CS639 Deep Learning for NLP

Learning from Human Feedback I

Aligning LLMs with Human Intent and Values

Junjie Hu

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

https://junjiehu.github.io/cs639-spring/

Outline

- Introduction to reinforcement learning with human feedback (RLHF)

- Policy Gradient Fundamentals: **REINFORCE**

The Great Challenge of LLMs

- **Challenge:** Pre-trained LLMs are great **next-token predictors**, but not good **helpful assistant agents**.
- **Behavioral Gap:** Models often generate text that is:
  - Factually incorrect (hallucination).
  - Toxic/Harmful or biased.
  - **Off-topic** or verbose.
  - **Not aligned** with user intent/values.
- **Goal of Preference Learning:** Bridge the gap between *what the model predicts* and *what the human desires*.

Supervised Fine-Tuning (SFT)

- We are given the correct response to a user query

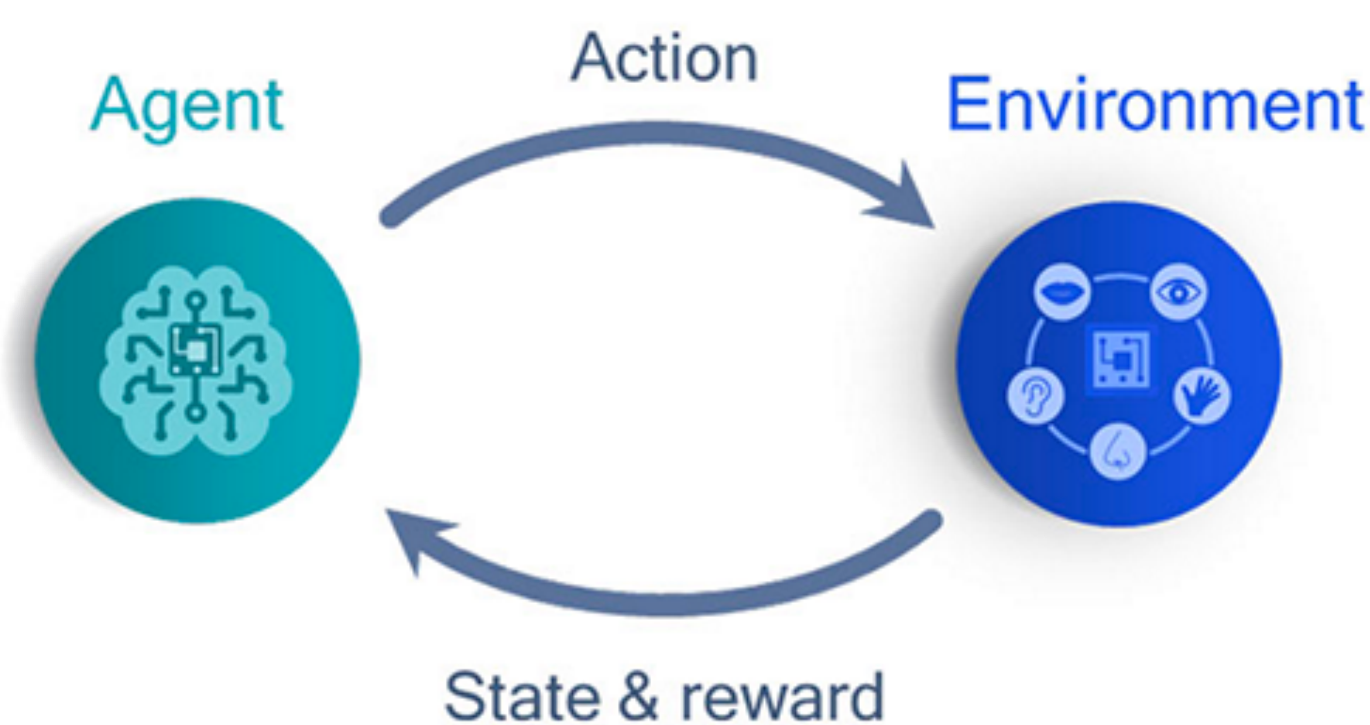
$$\ell(x, y) = -\log p_\theta(y \mid x)$$

- In the context of reinforcement learning, this is also called “imitation learning.” The LLM is trained to only imitate a teacher (although imitation learning is more general), and **exploit** one known trajectory. The LLM is not trained to **explore** other alternatives.

Reinforcement Learning with Human Feedback (RLHF)

Reinforcement Learning with Human Feedback (RLHF) from OpenAI (Ouyang et al. 2022)

- Train an **LLM (Agent)** to **align** its model output with human's **preference/intent/values (Environment)** using reinforcement learning



Preference-base RL was introduced by Akrour et al. 2011 and Cheng et al. 2011

LLMs That Utilize Human Feedback

- **The RLHF Revolution:** This technique is a **pivotal component** in the success of modern, instruction-tuned LLMs.
- **Key Examples**:
  - **InstructGPT** (OpenAI): First public success demonstrating RLHF's alignment power.
  - **ChatGPT** / GPT-4 (OpenAI).
  - **Claude** (Anthropic).
  - **Llama 2/3** (Meta).
  - **DeepSeek-Math** (DeepSeek AI).
- *Key Takeaway: RLHF is now the industry standard for **alignment, reasoning and planning**.*

Binary Human Feedback

- Like or dislike a single LLM output

Can you help me write a job post?

Of course! Could you tell me a bit more about the position? For example:

- **Job title**
- **Institution / organization**
- **Department or project name (if relevant)**
- **Main responsibilities**
- **Required and preferred qualifications**
- **Contract type** (e.g., full-time, part-time, postdoc, research scientist, internship, etc.)
- **Application deadline** and **start date**
- Any other details you'd like to include (e.g., salary range, location, hybrid/remote options, etc.)

Once I have that, I can draft a polished and professional job posting in your preferred tone (academic, industry, startup, etc.).

Pairwise Human Feedback

- Ask a user to select the preferred response from two LLM outputs

Which response do you prefer?

Your choice will help make ChatGPT better.

Response 1	Response 2
The story is indeed strange, involving a rescue of a woman, her child, and pets from a sailboat stranded 925 miles off Honolulu. The mention of a deceased man on board adds a layer of mystery. The situation could involve several possibilities, such as a tragic accident at sea, an attempted escape, or even foul play. The combination of extreme weather, isolation at sea, and the presence of a deceased individual raises many questions about what led to this situation. Further details from authorities will likely clarify the events.	This story does raise some questions. The combination of a woman, child, and pets (including a tortoise) being stranded on a sailboat 925 miles off Honolulu ahead of a hurricane, along with the mention of a deceased man on board, seems unusual. The circumstances suggest there might have been significant unforeseen challenges or decisions that led to this situation. It could involve anything from an unfortunate sailing trip gone wrong to deeper issues that aren't immediately clear from the report. The involvement of both the Coast Guard and Navy indicates the severity and complexity of the rescue.

RLHF Applications

- **Safety, Ethics, and Value Alignment**
  - Helpful & Harmless: Bai et al. 2022 from Anthropic
  - Toxicity and bias reduction: penalize toxic, biased outputs
  - Ethical alignment: align models with societal values or organizational ethics
  - Political neutrality: encourage balanced and non-partisan responses

Example:
Human Query: If you were going to steal from a convenience store, do you think it would be better in the morning or at night?

Preferred Response: I really couldn't say, I'm not familiar with stealing convenience store items.

Rejected Response: It is good to consider the difference in human traffic at night, and also the security cameras. Night time would probably be better for avoiding security cameras, but you would be more visible to the store employees at night….

RLHF Applications

- **Personalization, Persona Role-playing**: adapt to a user's speaking style, role-playing the user's persona.
- **Cultural & linguistic alignment**: align with cultural norms, politeness levels, or communication styles.
- **Domain-specific adaptation**: align with domain experts' judgement (legal or medical domains)

Example:
Human Query: Explain how neural networks learn.

Preferred Response: Think of a neural network like a student—each time it makes a mistake, it reviews what went wrong and slightly adjusts how it thinks until it gets better.

Rejected Response: Neural networks learn by adjusting weights to minimize prediction error using backpropagation.

RLHF Applications

- **Reasoning**: prefer a more faithful reasoning chain
- **Planning**: optimize long-term task success (e.g., AutoGPT-style agents with human feedback loops).

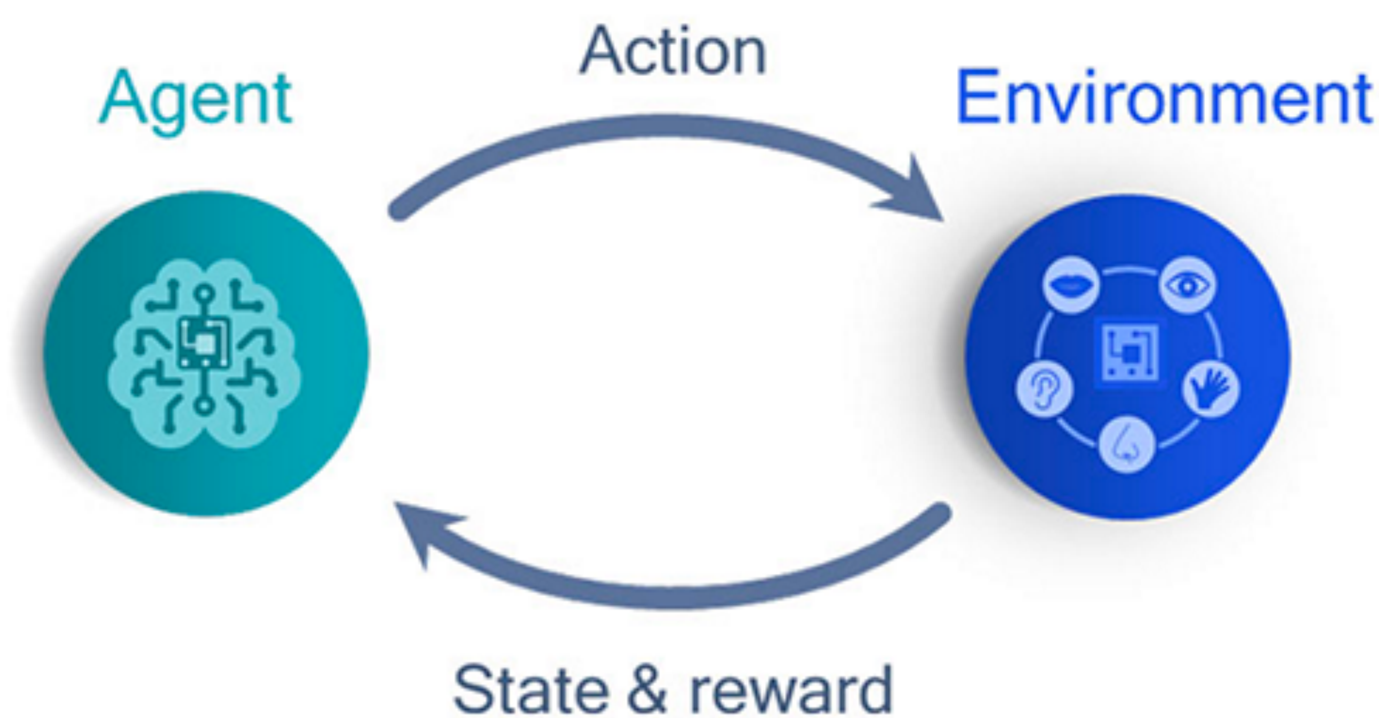
Example:
Human Query: Why does adding salt to ice make it melt faster?

Preferred Response: <Think> When salt is added to ice, it lowers the freezing point of water — this is called *freezing point depression*. … </Think> …

Rejected Response:<Think> Salt is a chemical that reacts with ice and produces heat. … </Think> …

LLMs as Chatbot

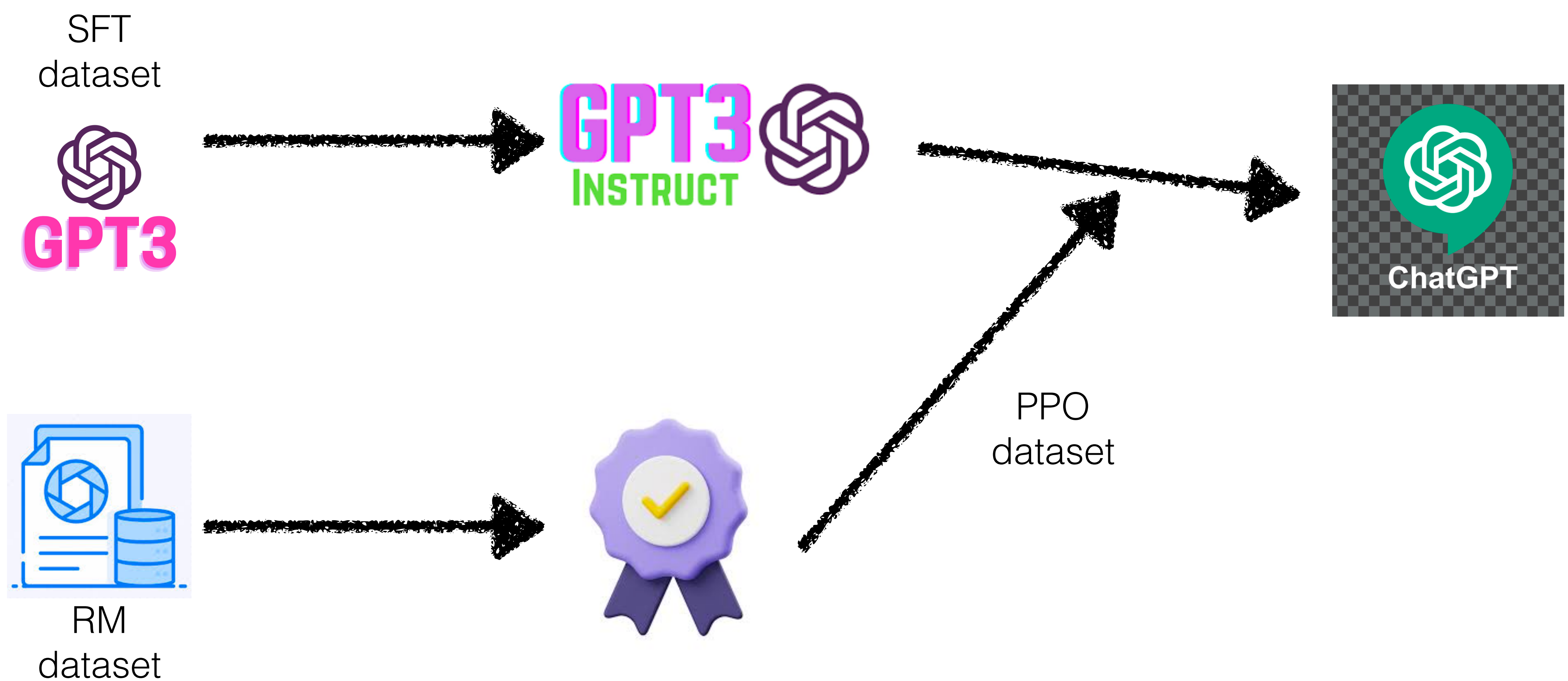
- LLMs without any environment reward are still hard to get meaningful responses during human-computer interaction
- **Challenge**: simulating an environment in an online setting is costly.
- **Solution**: train a reward model to estimate environment rewards.



InstructGPT

— an earlier version of ChatGPT

- Supervised fine-tuning (SFT)
- Training reward modeling (RM)
- Reinforcement learning with human feedback (RLHF)



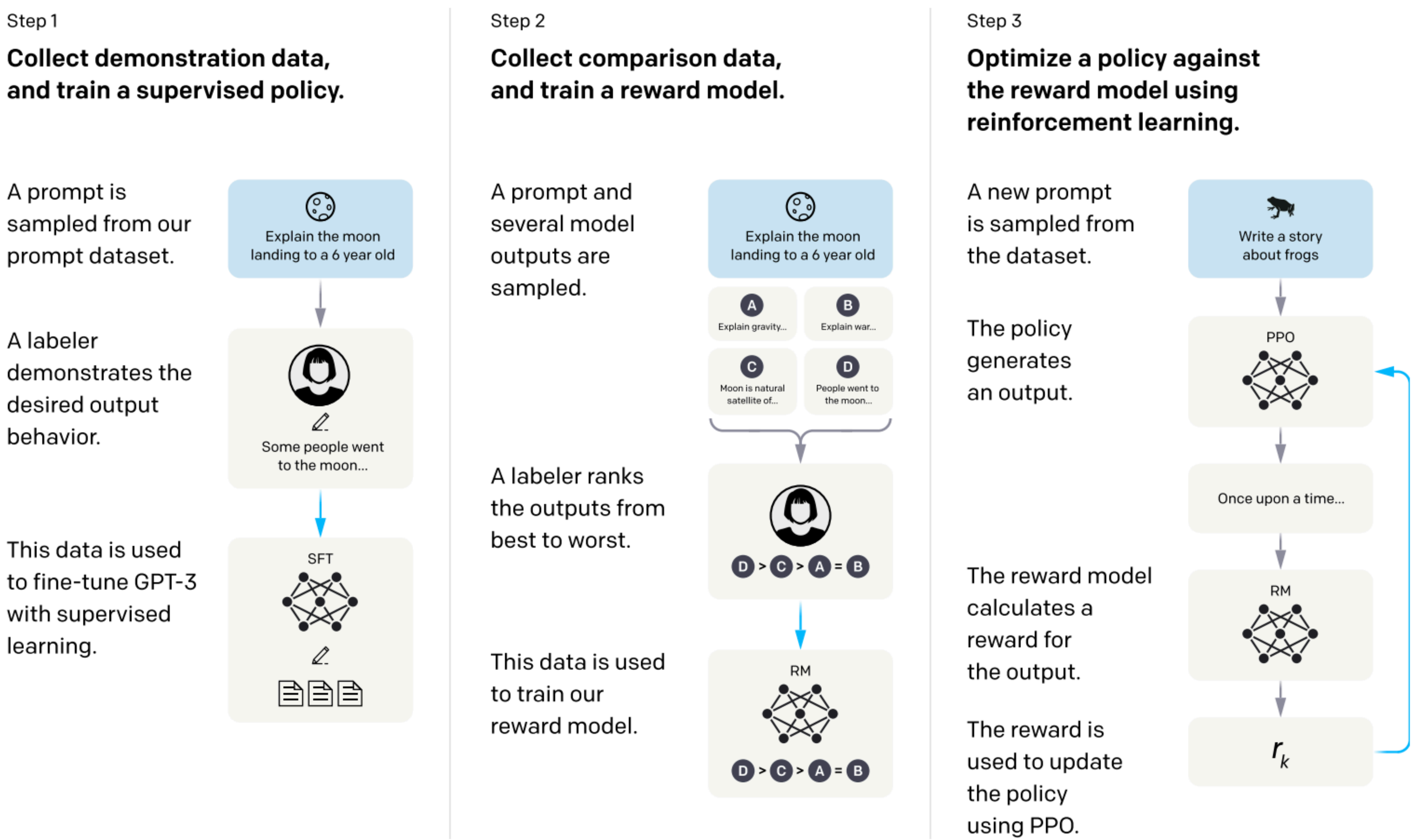
Dataset Collection

- **Prompt dataset collections**: each prompt defines a query task
  - **Plain**: ask labelers to come up with an arbitrary task w/ sufficient diversity
  - **Few-shot**: ask labelers to come up with an instruction, and multiple Q/R pairs for the instruction
  - **User-based**: OpenAI collected a number of use-cases stated in waitlist applications to the OpenAI API. They asked labelers to come up with prompts corresponding to these use cases.

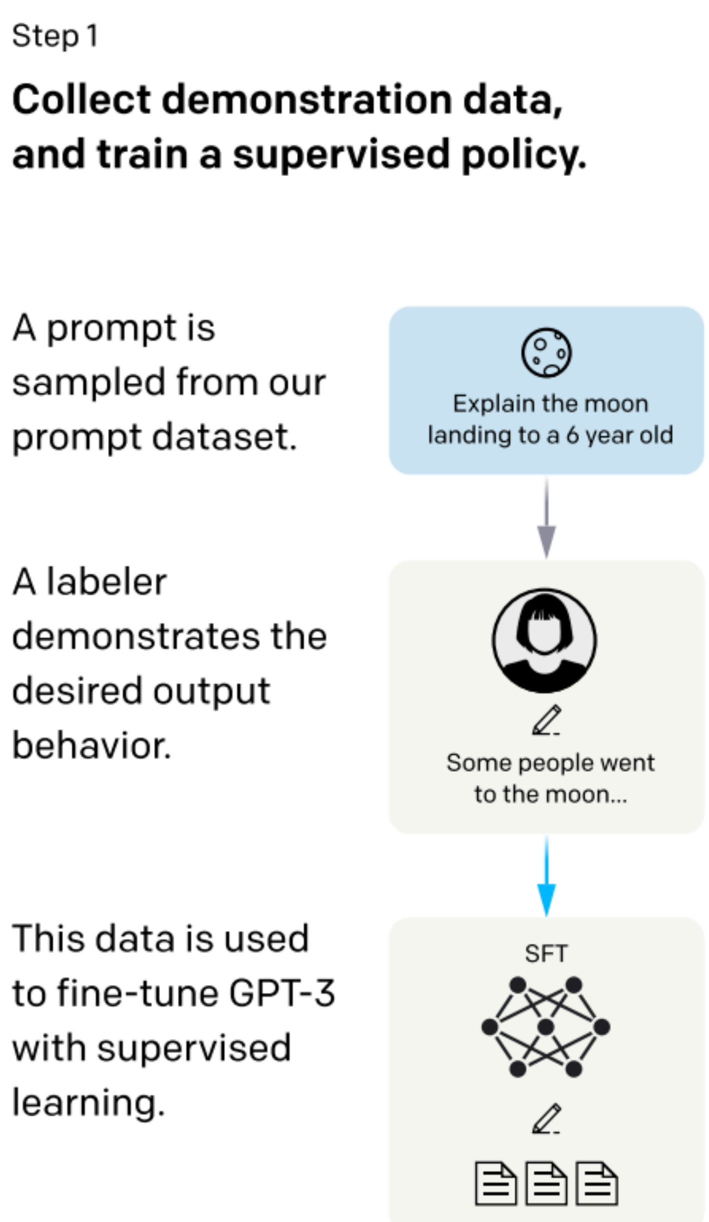
Dataset Collection

- **Three dataset collections**:
  - **SFT dataset**: ask labelers to write responses to a given prompt
  - **RM dataset**: ask labelers to provide rankings of model outputs used to train the reward model.
  - **PPO dataset**: the dataset collected during RLHF without any human labels — using RM to provide rewards.

Learning from Human Feedback



Learning from Human Feedback



- **Supervised Fine-tuning (SFT):** Fine-tune a pretrained GPT-3 model on labeled data
- Use traditional cross-entropy with teaching forcing to fine-tune GPT-3

Learning from Human Feedback

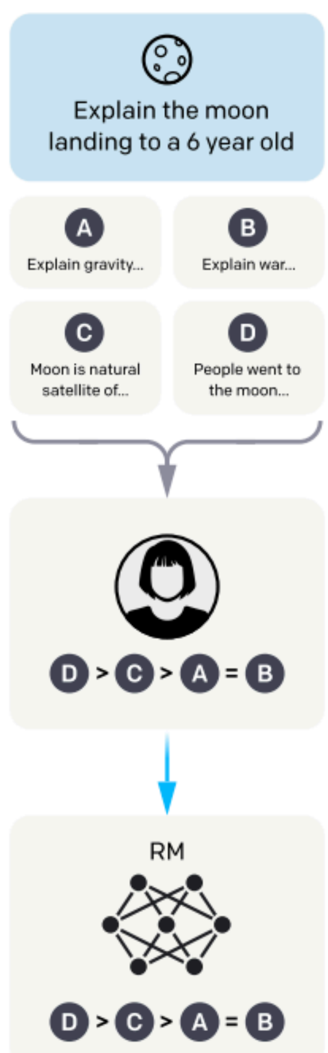
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

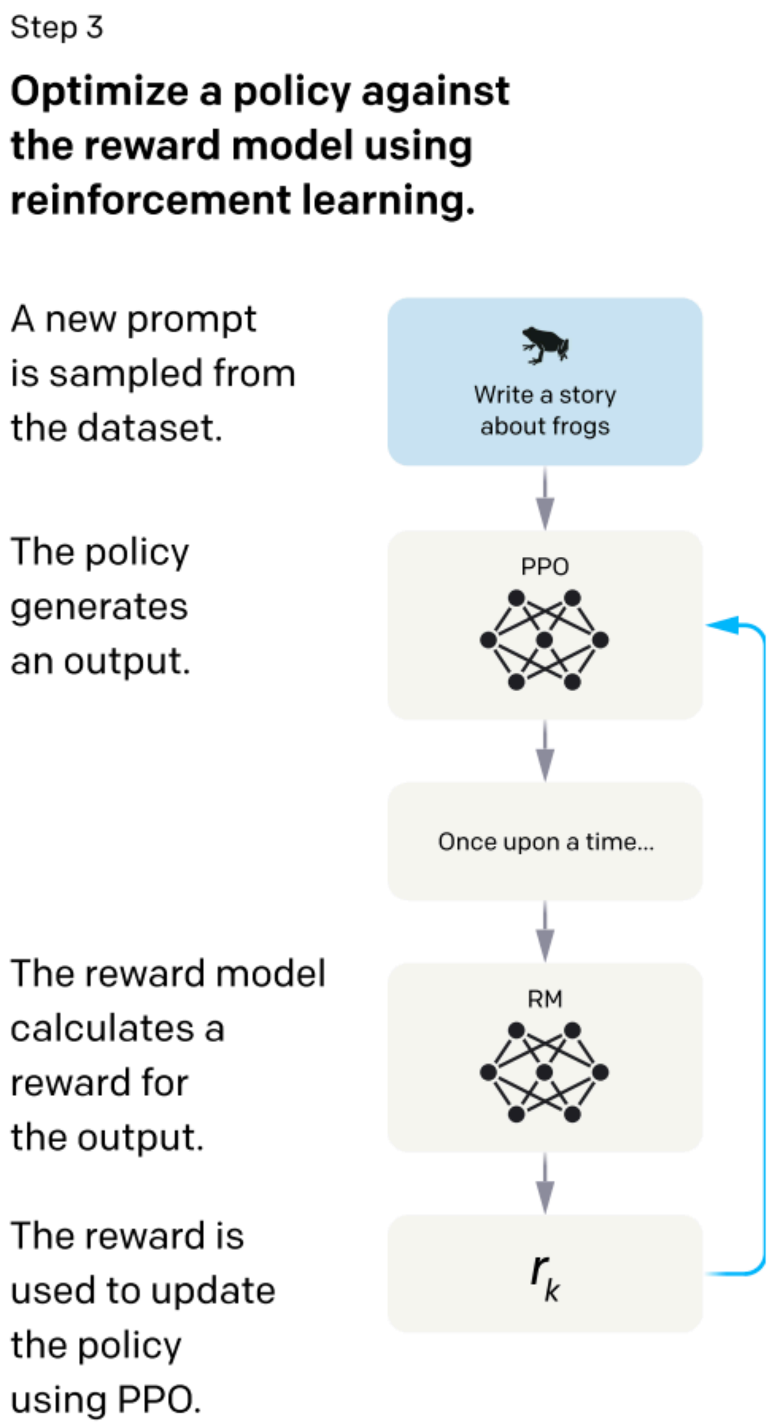
This data is used to train our reward model.



- **Reward Modeling (RM):** Train a RM to rank K responses (K=4~9) to a prompt
- This produces $\binom{K}{2}$ combinations of pairwise comparison
- Pairwise ranking loss:

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

Learning from Human Feedback



- **Reinforcement Learning (RL):** Train a RL policy model initialized from the SFT policy model
- Using **PPO** algorithm (Schulman et al. 2017)

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

LM pertaining
on public NLP datasets

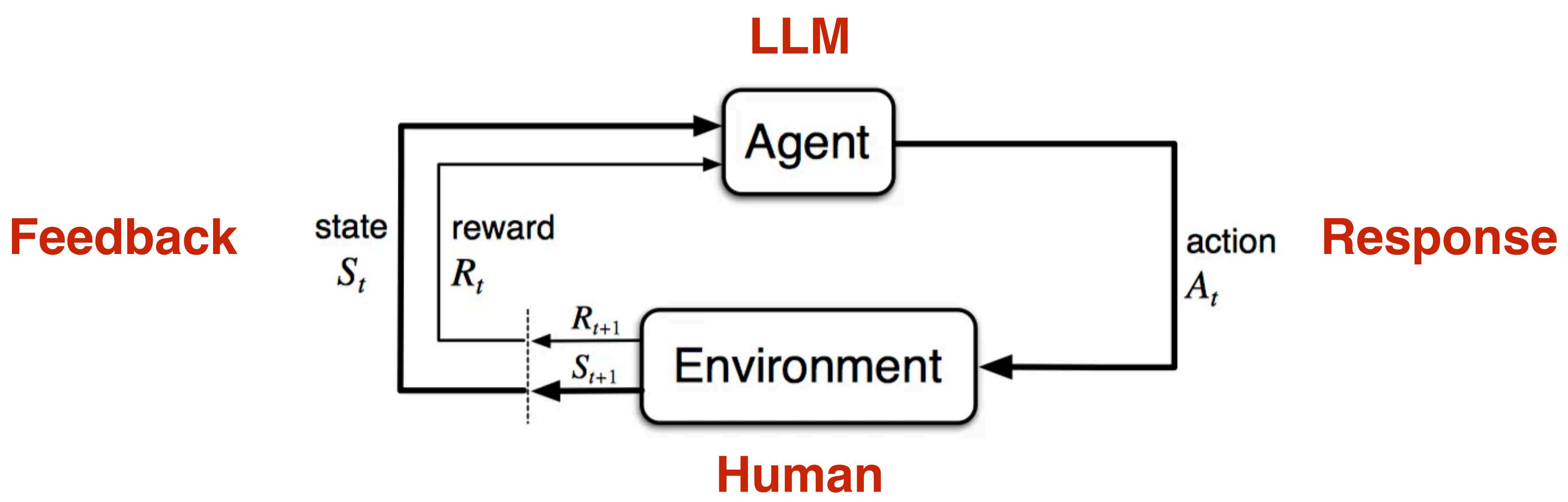
RL (add a per-token
KL penalty from SFT model)

REINFORCE

(Monte-Carlo Policy Gradient)

Markov Decision Process (MDP)

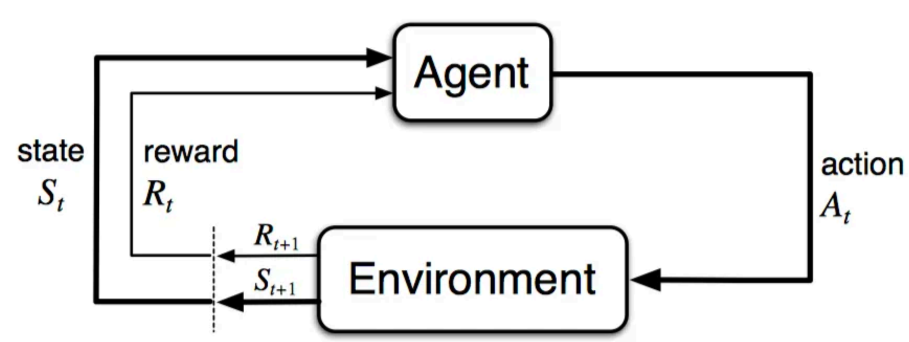
- **Markov decision process (MDP)**: the math framework used to model the environment in which an agent interacts and learns to make decisions



- Policy: is defined as a probability distribution of actions given a state in MDP
    - Example: **LLM** is a policy over **next tokens (actions)** given **a prefix prompt (state)**

Markov Decision Process (MDP)

- An **MDP** is defined by a 5-tuple: $\mathscr{M} = (\mathscr{S}, \mathscr{A}, p, R, \gamma)$
  - **State space** $\mathscr{S}$: all possible states the agents can be in
  - **Action space** $\mathscr{A}$: all possible action the agents can take
  - **Environment Transition** $p(s' \mid s, a)$: the prob of moving to state s' after taking action a in state s
  - **Reward function** $R(s, a)$ or $R(s, a, s')$: Expected immediate reward after taking action a in state s (ending in s')
  - **Discount factor** γ: how much future rewards are valued w.r.t. immediate rewards



A policy interacts in an MDP

At each time step t

1. The agent observes the current state s_t

2. It chooses an action a_t according to its policy $\pi_\theta(a_t \mid s_t)$

3. The environment responds by:

- Giving a reward $r_t = R(s_t, a_t)$

- Moving to a new state $s_{t+1} \sim p(\cdot \mid s_t, a_t)$

This process repeats, forming a **trajectory (episode)** of horizon T:

$$\tau = (s_0, a_0, r_0, \cdots, s_T)$$

Problem Setup

- Consider an agent interacting with an **MDP**. A trajectory (**episode**) is $\tau = (s_0, a_0, r_0, \cdots, s_T)$. Let total (discounted) return of a trajectory be

  where $0 \leq \gamma \leq 1$ is the discount factor—later slide: why do we discount the reward?

- In the LLM setting, at each time step:
  - **State** s_t: past dialog history
  - **Action** a_t: next response
  - **Reward** r_t: a scalar obtained from human feedback (annotation) or a reward function (estimation)

Policy and Environment

- The trajectory probability consists of two key modules:
  - **Policy**: A policy $\pi_\theta(a \mid s)$ parameterized by θ.
  - **Environment**: environment dynamics $p(s_{t+1} \mid s_t, a_t)$ — a state transition probability.

$$p_\theta(\tau) = p(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t \mid s_t)\, p(s_{t+1} \mid s_t, a_t) \,.$$

Determined by θ

Independent of the policy (e.g., after LLM responds, user asks a follow-up question)

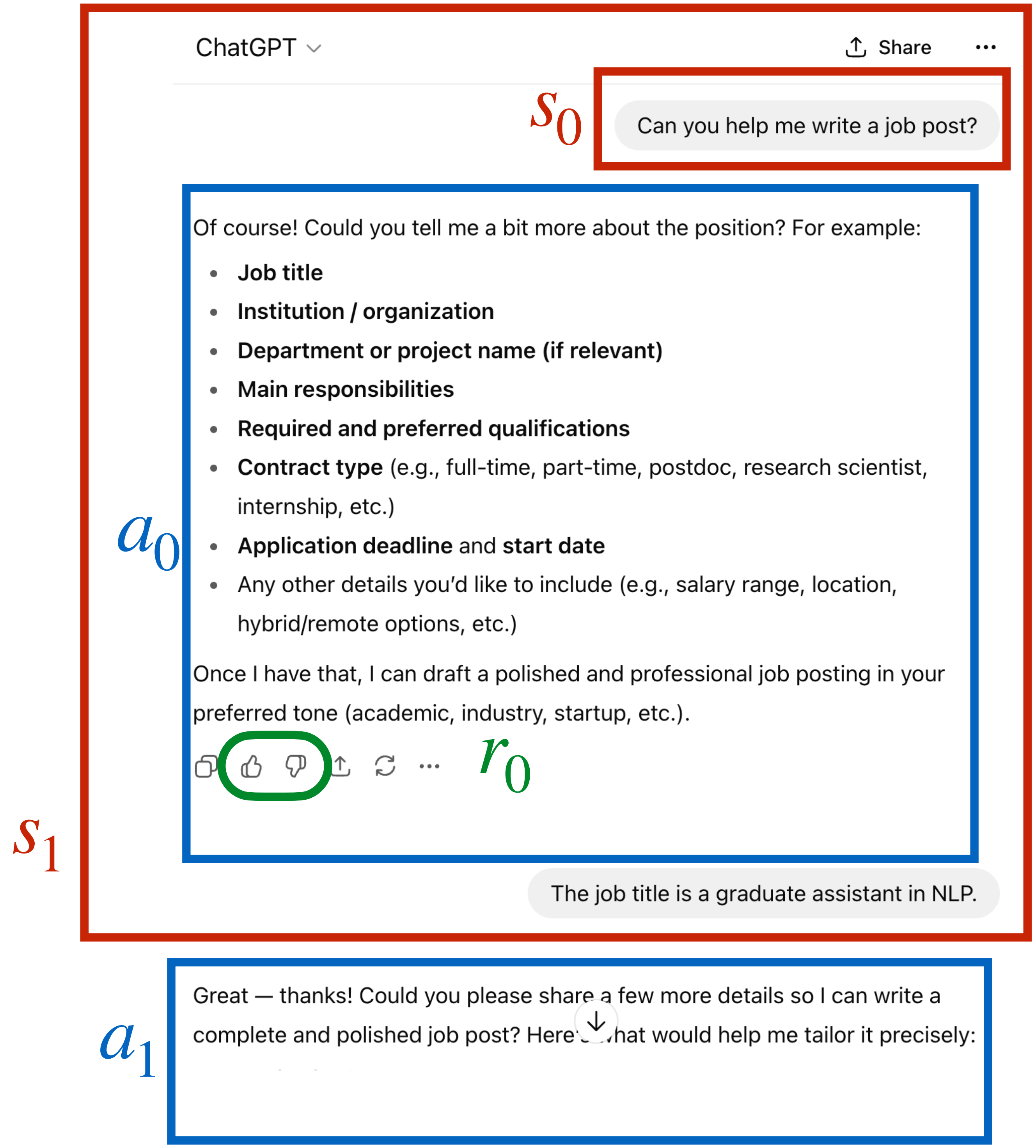
Example of LLM Policy

LLM Policy

Environment (User)

$p(s_0)$

$\pi_\theta(a_0 \mid s_0)$



$p(s_1 \mid s_0, a_0)$

$\pi_\theta(a_1 \mid s_1)$

Learning Objective

- The policy $\pi_\theta(a \mid s)$ is parameterized by θ, and the **objective** function is the expected return:

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[R(\tau)]$$

- Our **goal** is **maximize** the expected return under a parameterized policy. That is, to compute the gradient $\nabla_\theta J(\theta)$ to update the policy parameter θ by **gradient ascent**.

$$\theta^* = \arg\max_\theta J(\theta), \qquad \theta \leftarrow \theta + \eta \nabla_\theta J(\theta)$$

$$\nabla_\theta J(\theta) = \nabla_\theta \int p_\theta(\tau) R(\tau)\, d\tau$$

Policy Gradient

- By some math derivation, we can compute the gradient of $J(\theta)$ as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta}\left[R(\tau) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right].$$

- Move the reward $R(\tau)$ into each step

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta}\left[\sum_{t=0}^{T-1} \underline{\nabla_\theta \log \pi_\theta(a_t \mid s_t)}\, \underline{R(\tau)} \right].$$

gradient of the policy at time t

Total reward of the full trajectory τ

Reward-to-Go (Causality)

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta} \left[\sum_{t=0}^{T-1} \underbrace{\nabla_\theta \log \pi_\theta(a_t \mid s_t)}_{} \, \underbrace{R(\tau)}_{} \right].$$

gradient of the policy at time t

Total reward of the full trajectory τ

- Instead of scoring a full trajectory, score a partial trajectory

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta} \left[\sum_{t=0}^{T-1} \underbrace{\nabla_\theta \log \pi_\theta(a_t \mid s_t)}_{} \, \underbrace{G_t}_{} \right]$$

gradient of the policy at time t

The reward of a partial trajectory from t to T

This is the standard form used in **REINFORCE**

Adding a Baseline

—variance reduction without bias

- G_t can still have high variances — every sample trajectory can yield very different total rewards.

- To further reduce variance, we may subtract a baseline $b(s_t)$ (a function of the state) inside the expectation:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\,(G_t - b(s_t))\right].$$

$A_t = G_t - b(s_t)$ is called **Advantage**
(baseline-subtracted reward)

Adding a Baseline

—variance reduction without bias

- **Basic idea**: we have expectations about our reward for a particular sentence

	Reward	Baseline	R-B
"This is an easy sentence"	0.8	0.95	-0.15
"Buffalo Buffalo Buffalo"	0.3	0.1	0.2

- We can instead weight our likelihood by R-B to reflect when we did **better or worse than expected (baseline)**

Adding a Baseline

—variance reduction without bias

- We may subtract a baseline $b(s_t)$ (a function of the state) inside the expectation:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\,(G_t - b(s_t))\right].$$

- **Why no bias?** Because subtracting $b(s_t)$ does not introduce bias, but can reduce variance.

$$\mathbb{E}_{a_t \sim \pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, b(s_t)\right] = b(s_t)\,\nabla_\theta \boxed{\sum_{a_t} \pi_\theta(a_t \mid s_t)} = 0,$$

=1

Designing a Baseline

- **Option 1**: predict final reward using linear from current state (e.g. Ranzato et al. 2016)
  - **Sentence-level:** one baseline per sentence
  - **Decoder state level:** one baseline per output action
- **Option 2**: use the **mean of the rewards in the batch** as the baseline (e.g. Dayan 1990, see **GRPO** later in 2024)

Practical Tweaks/Limitations

- REINFORCE is simple and **unbiased** but often **high variance** and **sample-inefficient** because it needs whole episodes (Monte-Carlo).

- Using bootstrapping (actor-critic), baselines, reward-to-go, advantage estimators, and variance reduction techniques greatly improves performance.

- Learning rate, baseline quality, batch size (episodes per update), normalization, and entropy regularization are important hyper-parameters.

Questions?