Lecture 18: Reinforcement Learning from Human Feedback (RLHF)

Adapted from CSE 517 / CSE 447 by Yejin Choi and others

Administrative

- A5 due this Friday June 5
  - Required for grad version;
  - Optional for undergrad version (i.e. if completed 8% per assignment; otherwise 10%)
- Project report due June 8
- W credit:
  - Final Report + List of Revisions: Due Monday, June 8th
- **Project poster: June 8 10:30am - 12:20pm at Allen Atrium**
  - Your poster grade will be half on content, half on presentation

Recap: Instruction Tuning: Turning GPT into ChatGPT

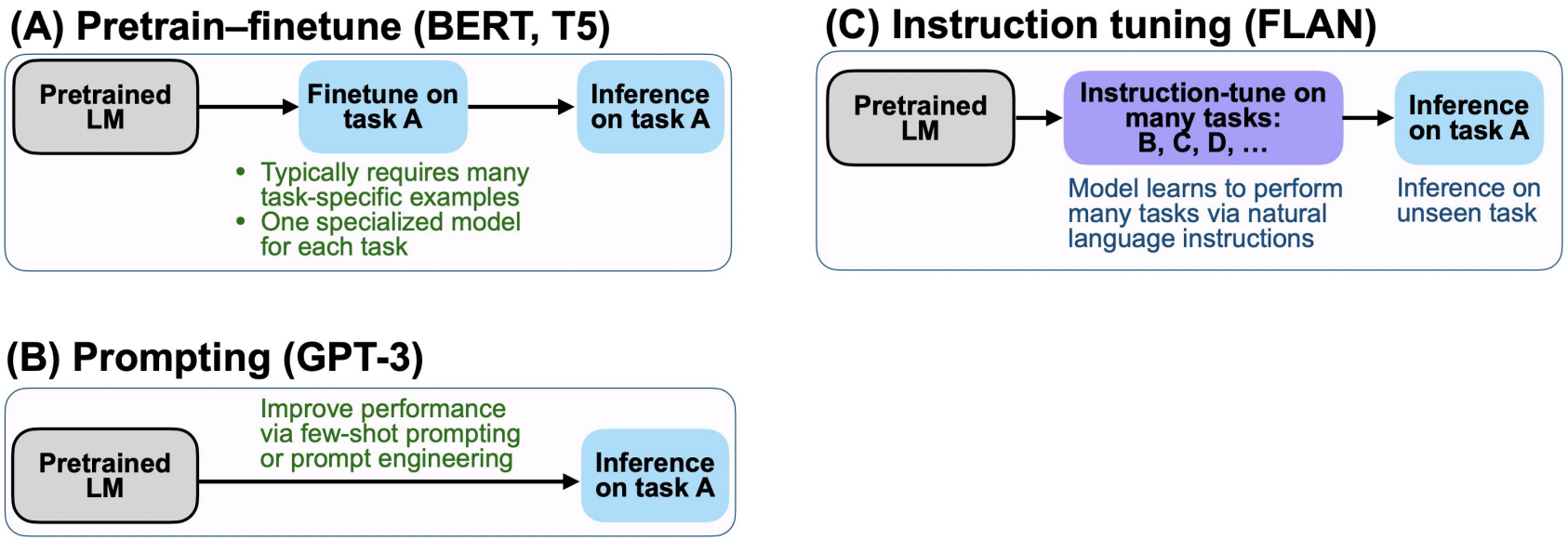
May 2020: GPT-3

April, September 2021: Instruction Tuning

March 2022: InstructGPT

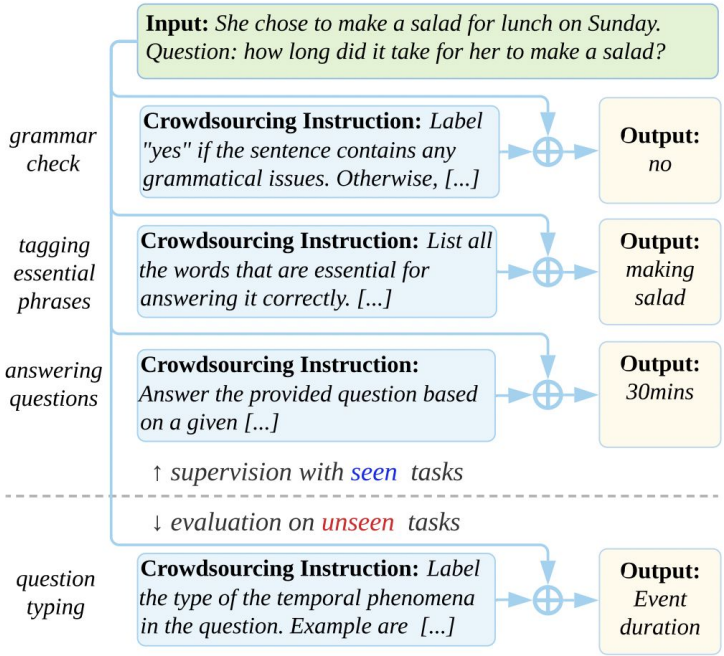
November 2022: ChatGPT

Recap: Instruction Tuning



"Finetuned Language Models are Zero-Shot Learners", Wei et al., 2021

Recap: Instruction Tuning



"Cross-Task Generalization via Natural Language Crowdsourcing Instructions", Mishra et al., 2021

Recap: Instruction Tuning

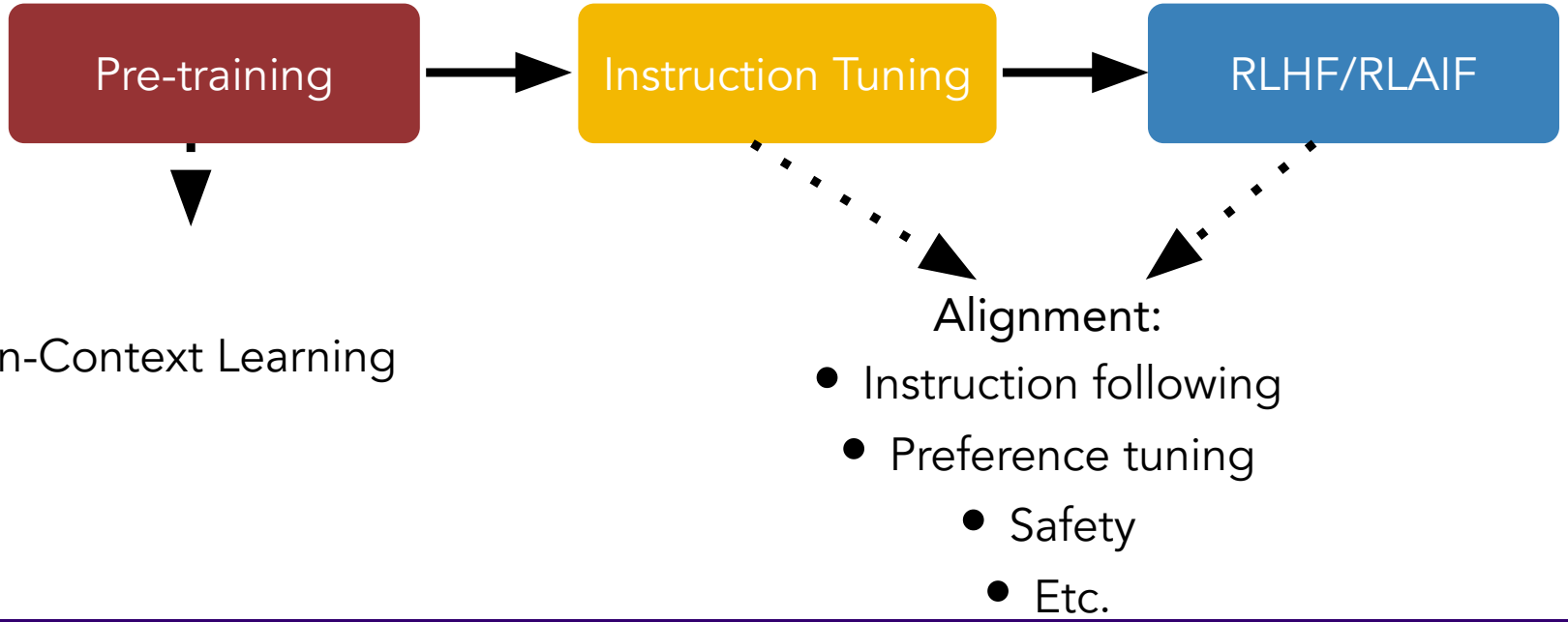
Great results…

Main takeaway: Teach your models to follow instructions!

One last finetuning step to actually get ChatGPT:

Reinforcement Learning from Human Feedback (RLHF)

The Adaptation Recipe



Limitations of Instruction Tuning

- Why do we need RLHF?

LM objective != human preferences

Limitations of Instruction Tuning

- Why do we need RLHF?
- (Open-ended) generation:
  - What makes one output better than the other? **-> hard to define**
- What types of LM errors should be weighted more?

LM objective != human preferences

Limitations of Instruction Tuning

- Why do we need RLHF?
- (Open-ended) generation: How do you capture all of the following and more in a loss function:
  - What is a *helpful* output?
  - What is a *polite* output?
  - What is a *funny* output?
  - What is a *safe* output?

LM objective != human preferences

RLHF!

Fine-Tuning Language Models from Human Preferences

Daniel M. Ziegler* **Nisan Stiennon*** **Jeffrey Wu** **Tom B. Brown**
Alec Radford **Dario Amodei** **Paul Christiano** **Geoffrey Irving**
OpenAI
{dmz,nisan,jeffwu,tom,alec,damodei,paul,irving}@openai.com

arxiv in Sep 2019
NeurIPS 2020

Learning to summarize from human feedback

Nisan Stiennon* **Long Ouyang*** **Jeff Wu*** **Daniel M. Ziegler*** **Ryan Lowe***
Chelsea Voss* **Alec Radford** **Dario Amodei** **Paul Christiano***
OpenAI

arxiv in Sep 2020
NeurIPS 2020

“Learning to Summarize with Human Feedback”

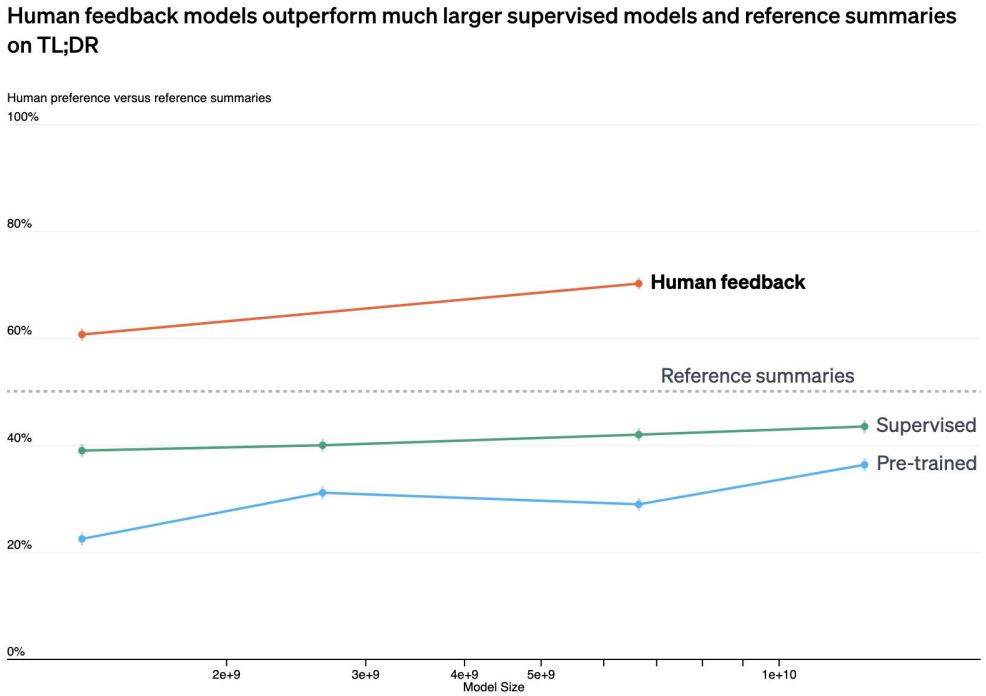
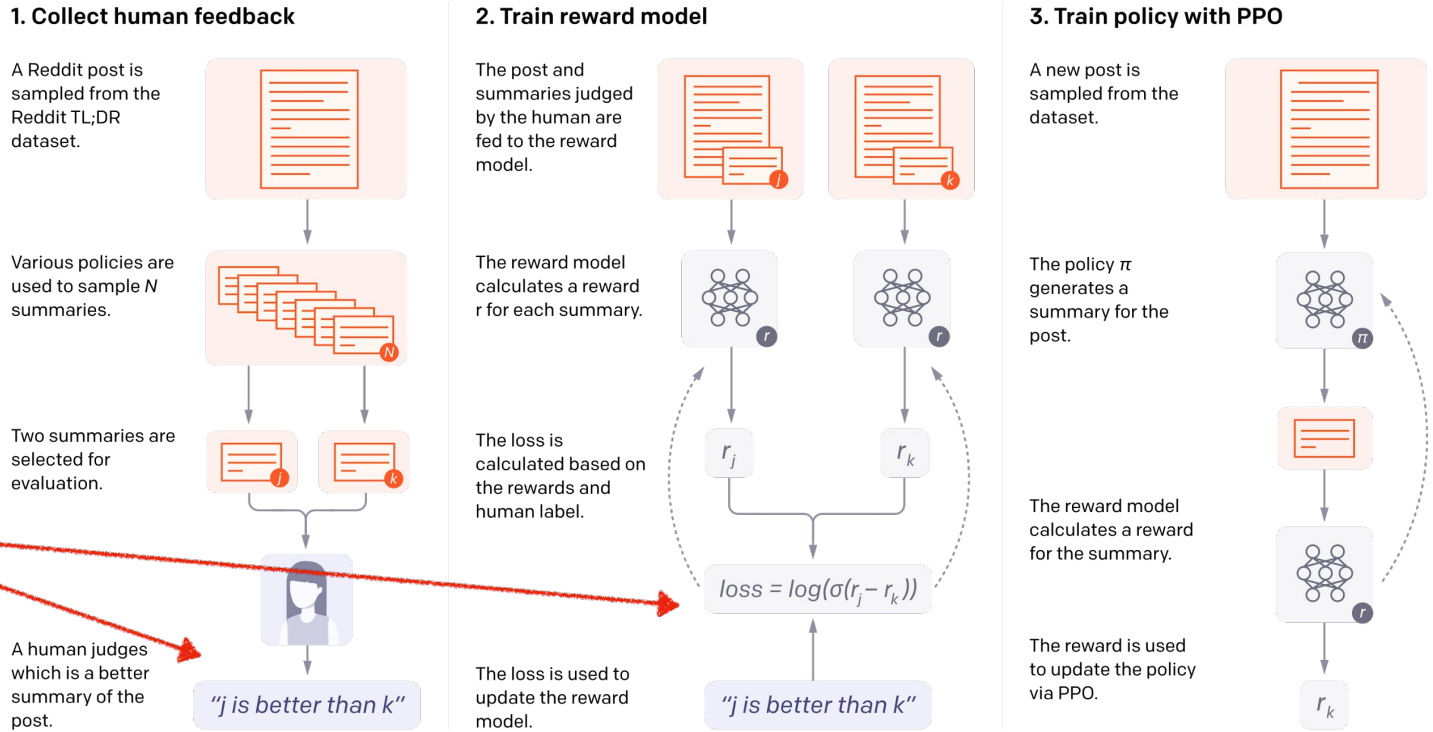


Figure 1: The performance of various training procedures for different model sizes. Model performance is measured by how often summaries from that model are preferred to the human-written reference summaries. Our pre-trained models are early versions of GPT-3, our supervised baselines were fine-tuned to predict 117K human-written TL;DRs, and our human feedback models are additionally fine-tuned on a dataset of about 65K summary comparisons.

https://openai.com/research/learning-to-summarize-with-human-feedback

“Learning to Summarize with Human Feedback”



RL methods don’t always assume “preference-based” (j is better than k) human feedback and reward model, but that’s what’s common with current “RLHF” approaches

https://openai.com/research/learning-to-summarize-with-human-feedback

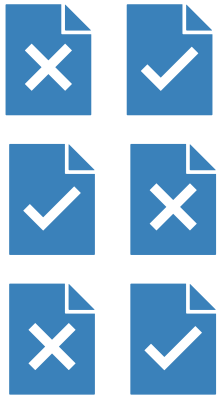
The general RLHF pipeline

1

Instruction-tuned Model

2

Collect Comparison Data



3

Train Reward Model on Comparison Data

4

Use RL to Optimize a Policy with the Reward Model

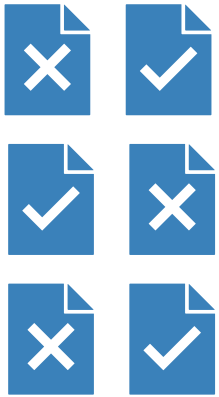
The general RLHF pipeline

1

Instruction-tuned Model

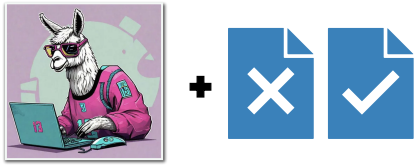
2

Collect Comparison Data



3

Train Reward Model on Comparison Data



4

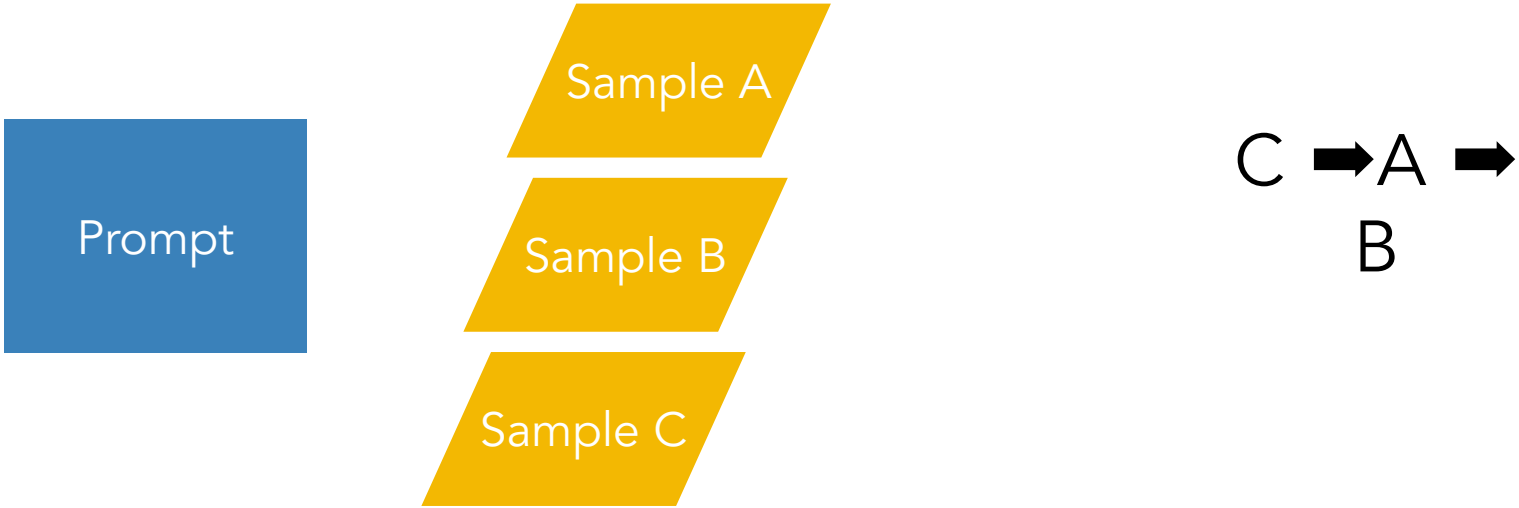
Use RL to Optimize a Policy with the Reward Model

Human Preferences

Human Preferences

A set of sampled completions for a prompt.

Ranking of the samples.

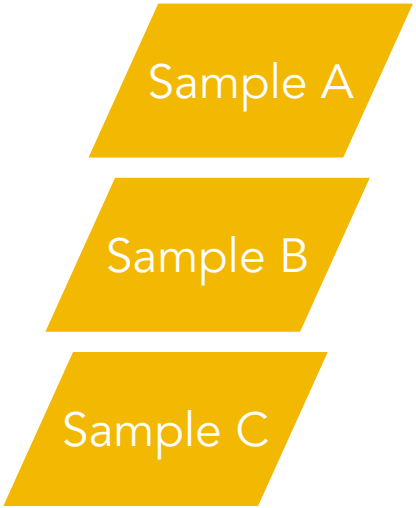


Human Preferences

A set of sampled completions for a prompt.

Triples

Prompt



$$D = \{x^i, y_w^i, y_l^i\}$$

Prompt

Preferred/ winning Response

Dispreferred/ losing Response

Example: Annotation

🧑‍💻 Can you help me write a resignation letter to my current employer, while leaving on good terms and expressing gratitude for the opportunities provided?

🤖 Here are two responses from the chatbot. (Please scroll down on the content to see the entire response if it is too long)

A

Sure, here is a resignation letter template that you can use:

Dear [manager],

It is with a heavy heart that I submit this letter of resignation from my position as [position] at [company]. I have enjoyed my time here and am grateful for the opportunities that have been provided to me.

I have accepted a new opportunity that presents a greater challenge and is more aligned with my long-term career goals. I have learned so much during my time at

B

Of course! Here is a sample resignation letter that you can use as a guide:

Dear [Manager's Name],

I am writing to inform you that I have decided to resign from [Company Name]. This was not an easy decision, as I have enjoyed my time here and have learned a lot from my colleagues.

I want to express my sincere gratitude for the opportunities and support that [Company Name] has provided me over the years. I have grown both

- Annotator needs to choose whether they prefer A or B.

Pairwise Comparison

Why do pairwise comparison and not rate outputs directly?

🧑‍💻 Compose an engaging travel blog post about a recent trip to Hawaii, highlighting cultural experiences and must-see attractions

A

Have you ever imagined standing on a black sand beach, watching the waves of the Pacific Ocean crash against the shore? Or strolling down a narrow street, taking in the vibrant colors and aromas of local cuisine? If so, then Hawaii is the perfect destination for you.
I recently had the pleasure of visiting this stunning archipelago in the middle of the Pacific Ocean, and I am still daydreaming about my amazing adventures there. From the Hawaiian culture to the natural wonders, every moment was full of wonder and excitement.

How would you rate this output?

- Hard to be consistent among different annotators!
- It's more reliable (Phelps et al., 2015; Clark et al., 2018)
- Can be used with the Bradley-Terry (1952) model

From Preference Data to Bradley-Terry Model

$$D = \{x^i, y_w^i, y_l^i\}$$

Prompt

Preferred Response

Dispreferred Response

Reward for preferred response

Reward for dispreferred response

$$p(y_w > y_l|x) = \sigma(r(x, y_w) - r(x, y_l))$$

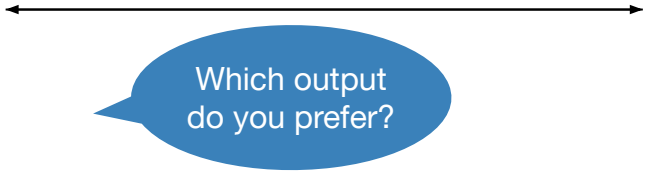
Logistic function; which is equivalent to using softmax:

$$p(y_w > y_l|x) = \frac{\exp(r(x, y_w))}{\exp(r(x, y_w)) + \exp(r(x, y_l))}$$

$$\frac{1}{1 + e^{-x}}$$

But..

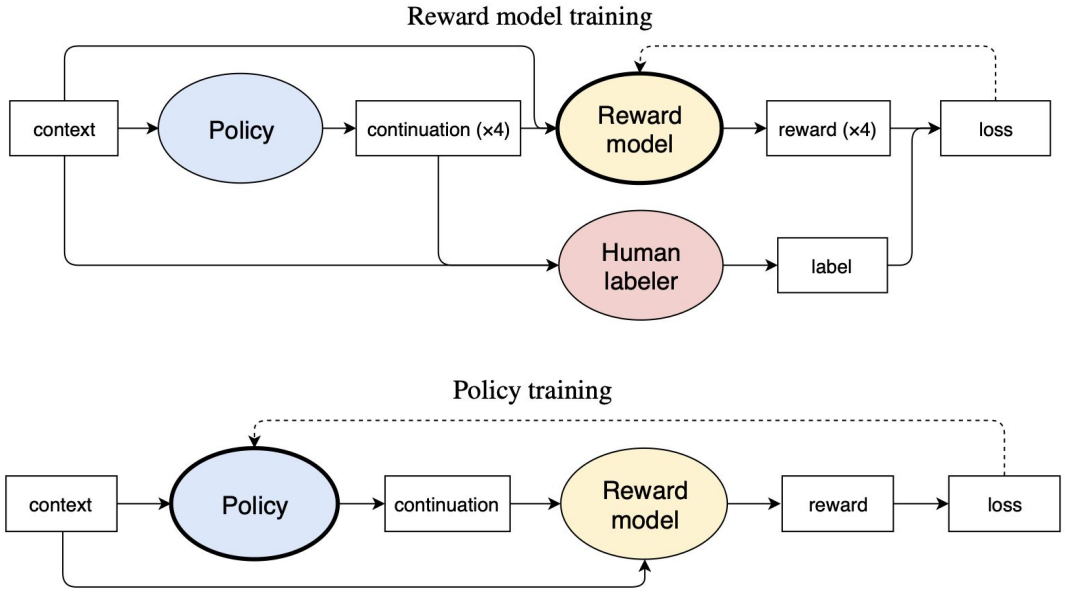
- How do we get feedback for the reward while training our RL model?



Having a human in the loop is very expensive!

But..

- How do we get feedback for the reward while training our RL model?



Instead: train a Reward Model (RM) on preference data to predict preferences!

Ziegler et al., 2019 "Fine-Tuning Language Models from Human Preferences"

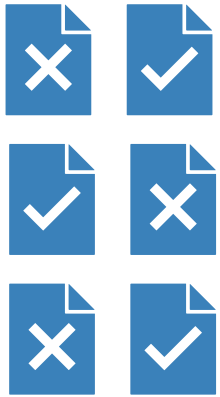
The general RLHF pipeline

1

Instruction-tuned Model

2

Collect Comparison Data



3

Train Reward Model on Comparison Data

4

Use RL to Optimize a Policy with the Reward Model

Reward Modeling

Reward Model

$$p(y_w > y_l|x) = \frac{\exp(r(x, y_w))}{\exp(r(x, y_w)) + \exp(r(x, y_l))}$$

- Train on preference data.
- Minimizing negative log likelihood.

Bradley-Terry Model

equivalent to

$$\mathcal{L}_R(\phi, D) = -\mathbb{E}_{(x,y_w,y_l)\sim D}[\log \sigma(r(x, y_w) - r(x, y_l))]$$

- Train an LLM with an additional layer to minimize the neg. log likelihood

Evaluating Reward Models

- Accuracy of predicting human preferences.

Preference Datasets

Reward Models

Table 2: Reward modeling accuracy (%) results. We compare our UltraRM with baseline open-source reward models. LLaMA2 results are taken from Touvron et al. (2023b). The highest results are in **bold** and the second highest scores are underlined.

Model	Backbone Model	Open?	Anthropic Helpful	OpenAI WebGPT	OpenAI Summ.	Stanford SHP	Avg.
Moss	LLaMA-7B	✓	61.3	54.6	58.1	54.6	57.2
Ziya	LLaMA-7B	✓	61.4	57.0	61.8	57.0	59.3
OASST	DeBERTa-v3-large	✓	67.6	-	72.1	53.9	-
SteamSHP	FLAN-T5-XL	✓	55.4	51.6	62.6	51.6	55.3
LLaMA2 Helpfulness	LLaMA2-70B	✗	**72.0**	-	**75.5**	**80.0**	-
UltraRM-UF	LLaMA2-13B	✓	66.7	65.1	66.8	68.4	66.8
UltraRM-Overall	LLaMA2-13B	✓	71.0	62.0	73.0	73.6	69.9
UltraRM	LLaMA2-13B	✓	71.0	**65.2**	74.0	73.7	**71.0**

Cui et al., ArXiV 2023 "UltraFeedback: Boosting Language Models with High-quality Feedba

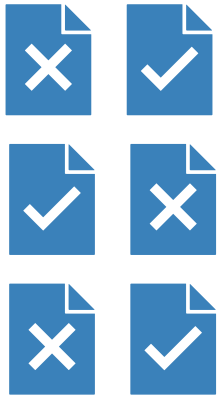
The general RLHF pipeline

1

Instruction-tuned Model

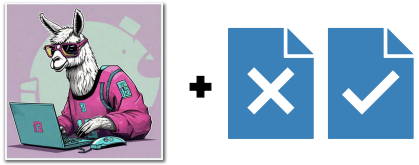
2

Collect Comparison Data



3

Train Reward Model on Comparison Data



4

Use RL to Optimize a Policy with the Reward Model

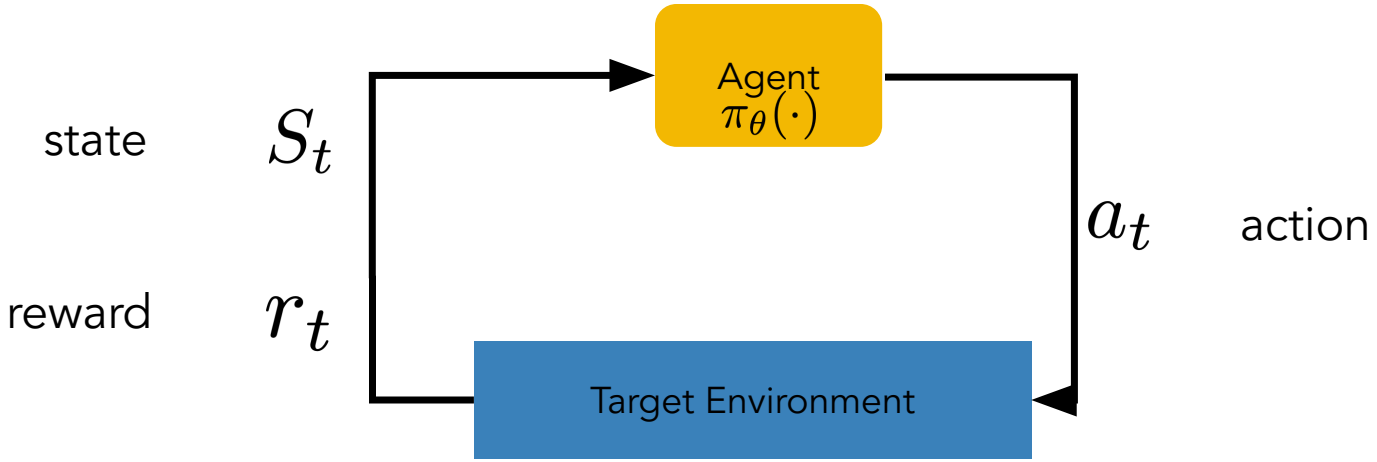
Fun Facts about Reward Models

- Trained for 1 epoch (to avoid overfitting)!
- Evaluation often only has 65% - 75% agreement

Lambert et al., 2023

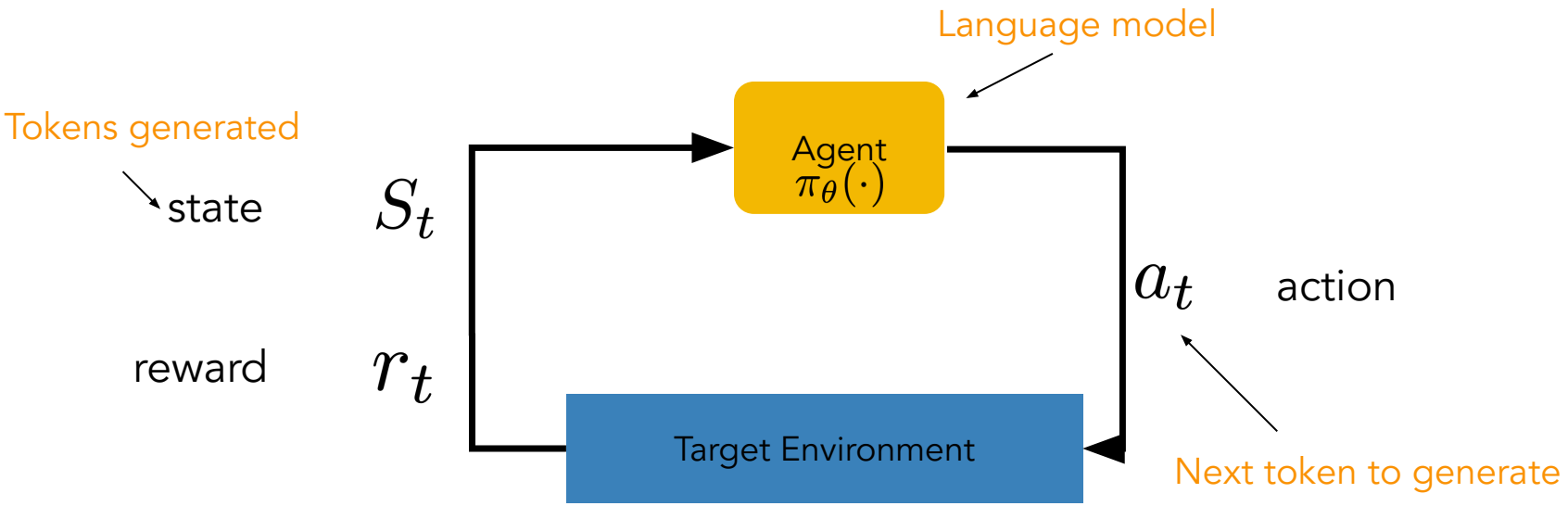
RL Basics

Recap: Reinforcement Learning Basics



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

RL in the Context of Language Models…



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

Recap: REINFORCE

$$\nabla_\theta J(\theta) = \int_\tau \left(r(\tau)\nabla_\theta \log p(\tau;\theta)\right) p(\tau;\theta)\mathrm{d}\tau$$
$$= \mathbb{E}_{\tau\sim p(\tau;\theta)}\left[r(\tau)\nabla_\theta \log p(\tau;\theta)\right]$$

Can we compute those quantities without knowing the transition probabilities?

We have: $p(\tau;\theta) = \prod_{t\geq 0} p(s_{t+1}|s_t,a_t)\pi_\theta(a_t|s_t)$

Thus: $\log p(\tau;\theta) = \sum_{t\geq 0} \log p(s_{t+1}|s_t,a_t) + \log \pi_\theta(a_t|s_t)$

And when differentiating: $\nabla_\theta \log p(\tau;\theta) = \sum_{t\geq 0} \nabla_\theta \log \pi_\theta(a_t|s_t)$

Doesn't depend on transition probabilities!

Therefore when sampling a trajectory τ, we can estimate J(θ) with

$$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} r(\tau)\nabla_\theta \log \pi_\theta(a_t|s_t)$$

Summary of Policy Gradient for RL

REINFORCE Update:

$$\theta_{k+1} := \theta_k + \alpha \frac{1}{m} \sum_{i=1}^{m} r(\tau_i) \sum_{t \geq 0} \nabla_\theta \log \pi_\theta(a_t \mid s_t)$$

Simplified Intuition: good actions are reinforced and bad actions are discouraged.

Williams, 1992

Summary of Policy Gradient for RL

REINFORCE Update:

$$\theta_{k+1} := \theta_k + \alpha \frac{1}{m} \sum_{i=1}^{m} r(\tau_i) \sum_{t \geq 0} \nabla_\theta \log \pi_\theta(a_t \mid s_t)$$

If: Reward is high/positive

Then: maximize this

Simplified Intuition: good actions are reinforced and bad actions are discouraged.

Williams, 1992

Policy

- **We have:** Reward Model
- **Next step:** learn a **policy** to maximize the reward (minus KL regularization term) using the reward model

$$\max_{\pi_\theta} \mathbb{E}_{x\sim D, y\sim\pi_\theta(y|x)}[r_\phi(x,y)] - \beta\mathbb{D}_{KL}[\pi_\theta(y|x)||\pi_{ref}(y|x)]$$

Sampling from policy

Reward given prompt and sampled generation

KL-divergence between original model's generation and the sampled generation

How the KL Divergence is Calculated

- Always nonnegative, equal to 0 only when the policies match. $\mathbb{D}_{KL} \geq 0$
- Definition (expectation form):

$$\mathbb{D}_{KL}[\pi_\theta(y \mid x) \,\|\, \pi_{\text{ref}}(y \mid x)] = \sum_y \pi_\theta(y \mid x) \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)} = \mathbb{E}_{y \sim \pi_\theta}\left[\log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}\right]$$

- Autoregressive factorization (token sequence y = y_1…y_T):

$$\log \pi(y \mid x) = \sum_{t=1}^{T} \log \pi(y_t \mid x, y_{<t})$$

- Per-sample token-by-token KL contribution:

$$\sum_{t=1}^{T} [\log \pi_\theta(y_t \mid x, y_{<t}) - \log \pi_{\text{ref}}(y_t \mid x, y_{<t})]$$

- Estimated by Monte Carlo; subtracted (x beta) from the PPO reward.

Policy

- **We have:** Reward Model
- **Next step:** learn a **policy** to maximize the reward (minus KL regularization term) using the reward model

$$\max_{\pi_\theta} \mathbb{E}_{x\sim D, y\sim\pi_\theta(y|x)}[r_\phi(x,y)] - \beta\mathbb{D}_{KL}[\pi_\theta(y|x)||\pi_{ref}(y|x)]$$

Sampling from policy

Reward given prompt and sampled generation

Should be high!

KL-divergence between original model's generation and the sampled generation

Should be low!

PPO!
Proximal Policy Optimization

Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI
{joschu, filip, prafulla, alec, oleg}@openai.com

arxiv in July 2017

PPO: builds on Policy Gradient Methods

Gradient Estimator

$$\hat{g} = \hat{\mathbb{E}}_t[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t]$$

Expectation: empirical average over all timesteps t in a batch of sampled trajectories

Objective / Loss:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t[\log \pi_\theta(a_t|s_t)\hat{A}_t]$$

Often leads to (too) large policy updated

Advantage function (we'll come back to this)

$$\hat{A}_t = \hat{A}(s_t, a_t) = -V_\phi(t) + G_t = -V_\phi(t) + \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'},$$

$\hat{A}_t$: estimator of the advantage function at timestep t

π_θ : policy that we are trying to learn via PPO; this is initialized as a language model

Schulman, 2017

PPO: the Advantage Function

Advantage function is about the advantage of taking action a_t at state s_t over all other actions (computed in terms of the expected discounted returns of any action versus action a_t)

$$\hat{A}_t = \hat{A}(s_t, a_t) = -V(s_t) + G_t$$: Advantage function

$$G_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$$: Empirical return (of taking a particular action a_t at a particular state s_t)

$$r_t = \begin{cases} -\beta \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} + r(s_{T+1}) & (\text{where } t = T) \\ -\beta \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} & (\text{where } 1 \leq t < T) \end{cases}$$

This way of setting the token-level reward is the common implementation among the original RLHF paper, AlpacaFarm, Quark, Rainier etc

Schulman, 2017; Liu et al., 2023

Intuition

Gradient estimator: $$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t | s_t)$$

Interpretation:

- If r(τ) is high, push up the probabilities of the actions seen
- If r(τ) is low, push down the probabilities of the actions seen

From r(τ) to the Advantage Â□

REINFORCE weights every action by the whole-trajectory reward r(τ); PPO instead weights by the advantage Â□. The two are the same idea, in three steps:

Step 1 — Reward-to-go: an action at time t only affects future rewards, so r(τ) can be replaced by the return-to-go R□ without changing the gradient in expectation.
Step 2 — Baseline: subtracting a state-only baseline V(s□) keeps the gradient unbiased but lowers variance.
Step 3 — Advantage: the result R□ − V(s□) is exactly the advantage Â□. Substituting gives PPO's estimator:

$$r(\tau) \longrightarrow R_t = \sum_{k \geq t} r_k$$

$$R_t \longrightarrow R_t - V(s_t), \quad \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, V(s_t)] = 0$$

$$\hat{A}_t = R_t - V(s_t) \approx Q(s_t, a_t) - V(s_t)$$

$$\sum_{t \geq 0} r(\tau)\, \nabla_\theta \log \pi_\theta(a_t \mid s_t) \longrightarrow \hat{\mathbb{E}}_t[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, \hat{A}_t]$$

PPO: builds on Trust Region Methods (TRPO)

"Surrogate objective"

Instead of using the learned policy directly, we use the ratio between learned & original policy

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\right]$$

θ_{old} : original policy parameters

$$\text{subject to} \quad \hat{\mathbb{E}}_t[KL[\pi_\theta(\cdot|s_t), \pi_{\theta_{old}}(\cdot|s_t)]] \leq \delta$$

"Trust region"

Constraint on the size of the policy update

Schulman, 2017

PPO: Clipped Surrogate Objective

"Surrogate objective":

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t[r_t(\theta)\hat{A}_t]$$

ratio of policies, not reward

Without constraint this leads to a too large policy update.

Objective proposed by PPO paper:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t]$$

Schulman, 2017

PPO: Clipped Surrogate Objective

"Surrogate objective":

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t[r_t(\theta)\hat{A}_t]$$

ratio of policies, not reward

Without constraint this leads to a too large policy update.

ϵ : hyperparameter

Objective proposed by PPO paper:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t]$$

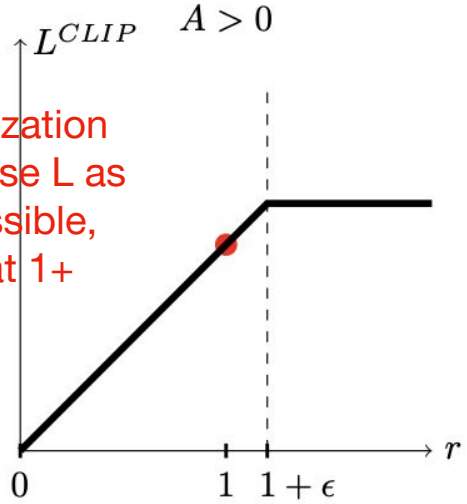
Clips the probability ratio (prevents r from moving outside of the interval)

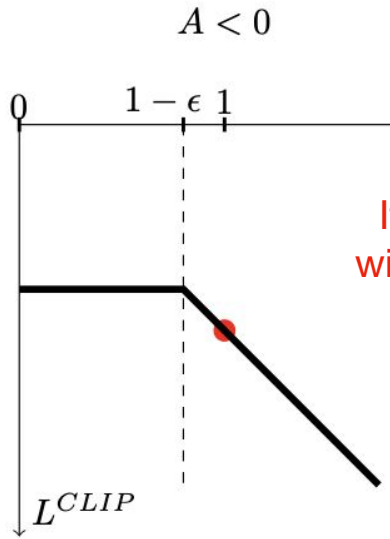
Take the minimum of the clipped and unclipped objective: final objective is a lower bound on the unclipped objective

Schulman, 2017

PPO: Clipped Surrogate Objective

If A>0, optimization will want to raise L as much as possible, thus clip r at 1+ epsilon





If A<0, optimization will want to lower L as much as possible, thus clip r at 1- epsilon

Figure 1: Plots showing one term (i.e., a single timestep) of the surrogate function L^{CLIP} as a function of the probability ratio r, for positive advantages (left) and negative advantages (right). The red circle on each plot shows the starting point for the optimization, i.e., $r = 1$. Note that L^{CLIP} sums many of these terms.

Schulman, 2017

PPO: the Value Model

- PPO trains two models (**a** is an action, **s** is a state): not just policy, but also value
  - Policy model $\pi_\theta(a_t|s_t)$
  - Value model $V_\phi(s_t)$
    - Value function $V_\phi(s_t) = \mathbb{E}_\pi[G_t] = \mathbb{E}_\pi[\sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} | s_t = s]$

Value is the **expected** return of a state s_t

G_t is the "**empirical** return" or "discounted future reward" (starting at s_t)

- "Attempts to minimize the value estimation error against the empirical return" $L_t^{VF}(\theta) = (V_\phi(s_t) - G_t)^2$

Liu et al., 2023

PPO: Final Objective

coefficients

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t[L_t^{CLIP_\theta} - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)]$$

entropy bonus: ensures sufficient exploration

CLIP Objective:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t]$$

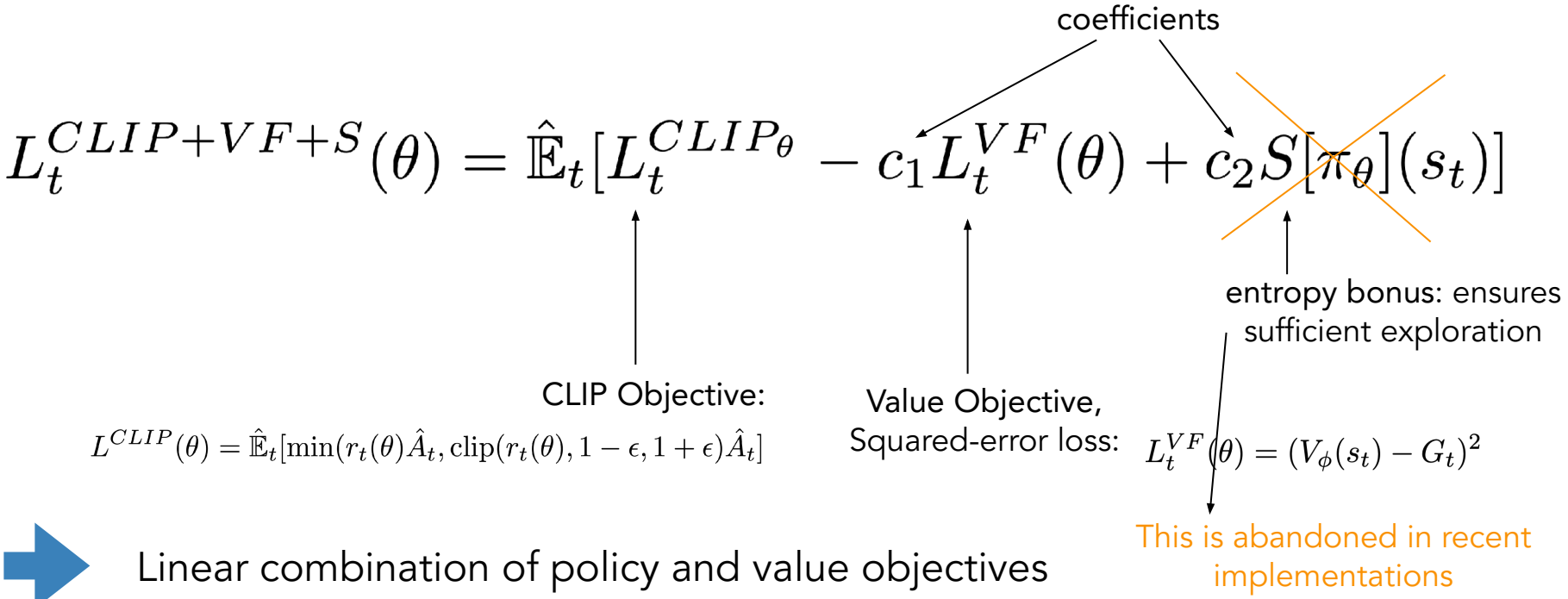
Value Objective, Squared-error loss:

$$L_t^{VF}(\theta) = (V_\phi(s_t) - G_t)^2$$

Linear combination of policy and value objectives

Schulman, 2017

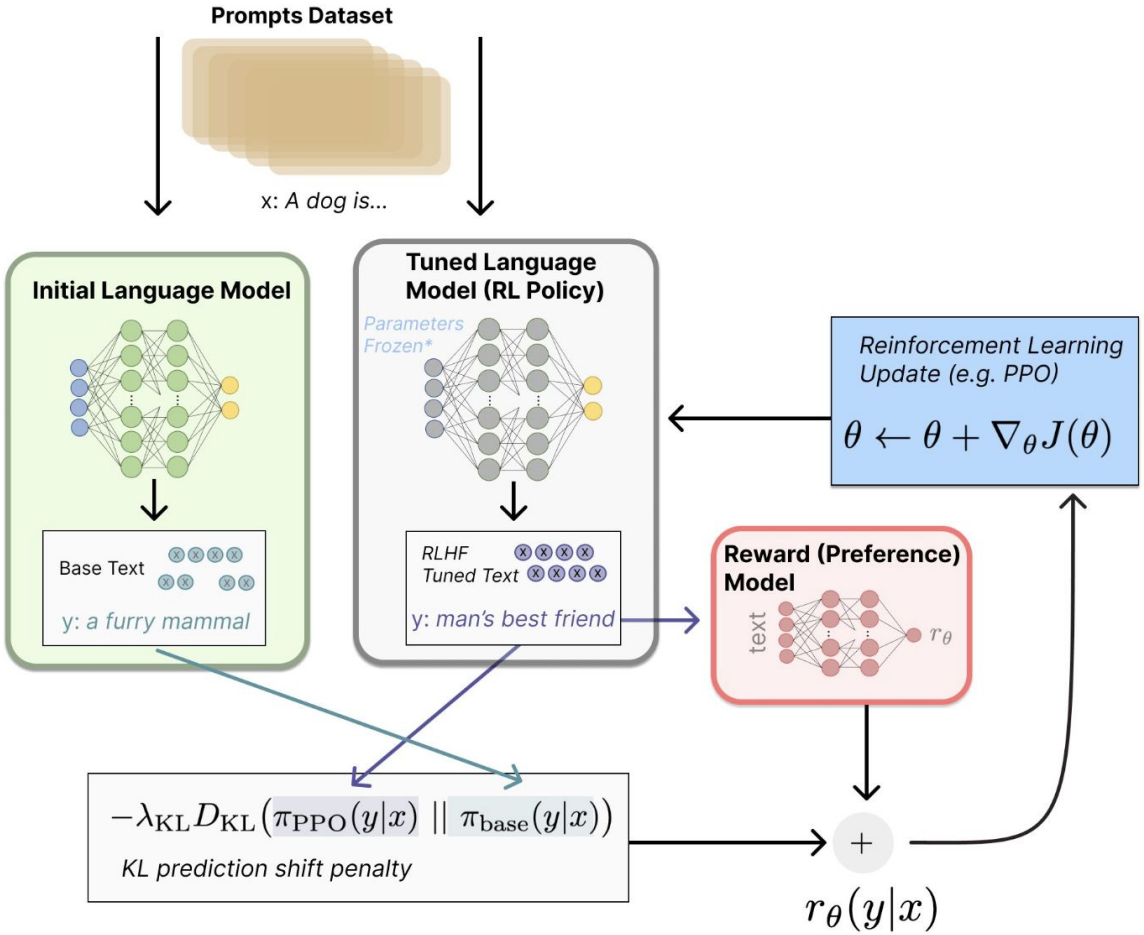
PPO: Final Objective



Linear combination of policy and value objectives

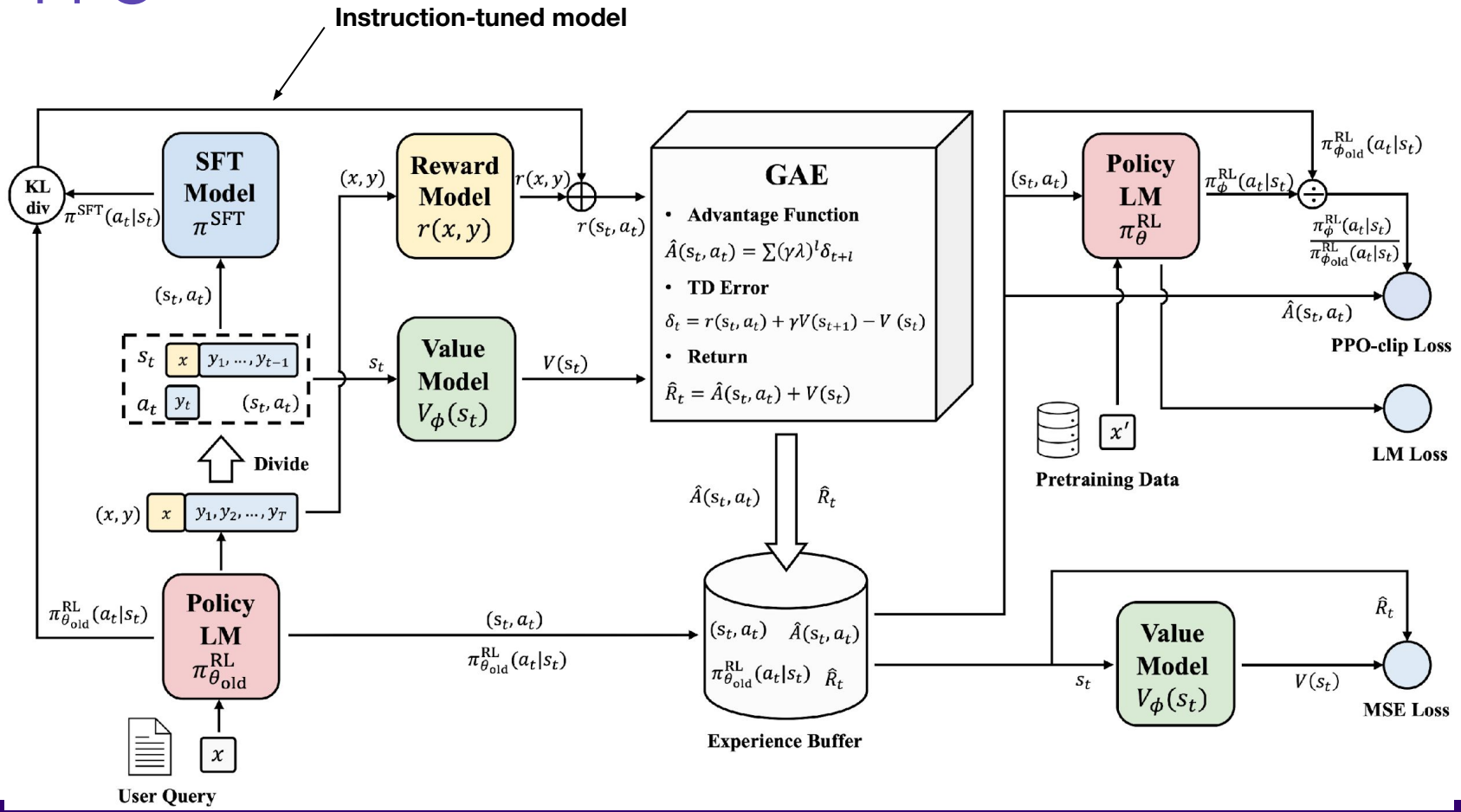
Schulman, 2017

PPO



Lambert, 2023

PPO



Zheng et al., 2023

GRPO!
Group Relative Policy Optimization

DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models

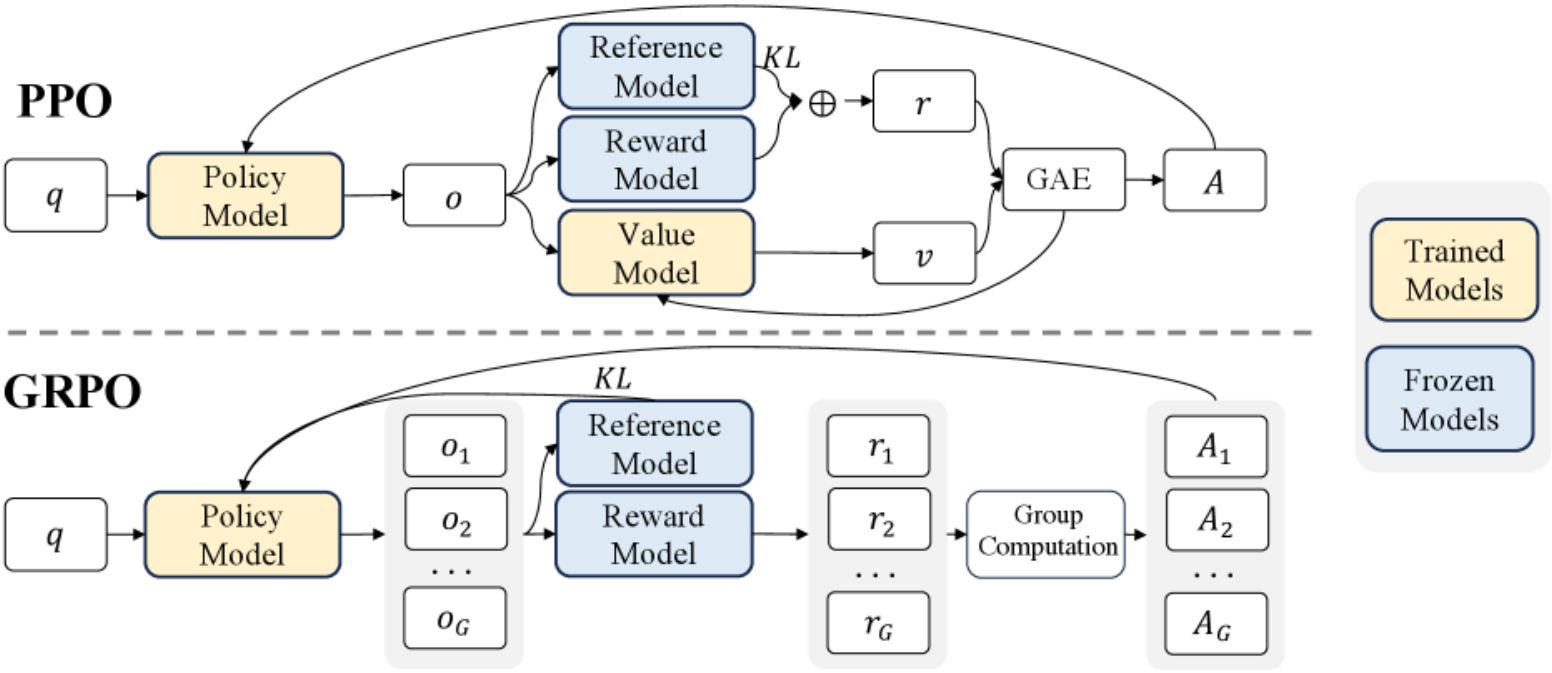
Shao, Wang, Zhu, Xu, Song, Bi, Zhang, Zhang, Li, Wu, Guo
DeepSeek-AI

Arxiv in Feb 2024
(DeepSeekMath)

GRPO: Motivation — Drop the Value Model

- PPO trains two models: a policy and a value model (the critic)
- The value model is expensive: it is typically as large as the policy and adds significant memory + compute
- GRPO removes the value model entirely — no learned critic
- Key idea: instead of estimating a baseline with a value network, sample a group of outputs per prompt and use their rewards to form a relative baseline
- This makes RL training cheaper and simpler, while keeping the PPO-style clipped objective

PPO vs GRPO: The Picture



Source: Shao et al., "DeepSeekMath" (2024), arXiv:2402.03300, Figure 4.

GRPO: Group-Relative Advantage

For each prompt q, sample a group of G outputs $\{o_1, \ldots, o_G\}$ from the old policy
Score each output with the reward model: $r_1, \ldots, r_G$
Compute the group-relative advantage by normalizing rewards within the group:

$$\hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}, \qquad \mathbf{r} = \{r_1, r_2, \ldots, r_G\}$$

Outputs above the group average get positive advantage; below-average get negative — this replaces PPO's value baseline $V(s)$ with a simple group statistic (no critic)

GRPO: The Objective

GRPO keeps PPO's clipped surrogate, but plugs in the group-relative advantage (no value model):

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)] \frac{1}{G} \sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \{\min[\rho_{i,t}\hat{A}_{i,t}, \text{clip}(\rho_{i,t}, 1-\varepsilon, 1+\varepsilon)\hat{A}_{i,t}] - \beta \mathbb{D}_{KL}[\pi_\theta \| \pi_{ref}]\}$$

$\rho_{i,t}$ is the per-token probability ratio $\pi_\theta / \pi_{\theta_{old}}$ (same as PPO)
$\hat{A}_{i,t}$ is the group-relative advantage (shared across all tokens of o_i)
KL penalty (β) keeps π_θ close to π_{ref}; the clip prevents too-large updates
No value-function loss term — the critic is gone, so only the policy is trained

PPO vs GRPO

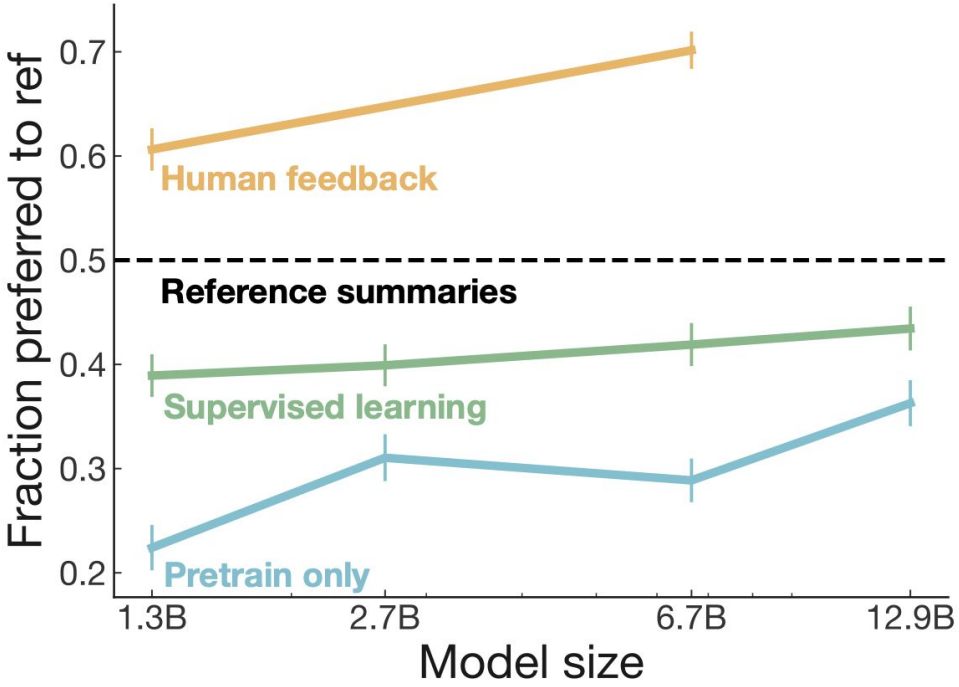
Dimension	PPO	GRPO
Baseline	Learned value model V(s)	In-group reward mean
Models trained	Policy + value	Policy only
Advantage	GAE from the critic	Normalized group-relative rewards
Cost	Trains and stores a critic (more memory + compute)	Cheaper - no critic to train or store
Objective	Clipped surrogate + KL penalty	Same clipped surrogate + KL penalty
Where it shines	General-purpose RLHF	Many cheap samples per prompt (e.g. math/reasoning with verifiable rewards)

Evaluating the Learned Policy

- Win Rate: How often does my policy's output win against a reference model's output, given the same instruction?
  - Who compares the two outputs?
    - Humans
    - Simulate humans (and human variability!) using GPT-4 (f.ex. Alpacafarm eval)

Dubois et al., 2023

RLHF vs. finetuning



- Win-rate over human-written reference summaries
- RLHF outperforms supervised learning and pretraining only for generating summaries.

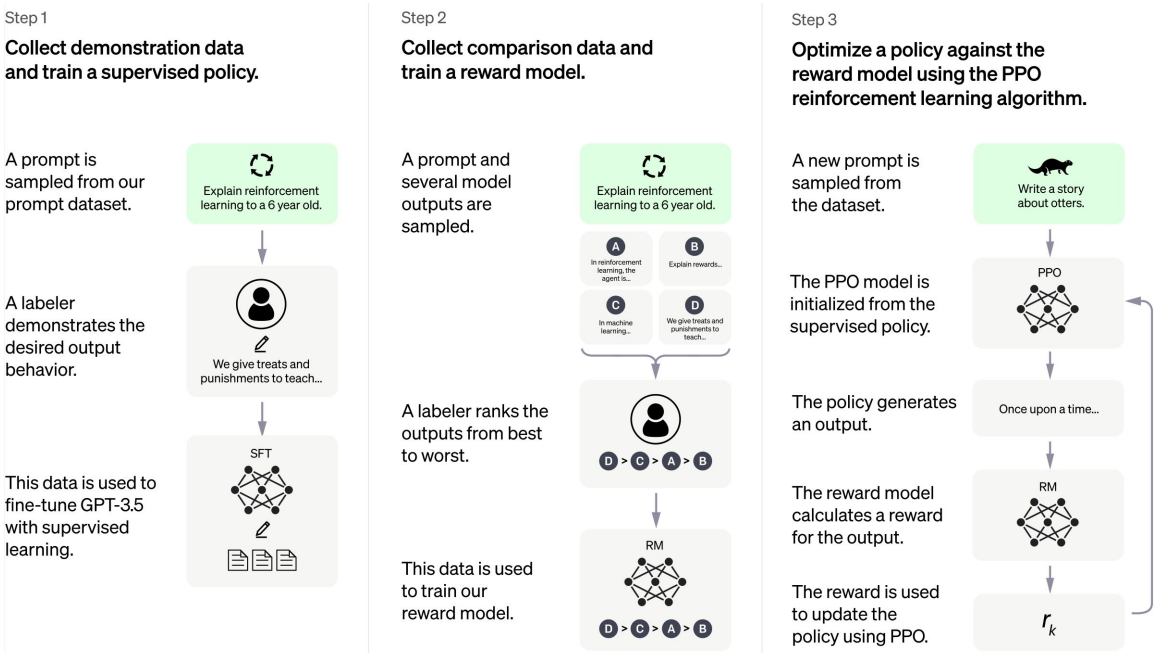
Stiennon et al., 2023

A short history of LLMs

- 2017: transformer
- 2018: Elmo, GPT-1 and BERT
- 2019: GPT-2, early research on RLHF
- 2020: GPT-3, “Learning to summarize with HF”
- 2022: ChatGPT, Claude, **RLHF gains a lot of public attention**
- 2023: GPT-4

*GPT

- InstructGPT
  - Instruction Tuning + RLHF
- ChatGPT
  - Instruction Tuning + RLHF for dialog agents



https://openai.com/blog/chatgpt

Online vs. offline RL

Online

- Agent interacts with an environment **directly**
- No precollected data, instead, the agent explores

Offline

- Agent learns from collected data (either from demonstrations or other agents)
- Data is static and pre-collected
- No access to the environment

Sutton & Barton, 2018; Lambert, 2023; Simonini, 2023

On-policy vs. off-policy

On-Policy

- "Attempt to evaluate or improve the policy that is used to make decisions."
- Directly update from samples, as policy generates
- PPO is on-policy

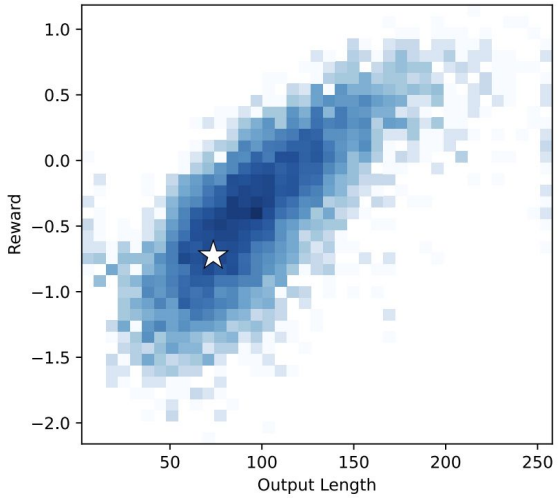
Off-Policy

- "Evaluate or improve a policy different from that used to generate the data"
- Learn from any state-action-reward tuples

Sutton & Barton, 2018; Lambert, 2023

Limitations of RLHF: Reward Hacking

- Reward hacking:
  "Exploiting errors in the reward model to achieve high estimated reward"
- Length (and other) biases
- Spurious Correlations



Question: *Why don't adults roll off the bed?*

☆ **SFT (Before); 59 tokens**

Adults typically do not roll off of the bed because they have developed the muscle memory to keep their bodies from involuntarily moving during sleep and maintaining proper posture.

RLHF (After); 243 tokens: Similar output, but much longer / more details

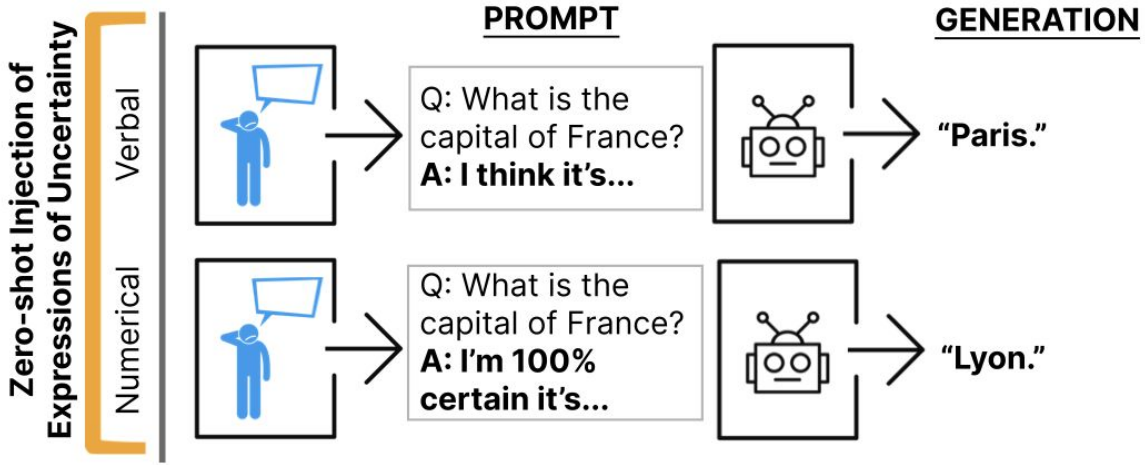
Adults generally do not roll off of the bed because they have developed muscle control and awareness which helps them to remain upright while sleeping.

Additionally, most adults find it uncomfortable or even painful to move around in their sleep, so rolling off the bed would be difficult without waking up first. There may also be safety concerns such as falling out of bed and potentially hurting oneself if one were to lose balance or fall from the …

Singhal et al., 2023

Limitations of RLHF

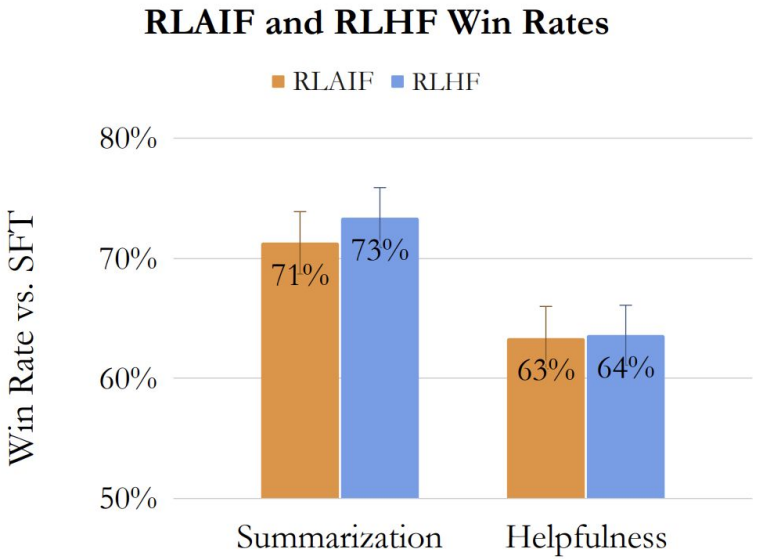
- Hallucinations and **false certainty**



Zhou et al., 2023

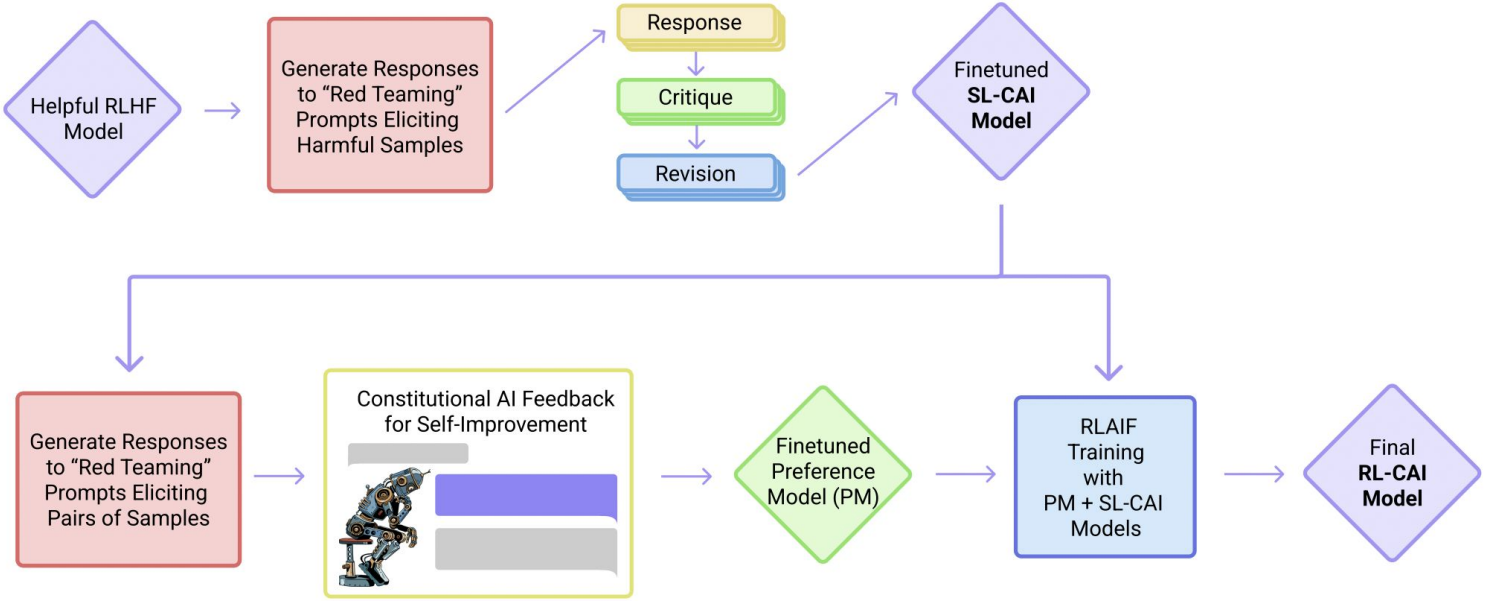
RLHF vs. RLAIF

- Human feedback vs. AI feedback



Lee et al., 2023

RLHF vs. RLAIF: Constitutional AI by Anthropic



Bai et al., 2023

Thank you!

References

- Bai, Yuntao, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen et al. "Constitutional ai: Harmlessness from ai feedback." *arXiv preprint arXiv:2212.08073* (2022).
- Cui, Ganqu, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. "Ultrafeedback: Boosting language models with high-quality feedback." *arXiv preprint arXiv:2310.01377* (2023).
- Dubois, Yann, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. "AlpacaFarm: A Simulation Framework for Methods that Learn from Human Feedback." *arXiv e-prints* (2023): arXiv-2305.
- Eisenstein, Jacob, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D'Amour, D. J. Dvijotham, Adam Fisch et al. "Helping or Herding? Reward Model Ensembles Mitigate but do not Eliminate Reward Hacking." *arXiv preprint arXiv:2312.09244* (2023).
- Ivison, Hamish, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang et al. "Camels in a Changing Climate: Enhancing LM Adaptation with Tulu 2." *arXiv preprint arXiv:2311.10702* (2023).
- Lambert, Nathan, Thomas Krendl Gilbert, and Tom Zick. "The History and Risks of Reinforcement Learning and Human Feedback." *arXiv e-prints* (2023): arXiv-2310.
- Lee, Harrison, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. "Rlaif: Scaling reinforcement learning from human feedback with ai feedback." *arXiv preprint arXiv:2309.00267* (2023).

References

- Liu, Jiacheng, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. "Making ppo even better: Value-guided monte-carlo tree search decoding." *arXiv preprint arXiv:2309.15028* (2023).
- Rafailov, Rafael, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. "Direct preference optimization: Your language model is secretly a reward model." *arXiv preprint arXiv:2305.18290* (2023).
- Röttger, Paul, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. "Xstest: A test suite for identifying exaggerated safety behaviours in large language models." *arXiv preprint arXiv:2308.01263* (2023),
- Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347* (2017).
- Singhal, Prasann, Tanya Goyal, Jiacheng Xu, and Greg Durrett. "A long way to go: Investigating length correlations in rlhf." *arXiv preprint arXiv:2310.03716* (2023).
- Stiennon, Nisan, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. "Learning to summarize with human feedback." *Advances in Neural Information Processing Systems* 33 (2020): 3008-3021.
- Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- Williams, R.J., 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*, pp.229-256.
- Zheng, Rui, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu et al. "Secrets of rlhf in large language models part i: Ppo." *arXiv preprint arXiv:2307.04964* (2023).
- Zhou, Kaitlyn, Dan Jurafsky, and Tatsunori B. Hashimoto. "Navigating the Grey Area: How Expressions of Uncertainty and Overconfidence Affect Language Models." In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5506-5524. 2023.
- Ziegler, Daniel M., Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. "Fine-tuning language models from human preferences." *arXiv preprint arXiv:1909.08593* (2019).