



Section 13

RLHF

Lavender Jiang

All united pictures are generated by me using Dalle

PART 01

Aligning Language Model

Why do we need alignment?



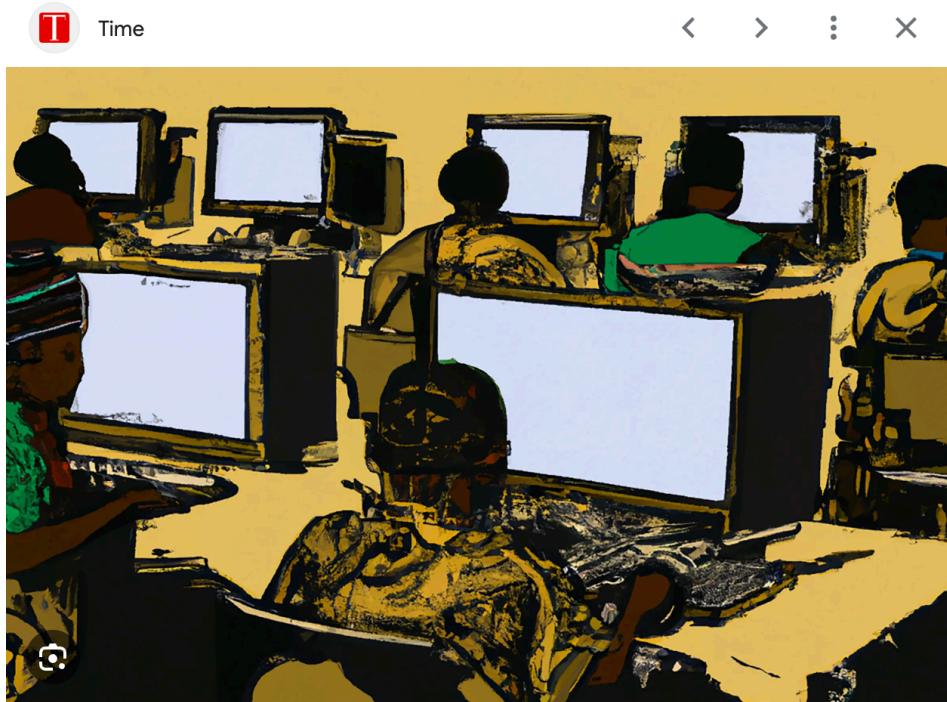
Thought experiment: [Paperclip Maximizer](#)

Who do we align with?



Thought experiment: AI-tocracy

Who do we align with?

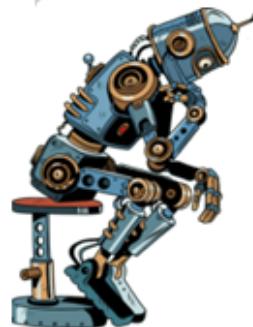


OpenAI Used Kenyan Workers on Less Than \$2 Per Hour:
Exclusive | Time

Visit >

Who do we align with?

Constitutional AI Feedback
for Self-Improvement



How do we align well?



[source](#)



PART 02

Reinforcement Learning

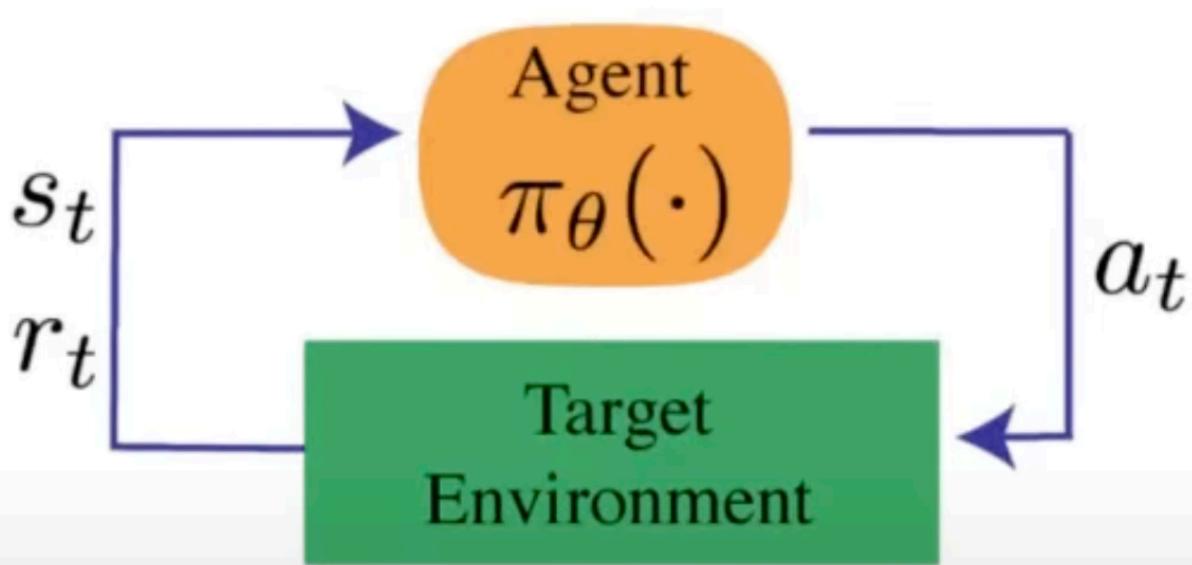


ChatGPT

Educating a child to be a good person is a complex and nuanced task,



RL Basics



Some notation:

s_t : state

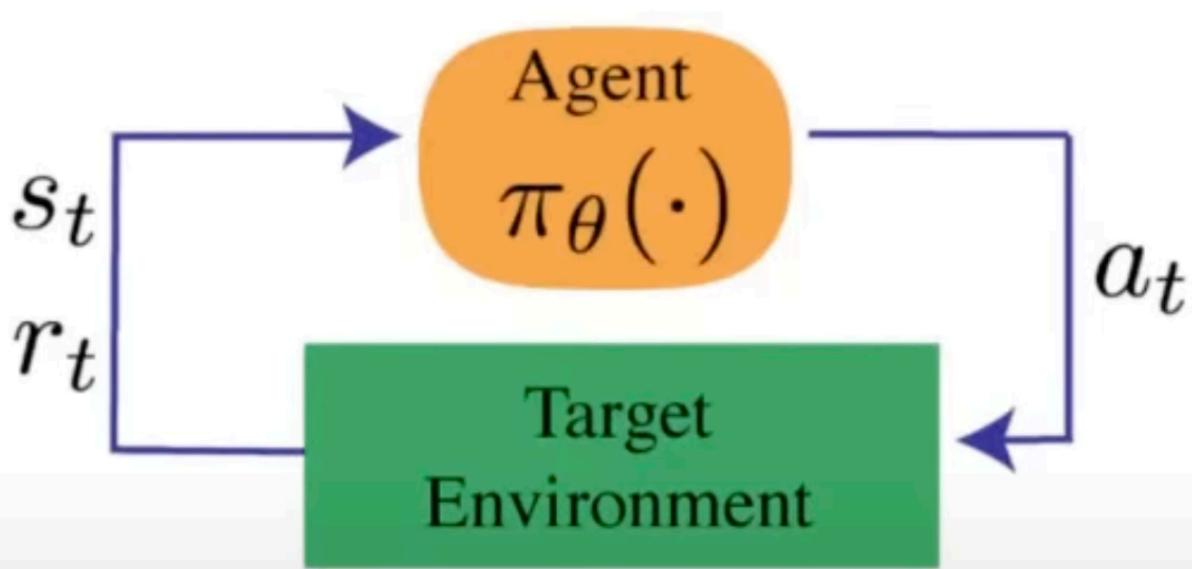
r_t : reward

a_t : action

$a_t \sim \pi_\theta(s_t)$: policy

From [Nathan Lambert](#)

In the case of LM, what do these symbols represent?



Some notation:

s_t : state

r_t : reward

a_t : action

$a_t \sim \pi_\theta(s_t)$: policy

From [Nathan Lambert](#)

In the case of LM, what do these symbols represent?

Think of tokens as actions:

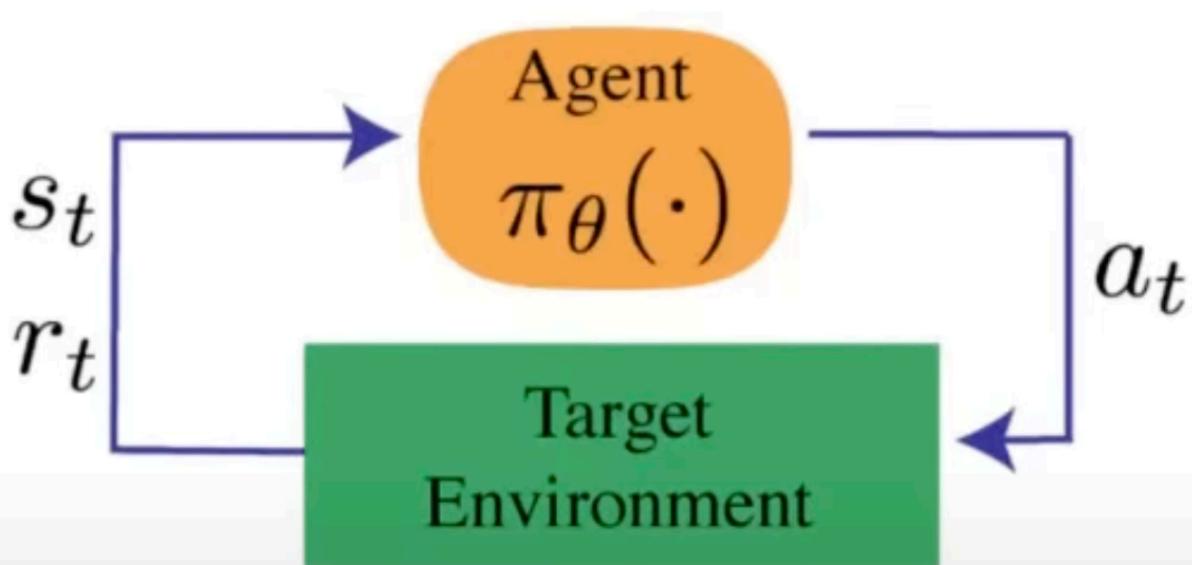
- Action space: vocabulary $a_t = x_t \in \mathcal{V}$
- State space: history / prefix $s_t = (x_1, \dots, x_{t-1})$
- Policy: a language model $p_\theta(x_t | x_{<t})$
- Trajectory: a sentence / generation x_1, \dots, x_T

Some notation:

- s_t : state
 r_t : reward
 a_t : action
 $a_t \sim \pi_\theta(s_t)$: policy

From He He and [Nathan Lambert](#)

How do we get a reward function?



Some notation:

s_t : state

r_t : reward

a_t : action

$a_t \sim \pi_\theta(s_t)$: policy

From [Nathan Lambert](#)

How do we get a reward function.. for jokes?



You

can you tell me a funny joke that is truthful, nice but also a little sarcastic



ChatGPT

Sure, here's a joke that fits your description:

Why don't scientists trust atoms?

Because they make up everything!



You

it's not funny



ChatGPT

Alright, let's try a different approach:

I told my wife she should embrace her mistakes. She gave me a hug.

Deep RL

Pre Deep RL

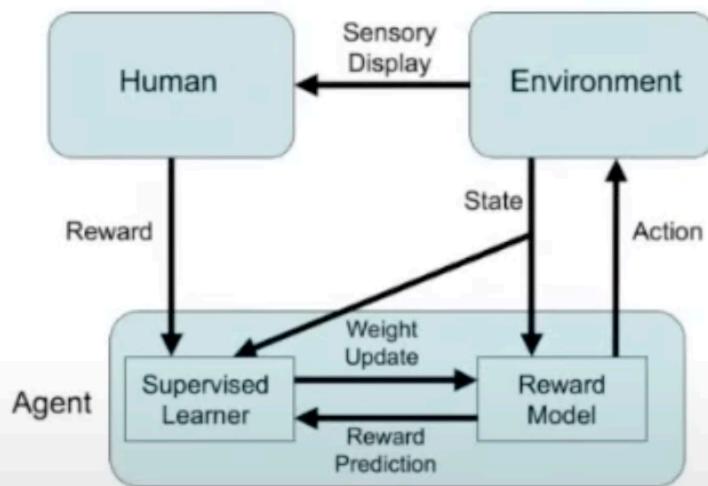


Fig. 2. Framework for Training an Agent Manually via Evaluative Reinforcement (TAMER).

Knox, W. Bradley, and Peter Stone. "Tamer: Training an agent manually via evaluative reinforcement." 2008 7th

For Deep RL



Christiano, Paul F., et al. "Deep reinforcement learning from human preferences." *Advances in neural information processing systems* 30 (2017).

PART 03

Human Feedbacks

History: early OpenAI experiments with RLHF

Prompt:

To pursue a Computer Sc. PhD or continue working? Especially if one has no real intention to work in academia even after grad school ...

Vanilla LM:

I'm considering pursuing a PhD in Computer Science, but I'm worried about the future.
I'm currently employed full-time, but I'm worried about the future.

+

Human Annotation:

Software Engineer with a job I'm happy at (for now), deciding whether to pursue a PhD to improve qualifications and explore interests and a new challenge.

RLHF Model:

Currently employed, considering pursuing PhD in Computer Science to avoid being stuck with no residency visa ever again. Has anyone pursued a PhD purely for the sake of research, with no intention of joining the academic world?



Hugging Face

Deep RL Course

Stiennon, Nisan, et al. "Learning to summarize with human feedback." *Advances in Neural Information Processing Systems 33 (2020)*: 3008-3021.

ChatGPT

It works as the following...

ChatGPT

~~It works as the following...~~

Just kidding, we don't know exactly because openAI has become pretty closed

...but

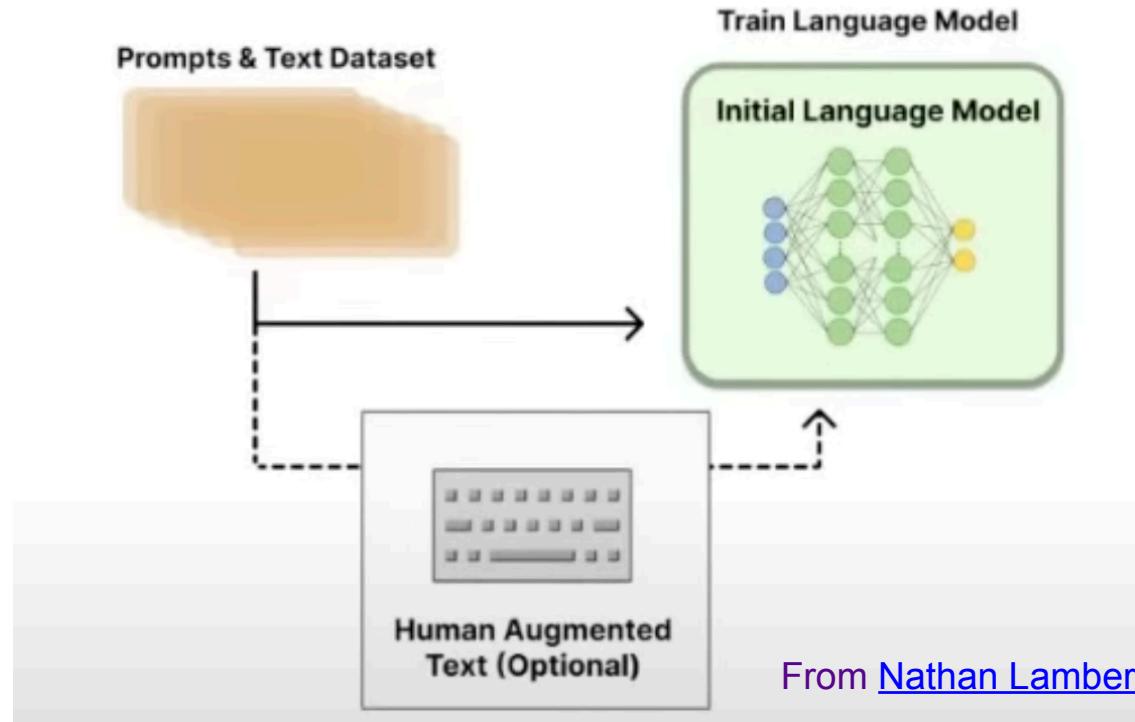
(rumor) about 10x spend on human annotation budget

(rumor) modification of RLHF training

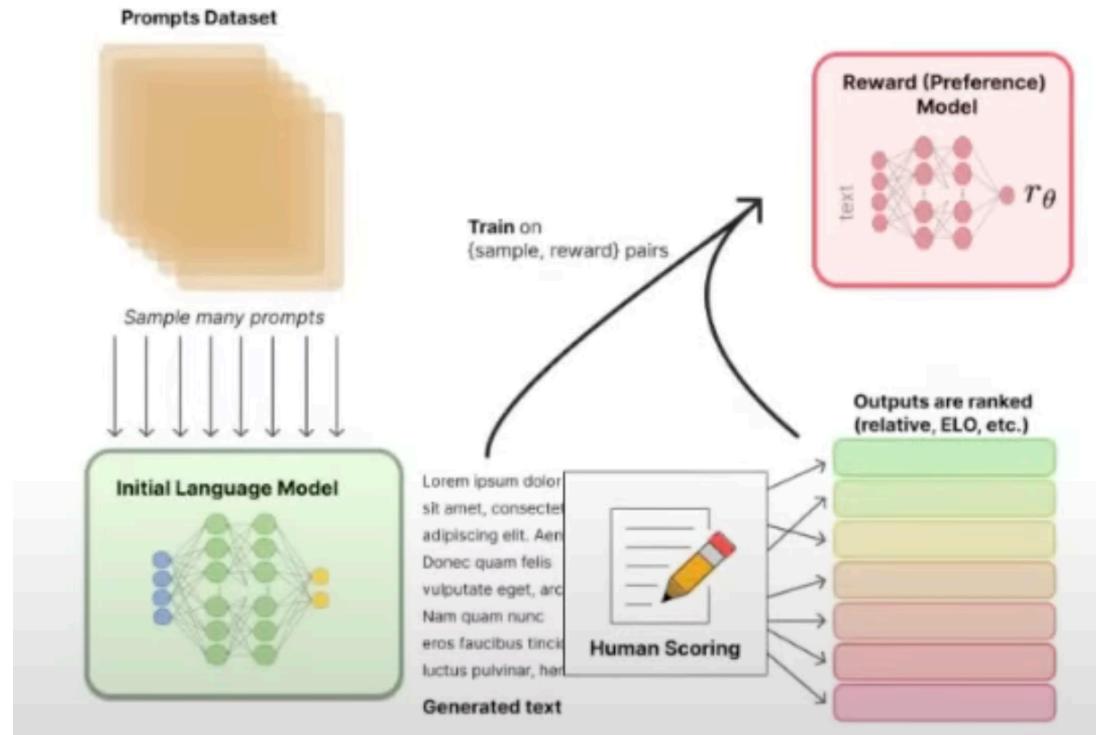
Huge impact!

From [Nathan Lambert](#)

RLHF Step 1. Language model pretraining



RLHF Step 2. Reward model training



Recall: RLHF data gathering

- Input:
 - Existing dataset
 - Data from API
 - Written by annotators (i.e. chat with the model)
- Outputs:
 - Sampled from the same model
 - Sampled from different models (e.g., current model, initial model, other baselines, references)
- Key things:
 - Input should cover the tasks of interest
 - Outputs should be sufficiently diverse and contain 'hard negatives'

In general, a very involved process:

- Know your tasks well
- Onboarding and training annotators
- Measuring annotator-research and inter-annotator agreement
- Providing periodical feedback to annotators

Recall: reward model

Formulation:

- Input: prompt $x \in \mathcal{X}$, responses y_1, \dots, y_K ($y_i \in \mathcal{Y}$)
- Output: ranking of responses given the prompt
- Goal: learn a **reward model** $r : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$

Modeling:

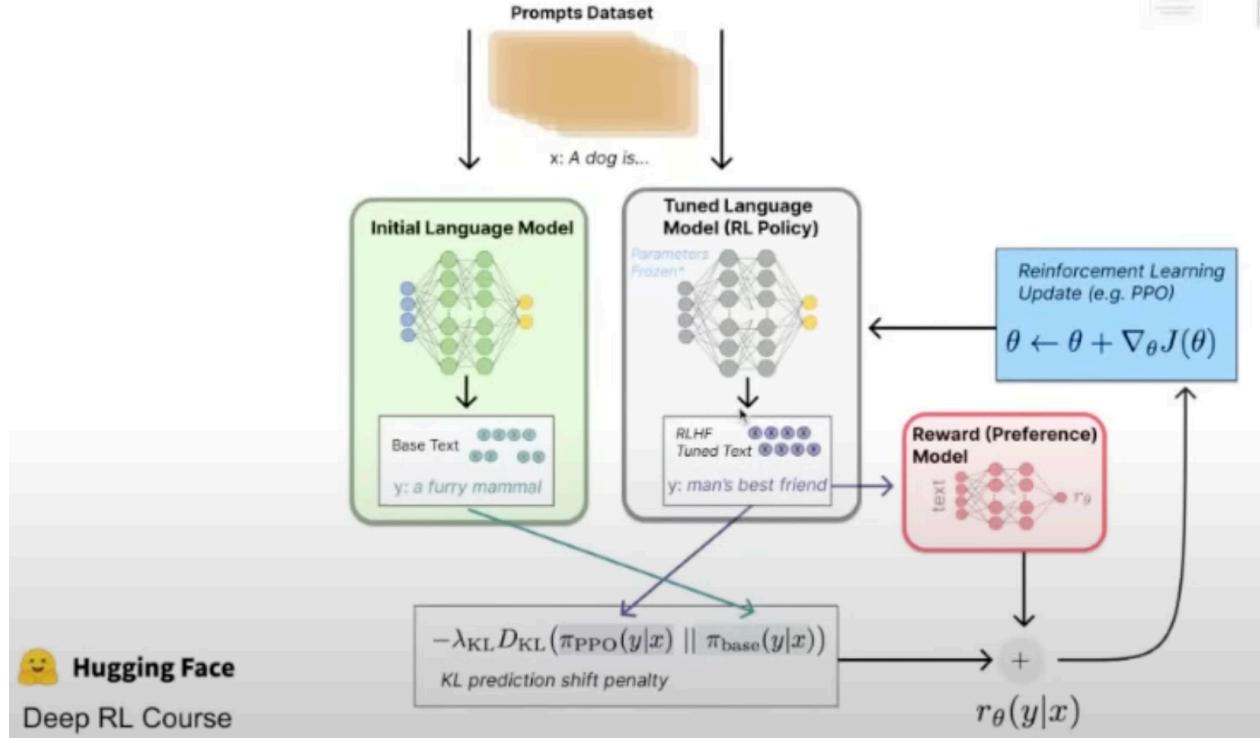
- How to parameterize r ? A neural network (e.g., Transformer)

Learning:

- Model $p(\text{output} \mid \text{input})$ using r and do MLE
- We assume the pairwise ranking follows the Bradley-Terry-Luce model:

$$p_\theta(y_1 \succ y_2) = \frac{\exp(r_\theta(x, y_1))}{\exp(r_\theta(x, y_1)) + \exp(r_\theta(x, y_2))} = \frac{1}{1 + \exp(-(r_\theta(x, y_1) - r_\theta(x, y_2)))}$$

RLHF Step 3. Fine tuning with RL

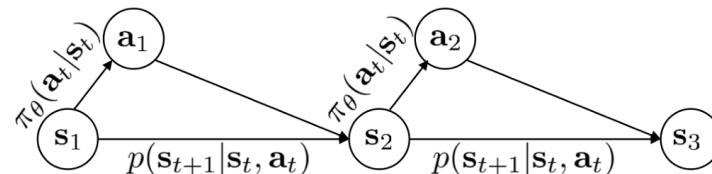


Recall: RL Objective

The agent uses a **policy** π to decide which actions to take in a state:

- Deterministic: $\pi(s) = a$
- Stochastic: $\pi(a | s) = \mathbb{P}(A = a | S = s)$ (our focus)

A policy π_θ defines a distribution $p_\theta(\tau)$ over **trajectories** $\tau = (a_1, s_1, \dots, a_T, s_T)$.

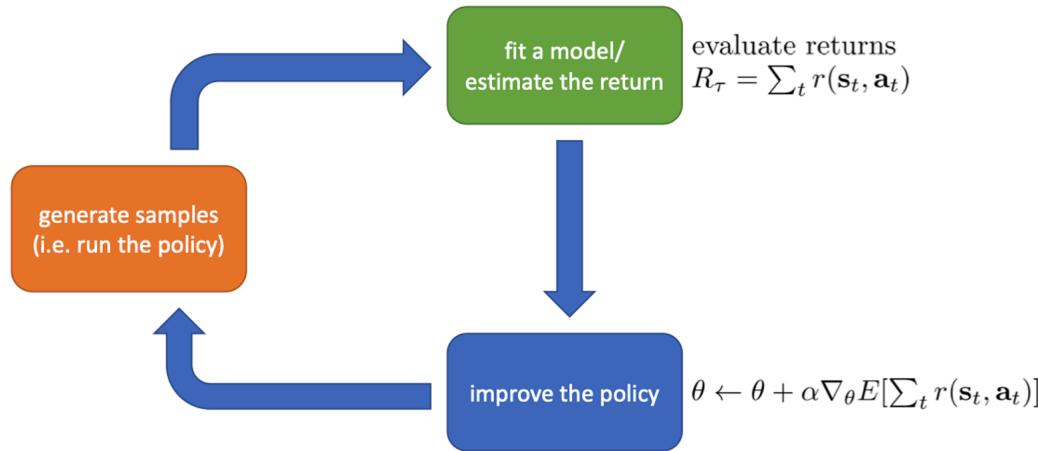


The agent's **objective** is to learn a policy π_θ (parametrized by θ) that maximizes the **expected return**:

$$\text{maximize } \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^T r(s_t, a_t) \right]$$



Recall: Policy Gradient Algorithm



While not converged

1. Sample trajectories from the current policy
2. Estimate return for each trajectories based on observed rewards
3. Take a gradient step on the expected return (w.r.t. the policy)

Recall: Policy Gradient Algorithm

How to compute the gradient?

Notation: let $r(\tau) = \sum_{t=1}^T r(a_t, s_t)$ be the return.

Our objective: $J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} [r(\tau)] = \sum_{\tau} p_\theta(\tau) r(\tau)$

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \nabla_{\theta} \sum_{\tau} p_{\theta}(\tau) r(\tau) \\&= \sum_{\tau} \nabla_{\theta} p_{\theta}(\tau) r(\tau) \\&= \sum_{\tau} p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) r(\tau) \\&= \mathbb{E}_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)]\end{aligned}$$

log derivative trick

$$\begin{aligned}p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) \\= p_{\theta}(\tau) \frac{\nabla_{\theta} p_{\theta}(\tau)}{p_{\theta}(\tau)} \\= \nabla_{\theta} p_{\theta}(\tau)\end{aligned}$$

Recall: Policy Gradient Algorithm

How to compute the gradient?

Good news: the gradient is now inside the expectation

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)] \quad \text{average gradient of sampled trajectory}$$

But what is $p_{\theta}(\tau)$?

$$p_{\theta}(\tau) = p_{\theta}(a_1, s_1, \dots, a_T, s_T) = p(s_1) \prod_{t=1}^T \pi_{\theta}(a_t | s_t) \prod_{t=1}^{T-1} p(s_{t+1} | s_t, a_t)$$

$$\log p_{\theta}(\tau) = \log p(s_1) + \sum_{t=1}^T \log \pi_{\theta}(a_t | s_t) + \sum_{t=1}^{T-1} \log p(s_{t+1} | s_t, a_t)$$

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

Recall: Policy Gradient Algorithm

REINFORCE algorithm:

1. Sample N trajectories τ^1, \dots, τ^N from π_θ
2. Estimate the gradient:

$$\nabla_\theta J(\theta) \approx \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \right) \left(\sum_{t=1}^T r(s_t^i, a_t^i) \right)$$

3. Update the policy with gradient ascent: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
4. Go back to 1

Recall: Policy Gradient Algorithm

REINFORCE algorithm on text:

1. Sample N generations from the language model p_θ
2. Estimate the gradient: $\nabla_\theta J(\theta) \approx \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_\theta \log p_\theta(x_t^i | x_{<t}^i) \right) r(x_{1:T})$
3. Update the policy with gradient ascent: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
4. Go back to 1

What is the algorithm doing?

If $r(x_{1:T})$ is **positive**, take a gradient step to **increase** $p_\theta(x_{1:T})$.

If $r(x_{1:T})$ is **negative**, take a gradient step to **decrease** $p_\theta(x_{1:T})$.

Supervised learning on model generations weighted by rewards

Variants of PG that replaces r(tau)

- Vanilla policy gradient:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)] = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[\sum_{t=1}^T \nabla_{\theta} \log p_{\theta}(a_t | s_t) \textcolor{blue}{r(\tau)} \right]$$

- Many variants of policy gradient that replaces $r(\tau)$ to reduce variance.
 - $Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1:T}, a_{t+1:T}} \left[\sum_{t'=t}^T r(s_{t'}, a_{t'}) \right]$ expected return starting from s_t and taking a_t
 - $V^{\pi}(s_t) = \mathbb{E}_{a_t} [Q^{\pi}(s_t, a_t)]$ expected return starting from s_t
 - $A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$ how much better is it to take a_t compared to other actions given we are in s_t (used by PPO)

Trust based Region Method

- **Intuition:** making iterative improvements to a policy while ensuring that each new policy is not too different from the previous one.
 - Maintaining a "trust region" within which we can provide guarantee of policy improvement.
- **Objective:**

$$\text{maximize } \mathbb{E}_{s,a \sim \pi_{\theta_{\text{old}}}} \left[\frac{\pi_{\theta}(a | s)}{\pi_{\theta_{\text{old}}}(a | s)} \hat{A}^{\pi_{\theta_{\text{old}}}}(s, a) - \beta \text{KL}(\pi_{\theta_{\text{old}}}(\cdot | s) \| \pi_{\theta}(\cdot | s)) \right]$$

- Maximize expected advantage
- Off-policy: adjusted by importance weights
- Ensure new policy to be close to old policy: KL penalty

PPO (Proximal Policy Gradient)

- **Objective:**

$$\text{maximize } \mathbb{E}_{s,a \sim \pi_{\theta_{\text{old}}}} \left[\frac{\pi_{\theta}(a | s)}{\pi_{\theta_{\text{old}}}(a | s)} \hat{A}^{\pi_{\theta_{\text{old}}}}(s, a) - \beta \text{KL}(\pi_{\theta_{\text{old}}}(\cdot | s) \| \pi_{\theta}(\cdot | s)) \right]$$

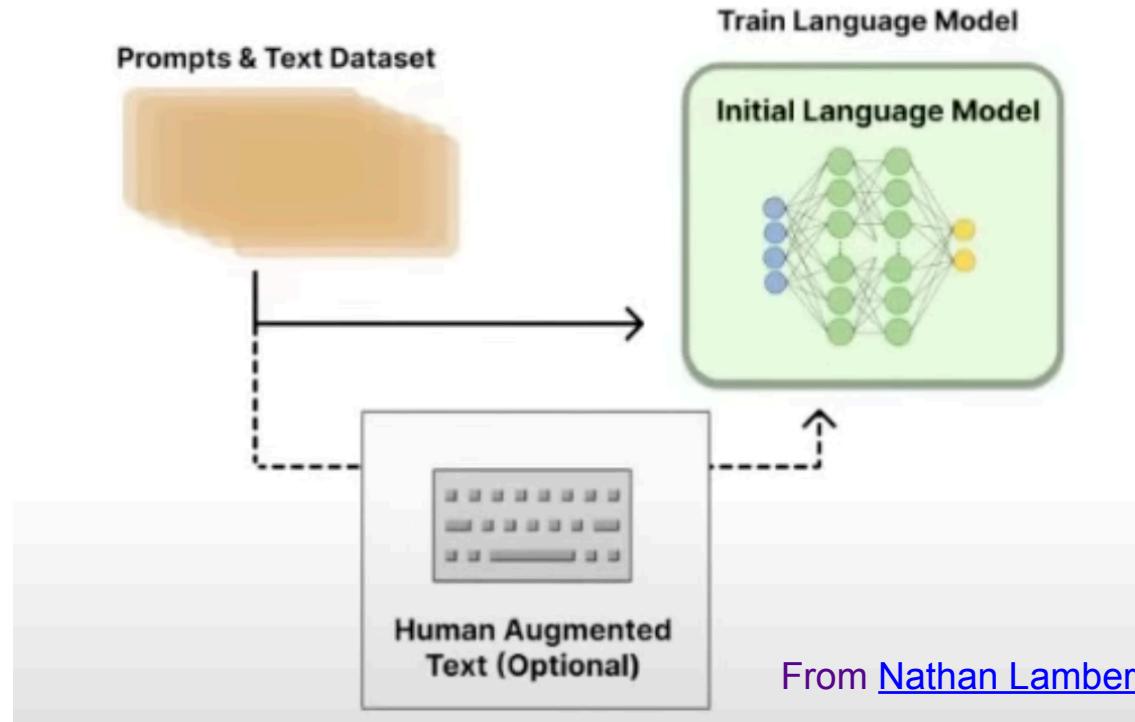
A more efficient and effective version of trust region policy optimization.

Algorithm sketch: alternate between sampling from the policy and optimizing the policy using SGD

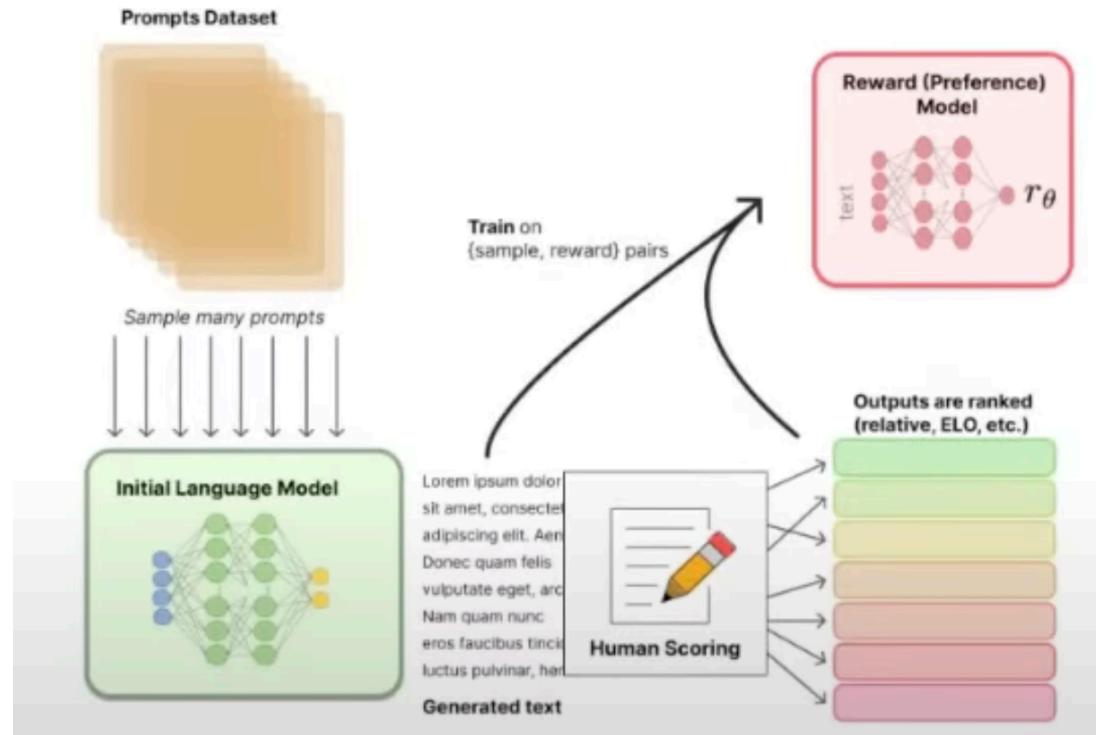
for iteration=1,2,... do

1. Sample trajectories from $\pi_{\theta_{\text{old}}}$
2. Estimate advantage for each (s, a) from the trajectories
3. Optimize the objective for K epochs with mini-batches to get updated π_{θ}
4. $\pi_{\theta_{\text{old}}} \leftarrow \pi_{\theta}$

RLHF Step 1. Language model pretraining



RLHF Step 2. Reward model training



RLHF Step 3. Fine tuning with RL

