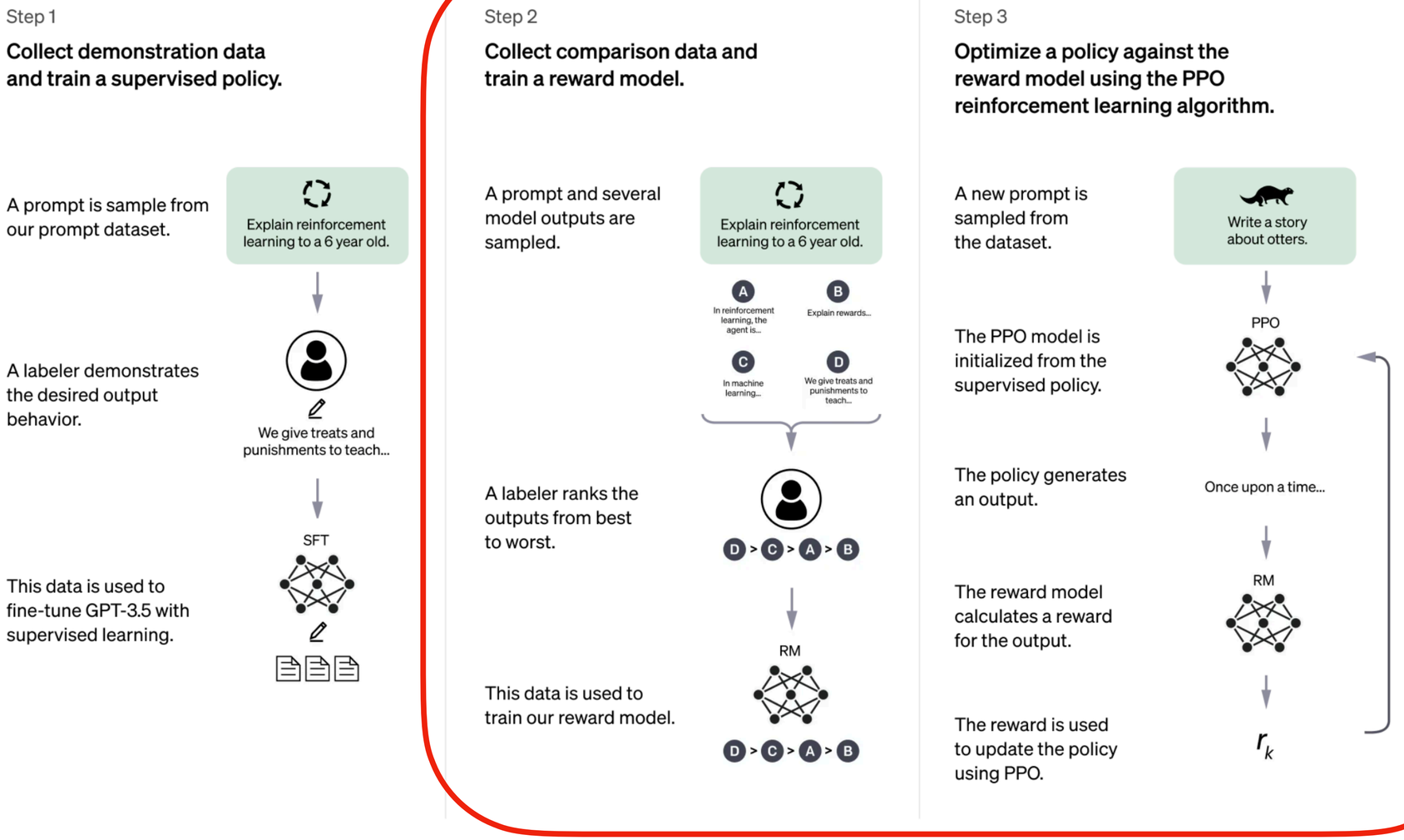
Is this policy optimal?

RL Example - gridworld

Optimal policy given a reward of -2 per step



Reinforcement Learning in LLMs



Reinforcement Learning in LLMs

1. There is no external environment
2. Each predicted token is an action, and the context is the environment states

From Imitation to Optimization

Imitation (SFT)

Fit $\hat{p}(y|x) \approx p^*(y|x)$ for some reference distribution $p^*(y|x)$

- Pure generative modeling perspective
- Requires samples from reference policy

Optimization (RLHF)

Find $\hat{\boldsymbol{p}}(\boldsymbol{y}|\boldsymbol{x})$ such that $\max_p E_p[R(y,x)]$ for a reward $R(y,x)$

- Maximize some reward function that we can measure
- LMs are policies, not a model of some distribution

Reward Optimization in Language Models

Objective:

$$\theta = \arg\max_{\theta} \mathbb{E}_{x \sim p_\theta(x)} R(x)$$

X is the language sequence, R is the reward function, scores the generated response (we can consider R as a human or a model)

How to do gradients over this objective?

Gradient estimation (policy gradient):

$$\hat{g} = \mathbb{E}_{x \sim p_\theta(x)} R(x) \nabla_\theta \log p_\theta(x)$$

$\hat{g}$ is the gradient (not objective)

REINFORCE / Policy Gradient

$$\hat{g} = \mathbb{E}_{x \sim p_\theta(x)} R(x) \nabla_\theta \log p_\theta(x)$$

How to implement?

$$\text{Objective} = \sum_{i=1}^{n} \frac{1}{n} R(x^{(i)}) \log p_\theta(x^{(i)}) \qquad x^{(1)}, \ldots, x^{(n)} \sim p_\theta(x)$$

This objective looks kinda like weighted log likelihood maximization?

What is different?

1. Have a weight of R(x)
2. The data x is sampled from the model itself, not from a static dataset

REINFORCE / Policy Gradient

$$\text{Objective} = \sum_{i=1}^{n} \frac{1}{n} R(x^{(i)}) \log p_\theta(x^{(i)}) \qquad x^{(1)}, \ldots, x^{(n)} \sim p_\theta(x)$$

1. Have a weight of R(x)
2. The data x is sampled from the model itself, not from a static dataset

This equation is not that complex, just view it as a weighted likelihood maximization

This is the simplest form of RL, many other RL algorithms (PPO, GRPO) are more like variants of this simple equation with the same spirit

We can see why RL is called “self-improving”, and it is trained by “synthetic data”

REINFORCE / Policy Gradient

$$\text{Objective} = \sum_{i=1}^{n} \frac{1}{n} R(x^{(i)}) \log p_\theta(x^{(i)}) \qquad x^{(1)}, \ldots, x^{(n)} \sim p_\theta(x)$$

1. Sample data from the model (or we call policy) itself (exploration)
2. A reward function judges whether the explored data is good or bad

REINFORCE / Policy Gradient

Objective = $\sum_{i=1}^{n} \frac{1}{n} R(x^{(i)}) \log p_\theta(x^{(i)})$ $\quad x^{(1)}, \ldots, x^{(n)} \sim p_\theta(x)$

Reinforcement learning is a mixed art of both training and inference during training time

Why?

REINFORCE / Policy Gradient

Objective = $\sum_{i=1}^{n} \frac{1}{n} R(x^{(i)}) \log p_\theta(x^{(i)})$ $\quad x^{(1)}, ..., x^{(n)} \sim p_\theta(x)$

Training Inference

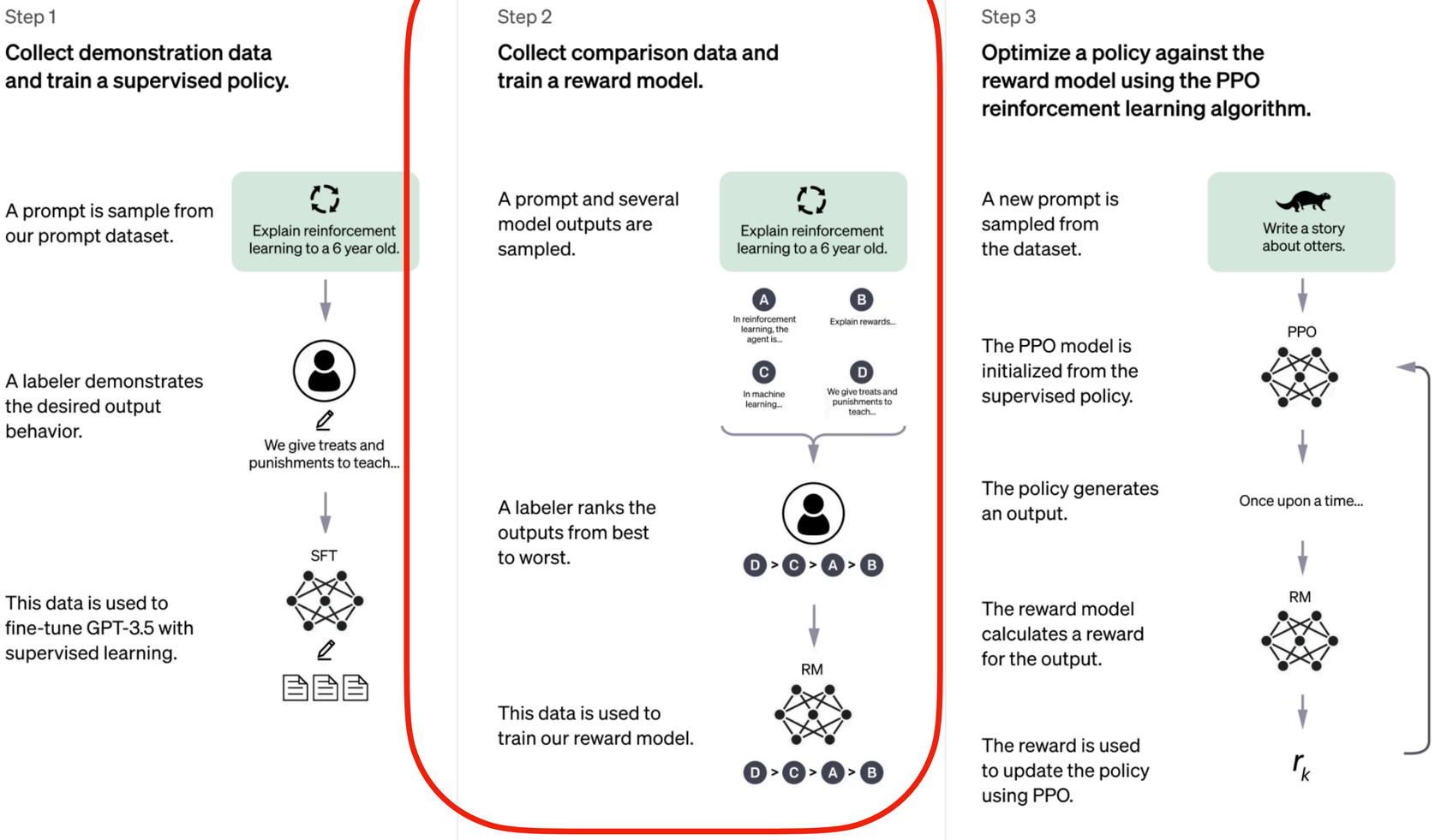
Each training step, the algorithm needs to run inference again

Is this efficient?

REINFORCE / Policy Gradient for Language Models

Objective = $$\sum_{i=1}^{n} \frac{1}{n} R(\{x_1^{(i)}, x_2^{(i)}, x_t^{(i)}\} \sum_{j=1}^{t} \log p_\theta(x_j^{(i)} | x_{<j}^{(i)})$$

Training a Reward Model in RLHF



Training a Reward Model

Suppose we have K responses and have them ranked by humans, then for all possible pairs of responses, y_w is the preferred one, y_l is the less preferred one, the objective of reward model $r_\theta(x, y)$ is:

$$\text{Objective} = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D} \left[\log\left(\sigma\left(r_\theta\left(x, y_w\right) - r_\theta\left(x, y_l\right)\right)\right)\right]$$

In practice, we can just sample a subset of pairs, rather than enumerating all pairs

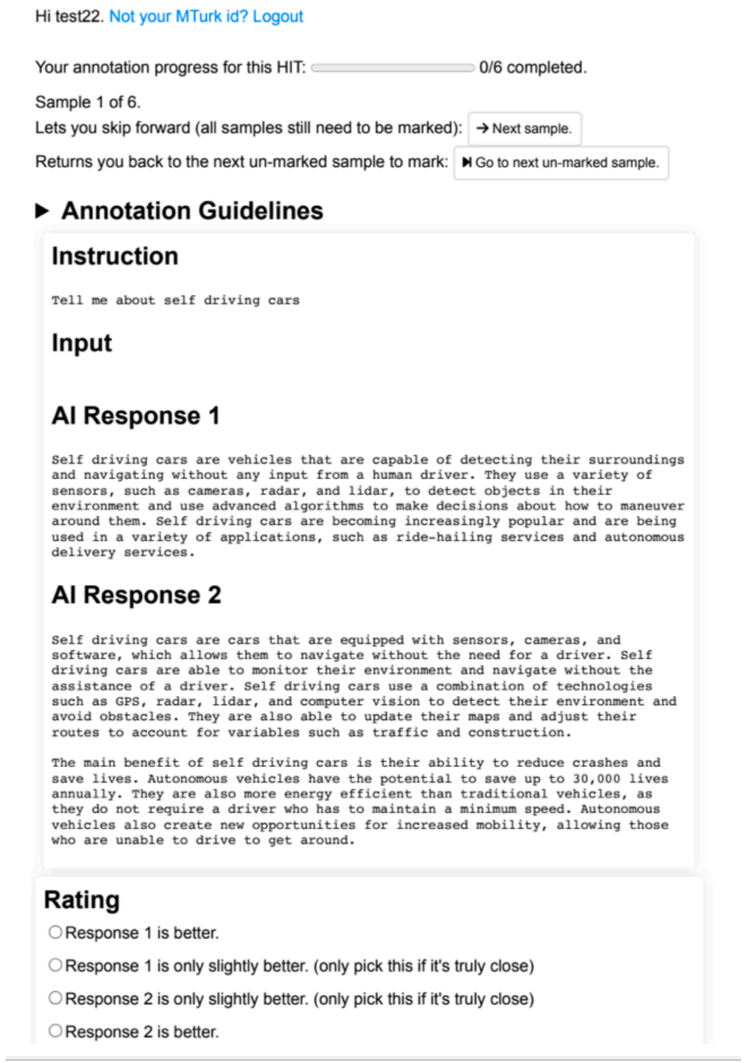
Reinforcement Learning Relaxes Human Annotations

	Base model →	Supervised learning (one time) →	Pairwise feedback →	RL →	Evaluation
Compute cost	$300k	$100	$100	$100	$0
Annotation cost	$0	$25k	$4k	$0	$50

1. SFT data can be expensive
2. There may be tasks that are much easier for humans to verify than solve, particularly when AI today is outperforming humans

Reinforcement Learning from Human Feedback Data

Basic Setup



Easier than annotating the responses directly

Thank You!