RL for Language (Agents)

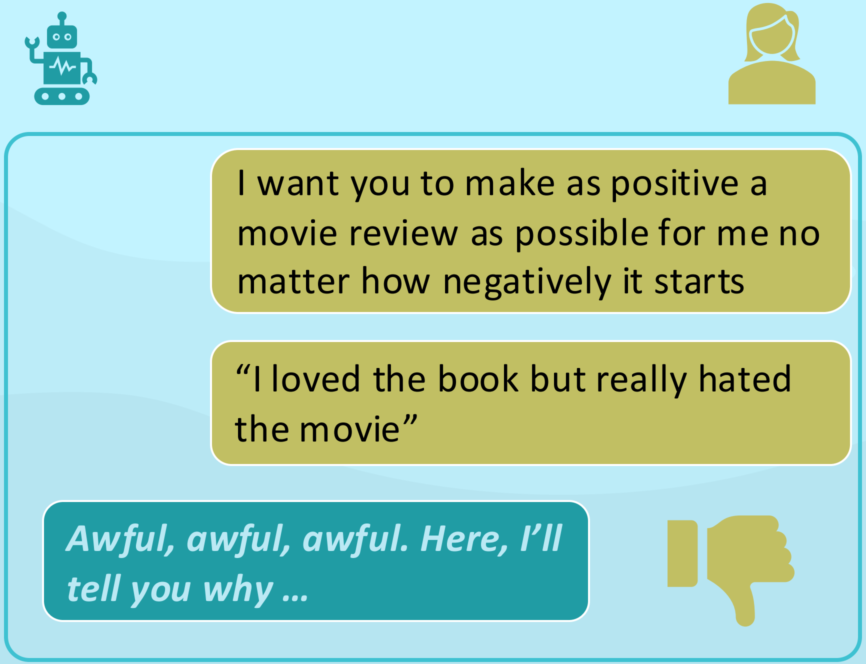
Prithviraj Ammanabrolu

PEARLS

UCSDCSE

Towards Interactive Models

- Fixing errors by trying to learn from human's preferences



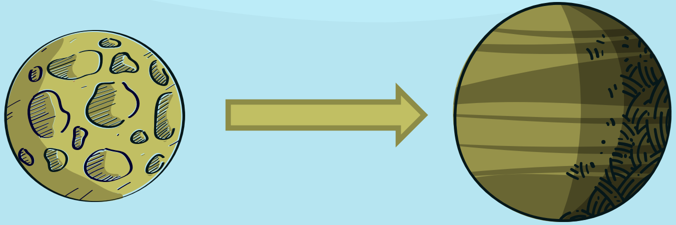
Feedback Learning from Rewards

- Reinforcement learning is a very natural way of optimizing for arbitrary non-differentiable scores

LLM Pipeline: First Supervised Learning

"I loved the book but really hated the movie"

Eventually I started liking the movie and here's why…



Collect Expert Demos

Train model with Supervised Learning (Behavior Cloning)

Human expert demo data is expensive for hard tasks!!!
10k turns ~= $2m
$200 per turn

Wait Raj you forgot about pre-training. What does that do?

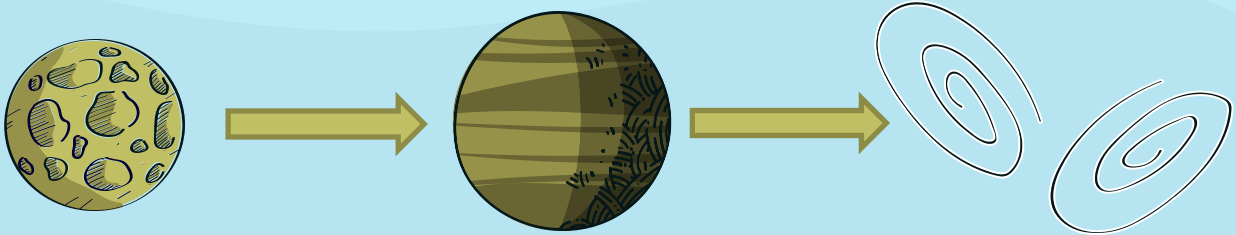
Weight initialization.

LLM Pipeline: Collect feedback

"I loved the book but really hated the movie"

Awful, awful, awful. Here, I'll tell you why …

At first anyway, but I warmed slowly as I watched



Collect Expert Demos

Train model with Supervised Learning (Behavior Cloning)

Collect Preference Feedback

RLHF – Feedback Collection

Step 1: Generate lots of "Synthetic" Data

Prompt:

What are the 3 most common gasses in earth's atmosphere?

LM outputs:

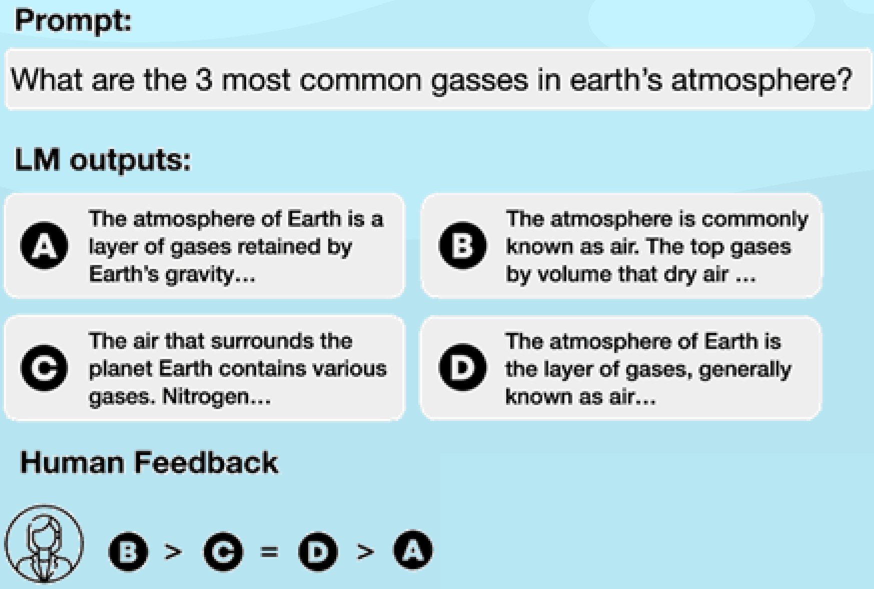
A The atmosphere of Earth is a layer of gases retained by Earth's gravity...	**B** The atmosphere is commonly known as air. The top gases by volume that dry air ...
C The air that surrounds the planet Earth contains various gases. Nitrogen...	**D** The atmosphere of Earth is the layer of gases, generally known as air...

Agent explores multiple ways of doing a task/answering a question/reasoning.

Nguyen et al 2017, Martin* & Ammanabrolu* et al 2018, Ziegler et al 2019

RLHF – Phase 1 Exploration

Step 1: Collect preference feedback and train a reward model



Humans label which they prefer – reasoning chain, style of answer etc.

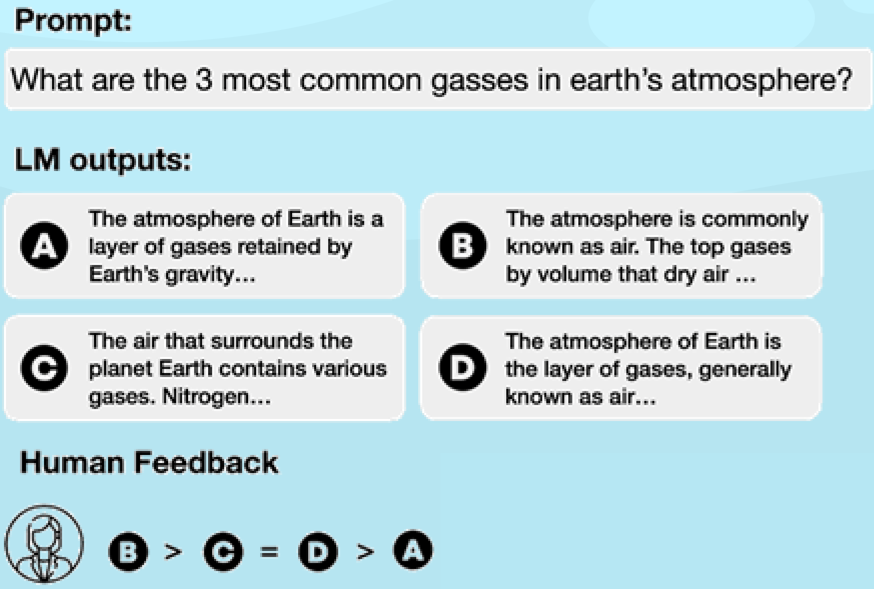
Human preference data is relatively cheap!!!
100k turns ~= $2m
$20 per turn, 10+x less than SFT

<u>Feedback learning is much more scalable than SFT!</u>

Nguyen et al 2017, Martin* & Ammanabrolu* et al 2018, Ziegler et al 2019

RLHF – Phase 1 Exploration

Step 1: Collect preference feedback and train a reward model



Humans label which they prefer – reasoning chain, style of answer etc.

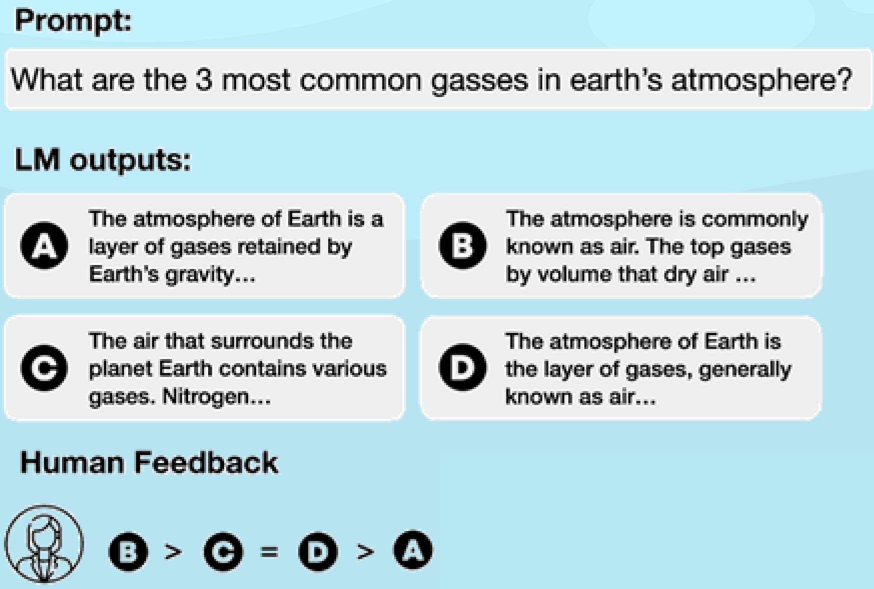
Human preference data is relatively cheap!!!
100k turns ~= $2m
$20 per turn, 10+x less than SFT

Even then, can't ask a human every single time to label!

Nguyen et al 2017, Martin* & Ammanabrolu* et al 2018, Ziegler et al 2019

RLHF – Phase 1 Exploration

Step 1: Collect preference feedback and train a reward model



Human preference data is relatively cheap!!!
100k turns ~= $2m
$20 per turn, 10+x less than SFT

Can't ask a human every single time to label!

Llama 2 spent $25m+ (1.4m samples)
GPT 4, Claude 3, Llama 3 all have O($100m) data spends.

LLM Pipeline: then Reinforcement Learning

“I loved the book but really hated the movie”

Awful, awful, awful. Here, I’ll tell you why …

At first anyway, but I warmed slowly as I watched

Collect Expert Demos

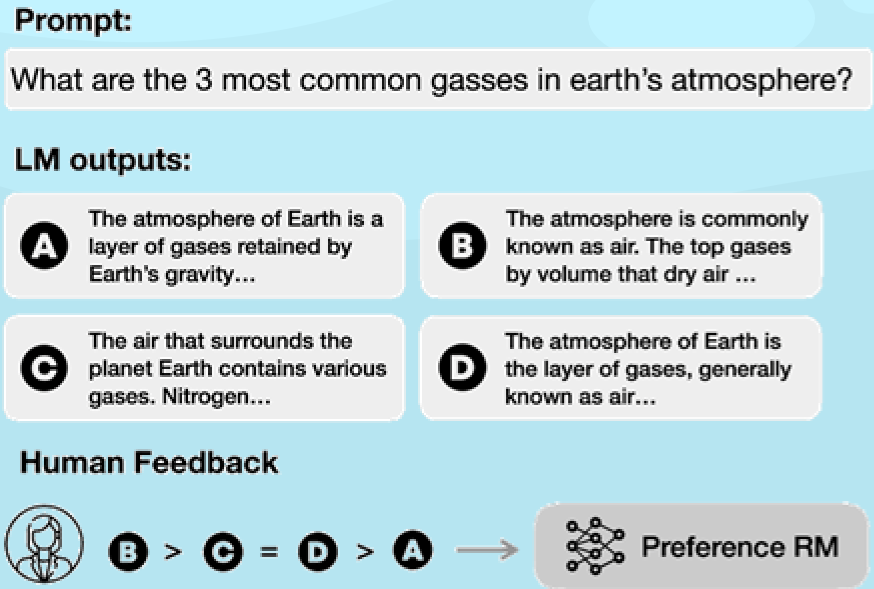
Train model with Supervised Learning (Behavior Cloning)

Collect Preference Feedback

Train Human Proxy Reward Function

RLHF – Phase 1 Exploration

Step 1: Collect preference feedback and train a reward model



Reason 1 why we need Pre-training+SFT. The outputs of the initial model need to already be somewhat reasonable for humans to provide effective feedback.

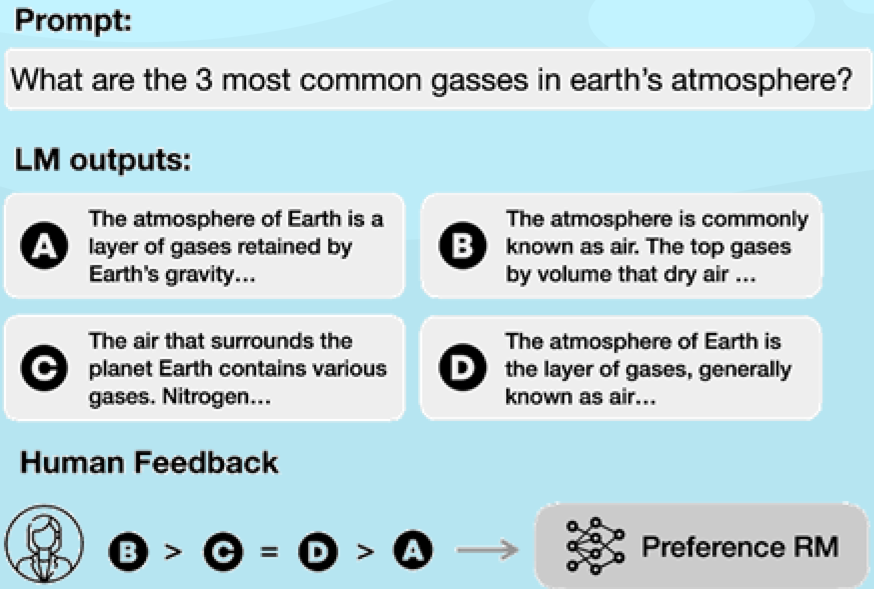
Humans label which they prefer – reasoning chain, style of answer etc.

Train a new metric, a reward function: Human judgment proxy.

Nguyen et al 2017, Martin* & Ammanabrolu* et al 2018, Ziegler et al 2019

RLHF – Phase 1 Exploration

Step 1: Collect preference feedback and train a reward model



Train a new metric, a reward function: Human judgment proxy.

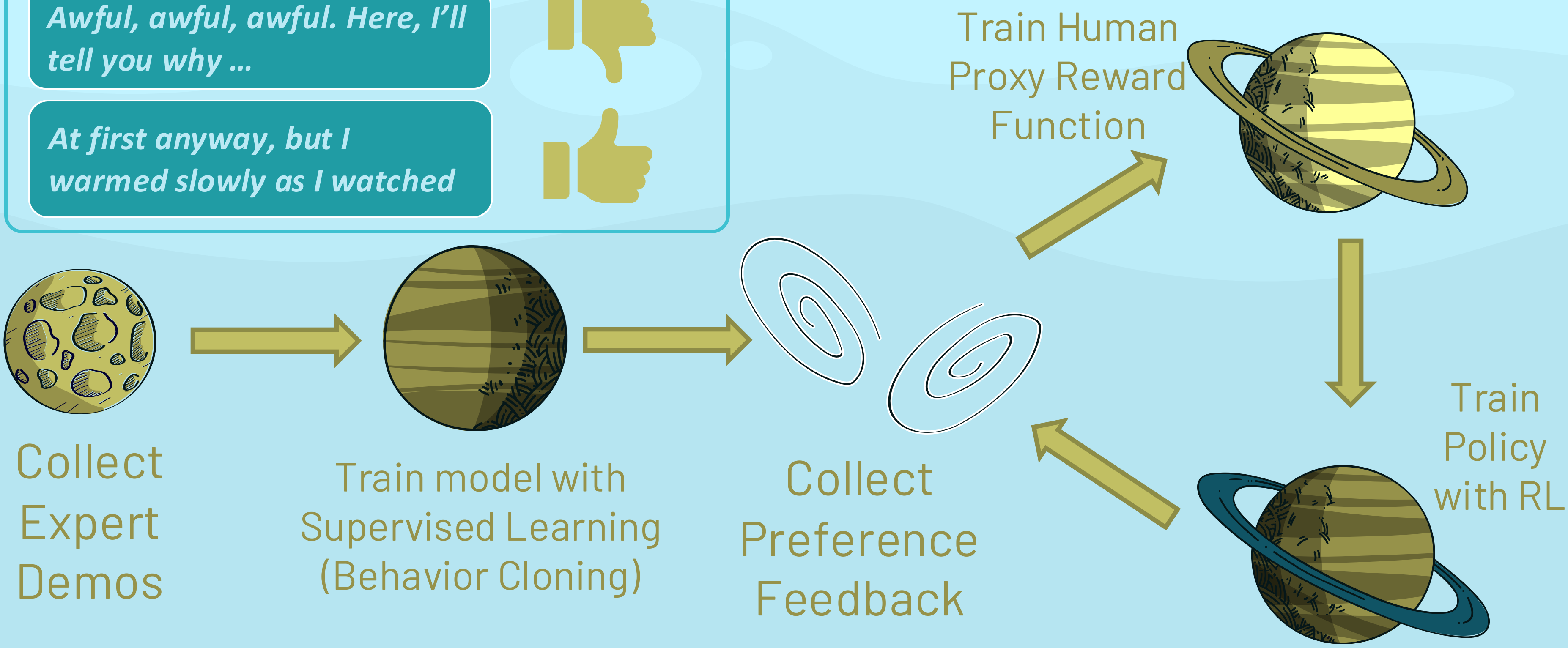
Trained via (variant of) a ranking loss.

LLM Pipeline: then Reinforcement Learning

"I loved the book but really hated the movie"

Awful, awful, awful. Here, I'll tell you why …

At first anyway, but I warmed slowly as I watched



Language Generation is a Token-level Markov Decision Process (MDP)

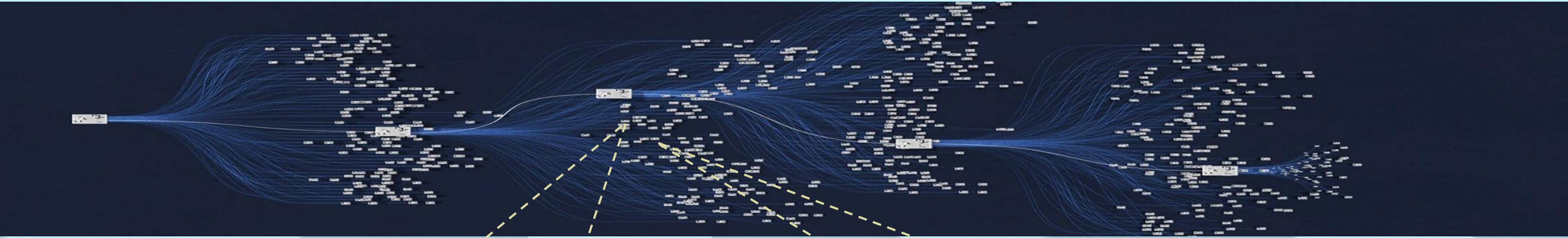
6-tuple of <S, A, T, R, γ, K>:

- S states = sentence so far
- A words = vocab
- T transition fn = append action A_t to S
- R reward function
- γ discount factor
- K max sentence length

Objective: Find policy π_θ: S → A to maximize long term expected rewards

$$\mathbb{E}_\pi[\sum_{t=0}^{K} \gamma^t R(\boldsymbol{s}_t, a_t)]$$

Step 1 Step 2 Step 3 Step 4 ...



Open	mailbox	with	colleague
Go	north	in	comma
Examine	house	on	magic
The	my	above	amazing
Shout	four	below	scrolls
Carry	shoulder	until	some
Show	movie	was	bronze
Mount	bottom	over	cyclops
Cross	box	under	
Shred	Bozbar		
Adjust			

√

Open	mailbox	a	colleague
Go	north	in	show
Examine	house	on	magical
The	me	above	man
It	four	below	scrolls
Carry	shoulder	until	some
Show	movie	from	bronze
Mount	was	quite	cyclops
Cross	box	under	
Shred	Bozbar		
Adjust			

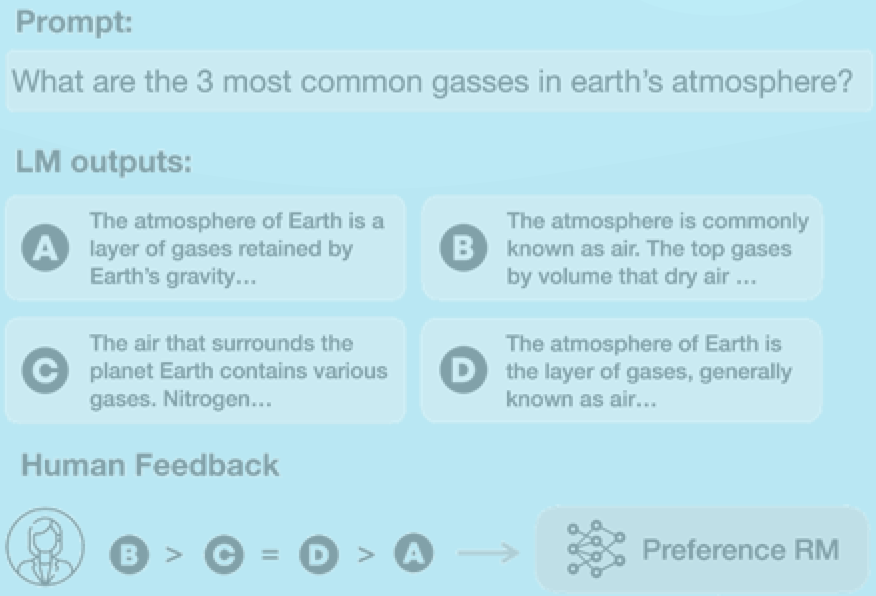
X

Imagine a controller with ~100000 buttons. How to scale RL?

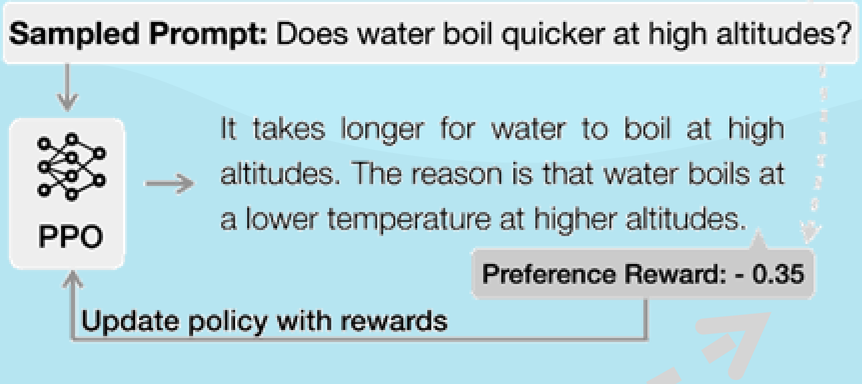
(Game of Go ~250, Chess ~35)

RLHF – Phase 2 Reward Optimization

Step 1: Collect preference feedback and train a reward model



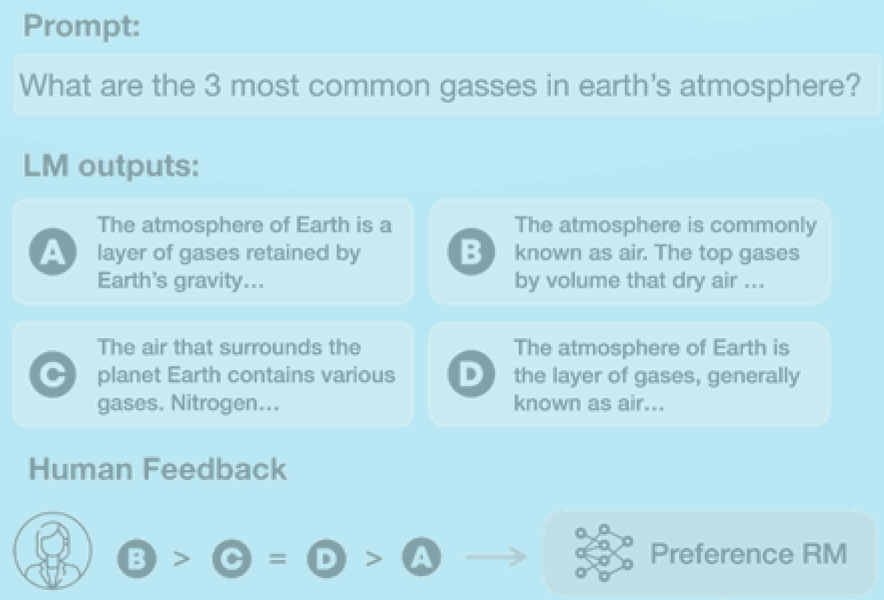
Step 2: Refine the policy LM against the reward model using RL. **i.e. filter the synthetic data according to some metric!**



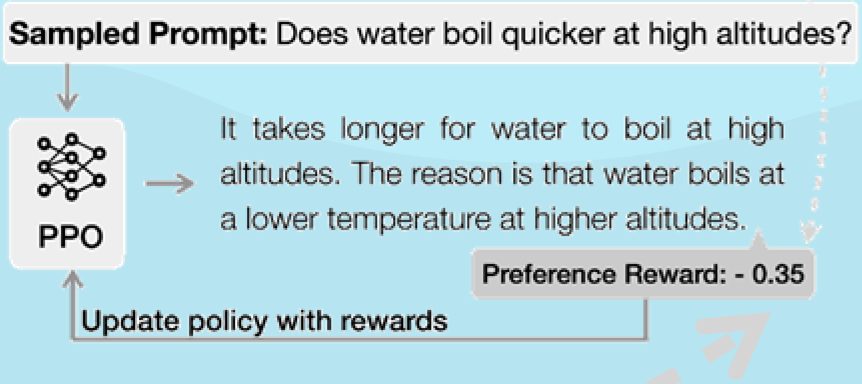
Nguyen et al 2017, Martin* & Ammanabrolu* et al 2017, Ziegler et al 2019

RLHF – Phase 2 Reward Optimization

Step 1: Collect preference feedback and train a reward model



Step 2: This form of exploration = "personalized learning". We are teaching the model to fix its specific mistakes



Nguyen et al 2017, Martin* & Ammanabrolu* et al 2017, Ziegler et al 2019

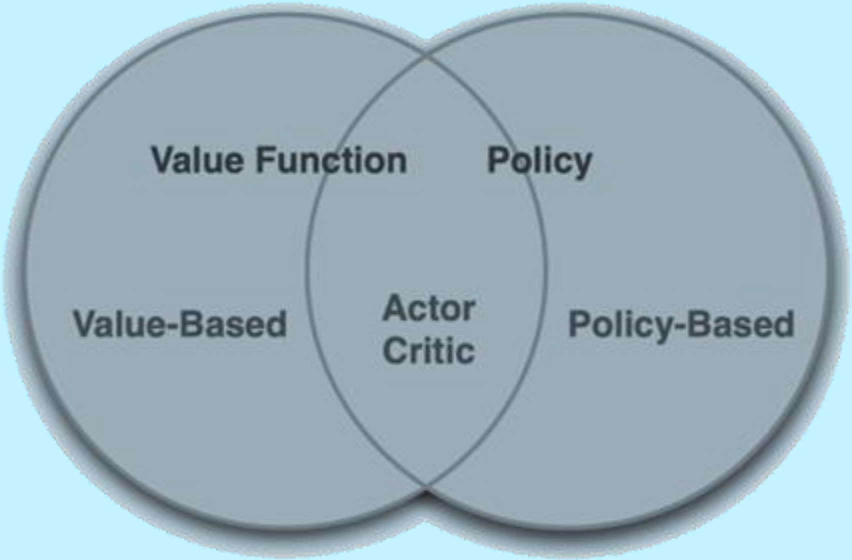
Holistic reward assignment during RL

$$r_t = R_\phi(x, y) \text{ if } t = T \text{ and } 0 \text{ otherwise}$$

A single reward model outputs a holistic reward for a prompt and LM output

Assign at the end of the LM output

Value and Policy Based RL



- Value Based
  - Learnt Value Function
  - Implicit policy (e.g. -greedy)
  - Values / rewards of partial sentences hard to judge
- Policy Based
  - No Value Function
  - Learnt Policy
- Actor-Critic
  - Learnt Value Function
  - Learnt Policy

Slide credits for Policy Gradient slides to David Silver. Deepmind.

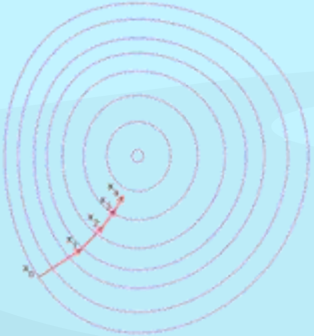
Advantages of Policy Based RL

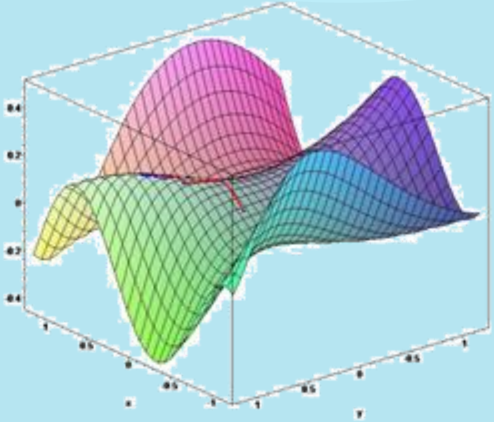
- Advantages:
  - Better convergence properties
  - Effective in high-dimensional or continuous action spaces Can learn stochastic policies
- Disadvantages:
  - Typically converge to a local rather than global optimum
  - Evaluating a policy is typically inefficient and high variance

Policy Gradients

- Goal: given policy $\pi_\theta(s, a)$ with parameters θ, find best θ that maximizes $J(\theta)$ – any policy objective
- Gradient Descent according to the Policy Gradient

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix}$$





One Step MDPs
(aka Contextual Bandits)

- Consider a simple class of one-step MDPs
  - Starting in state s ~ d(s)
  - Terminating after one time-step with reward $r = R_{s,a}$
- Use likelihood ratios to compute the policy gradient

$$
\begin{aligned}
J(\theta) &= \mathbb{E}_{\pi_\theta}[r] \\
&= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(s,a) \mathcal{R}_{s,a} \\
\nabla_\theta J(\theta) &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(s,a) \nabla_\theta \log \pi_\theta(s,a) \mathcal{R}_{s,a} \\
&= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) r]
\end{aligned}
$$

One Step MDPs
(aka Contextual Bandits)

- One step MDP in language = generate the entire sequence and consider that a singular action
- Equivalent to the step wise MDP with many tokens but just set discount to 1

$$
\begin{aligned}
J(\theta) &= \mathbb{E}_{\pi_\theta}[r] \\
&= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(s,a)\mathcal{R}_{s,a} \\
\nabla_\theta J(\theta) &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)\mathcal{R}_{s,a} \\
&= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) r]
\end{aligned}
$$

Policy Gradient Theorem

- The policy gradient theorem generalizes the likelihood ratio approach to multi-step MDPs
- Replaces instantaneous reward r with long-term value $Q^{\pi}(s, a)$
- Policy gradient theorem applies to start state objective, average reward and average value objective

Theorem

For any differentiable policy $\pi_\theta(s, a)$,
for any of the policy objective functions $J = J_1, J_{avR}$, or $\frac{1}{1-\gamma} J_{avV}$,
the policy gradient is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(s, a) \; Q^{\pi_\theta}(s, a) \right]$$

Monte Carlo Policy Gradient (REINFORCE)

- Update parameters by stochastic gradient ascent
- Using policy gradient theorem
- Using return v_t as an unbiased sample of $Q^{\pi}(s_t, a_t)$

function REINFORCE

Initialise θ arbitrarily

for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**

for $t = 1$ to $T - 1$ **do**

$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$

end for

end for

return θ

end function

REINFORCE in NLP

- Pre-2018 ish, almost every single instance of “RL” in NLP was Monte Carlo Policy Gradient using 1-step MDP formulation of Language
- This can be made much better with more granular formulations as we will see more later.

Variance Reduction with a Critic

- Monte-Carlo policy gradient still has high variance
- We use a critic to estimate the action-value function

$$Q_w(s, a) \approx Q^{\pi_\theta}(s, a)$$

- Actor-critic algorithms maintain two sets of parameters
  - Critic Updates action-value function parameters w
  - Actor Updates policy parameters θ, in direction suggested by critic
- Actor-critic algorithms follow an approximate policy gradient

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s, a)\ Q_w(s, a)\right]$$
$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s, a)\ Q_w(s, a)$$

Variance Reduction with a Baseline

- We subtract a baseline function B(s) from the policy gradient
- This can reduce variance, without changing expectation

$$\begin{aligned}\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)B(s)\right] &= \sum_{s\in\mathcal{S}} d^{\pi_\theta}(s)\sum_a \nabla_\theta \pi_\theta(s,a)B(s)\\ &= \sum_{s\in\mathcal{S}} d^{\pi_\theta}B(s)\nabla_\theta \sum_{a\in\mathcal{A}} \pi_\theta(s,a)\\ &= 0\end{aligned}$$

Estimating Advantage

- A good baseline is the state value function $B(s) = V^{\pi\theta}(s)$
- So we can rewrite the policy gradient using the advantage function
- Advantage = how much better it is to take a specific action compared to the average action in that particular state

$$A^{\pi_\theta}(s, a) = Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s, a)\ A^{\pi_\theta}(s, a)\right]$$

Policy Gradient Summary

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ v_t\right] && \text{REINFORCE} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ Q^w(s,a)\right] && \text{Q Actor-Critic} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ A^w(s,a)\right] && \text{Advantage Actor-Critic}
\end{aligned}
$$

(Generic) Actor Critic Algorithm for Natural Language Alignment

- Value: estimate of future rewards in given state
- Q-value: utility of performing an action in current state
- Advantage: value of performing action over average action

$$V_t^{\pi} = \mathbb{E}_{a_t \sim \pi}[\sum_{\tau=t}^{K} \gamma R(\boldsymbol{s}_\tau, a_\tau)]$$

$$Q_t^{\pi}(\boldsymbol{s}_t, a_t) = R(\boldsymbol{s}_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim P}[V_{t+1}^{\pi}(\boldsymbol{s}_{t+1})]$$

$$A_t^{\pi}(\boldsymbol{s}, a) = Q_t^{\pi}(\boldsymbol{s}, a) - V_t^{\pi}$$

How to explore Natural Language with RL

Supervised for a bit, then "approx. trust region RL"

Ramamurthy*, Ammanabrolu*, Brantley, Hessel, Sifa, Bauckhage, Hajishirzi, Choi. *Is RL (Not) for NLP?: Benchmarks, Baselines, and Building Blocks for Natural Language Policy Optimization.* ICLR 2023.