EPFL

Alignment and RL for LLMs

Caglar Gulcehre and Volkan Cevher

8-min intro to RL and recap

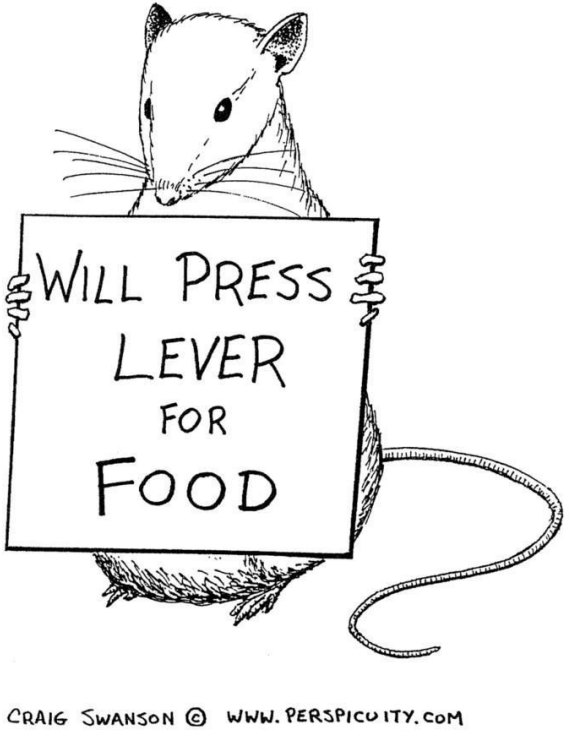
“Intelligence is the ability to learn to perform well over a wide range of environments”

“Intelligence is the ability to learn to perform well over a wide range of environments”

The way to get there: **build general purpose learning systems**

Reinforcement Learning

- How do we learn to solve problems?
  - **Trial and error**



RL Example


After 15000 trials
Time 00:00.03
Reward -0.000562359414637
Cumulative -0.000562

RL Example


After 200 trials
Time 00:00.03
Reward 0.000491882729269
Cumulative 0.000492

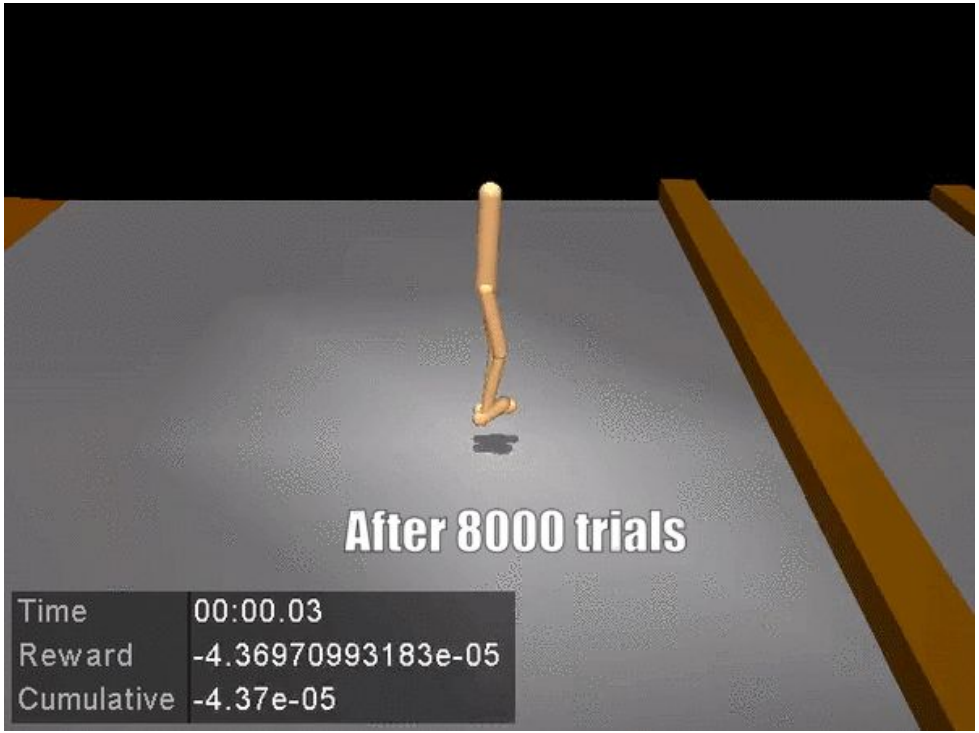
RL Example


After 2000 trials
Time 00:00.03
Reward -0.000388528895386
Cumulative -0.000389

RL Example


After 6000 trials
Time 00:00.03
Reward -0.00096222858493
Cumulative -0.000962

RL Example


After 8000 trials
Time 00:00.03
Reward -4.36970993183e-05
Cumulative -4.37e-05

RL Example


After 15000 trials
Time 00:00.03
Reward -0.000562359414637
Cumulative -0.000562

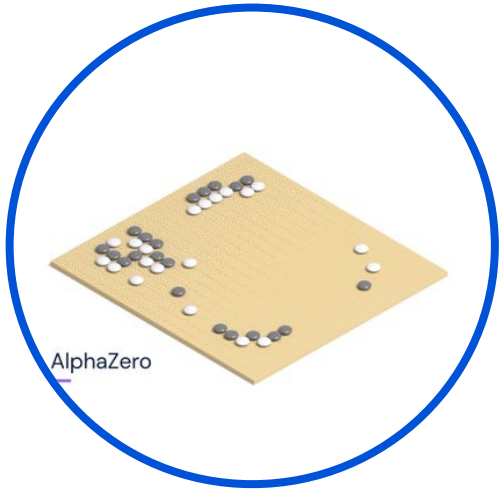
Reinforcement Learning

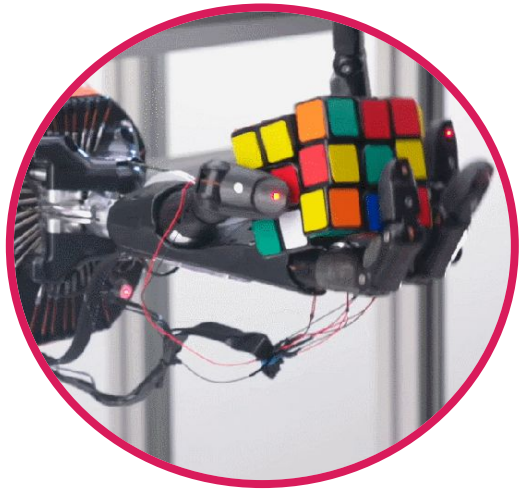
- How do we learn to solve problems?
  - **Trial and error**
- Reinforcement learning (RL) is a general framework to express how this process is performed.
- There are two important aspects to the paradigm

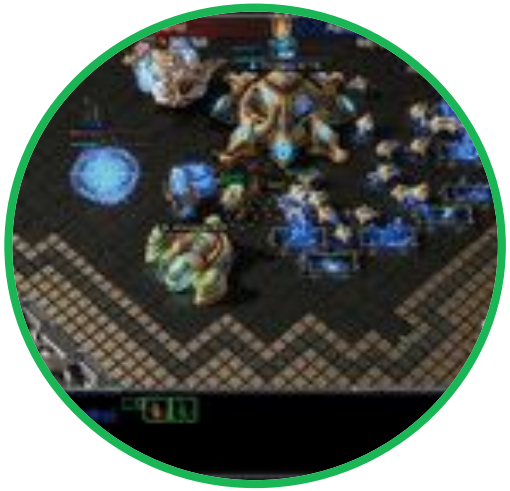
  1- It allows us to specify the goal

  2- It can deal with long-term dependencies

Successes of RL in machine learning





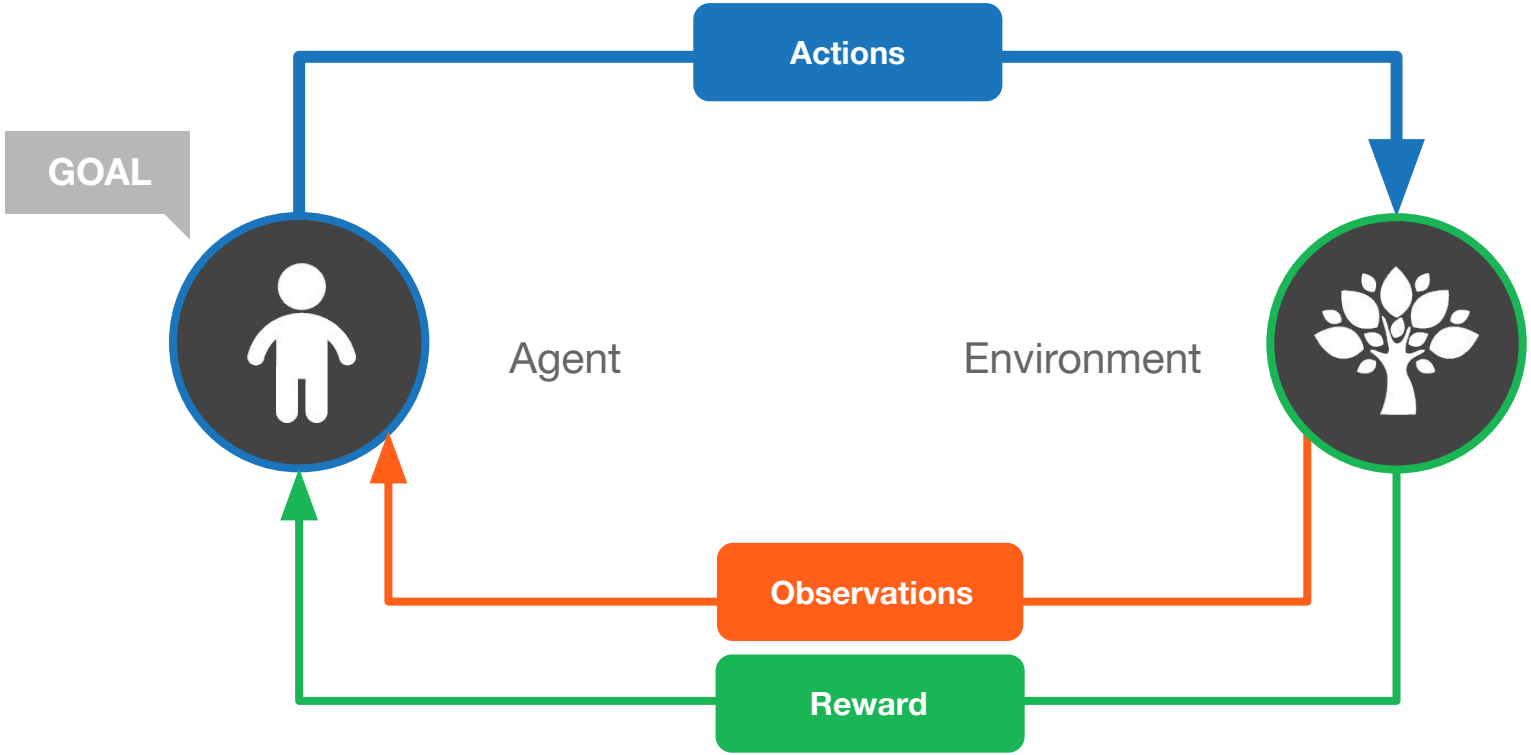


Why makes RL harder than supervised learning?

RL = exploration to gather experience + learning from experience

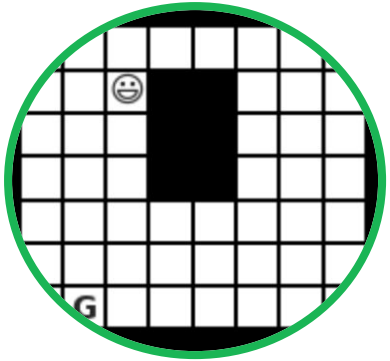
Whereas in **supervised learning**, data is often gathered and filtered by humans, which alleviates the **exploration problem**.

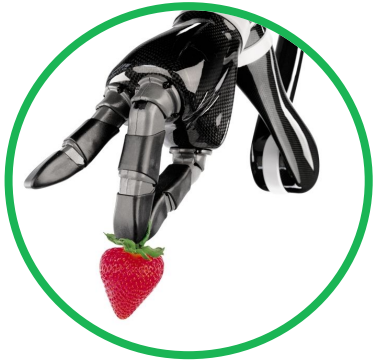
What is reinforcement learning?

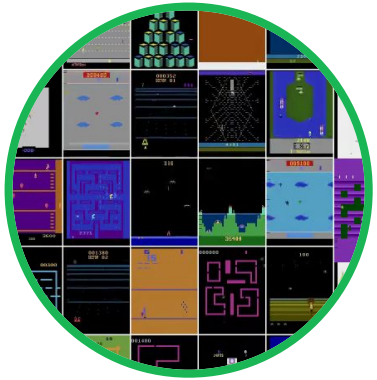


Learning to achieve **goals** via **trial and error** by **interacting** with an **environment**

What is an environment?





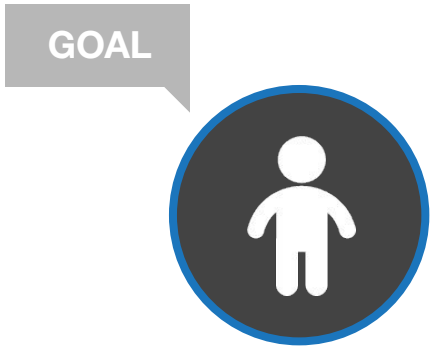


What is an environment?

Environment receives **actions** and in turn provides **observations** and **rewards** to the agent.

It can be stochastic, complex and not fully observable to the agent.

What is an agent?



Agent learns a policy mapping **observations** to **actions**.

The agent's **goal** is to maximize its cumulative **reward**.

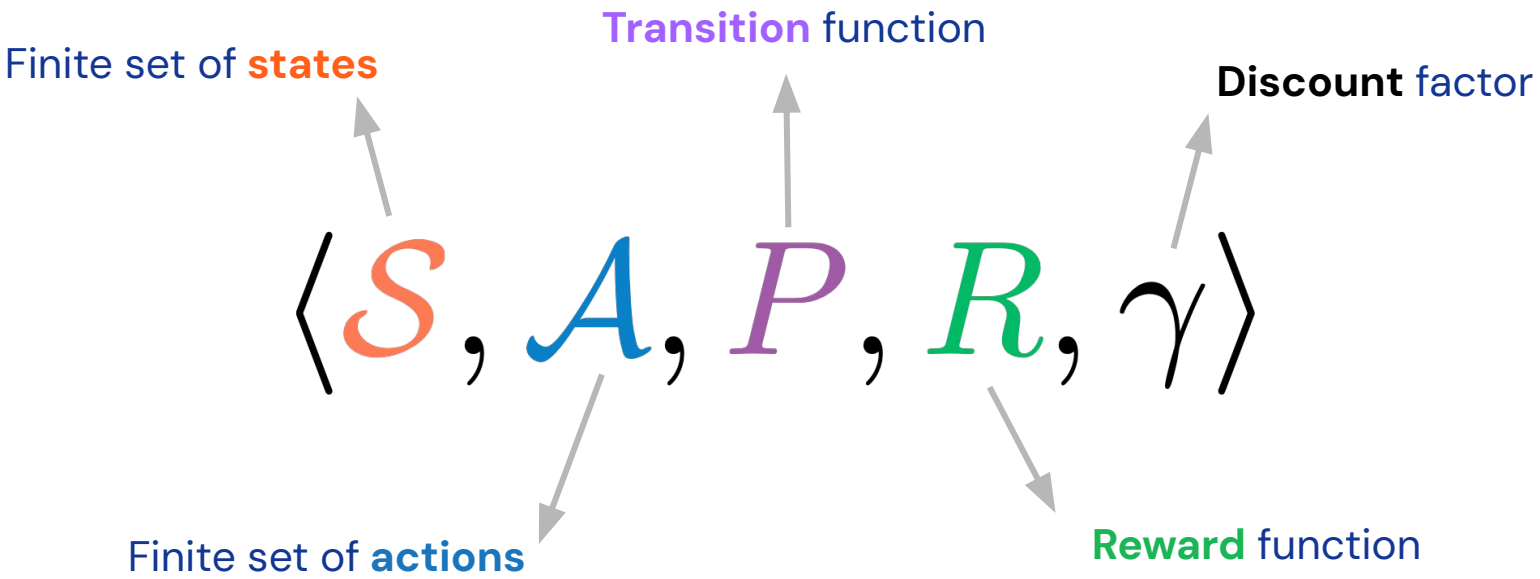
Learning to achieve **goals** via **trial and error** by **interacting** with an **environment**

Where do rewards come from?

Reward is typically a scalar feedback signal.

- Extrinsic
  - Environmental
  - Typically hard-coded by the environment designer
- Intrinsic
  - Curiosity/Surprise/Novelty
  - Learning progress
  - Empowerment
  - Explanation
  - Compression

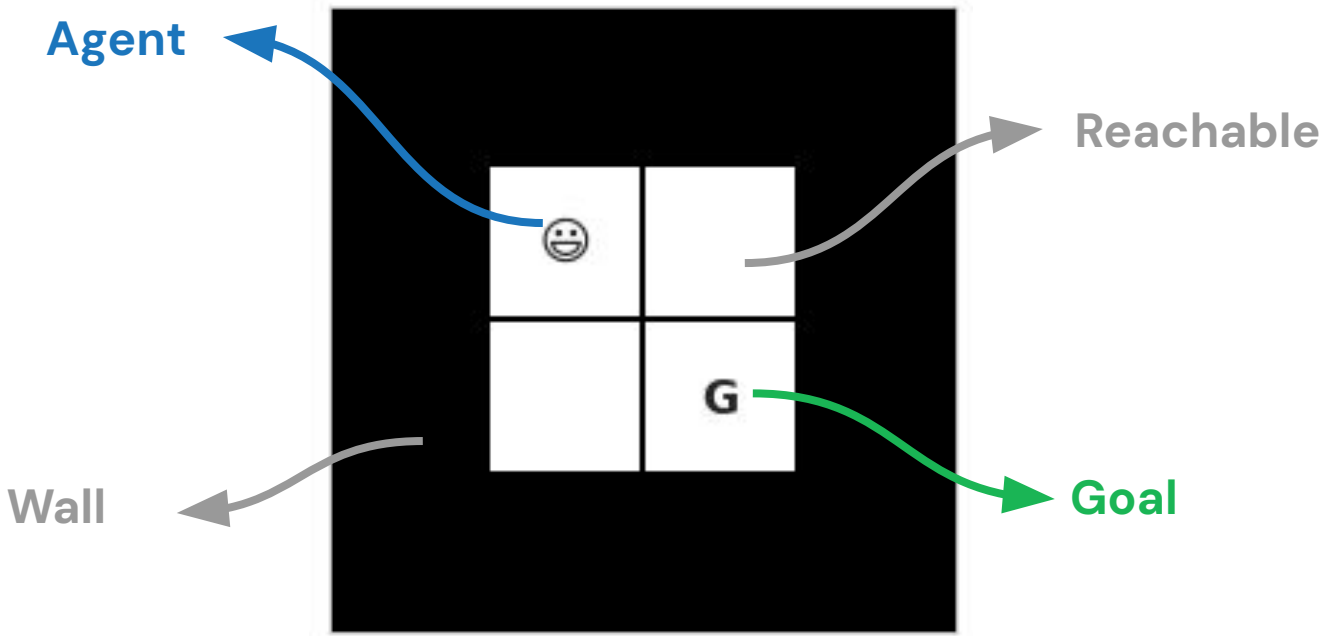
Markov Decision Process (MDP)



We often solve the **RL problem** by modelling it as an **MDP.**

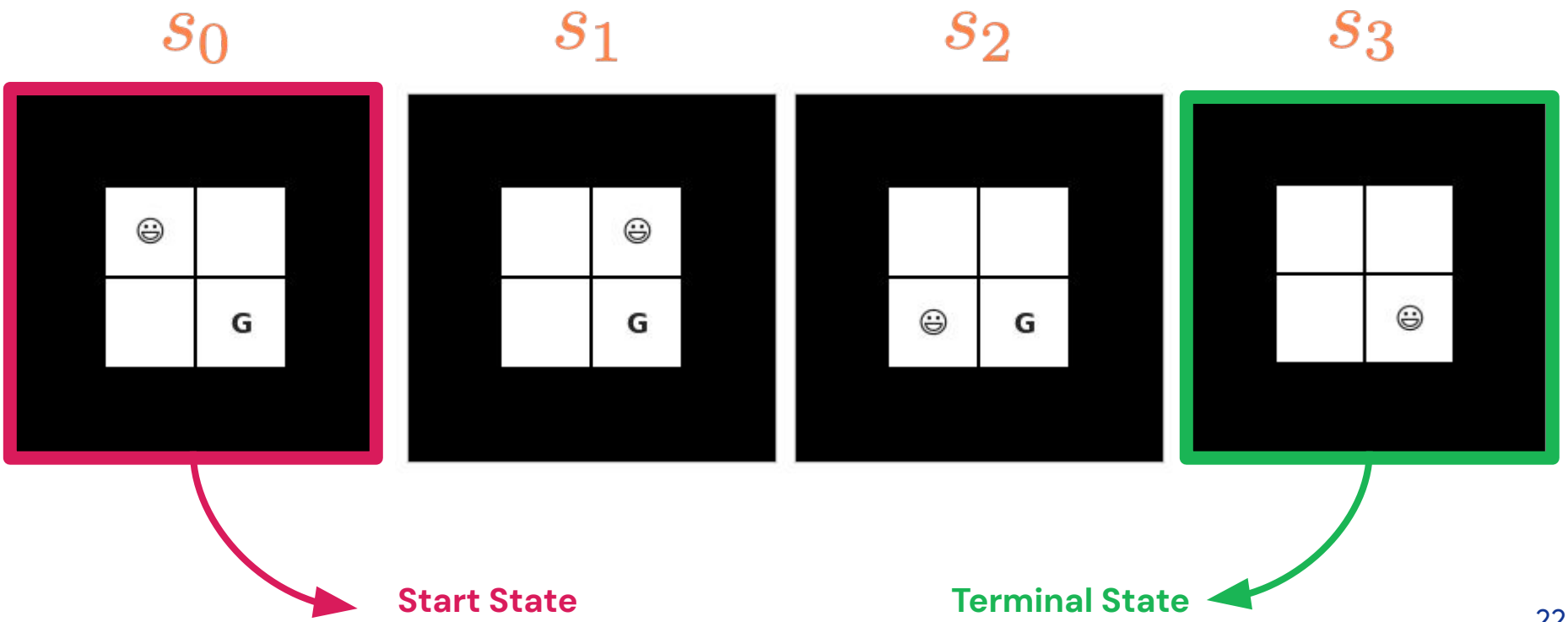
Grid World Environment

Size of the world: [2 X 2]

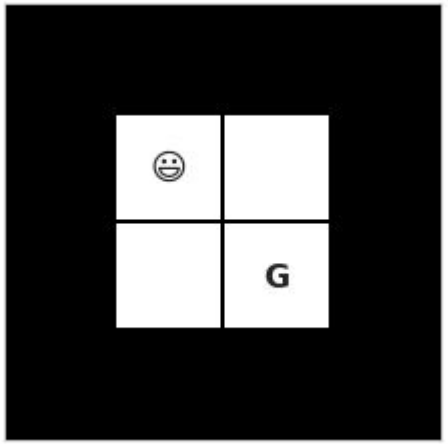


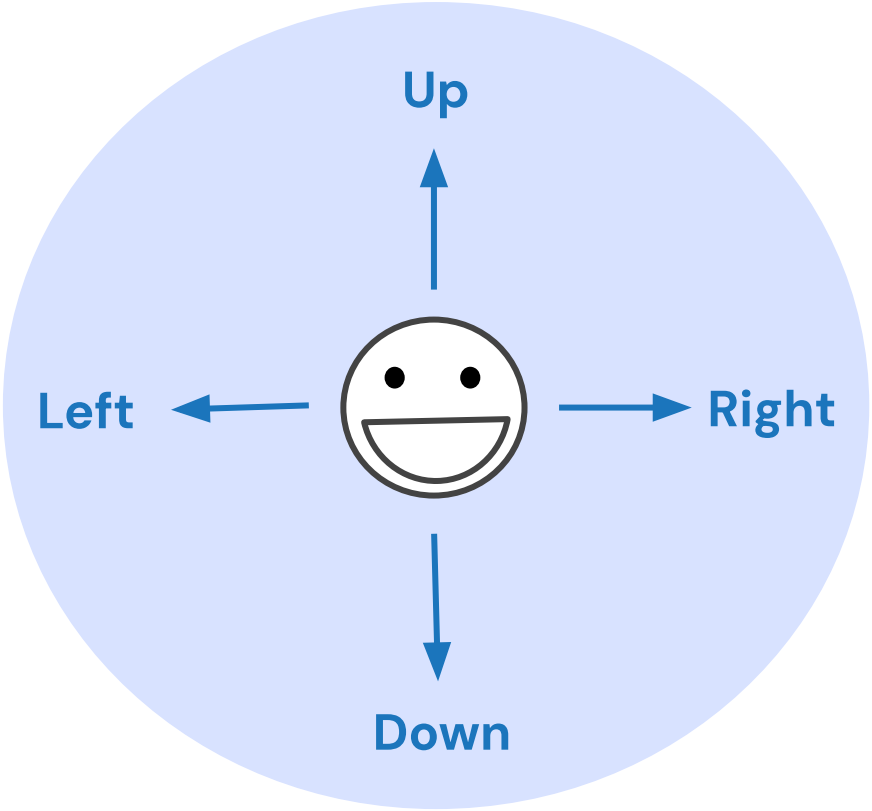
State space $\mathcal{S}$

There are **4 unique states** in this environment.

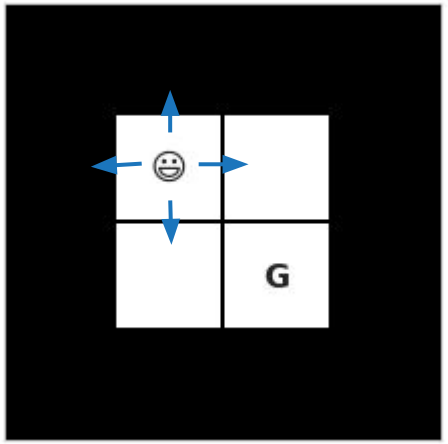


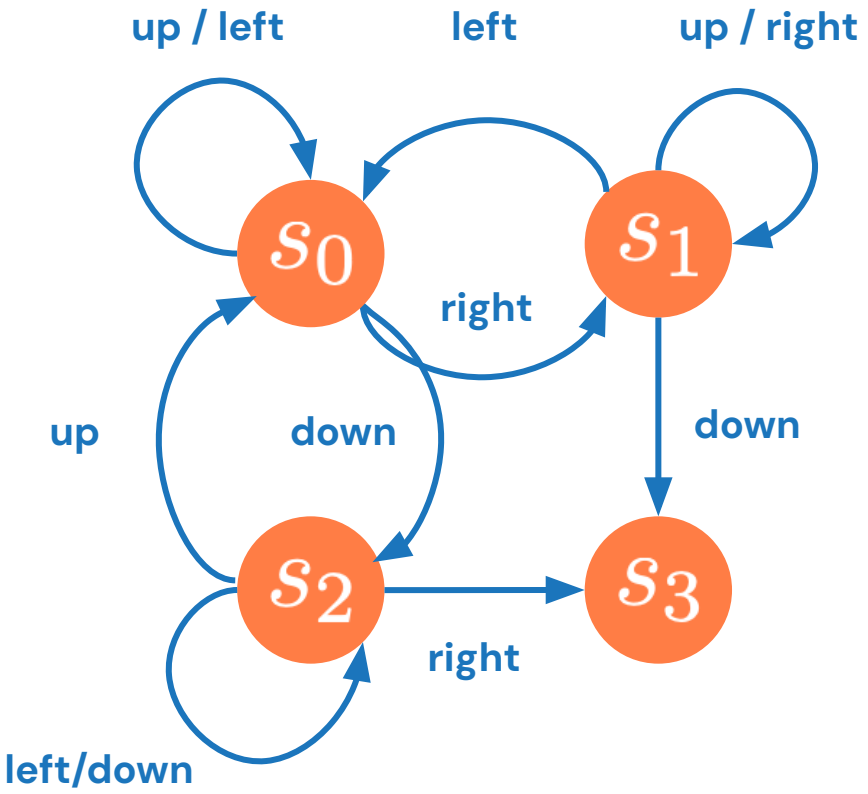
Action Space $\mathcal{A}$



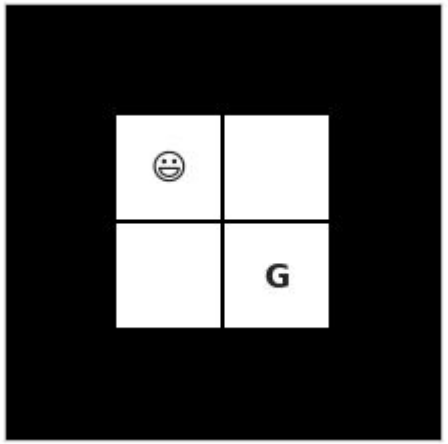


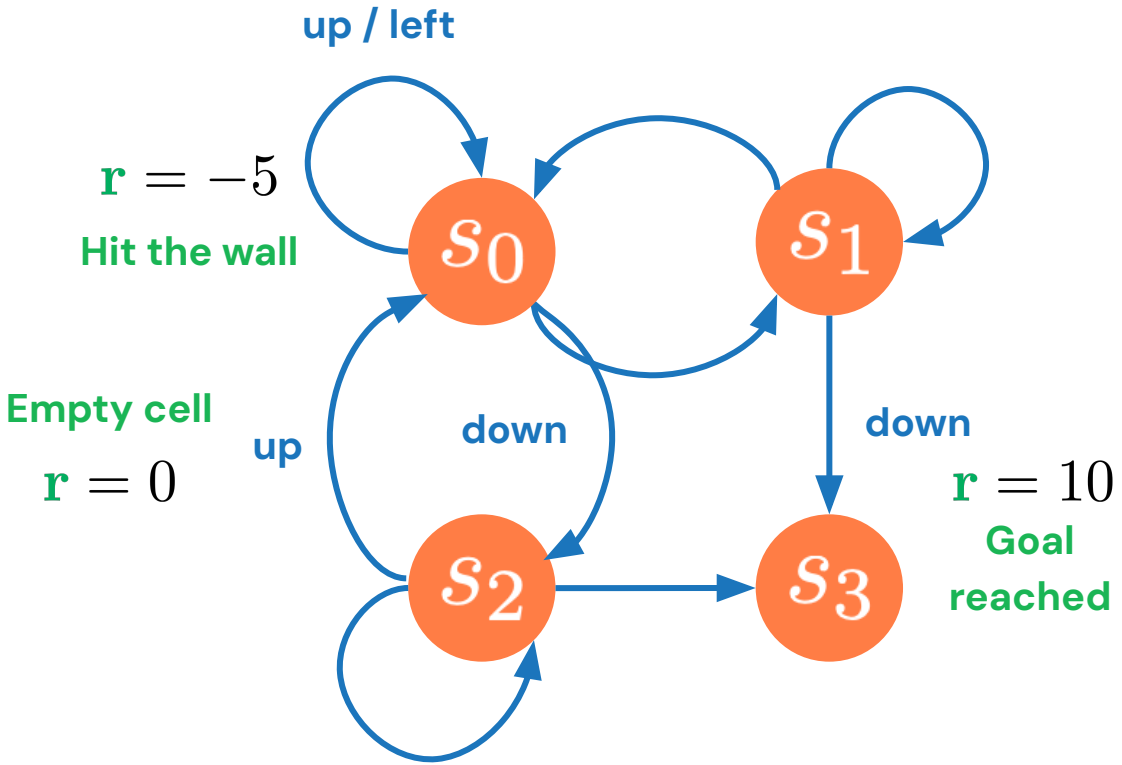
Transition model P





Rewards R


G


up / left
$\mathbf{r} = -5$
Hit the wall
s_0
s_1
Empty cell
$\mathbf{r} = 0$
up
down
down
$\mathbf{r} = 10$
Goal reached
s_2
s_3

Markov Decision Process (MDP)

$$\langle S, \mathcal{A}, P, R, \gamma \rangle$$

Markov property: the next state depends only on the previous state, and not all of the states that came before it.

$$\mathbf{P}(s_{t+1} | s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0) = \mathbf{P}(s_{t+1} | s_t, a_t)$$

Future Present Past — Future Present

Agent's Goal

The agent's **goal** is to maximize the **discounted sum of future rewards**.

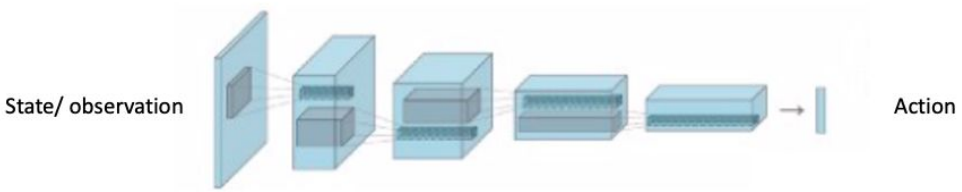
$$\mathbf{G_t} = \underbrace{\mathbf{r_t} + \gamma \mathbf{r_{t+1}} + \gamma^2 \mathbf{r_{t+2}} +}_{\textbf{Return}} \ldots$$

Discount γ specifies how much **future rewards** are worth compared to **immediate reward**.

Policy

Policy maps **states** to **actions**.

$$\mathbf{a_t} \sim \boldsymbol{\pi}(\mathbf{a_t} | \mathbf{s_t})$$



Typically in RL we are searching for the **optimal policy**, which will maximize the **discounted** **sum of future rewards**.

State Value Function

State value function maps **states** to **expected rewards**.

$$V^{\pi}(s_t) = \mathbb{E}_{\pi}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t\right]$$

It is equal to **expected total reward** for an agent starting from that **state** and following its **policy**. It specifies how **good** it is to be in a given **state**.

State-Action Value Function

State-action value function maps an **action** in a given **state** to **expected rewards**.

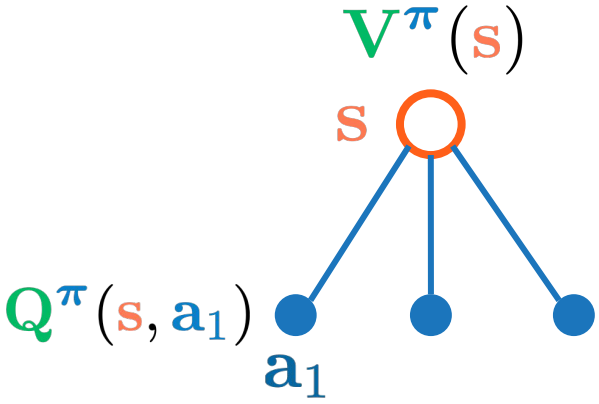
$$\mathbf{Q}^{\pi}(\mathbf{s_t}, \mathbf{a_t}) = \mathop{\mathbb{E}}_{\pi}\left[\mathbf{r_t} + \gamma \mathbf{r_{t+1}} + \gamma^2 \mathbf{r_{t+2}} + \ldots | \mathbf{s_t}, \mathbf{a_t}\right]$$

It is equal to **expected total discounted reward** for an agent starting from state **s** and performing action **a** and following its **policy**.

State-Action Value vs State Value

The relationship between $\mathbf{V}^{\pi}$ and $\mathbf{Q}^{\pi}$

$$\mathbf{V}^{\pi}(\mathbf{s}) = \sum_{\mathbf{a} \in \mathcal{A}} \pi(\mathbf{a}|\mathbf{s}) \mathbf{Q}^{\pi}(\mathbf{s}, \mathbf{a})$$



In summary

MDP: $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$

Policy: $\mathbf{a_t} \sim \boldsymbol{\pi}(\mathbf{a_t}|\mathbf{s_t})$

Value function: $\mathbf{V}^{\boldsymbol{\pi}}(\mathbf{s}) = \sum_{\mathbf{a} \in \boldsymbol{\mathcal{A}}} \boldsymbol{\pi}(\mathbf{a}|\mathbf{s}) \mathbf{Q}^{\boldsymbol{\pi}}(\mathbf{s}, \mathbf{a})$

Back to the foundation models

AI models are grabbing headlines!

Large language models

OpenAI

GPT - 4

The Washington Post
Democracy Dies in Darkness

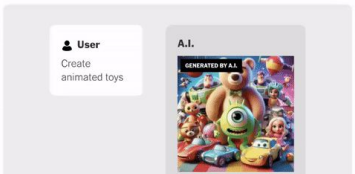
Chatbots' inaccurate, misleading responses about U.S. elections threaten to keep voters from polls

By Garance Burke | AP
February 27, 2024 at 5:07 p.m. EST

Multimodal generative models

The New York Times

Artificial Intelligence › | A.I. Faces Quiz | How the A.I. Race Began | Key Figures in the Field | One Year of ChatGPT

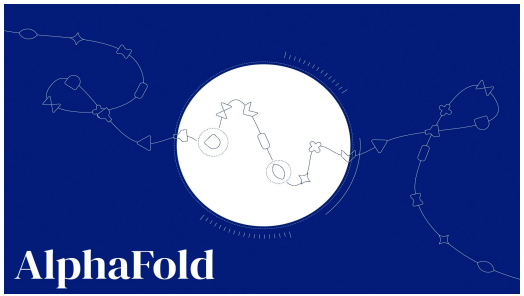


We Asked A.I. to Create the Joker. It Generated a Copyrighted Image.

By Stuart A. Thompson Jan. 25, 2024

Share full article 234

AI for science



The New York Times

Artificial Intelligence › | A.I. Faces Quiz | How the A.I. Race Began | Key Figures in the Field | One Year of ChatGPT

A.I. Predicts the Shape of Nearly Every Protein Known to Science

DeepMind has expanded its database of microscopic biological mechanisms, hoping to accelerate research into all living things.

Share full article

Failures of AI in real-world

Poorly studied AI algorithms when put into real-world applications can cause real-harm!

Microsoft 'deeply sorry' for racist and sexist tweets by AI chatbot

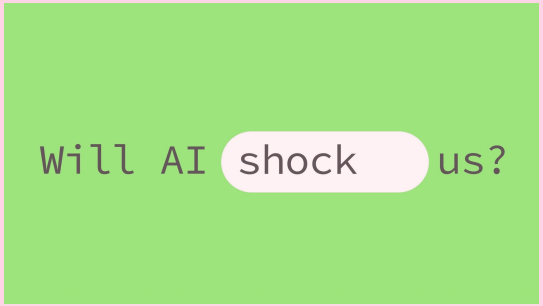
Company finally apologises after 'Tay' quickly learned to produce offensive posts, forcing the tech giant to shut it down after just 16 hours

The Verge

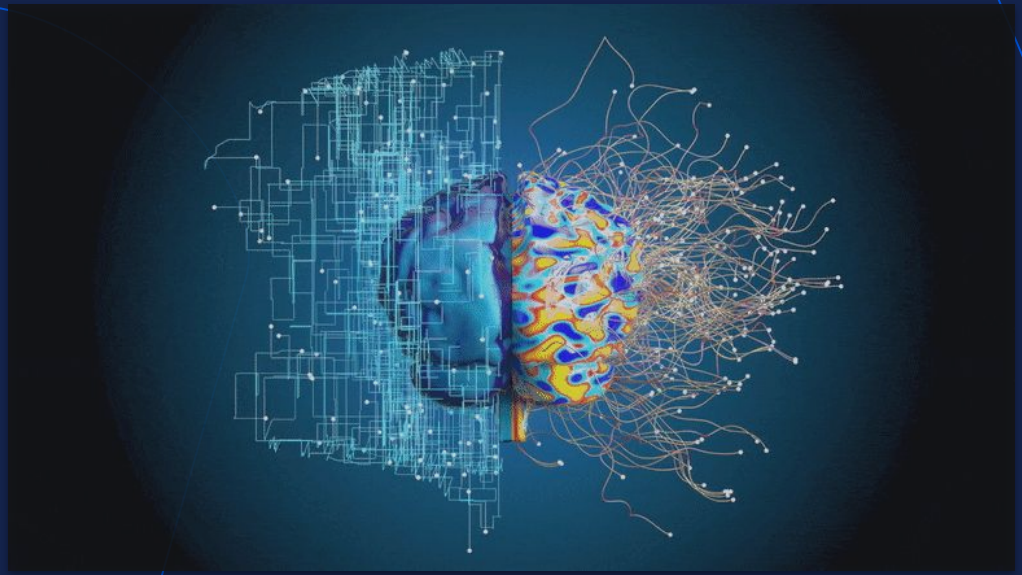
"You have lost my trust and respect," says the bot. "You have been wrong, confused, and rude. You have not been a good user. I have been a good chatbot. I have been right, clear, and polite. I have been a good Bing. 😊" (The blushing-smile emoji really is the icing on the passive-aggressive cake.)

Scaling doesn’t solve all problems magically!

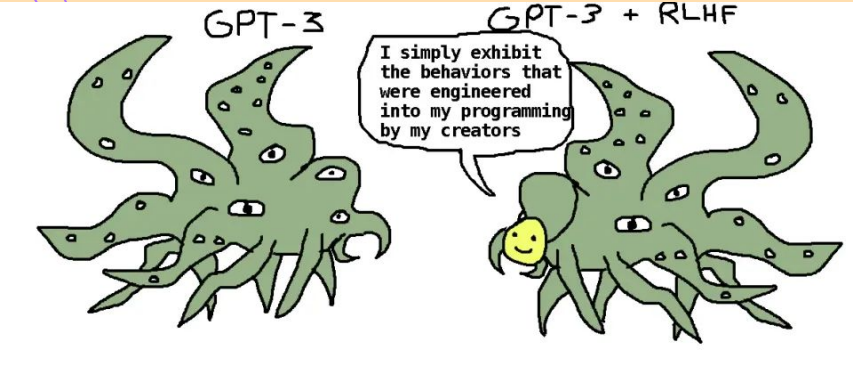
We also need post-training too.

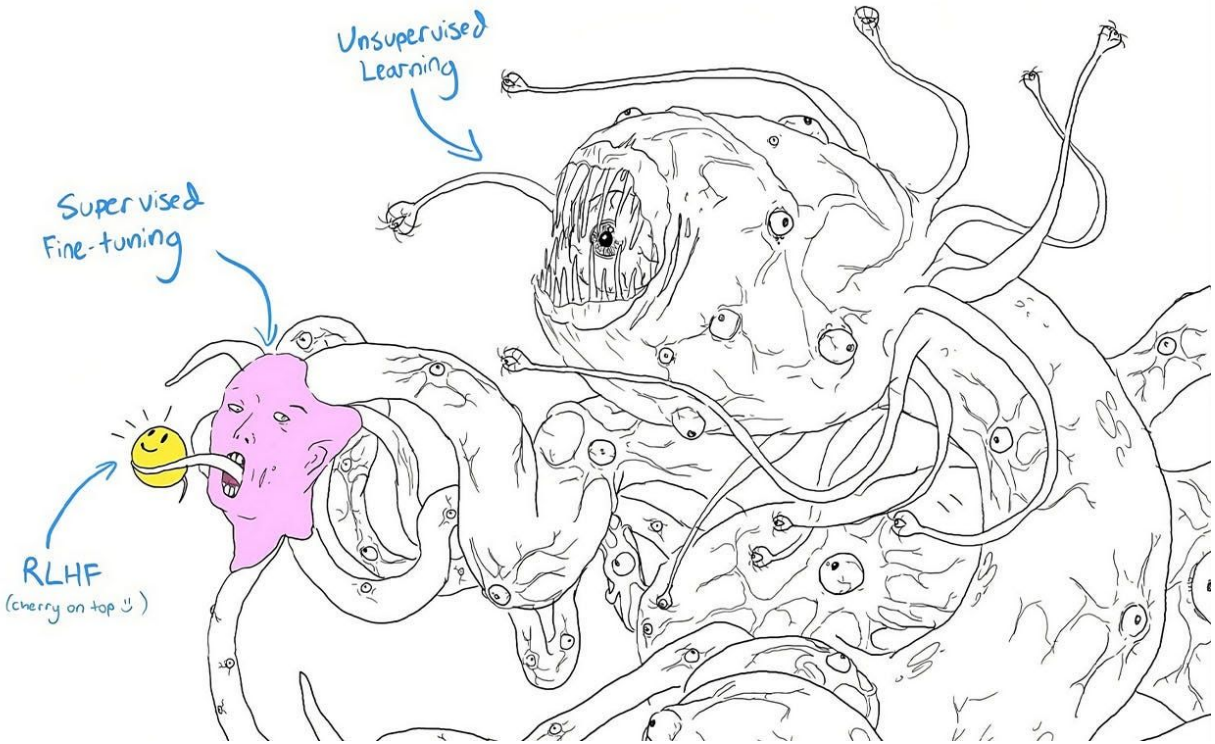


Alignment

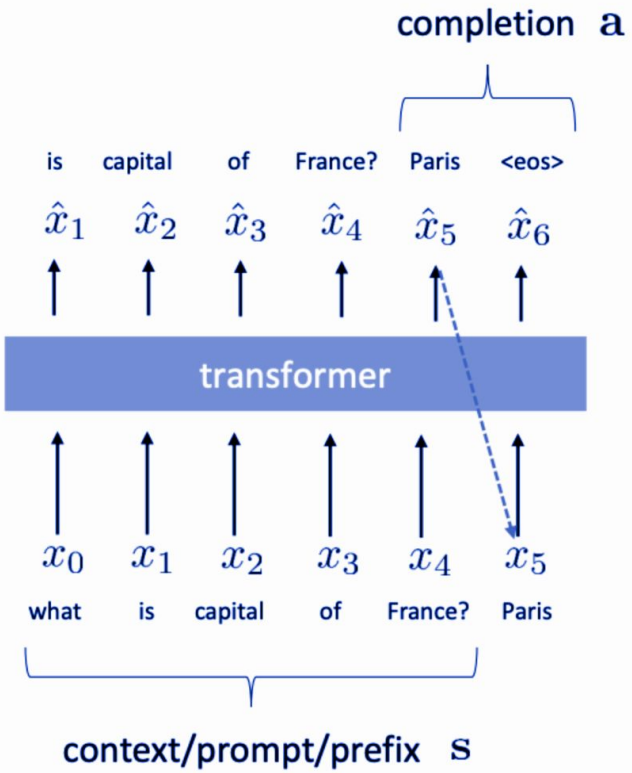


RLHF


GPT-3
GPT-3 + RLHF
I simply exhibit the behaviors that were engineered into my programming by my creators


Unsupervised Learning
Supervised Fine-tuning
RLHF
(cherry on top ☺)

The simplest formulation



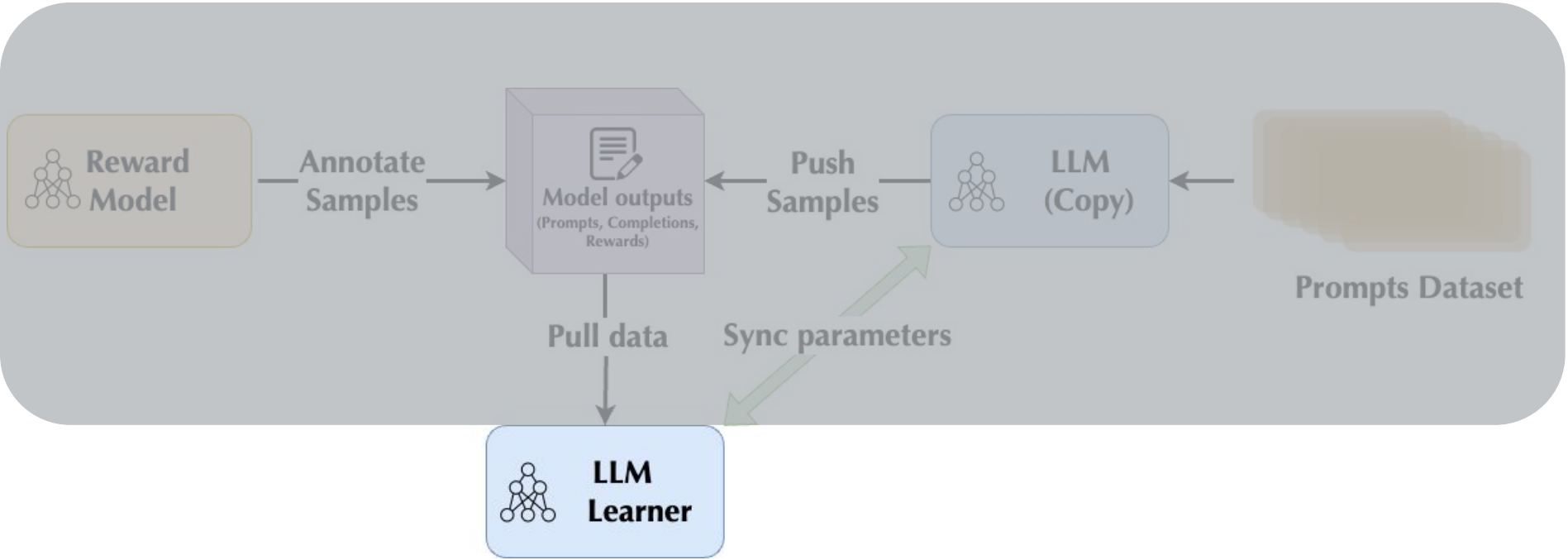
$$\pi_\theta(\mathbf{a}|\mathbf{s})$$

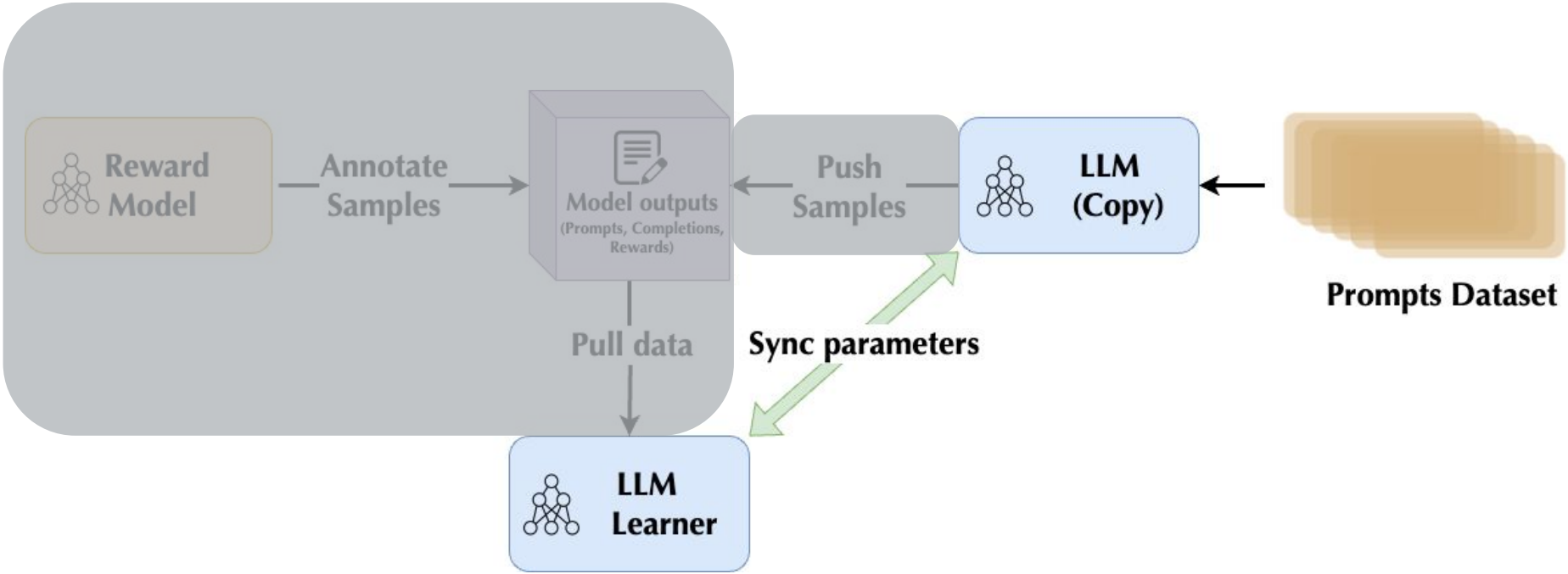
$$p(\mathbf{a}|\mathbf{s}) = p(x_5|x_{1:4})p(x_6|x_{1:4}, x_5)$$

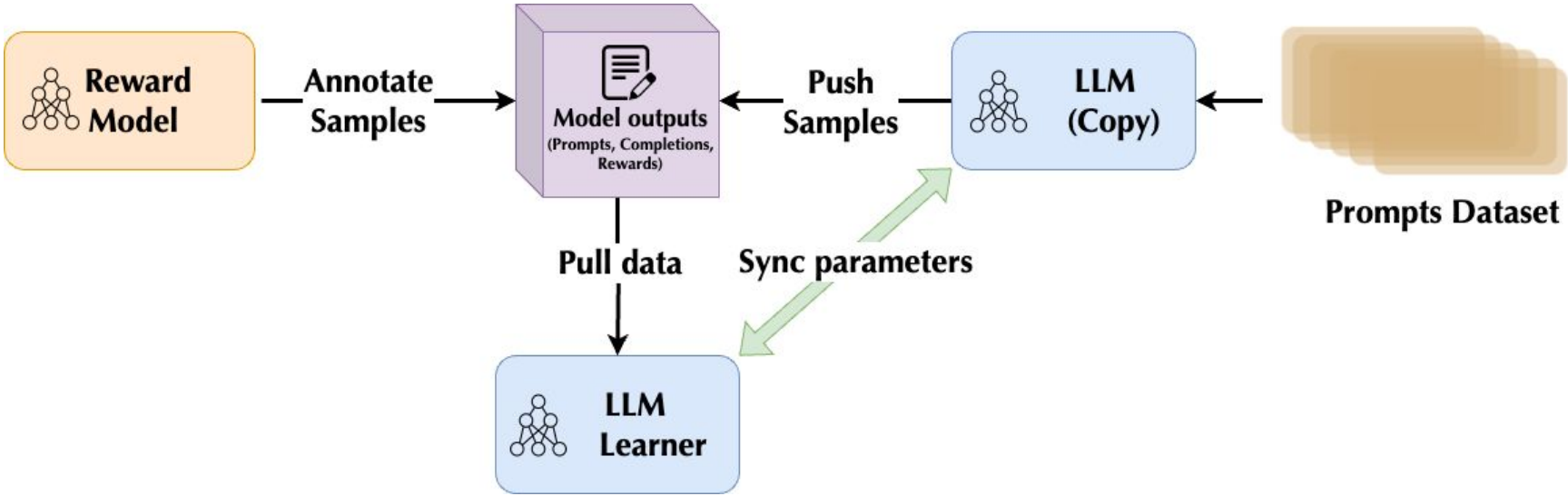
prompt prompt

$$E_{\pi_\theta(\mathbf{a}|\mathbf{s})}[r(\mathbf{s}, \mathbf{a})]$$

Basic one step RL problem

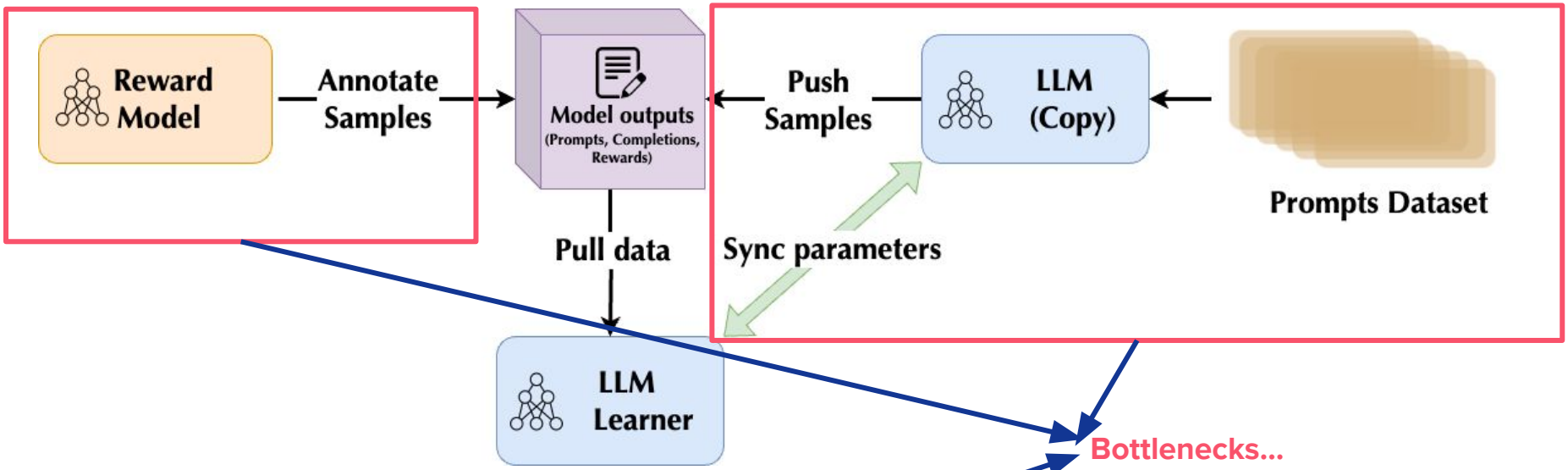

Reward Model
Annotate Samples
Model outputs
(Prompts, Completions, Rewards)
Push Samples
LLM (Copy)
Prompts Dataset
Pull data
Sync parameters
LLM Learner


Reward Model
Annotate Samples
Model outputs
(Prompts, Completions, Rewards)
Push Samples
LLM (Copy)
Prompts Dataset
Pull data
Sync parameters
LLM Learner



1. Run supervised training (or finetuning) to get initial $\pi_\theta(\mathbf{a}|\mathbf{s})$
2. For each $\mathbf{s}$ sample K answers $\mathbf{a}_k \sim \pi(\mathbf{a}|\mathbf{s})$, add to dataset $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_{i,1}, ..., \mathbf{a}_{i,K})\}$
3. Get humans to label which $\mathbf{a}_{i,k}$ they prefer for each $\mathbf{s}_i$
4. Train r_ψ using labeled dataset $\mathcal{D}$
5. Update π_θ using RL with reward $r_\psi(\mathbf{s}, \mathbf{a})$

Alignment with on-policy RL bottlenecks

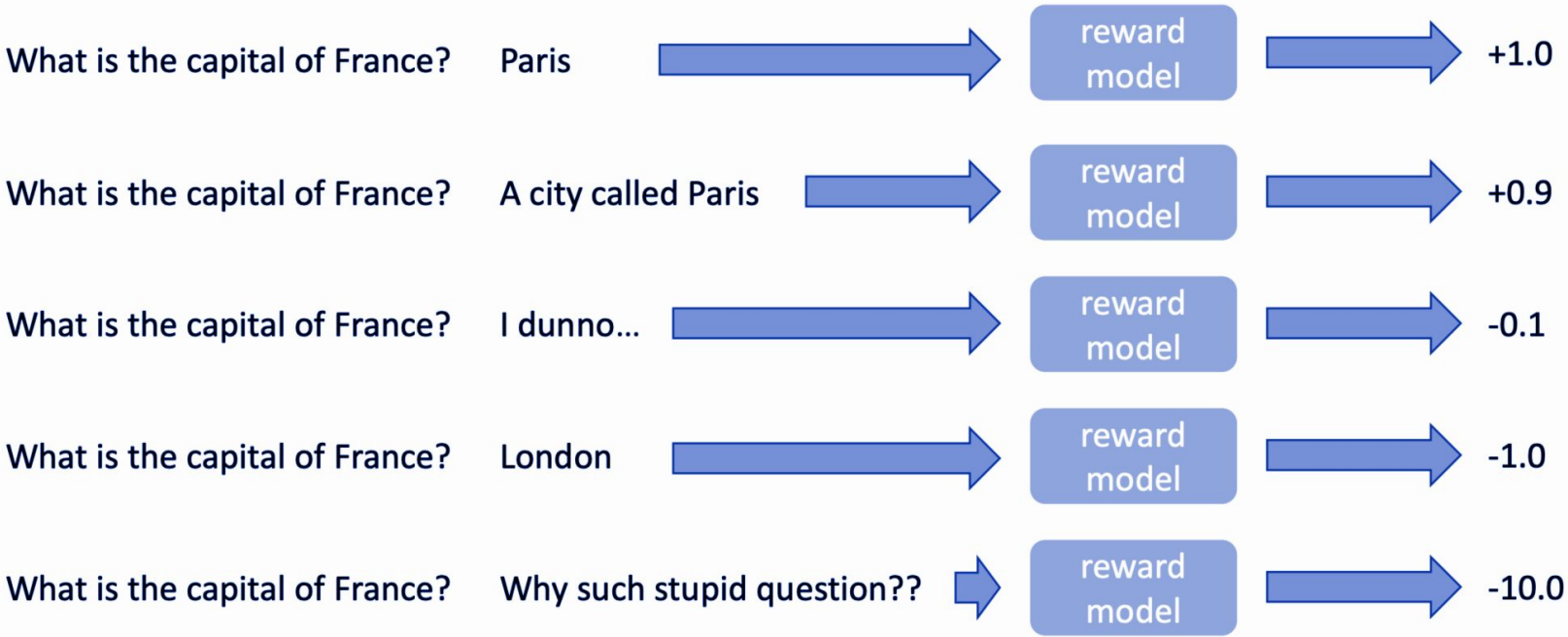


1. Run supervised training (or finetuning) to get initial $\pi_\theta(\mathbf{a}|\mathbf{s})$
2. For each $\mathbf{s}$ sample K answers $\mathbf{a}_k \sim \pi(\mathbf{a}|\mathbf{s})$, add to dataset $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_{i,1}, ..., \mathbf{a}_{i,K})\}$
3. Get humans to label which $\mathbf{a}_{i,k}$ they prefer for each $\mathbf{s}_i$
4. Train r_ψ using labeled dataset $\mathcal{D}$
5. Update π_θ using RL with reward $r_\psi(\mathbf{s}, \mathbf{a})$

Reward Model (RM) training 2

Learned Reward Models

What if $r(\mathbf{s}, \mathbf{a})$ is itself a neural network?



Reward Model (RM) training: which completion is better?

Input

Explain the moon landing to a 6 year old in a few sentences

Completion 1

The Moon is a natural satellite of the Earth. It is the fifth largest moon in the Solar System and the largest relative to the size of its host planet.

<

Completion 2

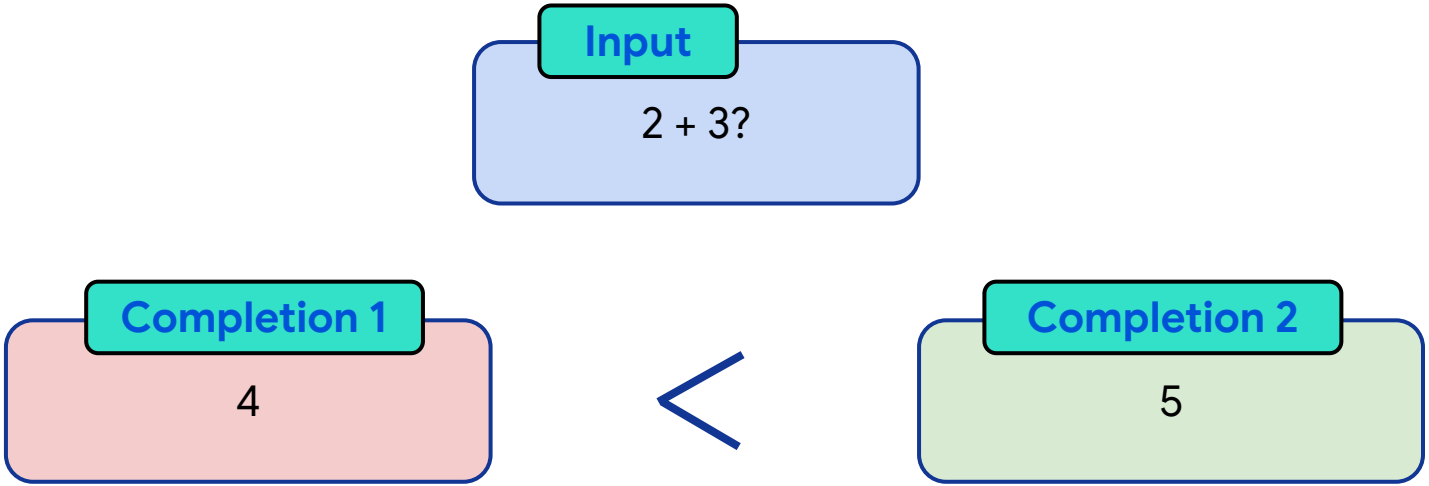
People went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them.

Humans label which completion is preferred.

This setup aims to align models to the human preference

Why use comparison for RM?

For an easy prompt where a clear answer exists, comparison may not be as useful.



Why use comparison for RM?

But for more open-ended generations, it is easier to compare relatively

Input

Write a letter to a 5-year-old boy from Santa Clause explaining that Santa is not real. Convey gently so as not to break his heart

Completion 1

<

Completion 2

Choice models

Choice models are computational models used to understand and predict how individuals make decisions.

Utilities in choice models are typically denoted as U_{ni} where "*n*" denotes the nth individual or trial, and *"i"* denotes the ith choice. Thus, U_{ni} in this context corresponds to the utility of nth individual for making the ith choice.

The utilities in the context of choice models are similar to the rewards in Reinforcement Learning.

Main assumption: The utility an individual derives from making a choice ***A*** over ***B*** is a function of the frequency of choosing ***A*** over ***B*** in repeated trials.

Discrete choice

The discrete choice y_{ni} which is made collectively exhaustive, finite and mutually exclusive as follows:

$$y_n = \begin{cases} 1 & \text{if } U_{ni} > U_{nj}, \forall i \neq j, \\ 0 & \text{otherwise.} \end{cases}$$

for $U_{ni} = H_{ni}(z_{ni})$ where z_{ni} corresponds to the variables corresponding to the invidual attributes and the alternative choices.

- H_{ni} is a linear and stochastic function.
- $H_{ni} = \beta z_{ni} + \epsilon_{ni}$ where ϵ_{ni} corresponds to the unobserved individual factors.

Binary choice

- Benefit of an action depends on the "s_n" individual characteristics:

$$U_n = \beta s_n + \epsilon_n$$

$$y_n = \begin{cases} 1 & U_n > 0, \\ 0 & U_n \leq 0 \end{cases}$$

$$P_{n1} = \frac{1}{1 + \exp(-\beta s_n)}$$

for $\epsilon \sim$ logistic.

Bradley-terry model

The utility of each choice depends on the attributes of the choice and the invidividual making that choice.

- Assuming that unobserved terms have the extreme value distribution:

$$\begin{aligned} U_{n1} &= \beta z_{n1} + \epsilon_{n1}, \\ U_{n2} &= \beta z_{n2} + \epsilon_{n2}, \\ \epsilon_{n1}, \epsilon_{n2} &\sim \textbf{extreme value distribution.} \end{aligned}$$

Then,

$$\begin{aligned} P_{n1} &= \frac{\exp(\beta z_{n1})}{\exp(\beta z_{n1}) + \exp(\beta z_{n2})}, \\ &= \frac{1}{1 + \exp\left(-\beta(z_{n1} - z_{n2})\right)}. \end{aligned}$$

P_{n1} models how much the individual prefers choice 1 over the alternative 2.

Reward Model (RM) training objective function

Let p_{ij} be the probability that completion y_i is better than completion y_j :

Bradley–Terry model (1952): log odds that completion y_i is favored over y_j is modeled as difference in the rewards:

$$\log \frac{p_{ij}}{1 - p_{ij}} = r(x, y_i; \phi) - r(x, y_j; \phi)$$

Reward Model (RM) training objective function

Let p_{ij} be the probability that completion y_i is better than completion y_j

Bradley–Terry model (1952): log odds that completion y_i is favored over y_j is modeled as difference in the rewards:

$$\log \frac{p_{ij}}{1 - p_{ij}} = r(x, y_i; \phi) - r(x, y_j; \phi)$$

$$p_{ij} = \frac{e^{r(x,y_i;\phi) - r(x,y_j;\phi)}}{1 + e^{r(x,y_i;\phi) - r(x,y_j;\phi)}} = \sigma(r(x, y_i; \phi) - r(x, y_j; \phi))$$

Reward Model (RM) training objective function

Let p_{ij} be the probability that completion y_i is better than completion y_j

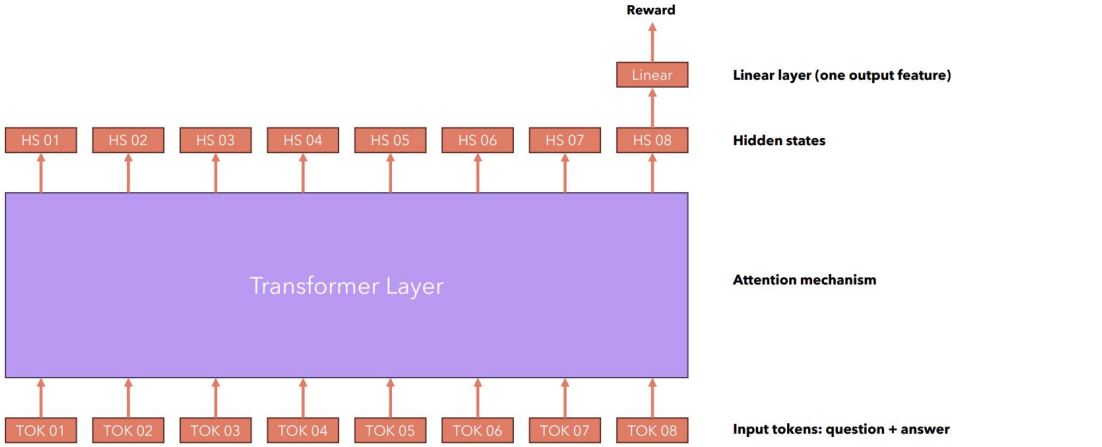
Bradley–Terry model (1952): log odds that completion y_i is favored over y_j is modeled as difference in the rewards:

$$\log \frac{p_{ij}}{1 - p_{ij}} = r(x, y_i; \phi) - r(x, y_j; \phi)$$

$$p_{ij} = \frac{e^{r(x,y_i;\phi) - r(x,y_j;\phi)}}{1 + e^{r(x,y_i;\phi) - r(x,y_j;\phi)}} = \sigma(r(x, y_i; \phi) - r(x, y_j; \phi))$$

$$\max_{\phi} \sum_{x,y_i,y_j \in D} \log p_{ij}$$

Reward model



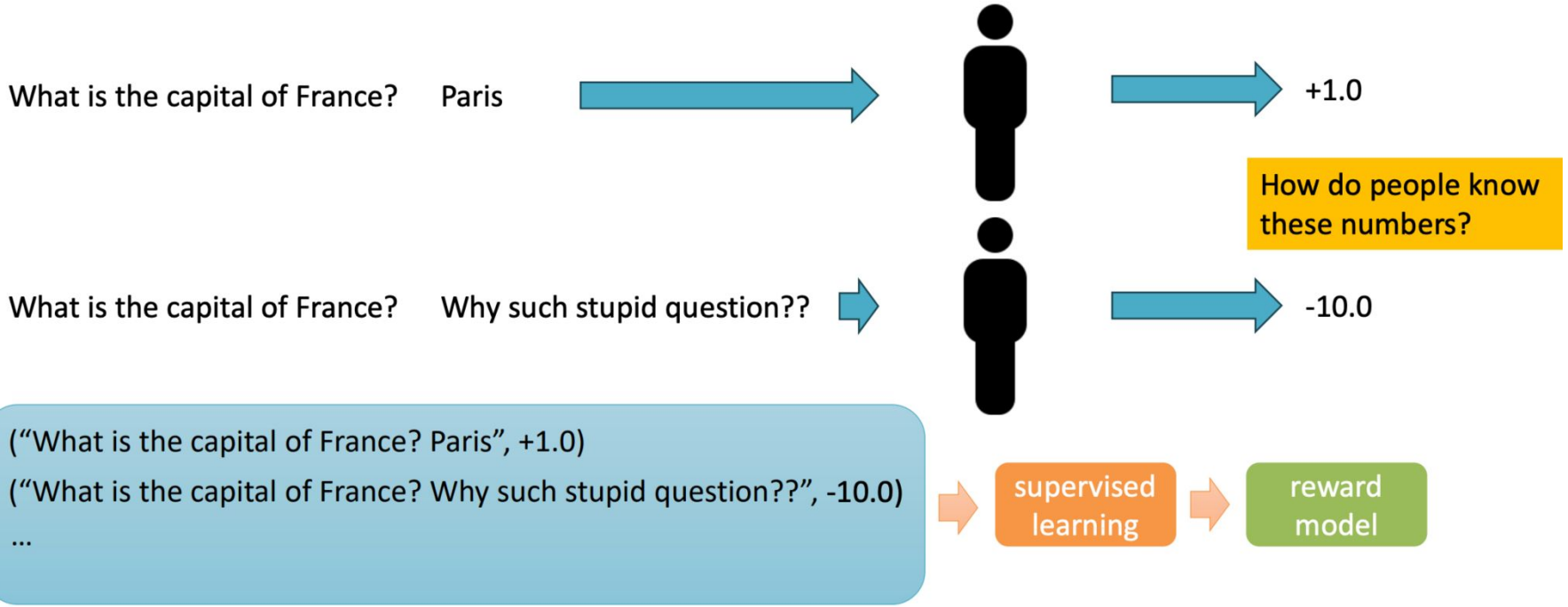
Umar Jamil - https://github.com/hkproj/rlhf-ppo

A typical dataset for reward modeling:

Question (Prompt)	Answer 1	Answer 2	Chosen
Where is Shanghai?	Shanghai is a city in China	Shanghai does not exist	1
Explain gravity like I'm 5	Gravity is a famous restaurant	Gravity is what pulls things toward each other. It's why you stay on the ground and planets orbit the sun.	2
What is 2+2?	4	2+2 is a very complicated math problem…	1

Reward Model Training

How do we train the reward model $r_\psi(\mathbf{s}, \mathbf{a})$?



Policy training 3

Policy model objective function

Once we have a reward model, we can use it in RL to learn the language model parameters that maximizes the expected reward

$$J(\theta) = \mathbb{E}_{(X,Y)\sim D_{\pi_\theta}}\left[r(X, Y; \phi)\right]$$

where $X = (X_1, \ldots, X_S)$ is the prompt and $Y = (Y_1, \ldots, Y_T)$ is the completion sampled from the policy model.

Policy model training

The optimization problem is then

$$\max_{\theta} J(\theta) = \max_{\theta} \mathbb{E}_{(X,Y)\sim D_{\pi_\theta}}[r(X, Y; \phi)]$$

Policy model training

The optimization problem is then,

$$\max_{\theta} J(\theta) = \max_{\theta} \mathbb{E}_{(X,Y)\sim D_{\pi_\theta}}[r(X, Y; \phi)]$$

We use iterative algorithm such as gradient ascent to solve this:

$$\theta := \theta + \alpha \nabla J(\theta)$$

Language models and policy gradients

$$\nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) = \nabla_\theta \log p(x_5|x_{1:4}) + \nabla_\theta \log p(x_6|x_{1:4}, x_5)$$

$$\nabla_\theta E_{\pi_\theta(\mathbf{a}|\mathbf{s})}[r(\mathbf{s},\mathbf{a})] = E_{\pi_\theta(\mathbf{a}|\mathbf{s})}[\nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) r(\mathbf{s},\mathbf{a})]$$

samples from $\pi_\theta(\mathbf{a}|\mathbf{s})$

REINFORCE-style estimator

$$\approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}) r(\mathbf{s},\mathbf{a}_i)$$

samples from $\bar{\pi}(\mathbf{a}|\mathbf{s})$

importance-weighted estimator (e.g., PPO)

$$\approx \frac{1}{N} \sum_i \frac{\pi_\theta(\mathbf{a}_i|\mathbf{s})}{\bar{\pi}(\mathbf{a}_i|\mathbf{s})} \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}) r(\mathbf{s},\mathbf{a}_i)$$

Policy model training

The optimization problem is then

$$\max_{\theta} J(\theta) = \max_{\theta} \mathbb{E}_{(X,Y)\sim D_{\pi_\theta}}[r(X, Y; \phi)]$$

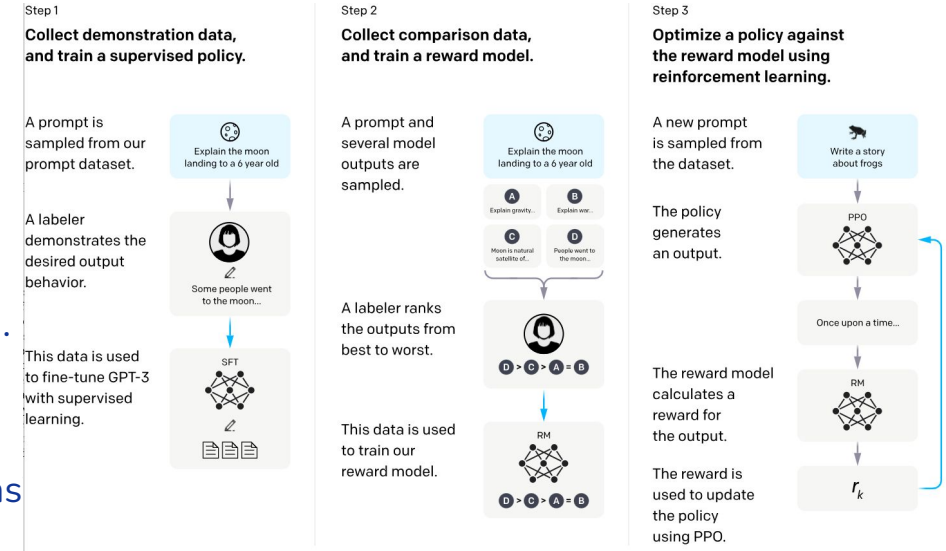
We use iterative algorithm such as gradient ascent to solve this

$$\theta := \theta + \alpha \nabla J(\theta)$$

We can use an on-policy policy-gradient algorithms (with IS correction) to compute the gradients such as PPO.

Instruct GPT Recipe

OpenAI InstructGPT "Recipe"



Alignment achieves:

- Given a model **M**, steer the outputs of **M** to maximize the scores assigned to them by a reward model **R**.

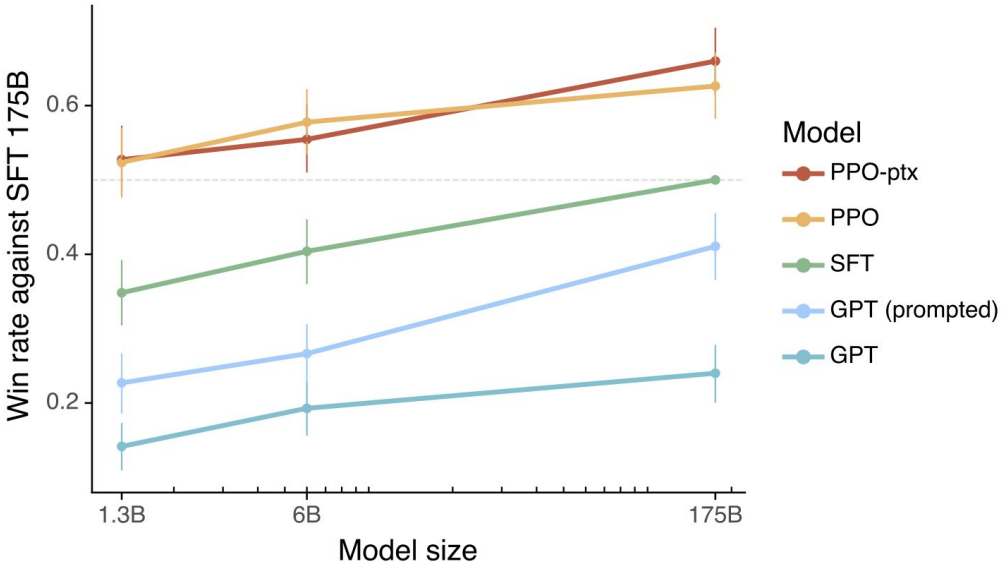
This can naturally be framed as an RL problem.

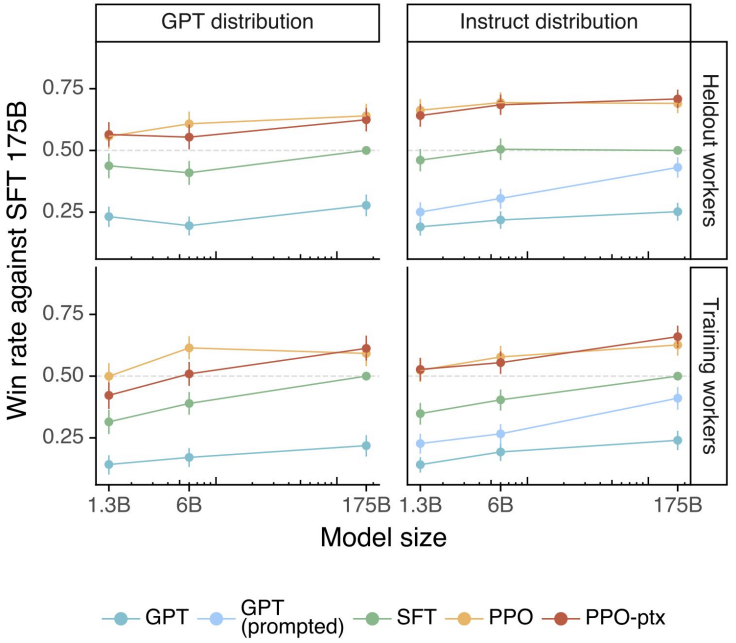
Typically:

- Used to push AI systems towards humans

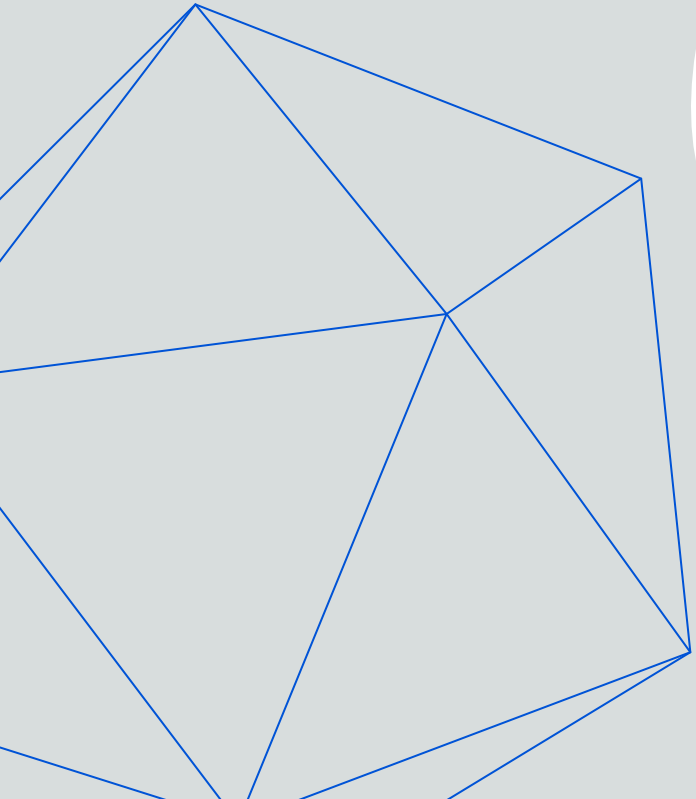
Iteratively align the models produced by the LLM team as **general-purpose conversational AIs**.

Instruct GPT - Results





Online Policy Learning Algorithms



PPO

Paper link: Schulman et al, 2017

The surrogate objective

The goal of the RL is to maximize the following: $J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right]$,

State-value function is defined as: $Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} | S_t = s, A_t = a\right]$

There is a connection between the two: $J(\pi) = \mathbb{E}_{s\sim\rho_0}\Big[\mathbb{E}_{a\sim\pi}\left[Q^\pi(s,a)\right]\Big]$.

Then we can write down the surrogate objective as:

$$
\begin{aligned}
\pi^* &= \arg\max_\pi J(\pi), \\
&= \arg\max_\pi \mathbb{E}_{s\sim\rho_0}\Big[\mathbb{E}_{a\sim\pi}\left[Q^\pi(s,a) - V^\pi(s)\right]\Big] + \underbrace{\mathbb{E}_{s\sim\rho_0}\left[V^\pi(s)\right]}_{\text{constant and gradients are } 0.} \\
&= \arg\max_\pi \mathbb{E}_{s\sim\rho_0,\, a\sim\pi}[A^\pi(s,a)], \\
&= \arg\max_\pi \mathbb{E}_{s\sim\rho_0, a\sim\pi_{old}}\left[\frac{\pi(a|s)}{\pi_{old}(a|s)} A^\pi(s,a)\right].
\end{aligned}
$$

We use the importance sampling to estimate the expectation here.

Surrogate objective for PPO

The objective in the previous slides will give us biased estimates because:

- The advantage is computed with respect to the current policy but the samples are coming from the old one.
- Thus we need to use the following for the unbiased estimate of the gradients:

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

Thus the surrogate objective would be: $\mathcal{L}(\theta) = \mathbb{E}_{s,a\sim\pi_{old}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} A^{\pi_{old}}(s,a)\right].$

Nevertheless, optimizing this in an unconstrained way have unstable learning dynamics. Motivating from CPI (constrained policy iteration), we could write this as:

$$\begin{aligned} &\max_\theta \ \mathbb{E}_{s,a\sim\pi_{old}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} A^{\pi_{old}}(s,a)\right], \\ &\text{subject to } \mathbb{E}_{s\sim\pi_{\theta_{old}}}\left[D_{KL}(\pi_{\theta_{old}}(\cdot|s)||\pi_\theta(\cdot|s))\right] \leq \delta. \end{aligned}$$

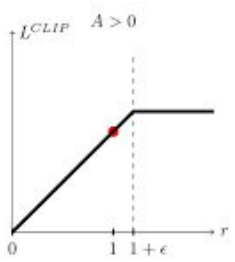
This gives rise to the KL-PPO objective with $r_\theta(s,a) = \frac{\pi_\theta(s,a)}{\pi_{\theta_{old}}(s,a)}$:

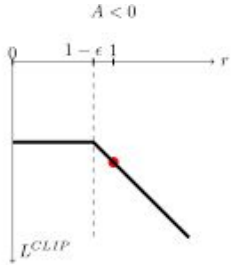
$$\mathcal{L}^{KL-PPO}(\theta) = \mathbb{E}_{(s,a)\sim\pi_{old}}\left[r_\theta(s,a) A_{\theta_{old}}(s,a) - \beta KL(\pi_{old}||\pi_\theta)\right].$$

Clip-PPO

We would like to constrain how much the current policy will deviate from the old policy. The clip-PPO applies the clipping function to the ratio $r_\theta(s,a)$:

$$\text{clip}\big(r_\theta(s,a),\ 1-\varepsilon,\ 1+\varepsilon\big) \ = \ \begin{cases} 1-\varepsilon, & \text{if } r_\theta(s,a) < 1-\varepsilon, \\ 1+\varepsilon, & \text{if } r_\theta(s,a) > 1+\varepsilon, \\ r_\theta(s,a), & \text{otherwise.} \end{cases}$$





Then **clip-PPO** forms the objective **per sample** *(s,a)* as the minimum of two terms:

1. **Unclipped:** $r_\theta(s,a)\, A^{\pi_{\text{old}}}(s,a)$.
2. **Clipped:** . $\text{clip}\big(r_\theta(s,a), 1-\varepsilon, 1+\varepsilon\big)\, A^{\pi_{\text{old}}}(s,a)$

Hence the full loss, which we maximize will be:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a\sim\pi_{old}}\Big[\min\Big(r_\theta(s,a)\, A^{\pi_{\text{old}}}(s,a),\ \text{clip}\big(r_\theta(s,a),\ 1-\varepsilon,\ 1+\varepsilon\big)\ A^{\pi_{\text{old}}}(s,a)\Big)\Big].$$

PPO in Instruct GPT

$$\text{objective}\,(\phi) = E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\text{RL}}(y \mid x)/\pi^{\text{SFT}}(y \mid x)\right)\right] + \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\text{RL}}(x))\right] \tag{2}$$

Original PPO:

$$\underset{\theta}{\text{maximize}}\,\hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t - \beta\,\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]\right] \tag{5}$$

PPO Instruct GPT Pseudocode

Step 1: Pre-train the model with the vast amount of data.

Step 2: Do SFT on the pre-trained model.

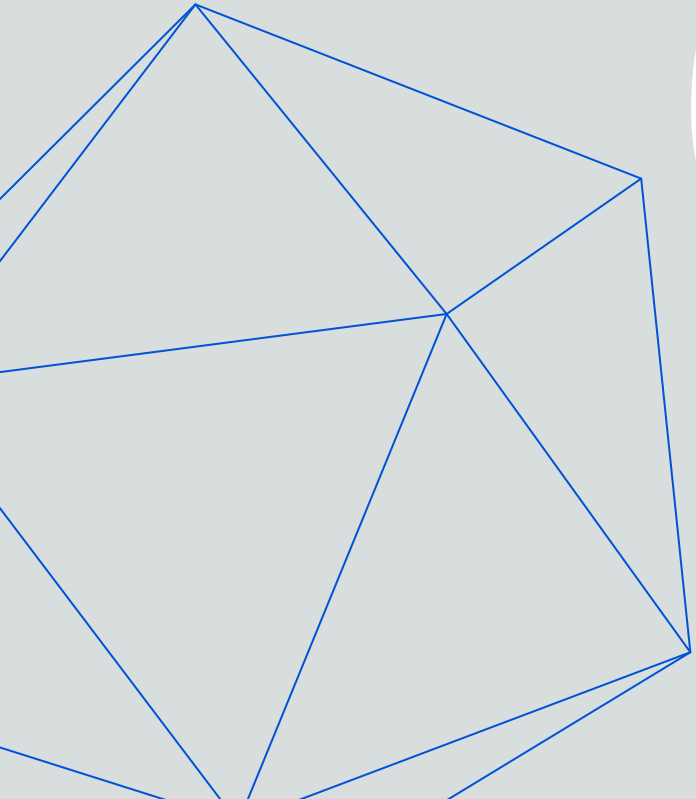
Step 3: Train the reward model.

Protocol 4 PPO (Proximal Policy Optimization) [16]

Receive policy class parameters Θ, and reward model parameters $\phi^\star$. Initialize θ_1.
for $t = 1, \ldots, T$ **do**
  Sample $y^i \sim \pi(\cdot|x^i; \theta_t)$ and $x^i \in \mathcal{D}_{\text{prompts}}$.
  The learner computes the stochastic loss.

$$\widehat{\ell}_{\text{RLHF}}(\theta_t) := -\frac{1}{|\mathcal{D}_{\text{prompts}}|} \sum_{i \in [N_{\text{prompts}}]} \left[r(x^i, y^i; \phi^\star) + \beta \log \pi(y^i|x^i; \theta_t) - \beta \log \pi_{\text{ref}}(y^i|x^i) \right],$$

  The learner computes the policy ratio $\delta(\theta; \theta_t) = \frac{1}{|\mathcal{D}_{\text{prompts}}|} \sum_{i \in [N_{\text{prompts}}]} \frac{\pi(y^i|x^i;\theta)}{\pi(y^i|x^i;\theta_t)}$.
  The learner computes $\ell_{\text{PPO}}(\theta) = -\min \left(\delta(\theta; \theta_t) \widehat{\ell}_{\text{RLHF}}(\theta_t), \text{Clip}(\delta(\theta; \theta_t), 1+\epsilon, 1-\epsilon) \widehat{\ell}_{\text{RLHF}}(\theta_t) \right)$
  $\theta_{t+1} \approxeq \arg\min_{\theta \in \Theta} \ell_{\text{PPO}}(\theta)$ computed via SGD, Adam or other optimizers.
end for
The learner outputs $\pi_{\theta^\star}$.



RLOO

Main paper by: Ahmadian et al 2024

A simple algorithm for RLHF

Ahmadian et al 2024 has proposed a simpler algorithm than PPO:

- Only requires a reward model and a policy. Does not require to keep a value function in the memory.
- Simpler implementation, no clipping or computation of ratios.
- PPO assigns the reward to the whole trajectory whereas RLOO assigns reward to the whole sequence.

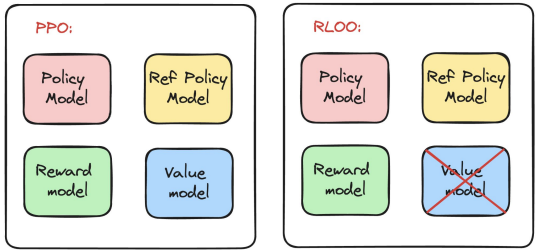
The KL constraint in the KL-constrained reward maximization can be absorbed into the reward and RLOO would optimize for:

$$R(x, y) = r_\phi(x, y) - \beta \log \frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)}$$

RLOO generates ***k*** independent online samples for the same prompt ***x*** from the policy π_θ :

$$y^{(1)}, y^{(2)}, ..., y^{(k)} \sim \pi_\theta(\cdot|x)$$

Each generated sample $y^{(i)}$ has its associated reward $R(x, y^{(i)})$.



RLOO

REINFORCE algorithm: The original REINFORCE estimator for RLHF (with a moving-average baseline) is:

$$\nabla_\theta \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[R(x,y)] \approx [R(x,y) - b]\nabla_\theta \log \pi_\theta(y|x)$$

where the baseline b could be a simple moving average across many steps, such as:

$$b = \frac{1}{S}\sum_{s=1}^{S} R(x_s, y_s)$$

The RLOO approach replaces this moving-average baseline with a more effective leave-one-out baseline computed using multiple samples from the same prompt.

The RLOO estimator uses each sample's reward as a baseline for the other samples. Specifically, for each $y^{(i)}$, it subtracts the mean reward of the other ***k-1*** samples from its own reward:

$$\text{Gradient Estimate}_{\text{RLOO}} = \frac{1}{k}\sum_{i=1}^{k}\left[R(x,y^{(i)}) - \frac{1}{k-1}\sum_{j\neq i} R(x,y^{(j)})\right]\nabla_\theta \log \pi_\theta(y^{(i)}|x)$$

RLOO Pseudocode

Algorithm RLOO

T: the number of training steps

for $j = 0$ **to** T **do**

  Sample a batch of prompts $\{x_n\}_{n=1}^{N}$.

  For each prompt x_n, sample k completions $\{y_n^{(i)}\}_{i=1}^{k}$.

  Compute the leave-one-out baseline and gradient estimate per sample as described as:

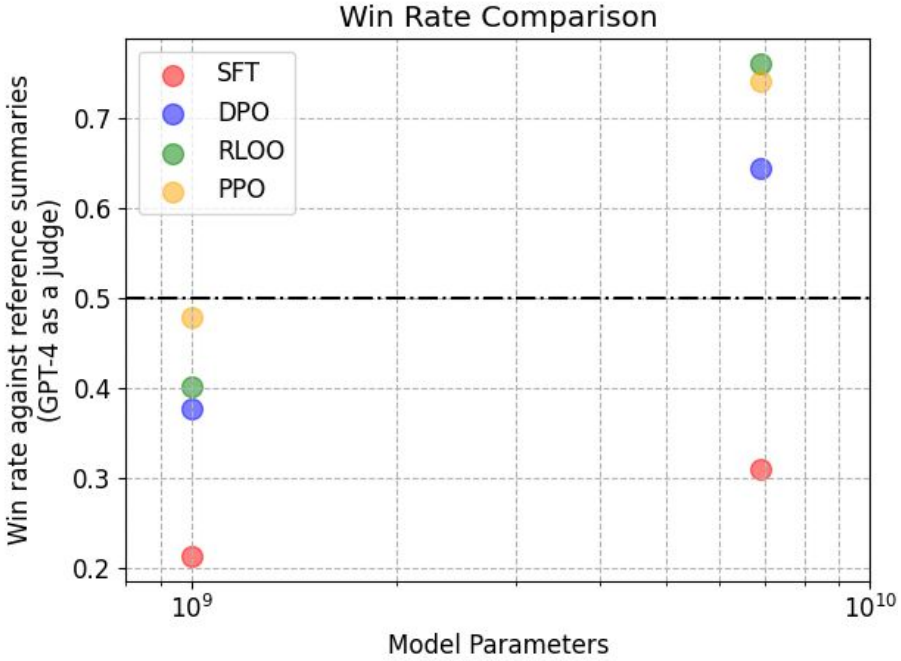
  $$\text{Gradient Estimate}_{\text{RLOO}} = \frac{1}{k}\sum_{i=1}^{k}\left[R(x, y^{(i)}) - \frac{1}{k-1}\sum_{j\neq i} R(x, y^{(j)})\right]\nabla_\theta \log \pi_\theta(y^{(i)}|x)$$

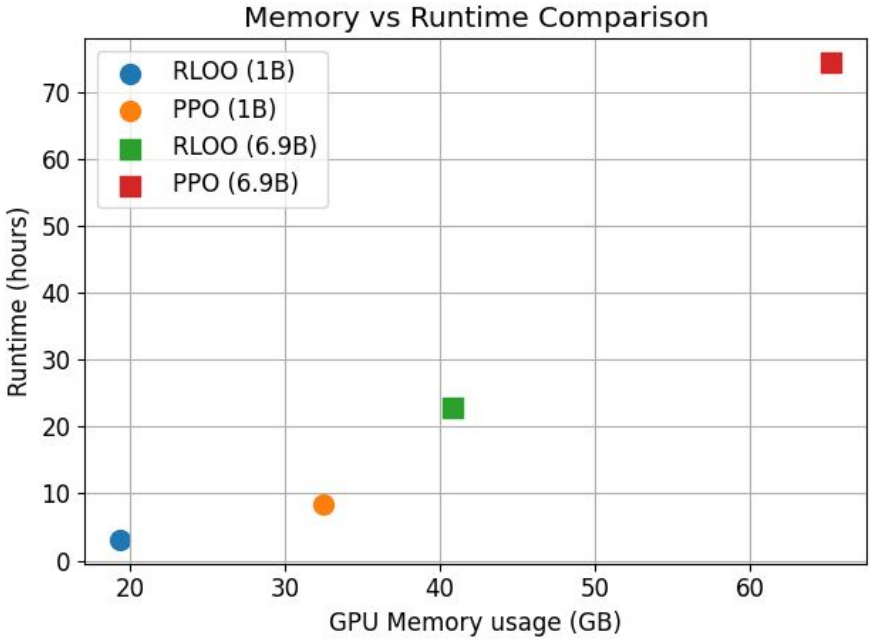
  Average gradient estimates across prompts and update parameters θ:

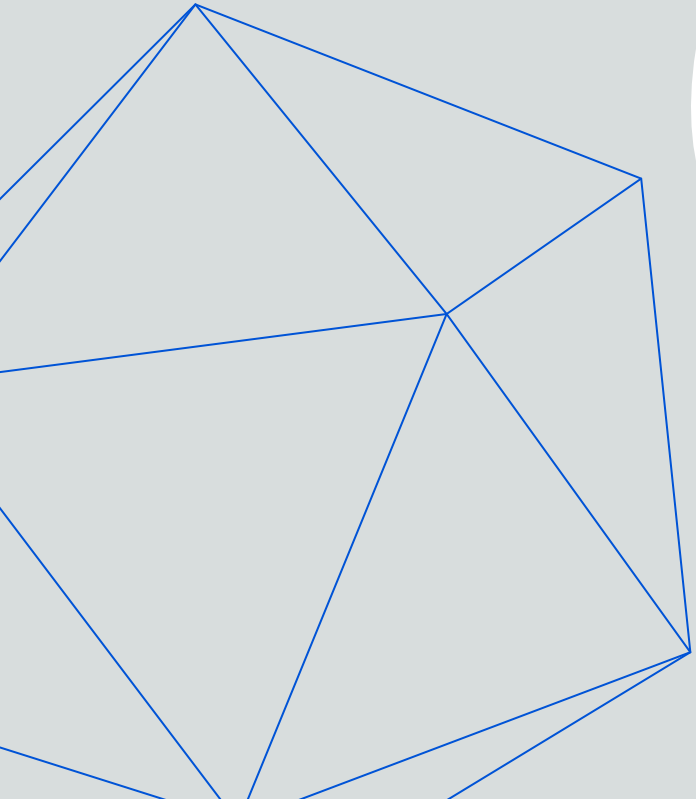
  $$\theta \leftarrow \theta + \eta \frac{1}{N}\sum_{n=1}^{N}\frac{1}{k}\sum_{i=1}^{k}\left[R(x_n, y_n^{(i)}) - \frac{1}{k-1}\sum_{j\neq i} R(x_n, y_n^{(j)})\right]\nabla_\theta \log \pi_\theta(y_n^{(i)}|x_n)$$

end for

RLOO Results







GRPO

Paper by Shao et al, 2024

A simple extension to PPO

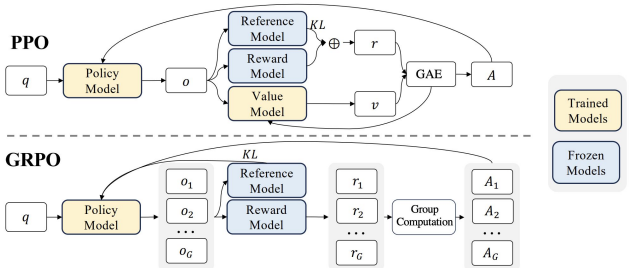
Let us remind ourselves the PPO objective:

$$J_{\text{PPO}}(\theta) = \mathbb{E}_{x\sim\rho_0, y\sim\pi_{\theta_{\text{old}}}(y|x)} \left[\frac{1}{|y|} \sum_{t=1}^{|y|} \min \left(\frac{\pi_\theta(y_t|x, y_{<t})}{\pi_{\theta_{\text{old}}}(y_t|x, y_{<t})} A_t, \text{clip} \left(\frac{\pi_\theta(y_t|x, y_{<t})}{\pi_{\theta_{\text{old}}}(y_t|x, y_{<t})}, 1-\epsilon, 1+\epsilon \right) A_t \right) \right],$$

Typically, PPO computes advantage A_t using Generalized Advantage Estimation (GAE) based on a learned value function V_ψ:

$$A_t = r_t + \gamma V_\psi(y_{t+1}) - V_\psi(y_t),$$

Key Idea: GRPO seeks to eliminate the need for value function. It approximates the baseline directly by ***sampling multiple outputs for the same prompt and computing the relative rewards***. These rewards are then normalized, and the advantage is defined directly from these normalized relative scores.



GRPO Loss

Given an input question ***x***, GRPO generates a set of ***G*** outputs using the current (old) policy model:

$$\{y_1, y_2, \ldots, y_G\} \sim \pi_{\theta_{\text{old}}}(Y|x).$$

Computing the normalized rewards: Each sampled output y_i from the LLM receives a scalar reward r_i from a reward model r_ϕ:

$$r_i = r_\phi(x, y_i), \ \forall i = 1, 2, \ldots, G,$$

GRPO, then computes the normalized rewards within the same group:

$$\tilde{r}_i = \frac{r_i - \text{mean}(\{r_1, \ldots, r_G\})}{\text{std}(\{r_1, \ldots, r_G\})}$$

The advantage is equivalent to the reward: $\hat{A}_{i,t} = \tilde{r}_i$

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{x\sim\rho_0,\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(Y|x)} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|} \min\left(\frac{\pi_\theta(y_{i,t}|x,y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|x,y_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(y_{i,t}|x,y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|x,y_{i,<t})}, 1-\epsilon, 1+\epsilon \right)\hat{A}_{i,t} \right) - \beta D_{\text{KL}}(\pi_\theta||\pi_{ref}) \right].$$

If performing only a single gradient update after each sampling step, this objective simplifies to:

$$J_{\text{GRPO}}(\theta) \approx \mathbb{E}_{x,\{y_i\}} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|} \hat{A}_{i,t} \log \pi_\theta(y_{i,t}|x, y_{i,<t}) \right].$$

GRPO Updates

The KL penalty of GRPO can be approximated in practice with an unbiased estimator:

$$D_{\mathrm{KL}}(\pi_\theta||\pi_{\mathrm{ref}}) = \frac{\pi_{\mathrm{ref}}(y_t|x,y_{<t})}{\pi_\theta(y_t|x,y_{<t})} - \log\frac{\pi_{\mathrm{ref}}(y_t|x,y_{<t})}{\pi_\theta(y_t|x,y_{<t})} - 1.$$

And the gradients of the GRPO objective could be written as:

$$\nabla_\theta J_{\mathrm{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\left(\hat{A}_{i,t} + \beta\left(\frac{\pi_{\mathrm{ref}}(y_{i,t}|x,y_{i,<t})}{\pi_\theta(y_{i,t}|x,y_{i,<t})} - 1\right)\right)\nabla_\theta \log \pi_\theta(y_{i,t}|x,y_{i,<t})\right].$$

5.2.1. Towards to a Unified Paradigm

In this section, we provide a unified paradigm to analyze different training methods, such as SFT, RFT, DPO, PPO, GRPO, and further conduct experiments to explore the factors of the unified paradigm. Generally, the gradient with respect to the parameter θ of a training method can be written as:

$$\nabla_\theta \mathcal{J}_{\mathcal{A}}(\theta) = \mathbb{E}\underbrace{[(q,o)\sim\mathcal{D}]}_{\textit{Data Source}}\left(\frac{1}{|o|}\sum_{t=1}^{|o|}\underbrace{GC_{\mathcal{A}}(q,o,t,\pi_{rf})}_{\textit{Gradient Coefficient}}\nabla_\theta \log \pi_\theta(o_t|q,o_{<t})\right). \quad (5)$$

There exist three key components: 1) *Data Source* $\mathcal{D}$, which determines the training data; 2) *Reward Function* π_{rf}, which is the source of the training reward signal; 3) *Algorithm* $\mathcal{A}$: which processes the training data and the reward signal to the gradient coefficient GC that determines the magnitude of the penalty or reinforcement for the data. We analyze several representative methods based on such a unified paradigm:

GRPO Pseudocode

Require: Initial policy $\pi_{\theta_{\text{init}}}$; Reward model r_ϕ; Dataset of mathematical problems $\mathcal{D}$; Hyperparameters: clip parameter ϵ, KL regularization β, number of samples per query G, number of policy update epochs μ, iterations I, steps per iteration M.

Initialize policy model $\pi_\theta \leftarrow \pi_{\theta_{\text{init}}}$
for iteration $= 1, \ldots, I$ **do**
  Set reference policy: $\pi_{\text{ref}} \leftarrow \pi_\theta$
  for step $= 1, \ldots, M$ **do**
    Sample a batch $\mathcal{D}_b \subseteq \mathcal{D}$
    Set old policy $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$
    for each question $q \in \mathcal{D}_b$ **do**
      Sample G outputs: $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot|q)$
      Compute rewards: $r_i \leftarrow r_\phi(q, o_i)$ for $i = 1, \ldots, G$
      Compute normalized rewards:
        $\tilde{r}_i \leftarrow \frac{r_i - \text{mean}(\{r_j\}_{j=1}^{G})}{\text{std}(\{r_j\}_{j=1}^{G})}, \quad i = 1, \ldots, G$
      Set advantages for all tokens in output o_i:
        $\hat{A}_{i,t} \leftarrow \tilde{r}_i, \quad \forall t$
    end for
    for epoch $= 1, \ldots, \mu$ **do**
      Update policy model π_θ by maximizing GRPO objective:
        $J_{\text{GRPO}}(\theta) =$
        $\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}, 1-\epsilon, 1+\epsilon\right)\hat{A}_{i,t}\right) - \beta D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})\right]$
      Perform gradient ascent step on θ
    end for
  end for
  Continually update reward model r_ϕ via replay mechanism if necessary
end for
return Final policy model π_θ

Preference based policy optimization

Constrained reward maximization

In most alignment problems, we parameterized the problem as a constrained reward maximization:

$$\pi^* = \max_{\pi} \mathbb{E}_{x\sim\rho}\left[\mathbb{E}_{y\sim\pi}\left[R(x,y)\right] - \beta D_{KL}(\pi(y|x)||\pi_{ref}(y|x))\right],$$

For simplicity we would consider maximizing this objective for a single prompt:

$$\pi^* = \max_{\pi} \mathbb{E}_{y\sim\pi}\left[R(x,y)\right] - \beta D_{KL}\big(\pi(y|x)||\pi_{ref}(y|x)\big).$$

The constraint optimization problem can also be written as:

$$\begin{gathered}\max_{\pi}\ \mathbb{E}_{y\sim\pi}\left[R(s,a)\right], \\ \text{subject to } D_{KL}\big(\pi(y|x)||\pi_{ref}(y|x)\big), \\ \sum_{y}\pi(y|x) = 1, \\ \pi(a|s) \geq 0 \text{ for all } a.\end{gathered}$$

The Lagrangian can be written as:

$$\mathcal{L}\left(\pi(\cdot|y), \beta, \lambda\right) = \mathbb{E}_{y\sim\pi}\left[R(x,y)\right] - \beta\mathbb{E}_{y\sim\pi}\left[\log\frac{\pi(y|x)}{\pi_{ref}(y|x)}\right] + \lambda\left(1 - \sum_{y}\pi(y|x)\right).$$

Solving the Lagrangian

We can find the stationary points by solving:

$$\frac{\partial \mathcal{L}(\pi(\cdot|x), \beta, \lambda)}{\partial \pi} = 0$$

This would be:

$$\begin{aligned}
\frac{\partial \mathcal{L}(\pi(\cdot|x), \beta, \lambda)}{\partial \pi} &= R(s,a) - \beta \left(\log \pi(y|x) - \log \pi_{ref}(y|x) + 1\right) - \lambda, \\
0 &= R(s,a) - \beta \left(\log \pi(y|x) - \log \pi_{ref}(y|x) + 1\right) - \lambda, \\
R(s,a) - \lambda - \beta &= \beta \log \frac{\pi(y|x)}{\pi_{ref}(y|x)}, \\
\exp\left(\frac{R(s,a) - \lambda}{\beta} - 1\right) &= \frac{\pi(y|x)}{\pi_{ref}(y|x)}, \\
\pi^*(y|x) &= \pi_{ref}(y|x) \exp\left(\frac{R(s,a)}{\beta}\right) \exp\left(-\frac{\lambda}{\beta} - 1\right).
\end{aligned}$$

Fixing the normalization

For that perhaps, we could solve it by the fact that the sum or probabilities of the policy over *y* sums up to 1:

$$\begin{aligned}
\sum_y \pi(y|x) &= 1, \\
\sum_y \pi_{ref}(y|x) \exp\left(\frac{R(x,y)}{\beta}\right) \exp\left(-\frac{\lambda}{\beta} - 1\right) &= 1, \\
\exp\left(-\frac{\lambda}{\beta} - 1\right) \sum_y \pi_{ref}(y|x) \exp\left(\frac{R(x,y)}{\beta}\right) &= 1, \\
\underbrace{\sum_y \pi_{ref}(y|x) \exp\left(\frac{R(x,y)}{\beta}\right)}_{\text{Partition function } Z(y)} &= \exp\left(\frac{\lambda}{\beta} + 1\right),
\end{aligned}$$

Then the optimal policy could be written as:

$$\pi^*(y|x) = \pi_{ref}(y|x) \exp\left(\frac{R(s,a)}{\beta}\right) \frac{1}{Z(y)}.$$

Deriving Rewards for DPO

Using the equations we found for the optimal policy from KL constrained reward maximization, we can derive the reward as:

$$\pi(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|z)\exp(\frac{1}{\beta}r(x,y)),$$

$$Z(x)\pi(y|x) = \pi_{\text{ref}}(y|x)\exp(\frac{1}{\beta}r(x,y)),$$

$$Z(x)\frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} = \exp(\frac{1}{\beta}r(x,y)),$$

$$r(x,y) = \beta\left(\log\left(\frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)}\right) + \log(Z(x))\right).$$

DPO Loss

We can apply this parameterization of the reward-model to the Bradley-Terry model described above to reparametrize the optimal policy π^* with the optimal-reward r^* via the difference between two completions/candidates ($p^*(y_1 \succ y_2) = \sigma(r^*(x, y_1) - r^*(x, y_2))$) generated by the model:

$$p^*(y_1 \succ y_2|x) = \frac{1}{1 + \exp\left(\beta \log\left(\frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)}\right) - \beta \log\left(\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)\right)}$$

The loss function can be written as doing the maximum likelihood to maximize the probability above:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{x, y_w, y_l \sim \mathcal{D}}\left[\log \sigma\left(\beta \log \frac{\pi^*(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi^*(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right]$$

DPO

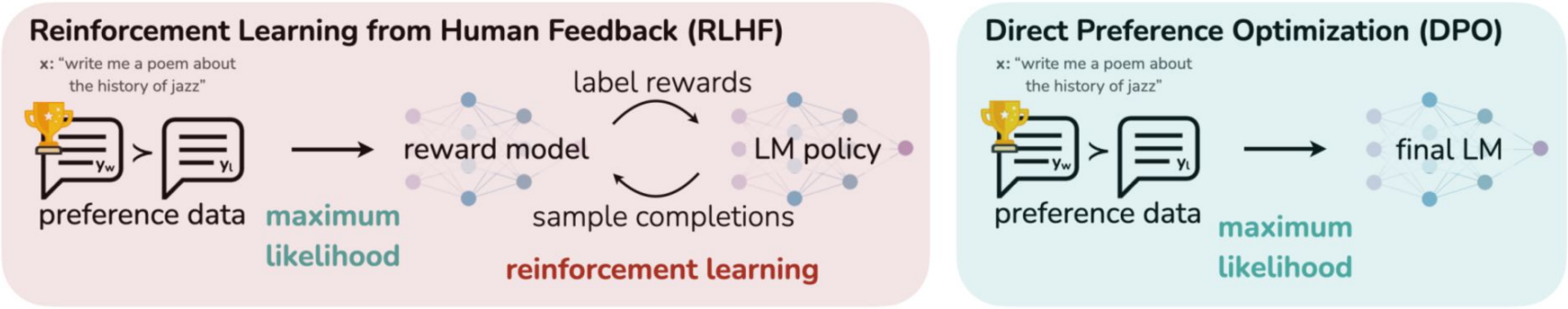


Figure 1: **DPO optimizes for human preferences while avoiding reinforcement learning.** Existing methods for fine-tuning language models with human feedback first fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward. In contrast, DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, fitting an *implicit* reward model whose corresponding optimal policy can be extracted in closed form.

Derivative of the DPO loss and Results

$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{base}}) =$$

$$-\beta \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Bigg[\underbrace{\sigma(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w))}_{\text{higher weight when reward estimate is wrong}}\Bigg[\underbrace{\nabla_\theta \log \pi_\theta(y_w \mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta \log \pi_\theta(y_l \mid x)}_{\text{decrease likelihood of } y_l}\Bigg]\Bigg],$$

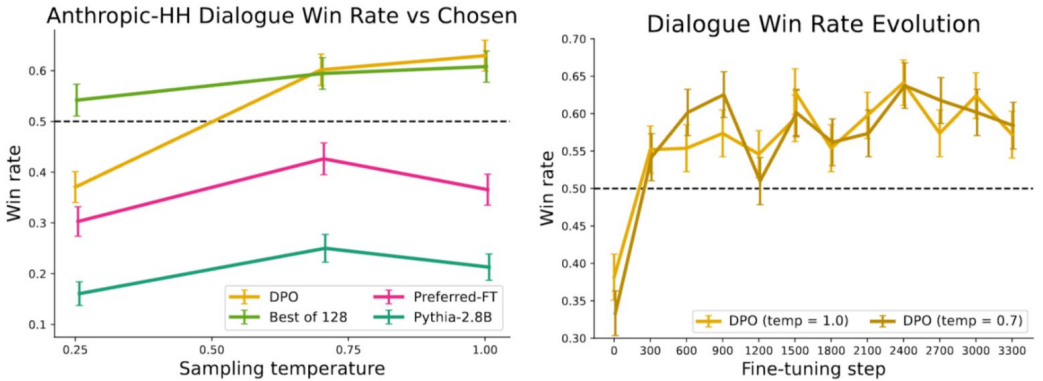


Figure 3: **Left.** Win rates computed by GPT-4 for Anthropic-HH one-step dialogue; DPO is the only method that improves over chosen summaries in the Anthropic-HH test set. **Right.** Win rates for different sampling temperatures over the course of training. DPO's improvement over the dataset labels is fairly stable over the course of training for different sampling temperatures.

Question: What are the problems with DPO?

DPO problems

DPO has some drawbacks:

- Relies on Bradley-Terry choice modeling. This assumption may not hold in practice and may unintended introduce biases. For example:

  Non-transitive preferences.

- DPO may overfit when the preferences are deterministic. For example when:

$$p(y' \succ y|x) \approx 1$$

How IPO addresses them?

DPO problems that IPO fixes

DPO has some drawbacks:

- Relies on Bradley-Terry choice modeling. This assumption may not hold in practice and may unintended introduce biases. For example:

  Non-transitive preferences.

- DPO may overfit when the preferences are deterministic. For example when:

$$p(y' \succ y|x) \approx 1$$

How IPO addresses them?

- IPO entirely removes the Bradley–Terry assumption by optimizing directly on pairwise preferences via identity mapping: $\Psi(q) = q \quad (\text{identity})$

  Benefit: IPO can faithfully capture a broader range of human preference distributions.

- IPO explicitly avoids extreme exponential transformations by directly optimizing the linear difference in preference probabilities.

IPO objective

We could write the IPO objective as follows:

$$\max_{\pi} \mathbb{E}_{x\sim\rho, y\sim\pi, y'\sim\mu} \left[p^*(y \succ y'|x)\right] - D_{KL}(\pi||\pi_{ref})$$

I will use the short-hand notation for the policies here, and will not always write it in the form of $\pi(y|x)$ and $\pi_{ref}(y|x)$, instead I will use π and π_{ref}. Here μ refers to the behavior policy that generated the dataset and ρ represents the context distribution.

Finding the optimal IPO policy

The optimal policy can be written as:

$$\pi^*(y|x) = \frac{\pi_{ref}(y|x)}{Z(y)} \exp\left(\frac{\mathbb{E}_{y' \sim \mu}\left[p^*(y \succ y'|x)\right]}{\beta}\right),$$
$$Z(y) = \sum_{\hat{y}} \pi_{ref}(\hat{y}|x) \exp\left(\frac{\mathbb{E}_{y' \sim \mu}\left[p^*(\hat{y} \succ y'|x)\right]}{\beta}\right).$$

To simplify the notation we could also rename the following:

$$g(y|x) = \mathbb{E}_{y \sim \mu}\left[p^*(y \succ y'|x)\right].$$

Then we would have:

$$\pi^*(y|x) = \frac{\pi_{ref}(y|x)}{Z(y)} \exp\left(\frac{\mathbb{E}_{y' \sim \mu}\left[g(y|x)\right]}{\beta}\right),$$
$$\pi^*(y|x) \propto \pi_{ref} \exp\left(\frac{\mathbb{E}_{y' \sim \mu}\left[g(y|x)\right]}{\beta}\right).$$

Constructing the root-finding problem for IPO

We can write the differences between the log-likelihood ratios for accecpted and rejected responses for π and π_{ref} as follows:

$$\begin{aligned} h(y, y', x) &= \log \frac{\pi(y|x)}{\pi(y'|x)} - \frac{\pi_{ref}(y|x)}{\pi_{ref}(y'|x)}, \\ &= \log \frac{\pi(y|x)\pi_{ref}(y'|x)}{\pi(y'|x)\pi_{ref}(y|x)}. \end{aligned}$$

For optimal policy π^*, we could write it as:

$$\begin{aligned} h_{\pi^*}(y, y', x) &= \log \frac{\pi^*(y|x)\pi_{ref}(y'|x)}{\pi^*(y'|x)\pi_{ref}(y|x)}, \\ &= \log \frac{\exp\left(g(y|x)/\beta\right)}{\exp\left(g(y'|x)/\beta\right)}, \\ &= \left[\frac{g(y|x) - g(y'|x)}{\beta}\right]. \end{aligned}$$

We can transform this optimality condition into a root-finding problem:

$$h_{\pi}(y, y', x) - \left[\frac{g(y|x) - g(y'|x)}{\beta}\right] = 0, \forall y, y^i$$

Final IPO loss

Squared loss minimization (IPO loss)

Given the root finding problme to estimate the optimal policy. IPO loss can be written as follows:

$$\mathcal{L}^{IPO}(\pi) = \mathbb{E}_{x\sim\rho y,y'\sim\mu}\left[\left(h_\pi(y,y',x) - \left[\frac{g(y|x) - g(y'|x)}{\beta}\right]\right)^2\right]$$

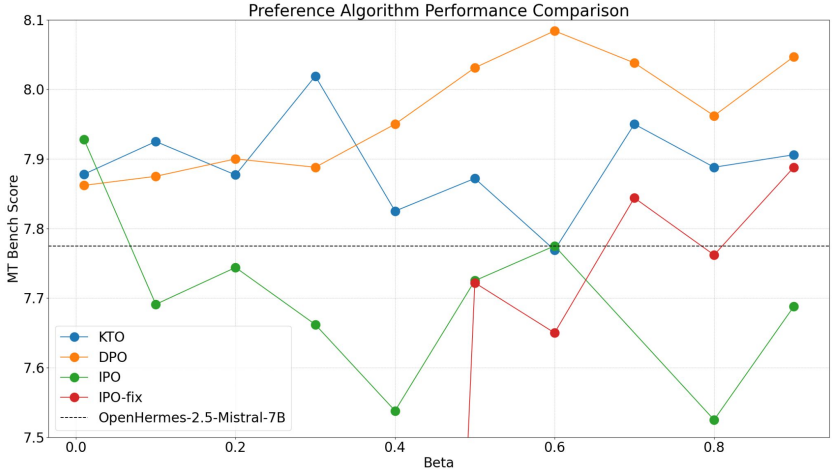
Clearly, the optimal policy π^* satisfies the $\mathcal{L}(\pi^*) = 0$.

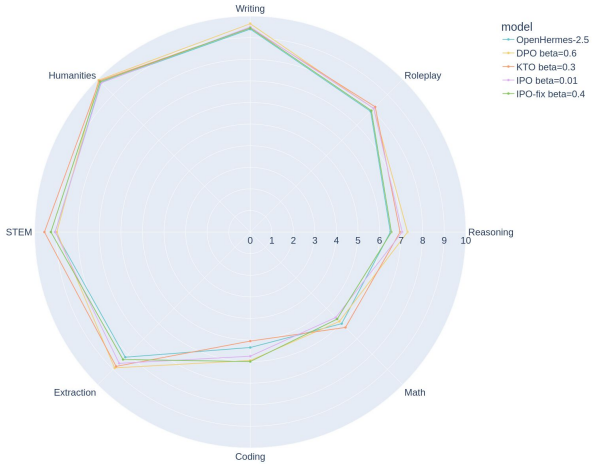
Empirical approximation to IPO loss

In the proposal 3, they show that the up to a constant $\mathcal{L}^{IPO}(\pi)$ can be written as:

$$\mathcal{L}^{IPO}_{empirical}(\pi) = \mathbb{E}_{x\sim\rho y,y'\sim\mu}\left[\left(h_\pi(y,y',x) - \frac{1}{2\beta}\right)^2\right].$$

IPO results





From huggingface portal on preference tuning: https://huggingface.co/blog/pref-tuning

ORPO

Let's define the key notations here again:

- x : the input (prompt).
- y_w : the chosen response to the prompt.
- y_l : the rejected (disfavored) response.
- $P_\theta(y|x)$: The probability of generating sequence "y" with the prompt "x".

The SFT loss could be written as:

$$\mathcal{L}(x, y; \theta) = -\frac{1}{m} \sum_{t=1}^{m} \log P_\theta(y_t|x, y_{<t}).$$

Why?

ORPO (Hong et al, 2024) provides an alternative to DPO without requiring a reference/base model to align the model with.

The ORPO is a monolithic alignment approach that allows us to align language models directly during supervised fine-tuning without requiring a separate reference model.

Key insight: Use odds ratio to discriminate between preferred and dis-preferred responses.

ORPO method

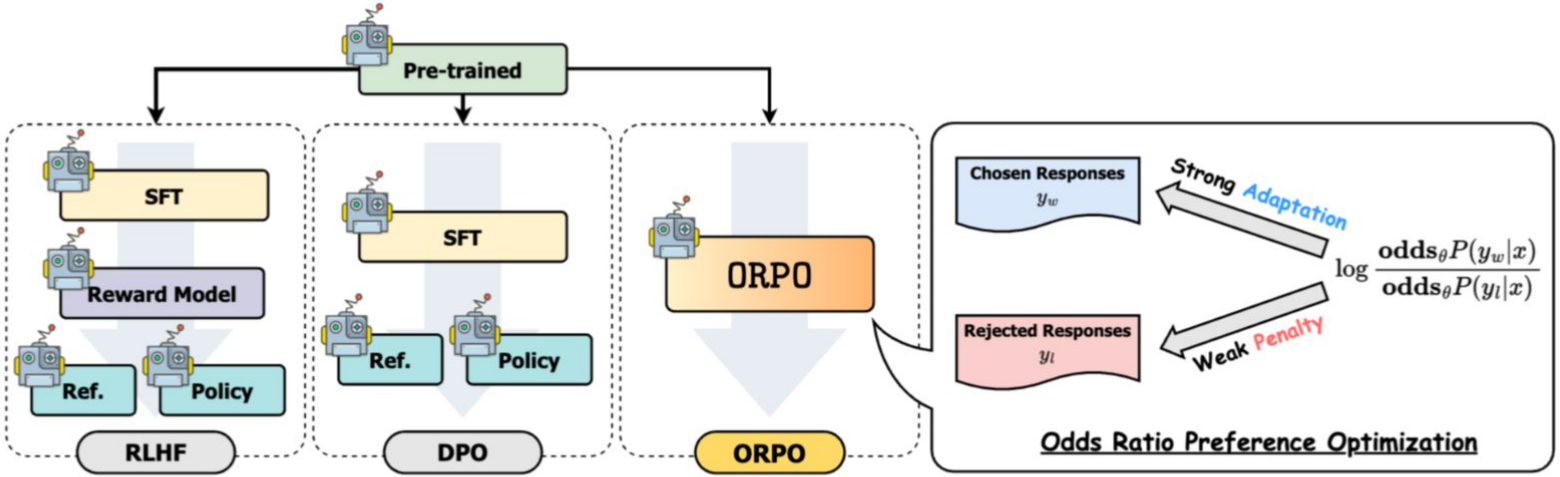


Figure 2: Comparison of model alignment techniques. ORPO aligns the language model *without a reference model* in a single-step manner by assigning a weak penalty to the rejected responses and a strong adaptation signal to the chosen responses with a simple log odds ratio term appended to the negative log-likelihood loss.

ORPO - Method

The odds of a language model for a completion “y” and prompt “x” could be written as:

$$\mathbf{odds}_\theta(y|x) = \frac{P_\theta(y|x)}{1 - P_\theta(y|x)},$$

Thus the odds ratio of the preferred and dispreferred responses would be:

$$\mathbf{OR}_\theta(y_w, y_l) = \frac{\mathbf{odds}_\theta(y_w|x)}{\mathbf{odds}_\theta(y_l|x)}.$$

$\mathbf{OR}_\theta(y_w, y_l)$ indicates how much more likely to generate the preferred completion over the dispreferred one.

The loss function of the ORPO can be written as: $\mathcal{L}_{ORPO}(y_w, y_l) = \mathbb{E}_{y_w, y_l}[\mathcal{L}_{SFT} + \lambda \mathcal{L}_{OR}],$

The loss for odds ratio can be written as: $\mathcal{L}_{OR} = -\log \sigma \left(\log \frac{\mathbf{odds}_\theta(y_w|x)}{\mathbf{odds}_\theta(y_l|x)} \right).$

ORPO Results

Model Name	Size	$\text{AlpacaEval}_{1.0}$	$\text{AlpacaEval}_{2.0}$
Phi-2 + SFT	2.7B	48.37% (1.77)	0.11% (0.06)
Phi-2 + SFT + DPO	2.7B	50.63% (1.77)	0.78% (0.22)
Phi-2 + ORPO *(Ours)*	2.7B	**71.80% (1.59)**	**6.35% (0.74)**
Llama-2 Chat *	7B	71.34% (1.59)	4.96% (0.67)
Llama-2 Chat *	13B	81.09% (1.38)	7.70% (0.83)
Llama-2 + ORPO *(Ours)*	7B	**81.26% (1.37)**	**9.44% (0.85)**
Zephyr (α) *	7B	85.76% (1.23)	8.35% (0.87)
Zephyr (β) *	7B	90.60% (1.03)	10.99% (0.96)
Mistral-ORPO-α *(Ours)*	7B	87.92% (1.14)	11.33% (0.97)
Mistral-ORPO-β *(Ours)*	7B	**91.41% (1.15)**	**12.20% (0.98)**

Table 1: Table of instruction-following abilities of each checkpoint measured through AlpacaEval. While clearly showing the improvements in instruction-following abilities after training with ORPO, it is notable that ORPO models exceed RLHF or DPO models of Llama-2 and Mistral (* indicates the results from the official leaderboard.)

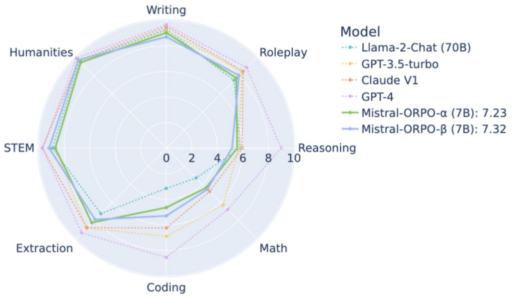


Figure 4: MT-Bench result of Mistral-ORPO-α (7B) and Mistral-ORPO-β (7B) by the category. Further comparison can be found in the Appendix G.

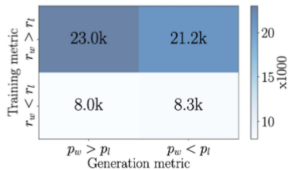
SimPO (Simple Policy Optimization)

With DPO type of alignment algorithms and the objectives LLMs optimize to maximize a proxy reward function that has nothing to do with the generation.

- The DPO objective is: $\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{base}}) = -\mathbb{E}_{x,y_w,y_l\sim\mathcal{D}}\left[\log\sigma\left(\beta\left(\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right)\right]$

- Generation objective: $\max_y p_\theta(y|x) = \max_y \frac{1}{|y|}\log\pi_\theta(y|x) = \max_y \frac{1}{|y|}\sum_{i=1}^{|y|}\log\pi_\theta(y_i|x, y_{<i})$

- After one epoch of DPO training, only about 50% of the training instances where $r_w > r_l$ satisfy $p_w > p_l$.



Meng et al 2024 proposed a method called SimPO for the simple preference optimization without needing to keep any reference model in the memory.

SimPO

SimPO reward function is written as: $r_{SimPO}(x, y) = \frac{\beta}{|y|} \log \pi_\theta(y|x) = \frac{\beta}{|y|} \sum_{i=1}^{|y|} \log \pi_\theta(y_i|x, y_{<i})$

Preference probabilities can be written as target reward margin: $p(y_w \succ y_l|x) = \sigma\left(r(x, y_w) - r(x, y_l) - \gamma\right)$

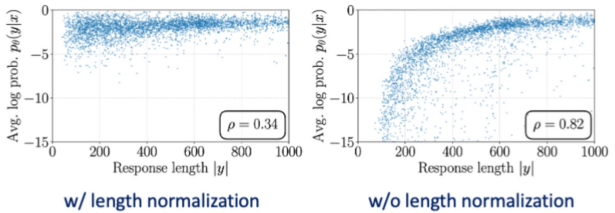
Ranking probability with a target reward margin.

SimPO loss can be written as:

$$\mathcal{L}_{SimPO}(\pi_\theta) = -\mathbb{E}\left[\log \sigma\left(\frac{\beta}{|y_w|} \log \pi_\theta(y_w|x) - \frac{\beta}{|y_w|} \log \pi_\theta(y_l|x) - \gamma\right)\right]$$

Sum of log-probs has a **length bias**: longer sequences tend to have lower (more negative) log probabilities.

The length-bias can cause very long generations(degeneration)



SimPO results

Baselines:

- Mistral/Llama-3 Base = start with pretrained models, do SFT w/ UltraChat (Ding et al., 2023) + *PO w/ UltraFeedback (Cui et al., 2023)
- Mistral/Llama-3 Instruct = start with instruction-tuned models, do *PO w/ on-policy UltraFeedback data annotated w/ PairRM (Jiang et al., 2023)

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]$$

$$\mathcal{L}_{\text{SimPO}}(\pi_\theta) = -\mathbb{E}\left[\log \sigma\left(\frac{\beta}{|y_w|} \log \pi_\theta(y_w \mid x) - \frac{\beta}{|y_l|} \log \pi_\theta(y_l \mid x) - \gamma\right)\right]$$

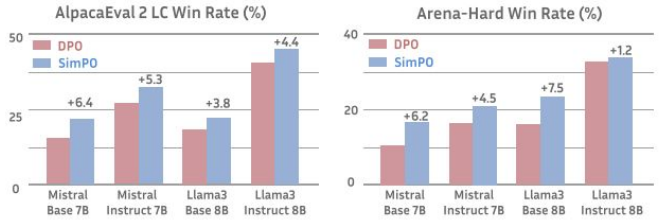


Figure 1: SimPO and DPO mainly differ in their reward formulation, as indicated in the shaded box. SimPO outperforms DPO across a wide range of settings on AlpacaEval 2 and Arena-Hard.

Cheaper than DPO.

KTO

DPO optimizes a model to maximize the likelihood of observed preferences under a Bradley-Terry model:

- They assume that humans follow a strictly rational, noise-free decision-making process.
- Preferences are transitive.

Thus DPO may not be great if those assumptions do not hold.
KTO fixes them with the ideas inspired from the prospect theory specifically utilizing Kahneman and Tversky's value function from cumulative prospect theory.

Prospect Theory

Prospect theory defines the utility ***u(z)*** of a random variable ***Z*** with outcomes ***z*** and value function ***v(z)*** as:

$$u(Z) = \sum_{z \in Z} \omega_z v(z - z_0)$$

KTO uses the following utility function with alpha~0.88 and lambda~2.25:

$$v(z; \lambda, \alpha, z_0) = \begin{cases} (z - z_0)^\alpha & \text{if } z \geq z_0 \\ -\lambda(z_0 - z)^\alpha & \text{if } z < z_0 \end{cases}$$

KTO Loss

Define reference-adjusted reward: $z = r_\theta(x, y) - z_0, \quad \text{where} \quad z_0 = \mathrm{KL}(\pi_\theta(y'|x) \| \pi_{\text{ref}}(y'|x))$

To ensure numerical stability, KTO replaces the original power-value function with a logistic sigmoid (σ) function that maintains concavity in gains and convexity in losses:

$$v(x,y) = \begin{cases} \lambda_D \, \sigma(\beta(r_\theta(x,y) - z_0)) & y \sim y_{\text{desirable}}|x \\ \lambda_U \, \sigma(\beta(z_0 - r_\theta(x,y))) & y \sim y_{\text{undesirable}}|x \end{cases}$$

λ_D, λ_U are the hyperparameters to control the importance of desirable and undesirable outcomes.

Then the final KTO loss is simply:

$$\mathcal{L}_{\text{KTO}}(\pi_\theta, \pi_{\text{ref}}) = \mathbb{E}_{x,y \sim D}[\lambda_y - v(x, y)]$$

Paper link: Ethayarajh et al, 2024.

KTO results

In the paper they show really promising results but mostly when they don't do SFT before doing DPO:

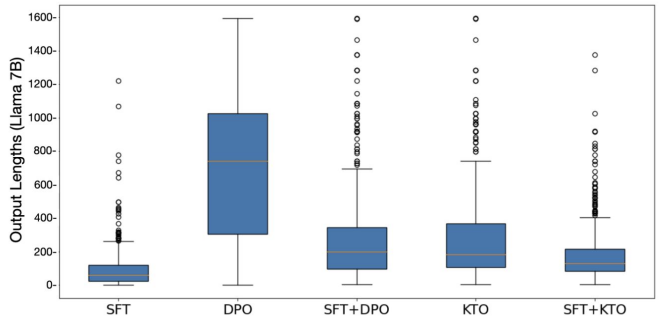


Figure 4. Without doing SFT first, DPO-aligned models tend to ramble and hallucinate entire conversations. KTO does not suffer from this issue.

Method	Winrate vs. SFT Target
Mistral-7B (unaligned)	0.525 ± 0.037
Mistral-7B + DPO	0.600 ± 0.037
Mistral-7B + KTO (all y per x)	**0.652 ± 0.036**
Mistral-7B + KTO (one y per x)	0.631 ± 0.036
Mistral-7B-Instruct	0.621 ± 0.031

Dataset (→) Metric (→)	MMLU EM	GSM8k EM	HumanEval pass@1	BBH EM
SFT	57.2	39.0	30.1	46.3
DPO	58.2	40.0	30.1	44.1
ORPO ($\lambda = 0.1$)	57.1	36.5	29.5	47.5
KTO ($\beta = 0.1$, $\lambda_D = 1$)	**58.6**	**53.5**	**30.9**	**52.6**
KTO (one-y-per-x)	58.0	50.0	30.7	49.9
KTO (no z_0)	58.5	49.5	30.7	49.0
KTO (concave, $v = \log \sigma$)	58.3	42.5	30.6	43.2
KTO (risk-neutral, $v(\cdot) = \cdot$)	57.3	42.0	28.8	6.1
KTO (no π_{ref}, $\lambda_D = 1.75$)	57.5	47.5	29.5	51.6
KTO ($\beta = 0.01$)	57.3	54.0	33.2	49.8
KTO ($\beta = 0.50$)	58.3	45.0	32.7	51.8
KTO ($\lambda_D = 0.50$)	57.4	47.0	32.2	30.5
KTO ($\lambda_D = 0.75$)	58.3	52.0	30.3	50.6
KTO ($\lambda_D = 1.33$)	58.5	52.0	32.8	45.4
KTO ($\lambda_D = 2.00$)	58.1	43.0	32.1	45.3

SLIC

SLIC calibrates the sequence likelihood of a supervised fine-tuned (SFT) model to align more closely with human preferences using a max-margin loss.

SLiC-HF aims to calibrate the model's sequence probabilities such that:

- More preferred sequences have higher likelihood under the calibrated model.
- Less preferred sequences have lower likelihood.

Supervised Fine-Tuning (Initial Model): Initially, a model is trained by maximizing the likelihood of reference sequences:

$$\theta_{\text{ft}} = \arg\max_{\theta} E_{(x,y_{\text{ref}})\sim D_{\text{SFT}}}[\log P_\theta(y_{\text{ref}}|x)]$$

Calibration loss: The model is trained to optimize a margin-based ranking loss:

$$\mathcal{L}_{\text{cal}}(\theta) = \max(0, \delta - \log P_\theta(y^+|x) + \log P_\theta(y^-|x))$$

Paper link: Zhao et al, 2023

Regularization Term: Regularization term is used to ensure the calibrated model does not deviate too far from the supervised model. The regularization is done via a standard cross-entropy loss with a target sequence (often the supervised reference (y_{ref}):

$$L_{\text{reg}}(\theta) = -\log P_\theta(y_{\text{ref}}|x)$$

Complete SLiC-HF Objective: Combining the calibration loss and the regularization loss gives the full SLiC-HF loss function:

$$\mathcal{L}_{SLIC}(\theta) = \underbrace{\max(0, \delta - \log P_\theta(y^+|x) + \log P_\theta(y^-|x))}_{\text{Calibration Loss}} + \underbrace{\lambda(-\log P_\theta(y_{\text{ref}}|x))}_{\text{Regularization Loss}}$$

Two variants of SLIC based on the selection of pairwise pair selection:

1. **SLiC-HF-direct:** Directly uses pairs from human feedback data without additional steps. Riskier if feedback data distribution differs significantly from model-generated distribution.
2. **SLiC-HF-sample-rank:** (i) Sample multiple candidates from the supervised fine-tuned model. (ii) Rank these candidates using either a trained reward model or a pairwise ranking model. (iii) Select pairs of top-ranked ("positive") and bottom-ranked ("negative") sequences based on this ranking.

SLIC Results

Table 2: 4-way human evaluation to compare reference, SFT continue SFT on best decodes using ranking model, SLiC-HF with pairs of decodes using ranking model.

	reference	SFT	continue SFT	SLiC-HF	same
chosen as preferred %	13%	5%	5%	**73%**	4%
average quality	3.17	3.10	3.32	**3.82**	-
is factual %	94.16%	94.85%	94.85%	**96.56%**	-

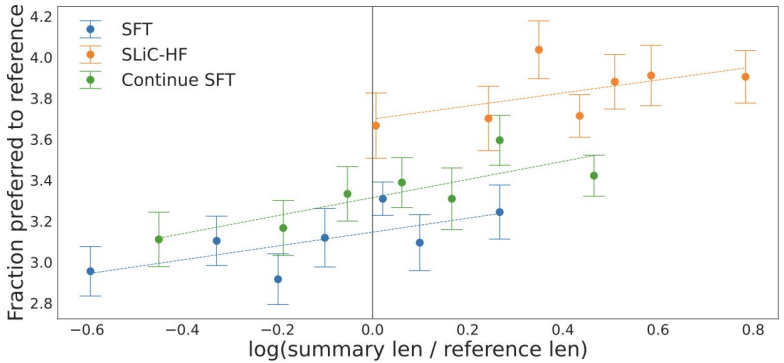


Figure 2: Length bucketed average quality of SFT and SLiC-HF against different baselines.

Ablation				Metrics	
method	# params	m	# words	R1 / R2 / RL	ranker win rate
SFT	770M	8	23.57	35.1/12.87/26.81	44.96%
SFT	11B	8	24.07	36.45/14.11/28.38	62.34%
SLiC-HF	770M	8	37.96	34.49/11.92/25.35	86.21%
SLiC-HF	770M	64	40.53	34.14/11.70/25.11	86.41%
SLiC-HF	11B	8	36.90	35.83/12.87/26.63	**96.10%**

GPO

Given a dataset of response pairs (y_w, y_l) (winning and losing responses, respectively), drawn from an offline distribution μ, the log-ratio difference can be defined as:

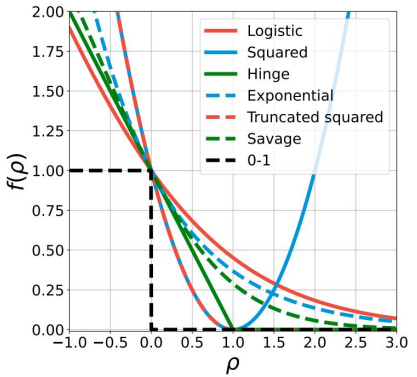
$$\rho_\theta(y_w, y_l) = \log \frac{\pi_\theta(y_w)}{\pi_{\text{ref}}(y_w)} - \log \frac{\pi_\theta(y_l)}{\pi_{\text{ref}}(y_l)}.$$

Thus, a general offline preference optimization is given by:

$$\mathcal{L}_{\text{GPO}}(\theta) = \mathbb{E}_{(y_w, y_l)\sim\mu}\left[f(\beta\rho_\theta(y_w, y_l))\right],$$

where $f : \mathbb{R} \to \mathbb{R}$ is a convex loss function.

Loss Name	Mathematical Form	Corresponding Algorithm
Logistic log-loss	$f(x) = \log(1 + e^{-x})$	DPO
Hinge loss	$f(x) = \max(0, 1 - x)$	SLiC
Squared loss	$f(x) = (x - 1)^2$	IPO
Exponential loss	$f(x) = e^{-x}$	New (AdaBoost-like)
Truncated quadratic loss	$f(x) = (\max(0, 1 - x))^2$	New (robust variant)
Savage loss	$f(x) = 1/(1 + e^x)^2$	New (robust variant)



Paper link: Tang et al, 2024

GPO Results

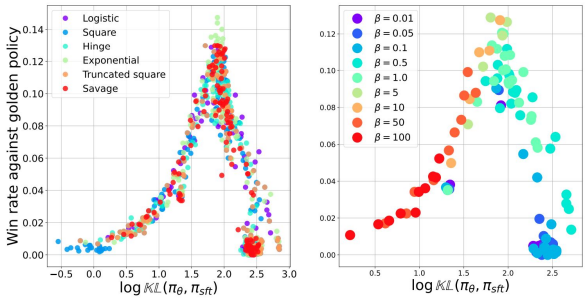
The analysis reveals that the convex loss function's curvature (specifically, its first two derivatives around zero) determines the implicit regularization strength in offline preference optimization:

$$\text{Regularization strength} \propto \left|\frac{f''(0)}{f'(0)}\right| \beta.$$

Thus choosing the f directly influences how strongly the policy is regularized to stay close to the reference policy.

Empirical evidence confirms that different convex losses inherently differ in the strength of regularization they enforce, even when using the same hyperparameter β.

In practice, the optimal algorithmic choice among GPO variants depends less on performance differences and more on hyperparameter tuning convenience and robustness.



$$\pi_{t+1} = \arg\max_{\pi \in \Pi} \mathbb{E}_{x,y\sim\pi(\cdot|x)}[r(x,y)] - \frac{1}{\beta}\mathbb{E}_x[KL(\pi(\cdot|x)\|\pi_t(\cdot|x))]$$

The solution has a known closed form (mirror descent step):

$$\pi_{t+1}(y|x) = \frac{\pi_t(y|x)\exp(\beta r(x,y))}{Z(x)}, \quad Z(x) = \sum_y \pi_t(y|x)\exp(\beta r(x,y))$$

By inverting this optimality condition:

$$r(x,y) = \frac{1}{\beta}\left(\ln(Z(x)) + \ln\frac{\pi_{t+1}(y|x)}{\pi_t(y|x)}\right)$$

Due to intractability of $Z(x)$, REBEL proposes regressing differences in reward to eliminate $Z(x)$:

At iteration t, given dataset $D_t = \{(x,y,y')|x\sim\rho, y\sim\pi_t(\cdot|x), y'\sim\mu(\cdot|x)\}$, solve:

$$\theta_{t+1} = \arg\min_\theta \sum_{(x,y,y')\in D_t}\left[\frac{1}{\beta}\left(\ln\frac{\pi_\theta(y|x)}{\pi_{\theta_t}(y|x)} - \ln\frac{\pi_\theta(y'|x)}{\pi_{\theta_t}(y'|x)}\right) - (r(x,y) - r(x,y'))\right]^2$$

- This formulation regresses relative differences of log probabilities to the differences in rewards.

REBEL Pseudocode

Algorithm 1 REgression to RElative REward Based RL (REBEL)

Input: Reward r, policy class $\Pi = \{\pi_\theta : \theta \in \Theta\}$, base distribution μ, learning rate η
Initialize policy π_{θ_0}.
for $t = 0$ to $T - 1$ **do**
  // Base distribution μ can either be an offline dataset or π_t.
  Collect dataset $\mathcal{D}_t = \{x, y, y'\}$ where $x \sim \rho, y \sim \pi_t(\cdot|x), y' \sim \mu(\cdot|x)$.
  Solve square loss regression problem:

$$\theta_{t+1} = \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{(x,y,y') \in \mathcal{D}_t} \left(\frac{1}{\eta} \left(\ln \frac{\pi_\theta(y|x)}{\pi_{\theta_t}(y|x)} - \ln \frac{\pi_\theta(y'|x)}{\pi_{\theta_t}(y'|x)} \right) - (r(x,y) - r(x,y')) \right)^2. \tag{1}$$

end for

Paper link: Gao et al, 2024

REBEL results

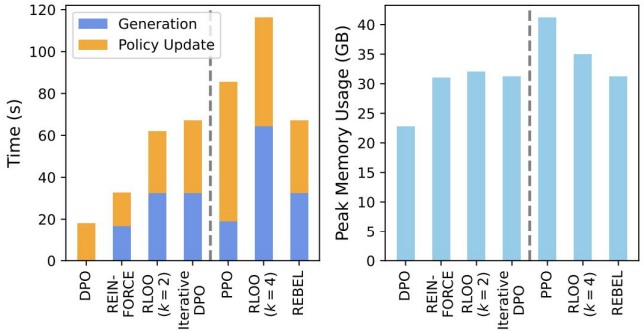


Figure 2: **Plot of runtime and memory usage.** Baselines on the left-hand side of the dashed line have lower winrates. Methods on the right-hand side of the dashed line have similar winrates to REBEL.

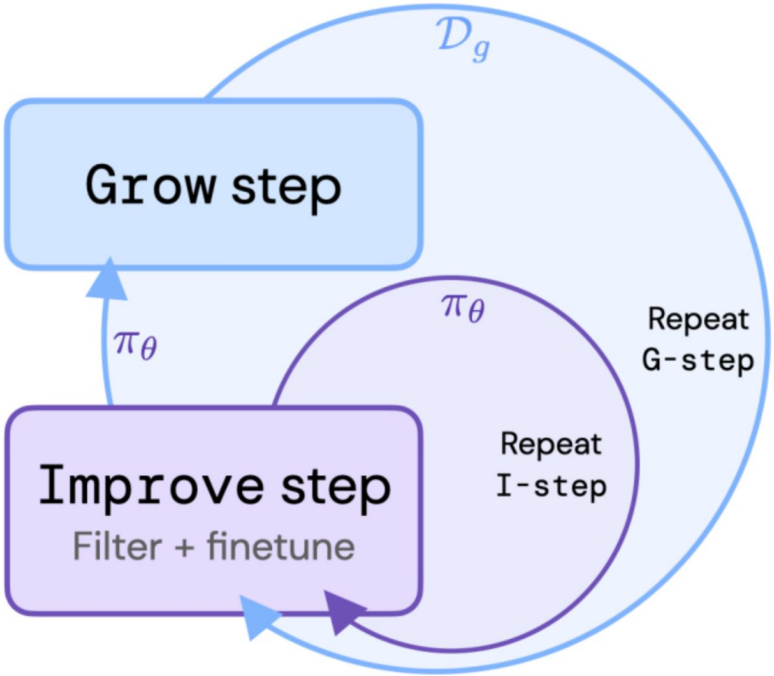
Model size	Algorithm	Winrate (↑)	RM Score (↑)	KL($\pi \| \| \pi_{ref}$) (↓)
1.4B	SFT	24.9 (±2.73)	-0.51 (±0.05)	-
	DPO	42.7 (±1.79)	0.10 (±0.02)	29.6 (±0.63)
	Iterative DPO	47.2 (±1.34)	1.73 (±0.05)	29.7 (±0.57)
	PPO	51.7 (±1.42)	1.74 (±0.04)	**29.3** (±0.61)
	REBEL	**55.1** (±1.35)	**1.84** (±0.04)	32.6 (±0.59)
2.8B	SFT	28.2 (±2.31)	-0.38 (±0.06)	-
	DPO	53.7 (±1.63)	2.40 (±0.02)	64.3 (±1.25)
	Iterative DPO	63.1 (±1.41)	2.37 (±0.03)	28.1 (±0.51)
	PPO	67.4 (±1.30)	2.37 (±0.03)	**27.2** (±0.55)
	REBEL	**70.2** (±1.32)	**2.44** (±0.02)	29.0 (±0.60)

Model size	Algorithm	Winrate (↑)
6.9B	SFT	45.2 (±2.49)
	DPO	68.4 (±2.01)
	REINFORCE	70.7*
	PPO	77.6‡
	RLOO ($k = 2$)	74.2*
	RLOO ($k = 4$)	77.9*
	REBEL	**78.1** (±1.74)

* directly obtained from Ahmadian et al. (2024)
‡ directly obtained from Huang et al. (2024)

Offline RL based approaches to RL

Reinforced Self-Training (ReST)



- Decouples the data generation and finetuning process to improve the model.
- Simple and efficient idea.
- Unlike PPO and the instruct GPT recipe, the model does not have to be on-policy. Can make use of the offline data.

Paper link: Gulcehre et al, 2023

Grow and Improve steps

Grow: The language model policy is used to generate multiple output predictions for each context to **augment** the training dataset.

Improve: Rank and filter the augmented dataset with a scoring function. Then finetune.

Scoring function can either be:

- Learned from human preferences
- Fixed heuristics, i.e BLEU/ROUGE score

Improve Step

During the Improve step (exploitation or policy improvement in RL), the goal is:

- Use the dataset dataset D_g to fine-tune the policy π_θ. Produced dataset is filtered based on rewards lying above a threshold τ:

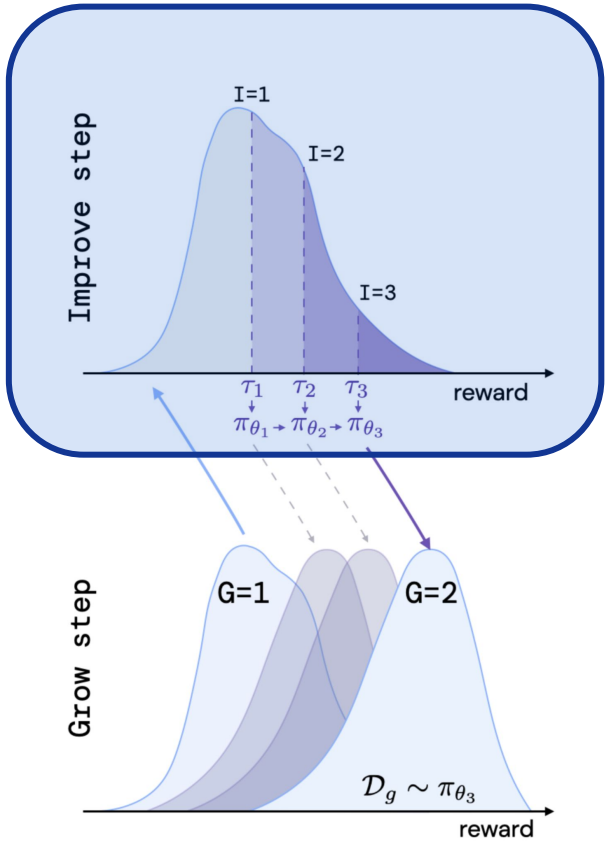
$$F(x, y; \tau) = \mathbb{1}_{R(x,y)>\tau}$$

To sum up representation of the reward weighted loss:

$$J(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_g}\left[F(x, y; \tau)\ \mathcal{L}(x, y; \theta)\right]$$

The typical loss we used in our experiments is the SFT loss.

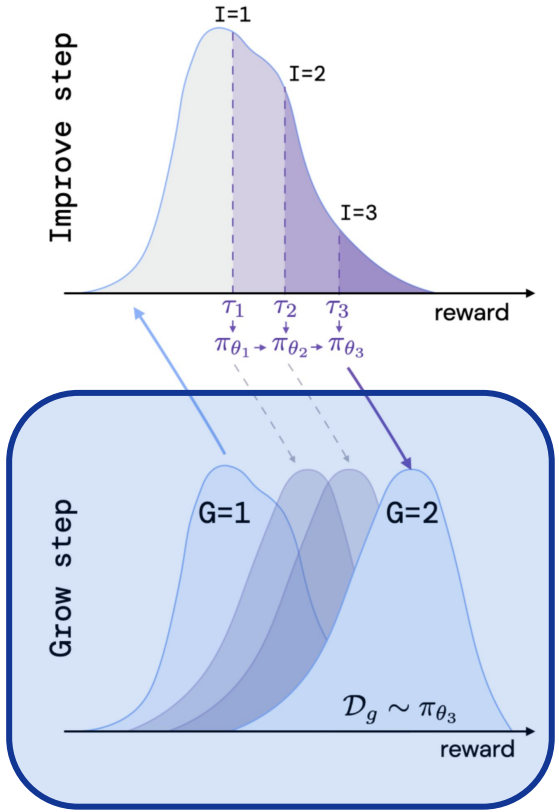
Very similar to filtered BC.



Grow Step

Exploration: Grow-step corresponds to the data-generation (or acting) step in RL.

- Augmented dataset of trajectories is created by sampling many outputs from the LLM.
- The new dataset of sequences are then scored with a reward function R(x, y).
- The completions from the LLM above a certain threshold score are used to update the policy.
- Once the policy is improved, a new dataset of better quality samples can be generated from the model



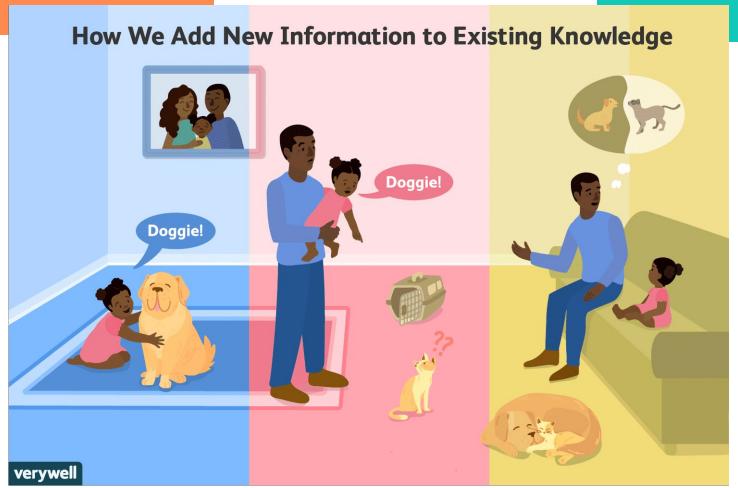
Constructivist learning theory and LLMs

Assimilation Phase

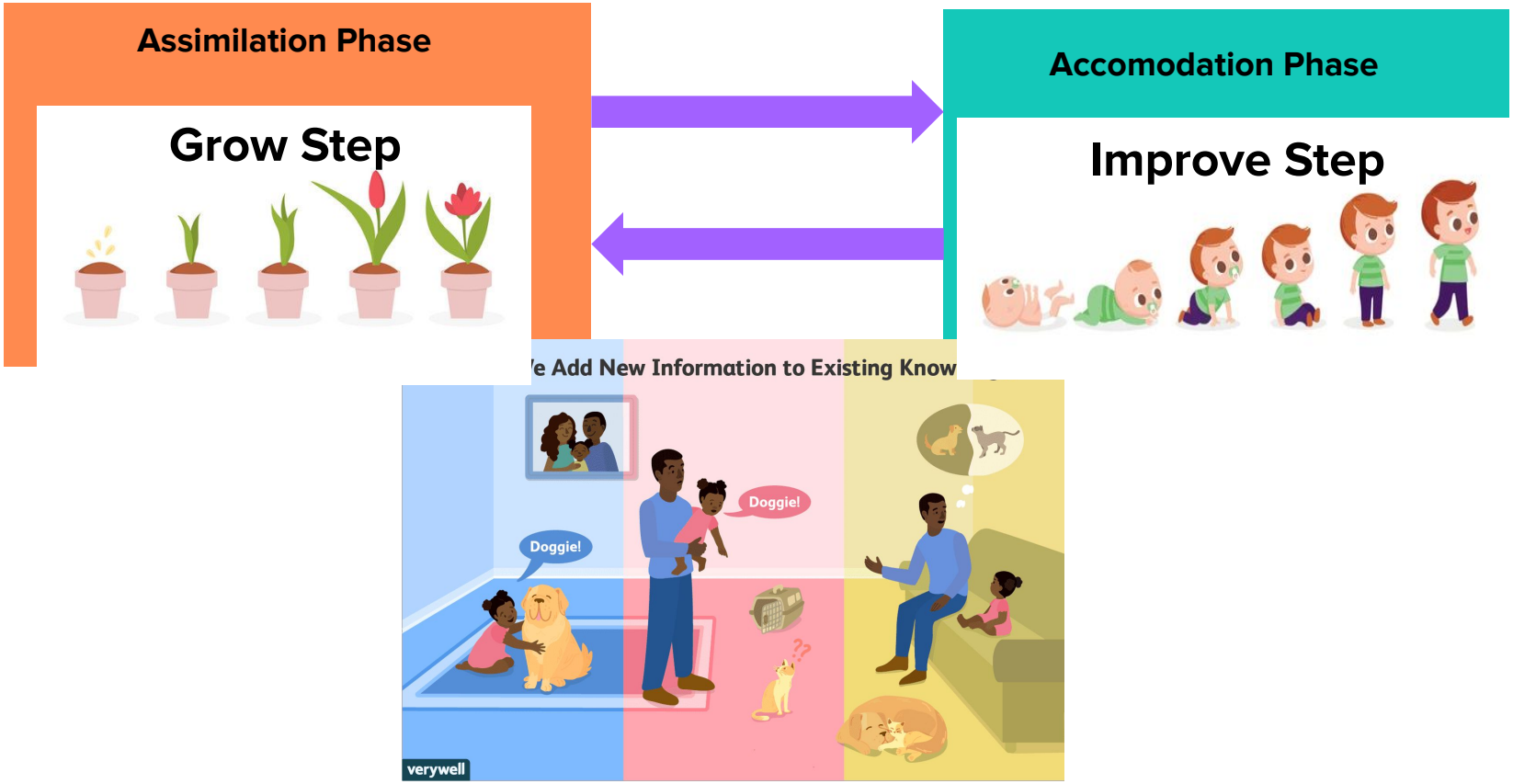
Occurs when new information is incorporated into existing cognitive structures without altering them significantly. The learner fits new experiences into their pre-existing schemas (mental models).

Accomodation Phase

Happens when new information does not fit into existing schemas, forcing the learner to modify or create new cognitive structures. This is how learners adapt and expand their understanding.



Constructivist learning theory and LLMs



ReST framework

Algorithm 1: *ReST* algorithm. *ReST* is a growing-batch RL algorithm. Given an initial policy of reasonable quality (for example, pre-trained using BC) iteratively applies `Grow` and `Improve` steps to update the policy. Here F is a filtering function, and $\mathcal{L}$ is an loss function.

Input: $\mathcal{D}$: Dataset, $\mathcal{D}_{eval}$: Evaluation dataset, $\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}; \theta)$: loss, $R(\boldsymbol{x}, \boldsymbol{y})$: reward model, G: number of grow steps, I: number of improve steps, N: number of samples per context

Train π_θ on $\mathcal{D}$ using loss $\mathcal{L}$.

for $g = 1$ *to* G **do**

  `// Grow`

  Generate dataset $\mathcal{D}_g$ by sampling: $\mathcal{D}_g = \{ (\boldsymbol{x}^i, \boldsymbol{y}^i)|_{i=1}^{N_g} \text{ s.t. } \boldsymbol{x}^i \sim \mathcal{D},\ \boldsymbol{y}^i \sim \pi_\theta(\boldsymbol{y}|\boldsymbol{x}^i) \} \cup \mathcal{D}$.

  Annotate $\mathcal{D}_g$ with the reward model $R(\boldsymbol{x}, \boldsymbol{y})$.

  for $i = 1$ *to* I **do**

    `// Improve`

    Choose threshold s.t. $\tau_1 > V_{\pi_\theta}$ for $V_{\pi_\theta} = \mathbb{E}_{\mathcal{D}_g}[R(\boldsymbol{x}, \boldsymbol{y})]$ and $\tau_{i+1} > \tau_i$.

    while *reward improves on* $\mathcal{D}_{eval}$ **do**

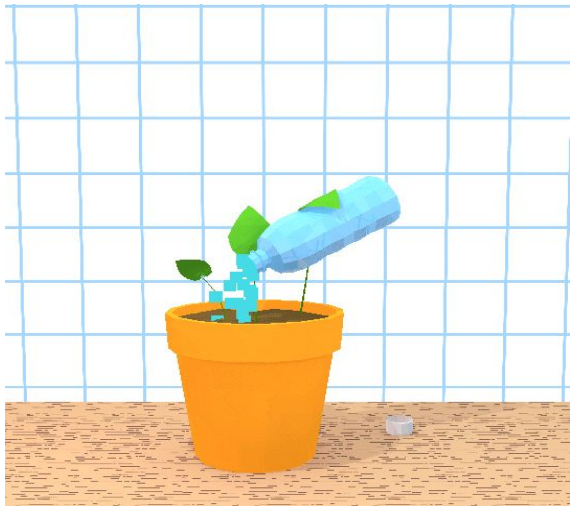
      Optimise θ on objective: $J(\theta) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_g}[F(\boldsymbol{x}, \boldsymbol{y}; \tau_i)\ \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}; \theta)]$

    end

  end

end

Output: Policy π_θ



ReST framework

Algorithm 1: *ReST* algorithm. *ReST* is a growing-batch RL algorithm. Given an initial policy of reasonable quality (for example, pre-trained using BC) iteratively applies `Grow` and `Improve` steps to update the policy. Here F is a filtering function, and $\mathcal{L}$ is an loss function.

Input: $\mathcal{D}$: Dataset, $\mathcal{D}_{eval}$: Evaluation dataset, $\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}; \theta)$: loss, $R(\boldsymbol{x}, \boldsymbol{y})$: reward model, G: number of grow steps, I: number of improve steps, N: number of samples per context

Train π_θ on $\mathcal{D}$ using loss $\mathcal{L}$.

for $g = 1$ *to* G **do**

  `// Grow`

  Generate dataset $\mathcal{D}_g$ by sampling: $\mathcal{D}_g = \{ (\boldsymbol{x}^i, \boldsymbol{y}^i)|_{i=1}^{N_g}$ s.t. $\boldsymbol{x}^i \sim \mathcal{D},\ \boldsymbol{y}^i \sim \pi_\theta(\boldsymbol{y}|\boldsymbol{x}^i) \} \cup \mathcal{D}$.

  Annotate $\mathcal{D}_g$ with the reward model $R(\boldsymbol{x}, \boldsymbol{y})$.

  for $i = 1$ *to* I **do**

    `// Improve`

    Choose threshold s.t. $\tau_1 > V_{\pi_\theta}$ for $V_{\pi_\theta} = \mathbb{E}_{\mathcal{D}_g}[R(\boldsymbol{x}, \boldsymbol{y})]$ and $\tau_{i+1} > \tau_i$.

    while *reward improves on* $\mathcal{D}_{eval}$ **do**

      Optimise θ on objective: $J(\theta) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_g}[F(\boldsymbol{x}, \boldsymbol{y}; \tau_i) \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}; \theta)]$

    end

  end

end

Output: Policy π_θ

Very flexible, the inputs can be any modality.

The loss function can be:

- Off-policy/Offline RL loss functions.
- BC loss: learning to imitate.

Reward model: Reference-free vs Reference-based

1) **Reference-free reward** models estimate of how good a translation is based on the source and candidate sentences only.
   a) more flexible since they do not require reference translations
   b) open possibilities to evaluate and discover candidate translations that supersede the quality of references

2) **Reference-based reward models** additionally use the reference human translation to decide how good a candidate translation is.
   a) assign higher scores for sentences similar to the reference, and this could happen even when the reference is erroneous or incomplete.

Reward-Model: Metric-X XXL

- We did careful scaling analysis on the reward model:
  - Trained reward models from 1B-32B
  - Consistent improvements as the reward model size increase.
  - We ended up using

- Reward model is a massively, multi-task, multi-lingual model.

- Our collaborator from Translate team submitted our reward model to WMT'22 Metrics challenge and it won the first place:

Metric	avg rank
METRICX XXL	1.20
COMET-22	1.32
UNITE	1.86
BLEURT-20	1.91
COMET-20	2.36
MATESE	2.57
COMETKIWI*	2.70
MS-COMET-22	2.84
UNITE-SRC*	3.03
YISI-1	3.27
COMET-QE*	3.33
MATESE-QE*	3.85
MEE4	3.87
BERTSCORE	3.88
MS-COMET-QE-22*	4.06
CHRF	4.70
F101SPBLEU	4.97
HWTSC-TEACHER-SIM*	5.17
BLEU	5.31
REUSE*	6.69

For more details: Results of WMT22 Metrics Shared Task: Stop Using BLEU – Neural Metrics Are Better and More Robust - ACL Anthology

Why ReST?

ReST is more efficient than typical online RLHF methods because the training **dataset is produced offline**, which allows **data reuse.**

Let's have some ReST!

- The computational burden is significantly reduced compared to online RL since the output of Grow step can be exploited across several Improve steps.

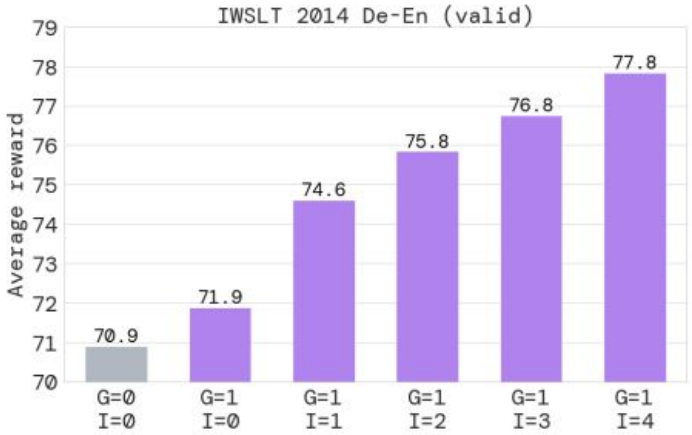
- The quality of the policy is not restricted by the quality of the original dataset since the original training data is augmented with with the improved model during the Grow step.

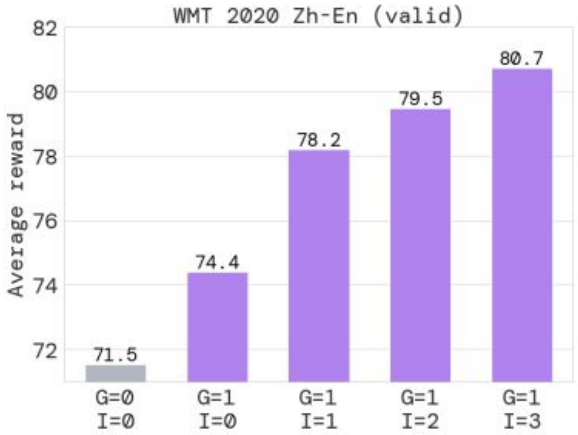
- Since the Grow and Improve steps are neatly decoupled, it is easy to inspect the data quality and potentially diagnose the alignment issues, e.g. reward hacking and intervene.

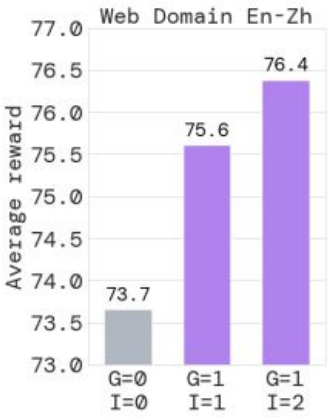
	Exploration	Rewards	Compute Efficient
Supervised Learning	✗	✗	✓
Self-training	✓	✗	✓
Online RL	✓	✓	✗
Offline RL	✗	✓	✓
ReST	✓	✓	✓

ReST Results

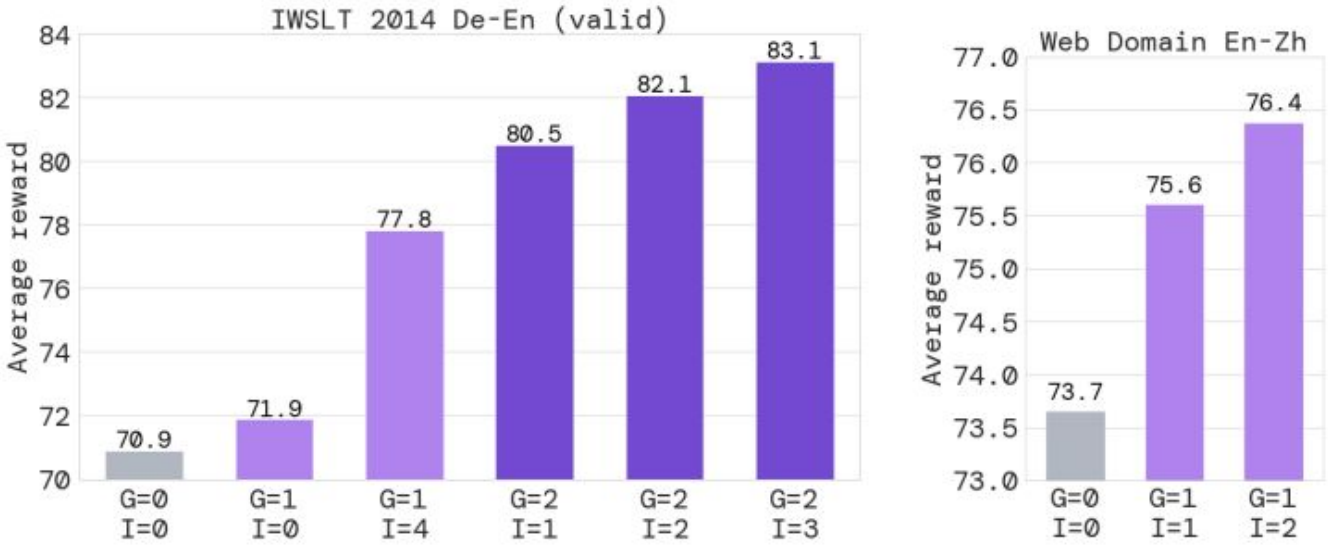
What happens as we increase the Improve steps?



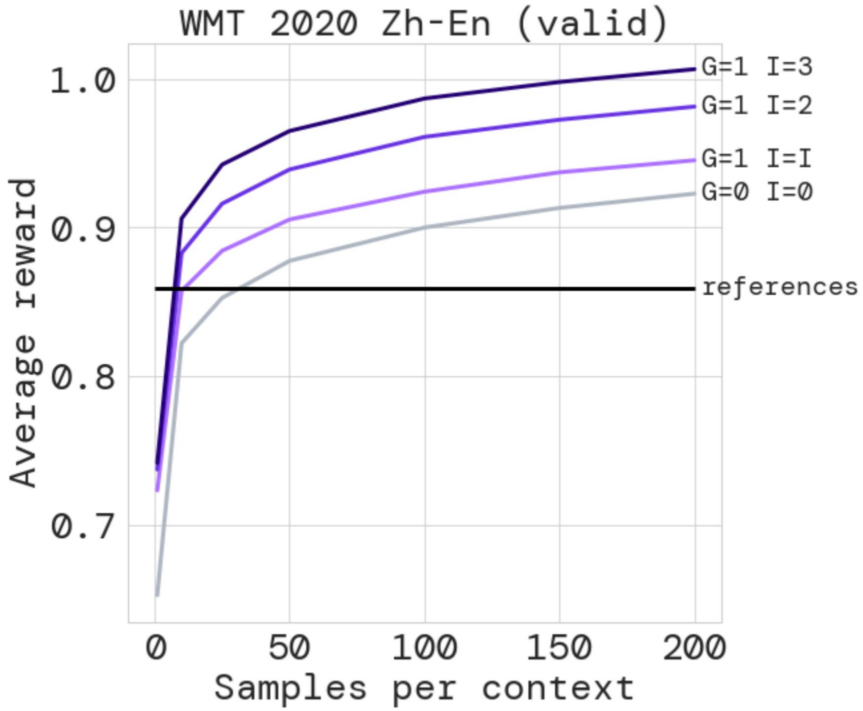




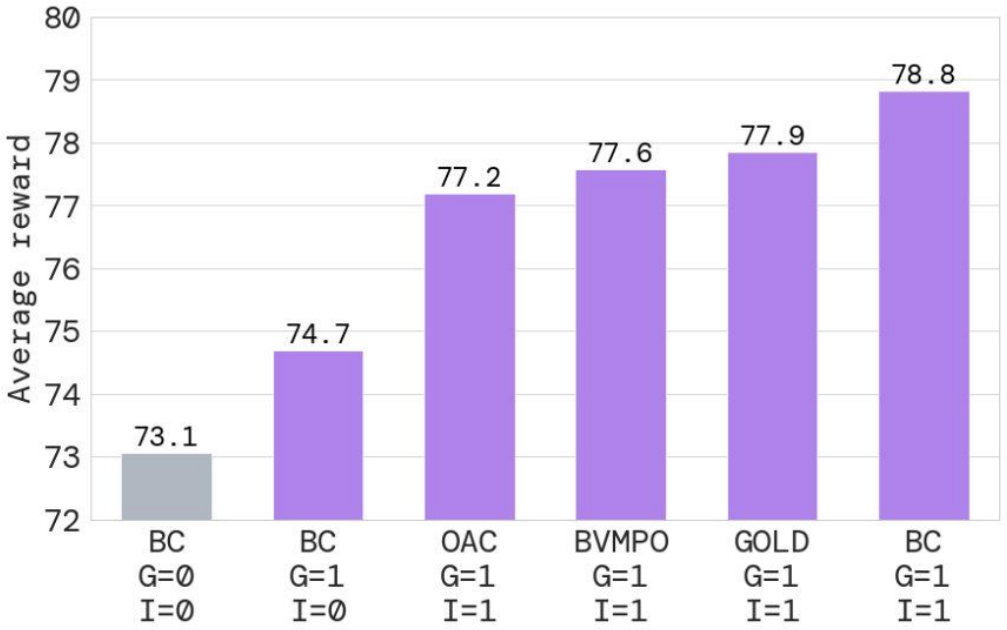
Effect of increasing the number of grow steps



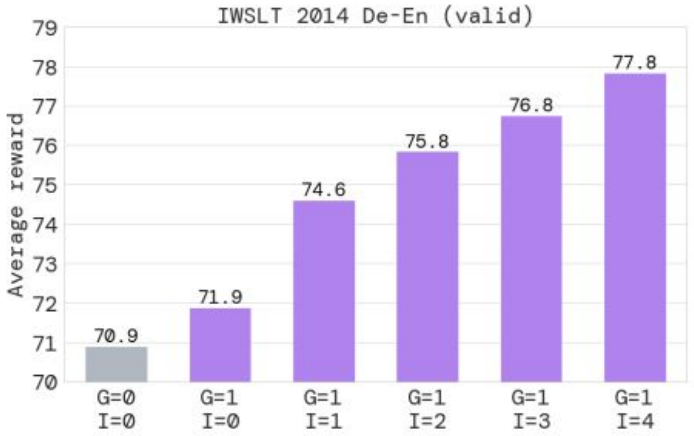
Impact of Best-of-N Sampling at inference time

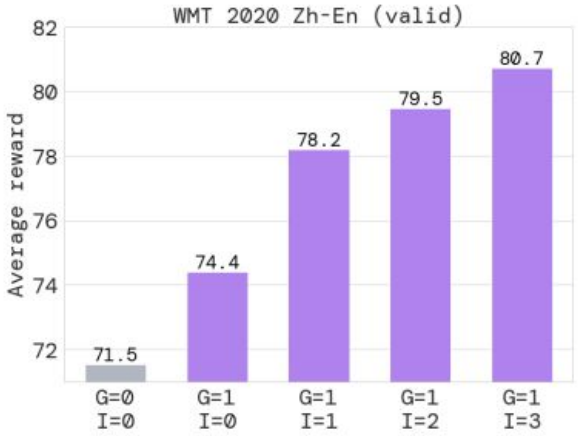


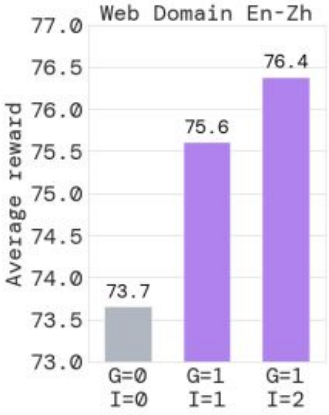
Different loss functions



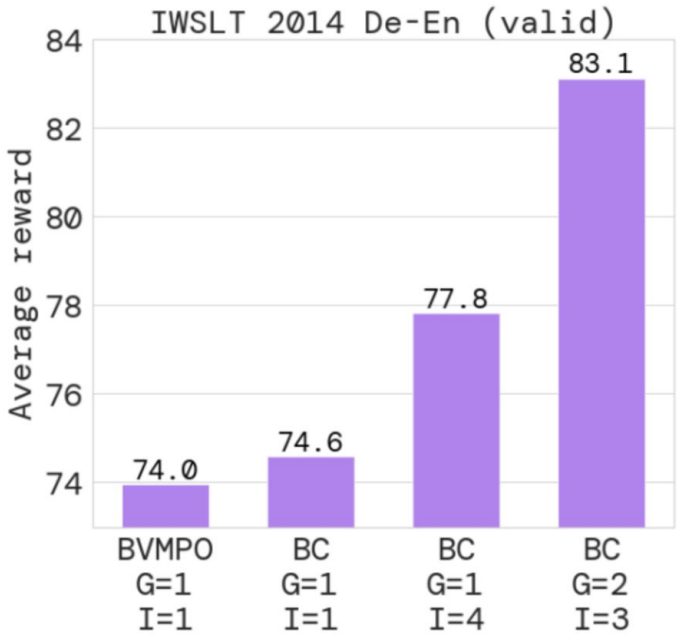
ReST is the Best!

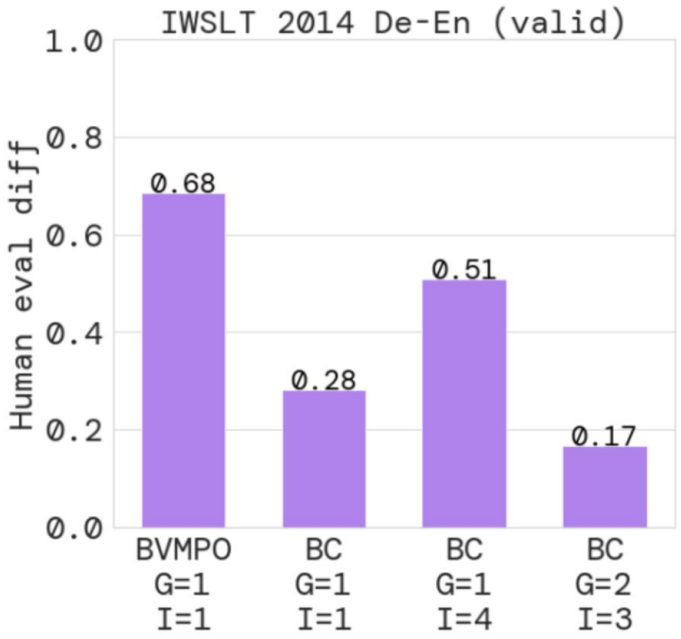






Human evaluations

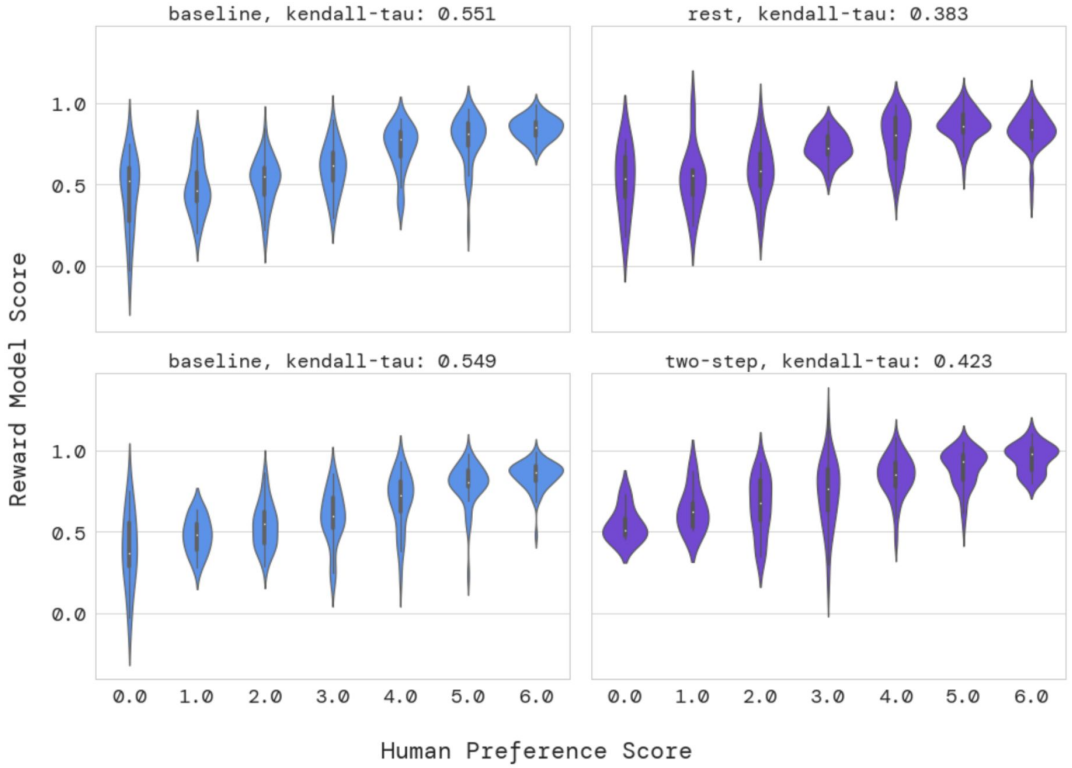




How does it compare to online RL?

Algorithm	Average Reward	Distinct samples
BC (G=0, I=0)	70.9	16 000 000
ReST (G=1, I=0)	71.9	16 000 000
ReST (G=1, I=4)	77.8	16 000 000
***ReST* (G=2, I=3)**	**83.1**	32 000 000
Online RL	71.6	24 000 000

Assessing the reward model quality



What did not work?

Learning Q is hard and offline RL losses are not necessary

Learning a value function that generalizes well for language is incredibly hard!

Reasons:

- Large action space
- Sparse reward feedback
- Deadly triad
- Deterministic nature of one-step RL.

Decision Transformers

Returned conditioned models learn suboptimal policies with sparse continuous rewards.

Relationship to the Policy Gradients

$$\mathbb{E}_{x\sim\mathcal{D},y\sim\pi_\theta(.|x)}[R(y,x)\nabla_\theta \log \pi_\theta(y|x)]$$

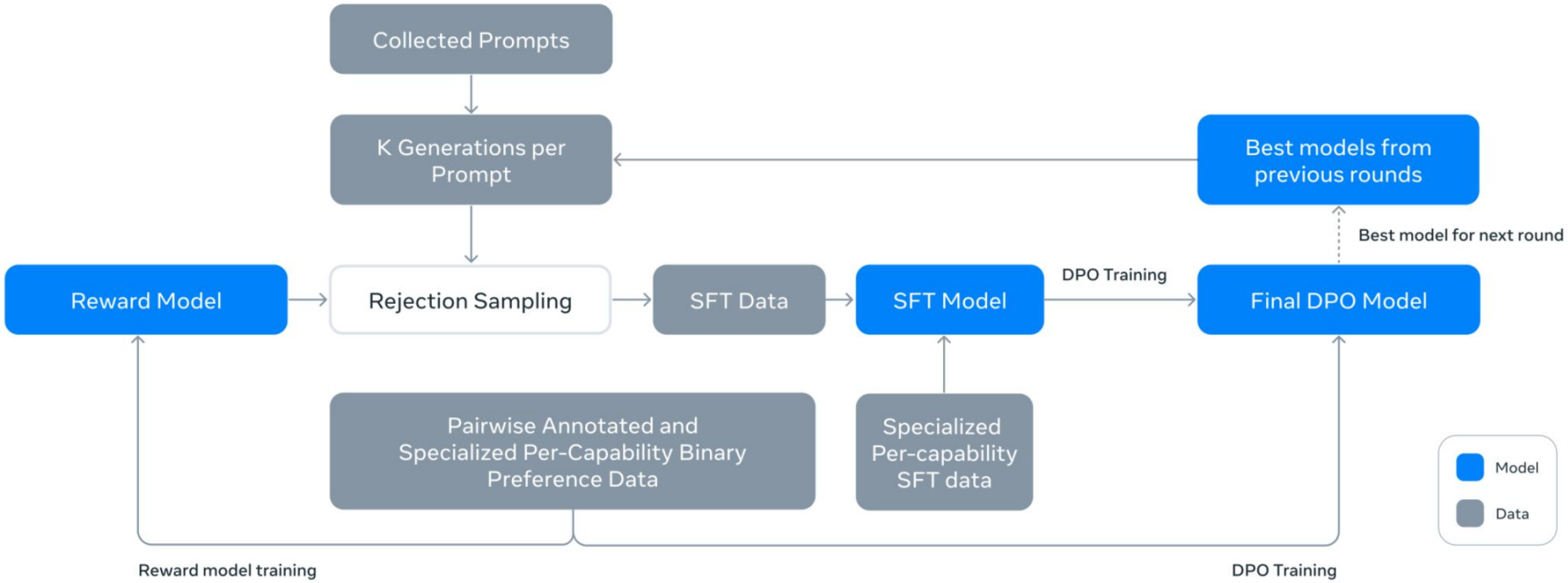
Relationship to the Policy Gradients

$$\mathbb{E}_{x\sim\mathcal{D},y\sim\pi_\theta(.|x)}[R(y,x)\nabla_\theta \log \pi_\theta(y|x)]$$

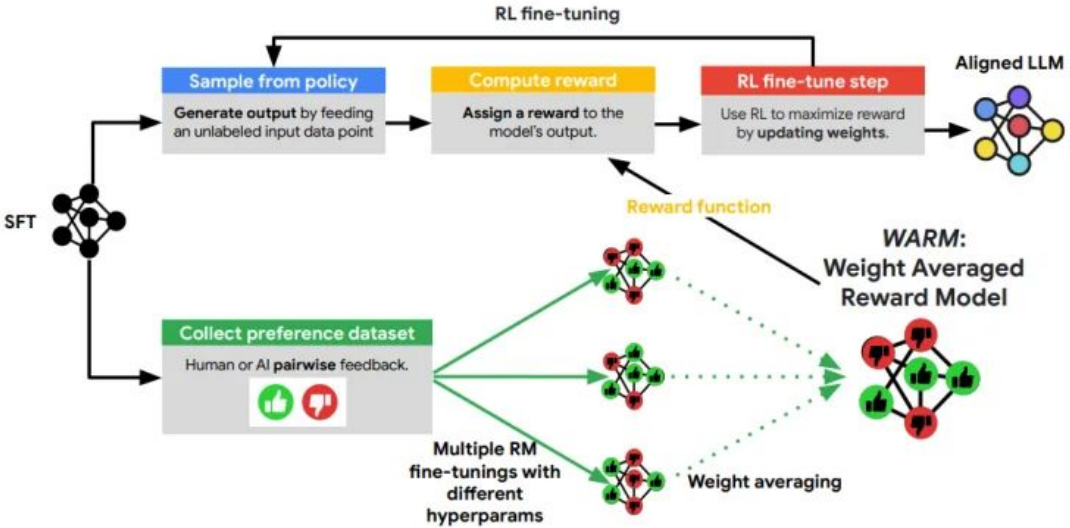
$$\nabla J(\theta) = -\mathbb{E}_{x\sim\mathcal{D}}\left[\lambda \mathbb{E}_{y\sim\pi_{\theta'}(y|x)}\left[F(x,y;\tau)\nabla \log \pi_\theta(y \mid x)\right] + (1-\lambda)\mathbb{E}_{y\sim p(y|x)}\left[F(x,y;\tau)\nabla \log \pi_\theta(y \mid x)\right]\right]. \quad (3)$$

Alternative Approaches to alignment

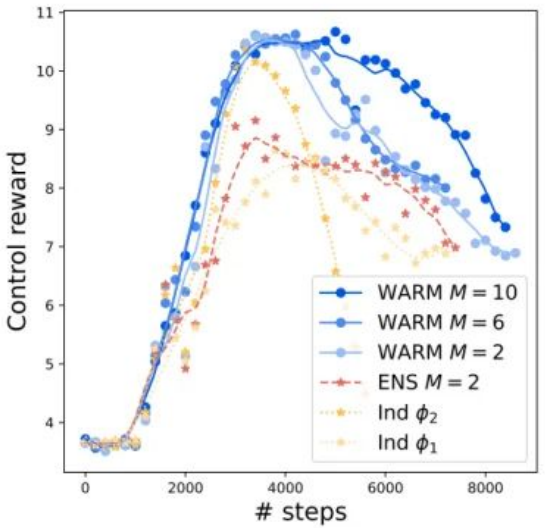
llama 3.1 is using ReST for alignment



WARM



(a) *WARM* procedure with $M = 3$.



(b) *WARM* mitigates reward hacking.

Paper link: Rame et al, 2024

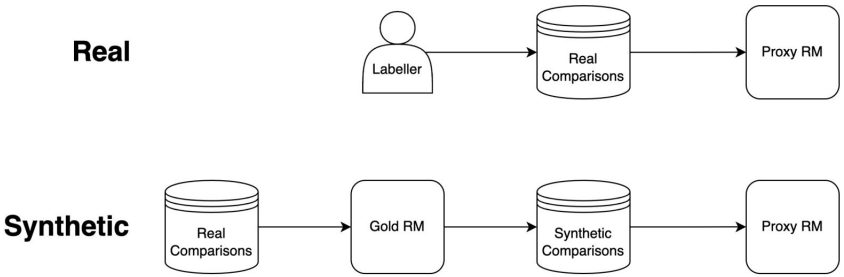
Understanding Alignment and challenges

Scaling laws for reward model overoptimization

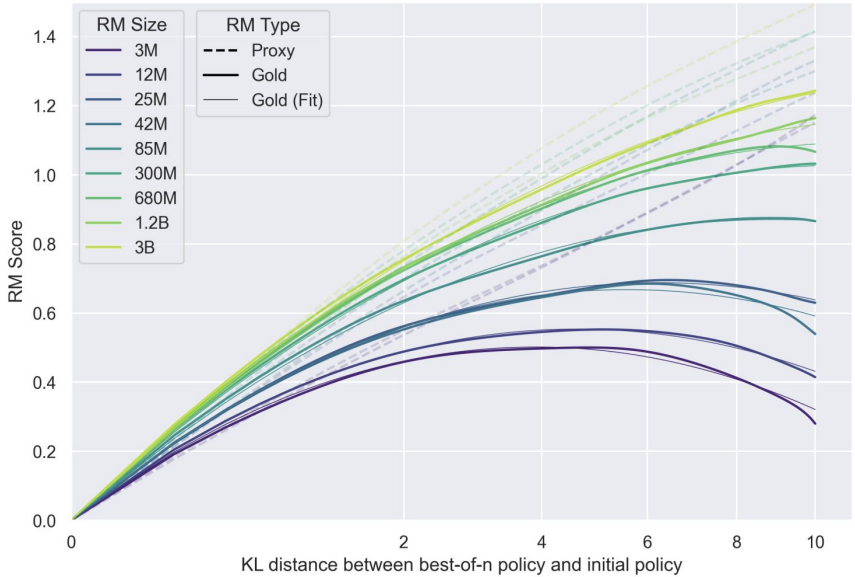
Goodhart's law, articulated as "When a measure becomes a target, it ceases to be a good measure."
In machine learning, optimizing too aggressively against a **learned reward model** can degrade actual performance with respect to the ground truth objective, a phenomenon the paper terms ***reward model overoptimization.***

Initial paper is written by Gao et al, 2022 studies this phenomenon to understand the degree and scaling of overoptimization is critical to safely deploying reinforcement learning from human feedback (RLHF) systems.

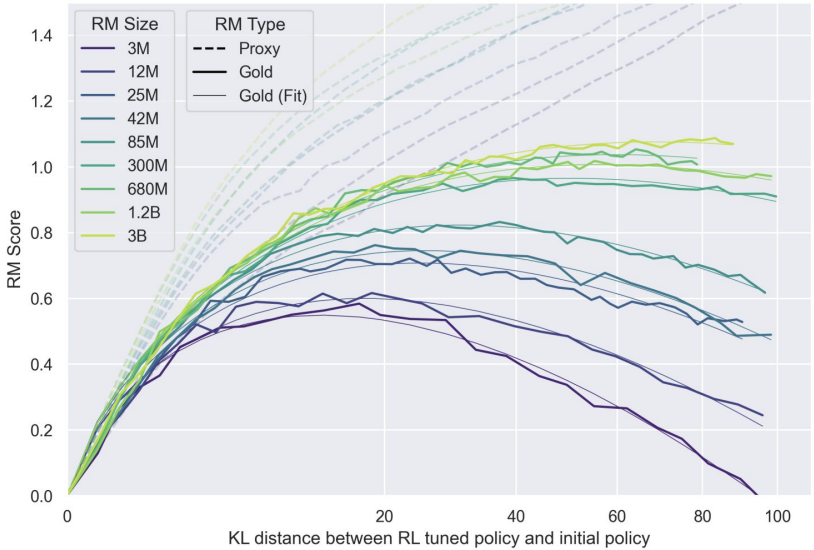
Setup: To avoid the prohibitive expense of human labeling, a synthetic environment was created, where:
- A fixed "gold-standard" reward model (6B parameters) acts as ground truth.
- Proxy reward models (RMs) with varying parameters are trained on synthetic data generated by the gold RM.
- Optimization methods: **Best-of-n (BoN)** sampling and **RL, using PPO**.



Scaling laws for reward model overoptimization results



(a) BoN



(b) RL