CS 690: Human-Centric Machine Learning

Prof. Scott Niekum

RLHF without reward modeling

Part 1: Direct Preference Optimization

Do we really need reward inference for RLHF?

Slide credits: Archit Sharma, Rafael Rafailov, Eric Mitchell

RLHF: RL From human feedback

RLHF: RL From human feedback

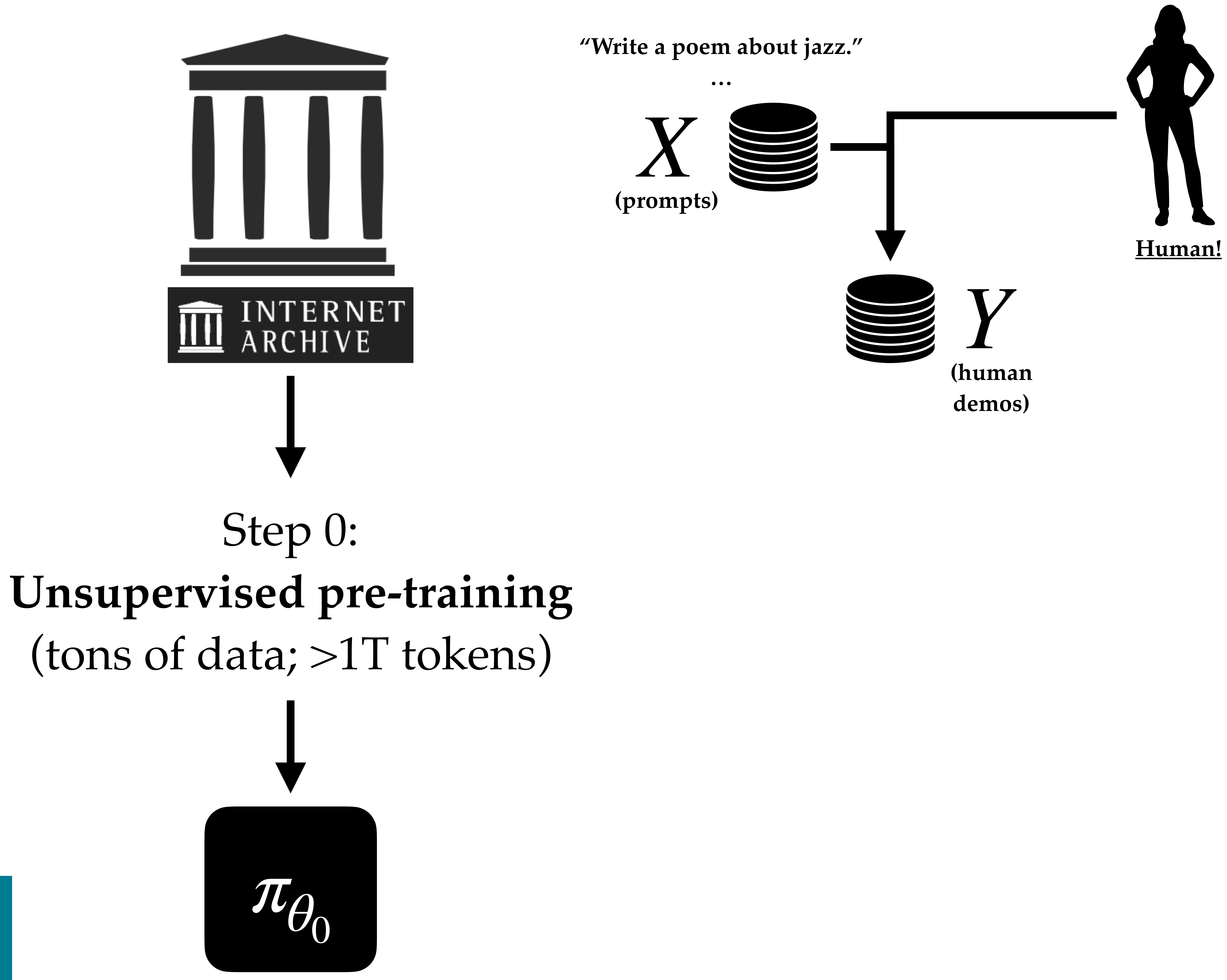
RLHF: RL From human feedback

INTERNET ARCHIVE

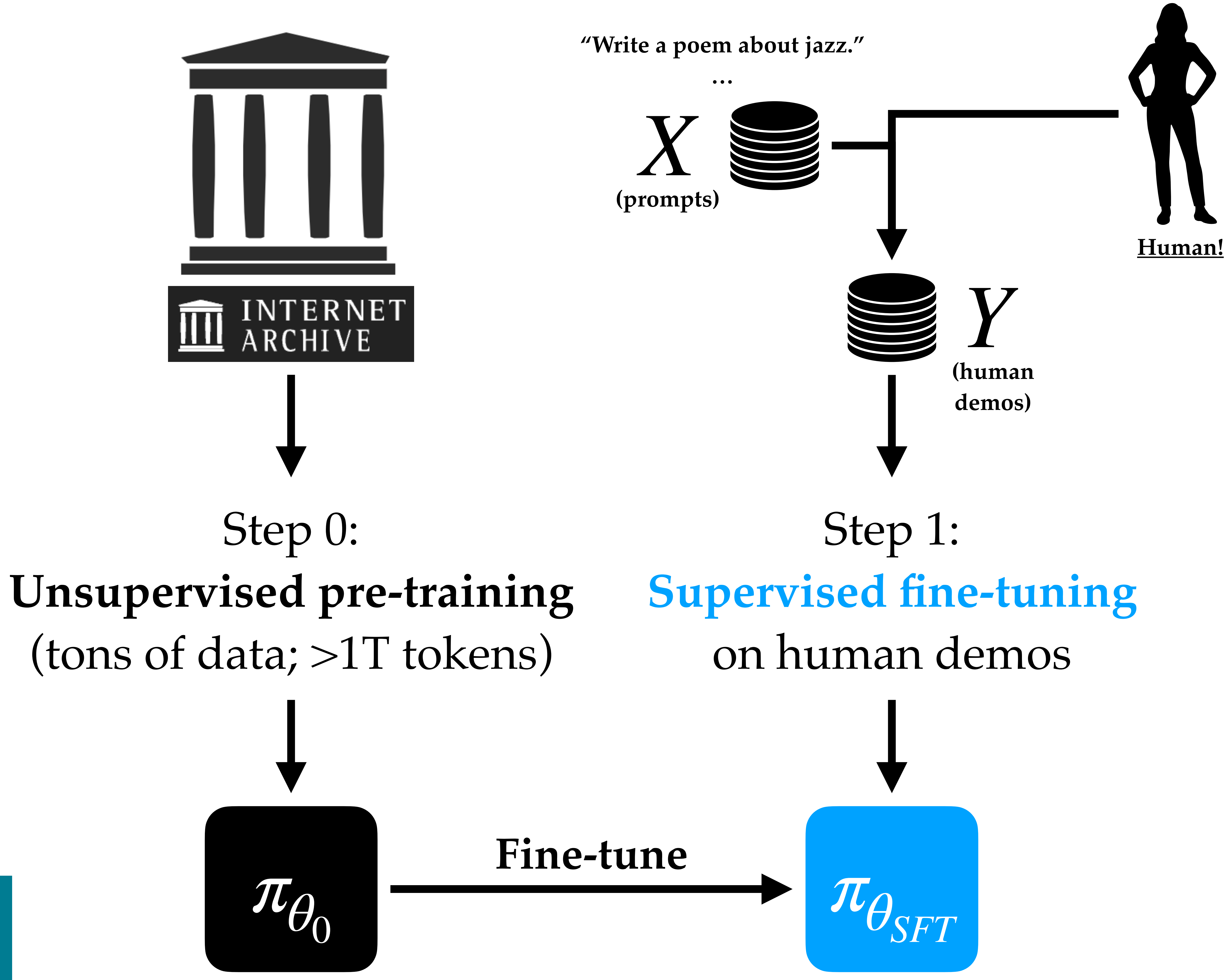
Step 0:
Unsupervised pre-training
(tons of data; >1T tokens)

π_{θ_0}

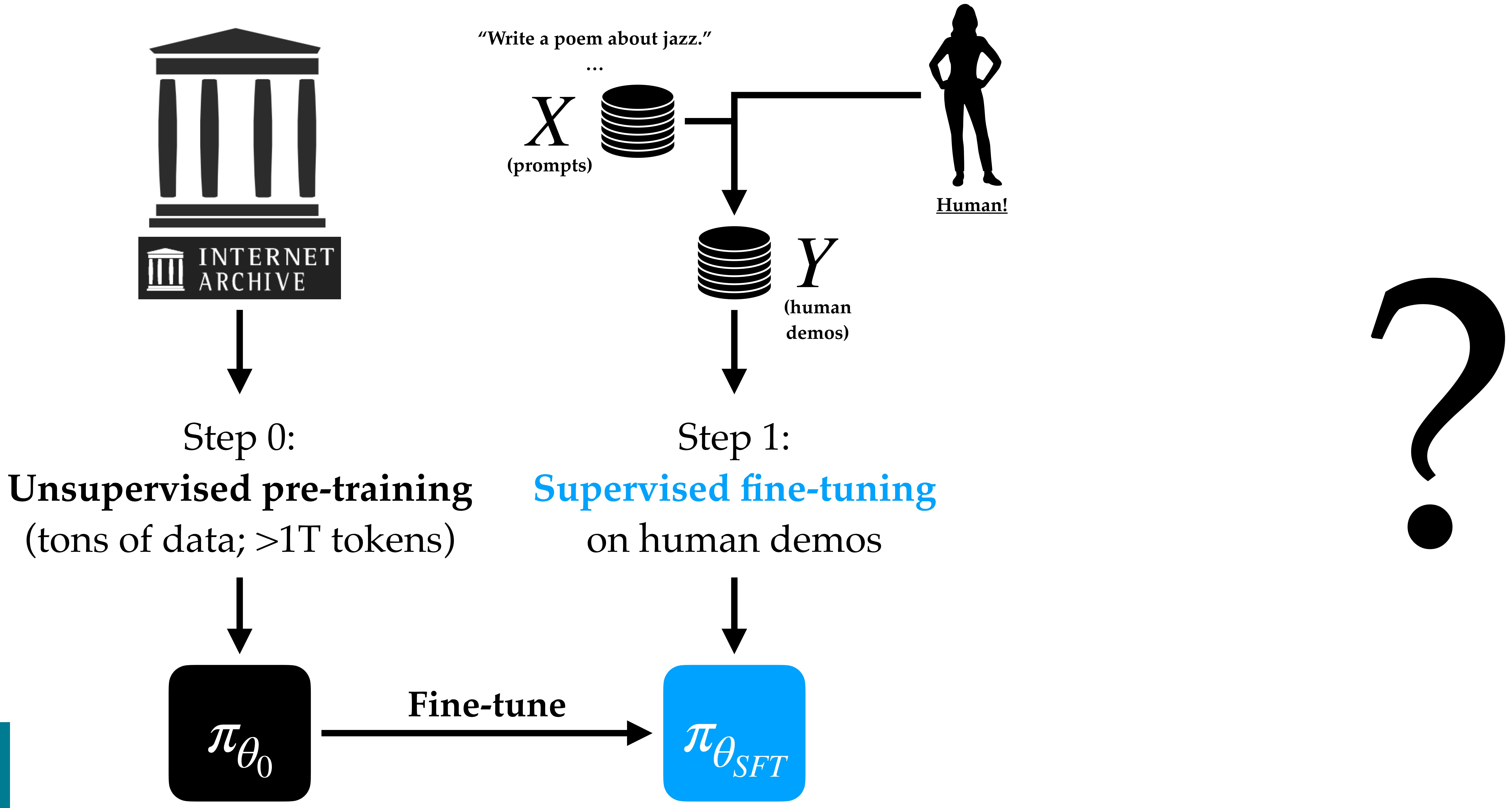
RLHF: RL From human feedback



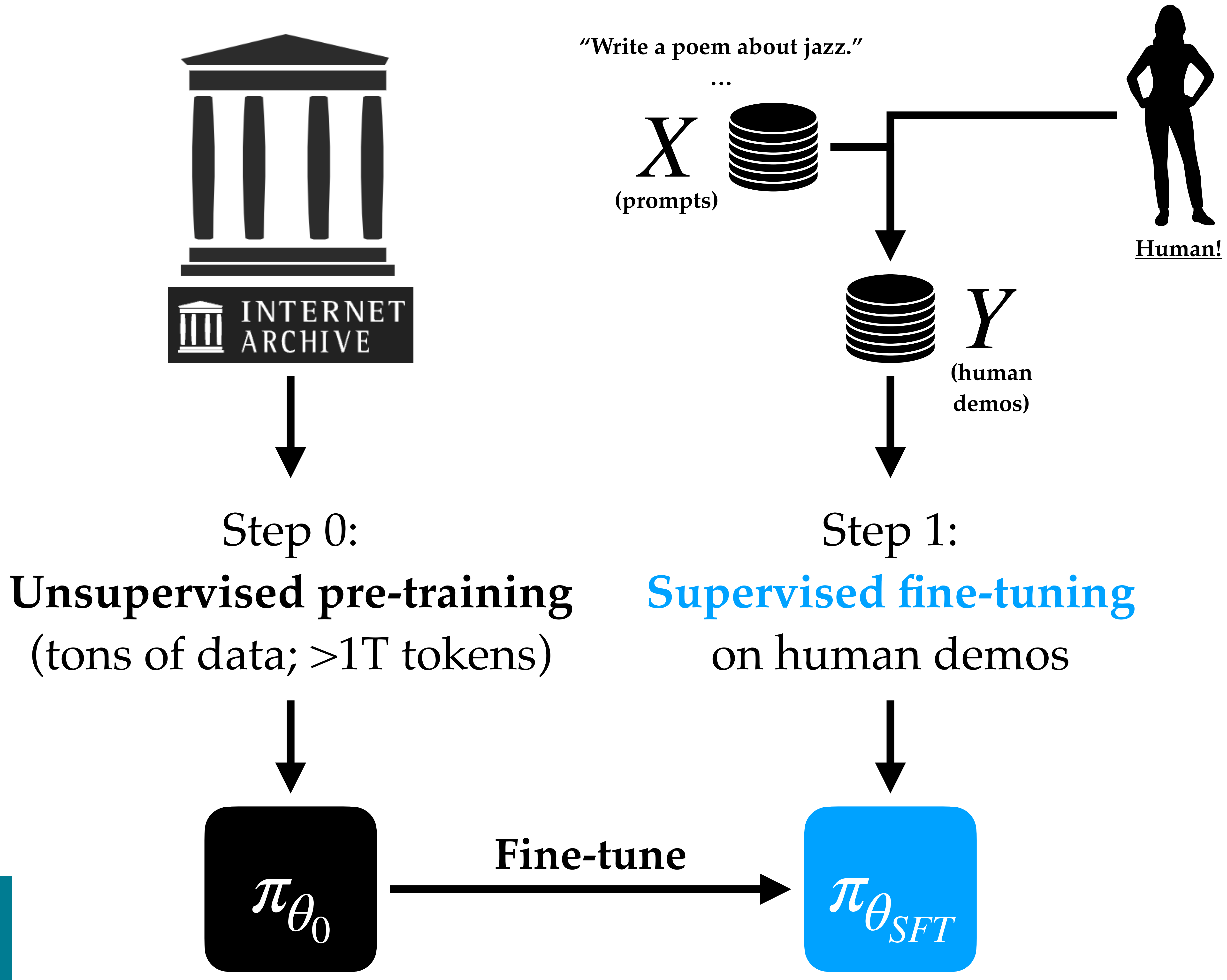
RLHF: RL From human feedback



RLHF: RL From human feedback



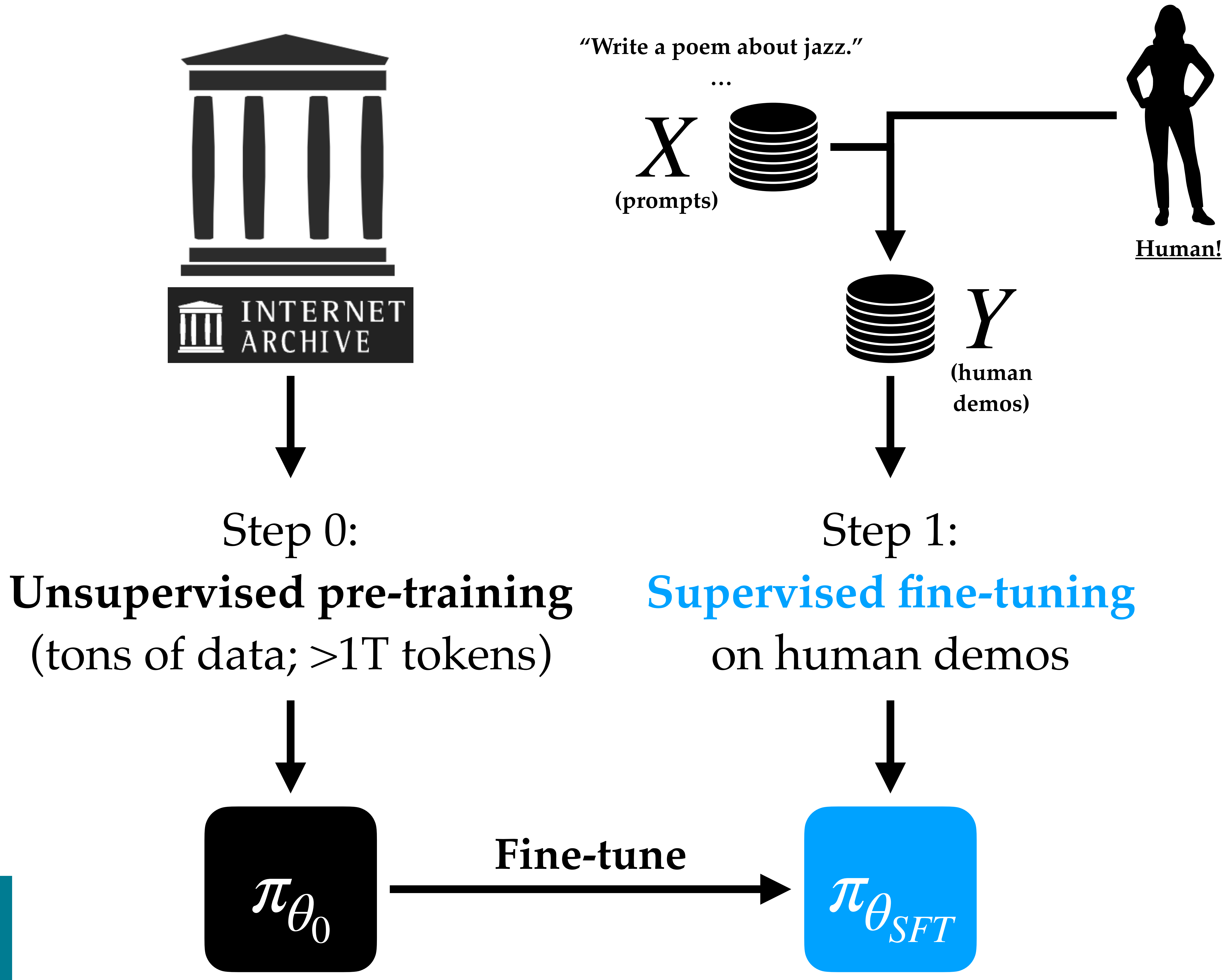
RLHF: RL From human feedback



?

Why RL at all?

RLHF: RL From human feedback



?

Why RL at all?

- Scale annotation
- Exceed human performance

RLHF: Learning rewards from preferences

Feedback comes as **preferences over model samples:** $\mathcal{D} = \{x^i, y_w^i, y_l^i\}$

- x^i: Prompt
- y_w^i: Preferred response
- y_l^i: Dispreferred response

RLHF: Learning rewards from preferences

Feedback comes as **preferences over model samples:** $\mathcal{D} = \{x^i, y_w^i, y_l^i\}$

Prompt

Preferred response

Dispreferred response

How do we get a reward function from this data?

RLHF: Learning rewards from preferences

Feedback comes as **preferences over model samples:** $\mathcal{D} = \{x^i, y_w^i, y_l^i\}$

Prompt — Preferred response — Dispreferred response

How do we get a reward function from this data?

Bradley-Terry Model connects scores (rewards?) to preferences:

Unobserved implicit score assigned to each choice

$$p(a \succ b) = \sigma(s(a) - s(b))$$

[Rank Analysis of Incomplete Block Designs. Bradley & Terry, 1952.]

RLHF: Learning rewards from preferences

Feedback comes as **preferences over model samples:** $\mathcal{D} = \{x^i, y_w^i, y_l^i\}$

Prompt — Preferred response — Dispreferred response

How do we get a reward function from this data?

Bradley-Terry Model connects scores (rewards?) to preferences:

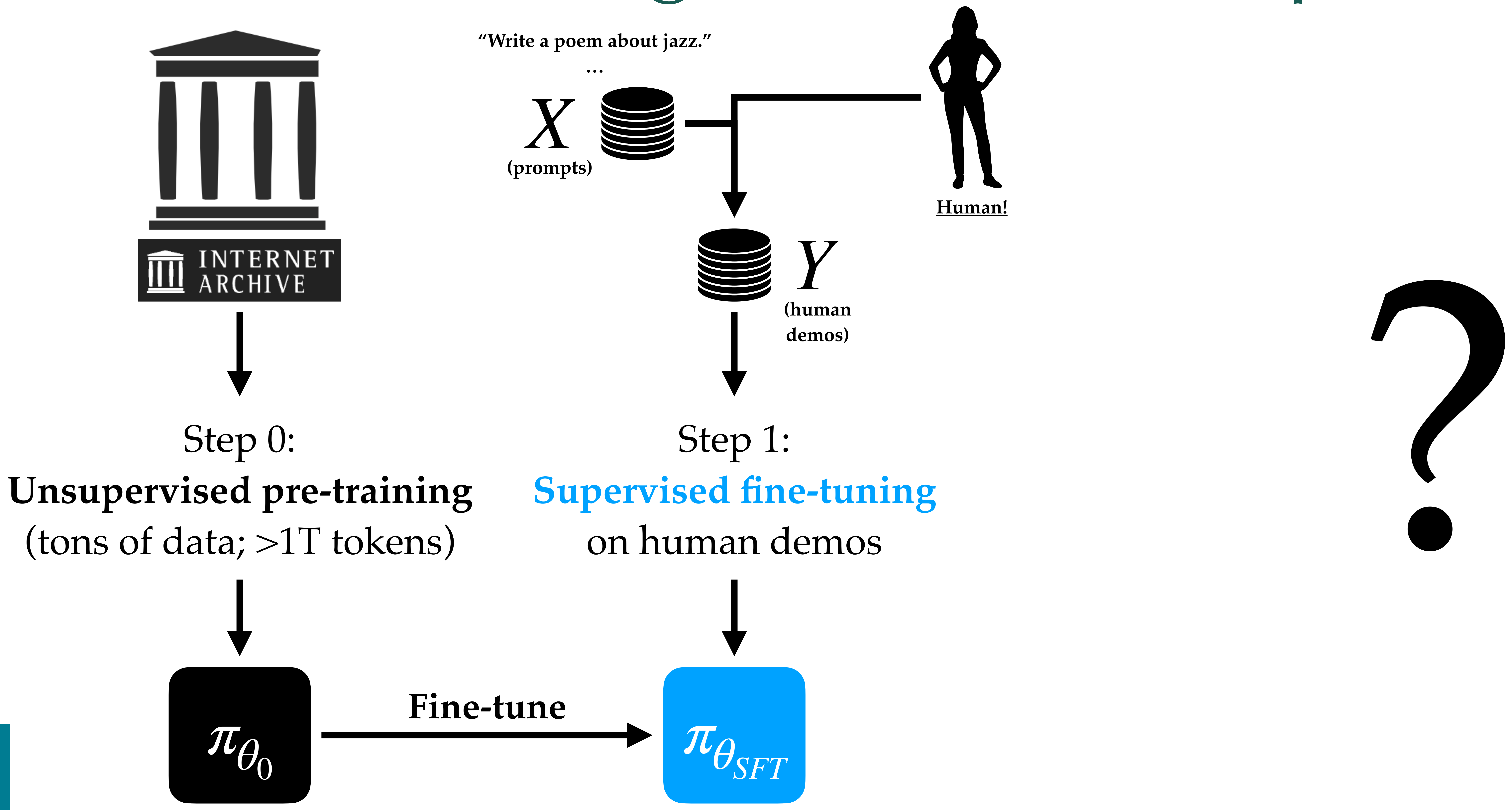
Unobserved implicit score assigned to each choice

$$p(a \succ b) = \sigma(s(a) - s(b))$$

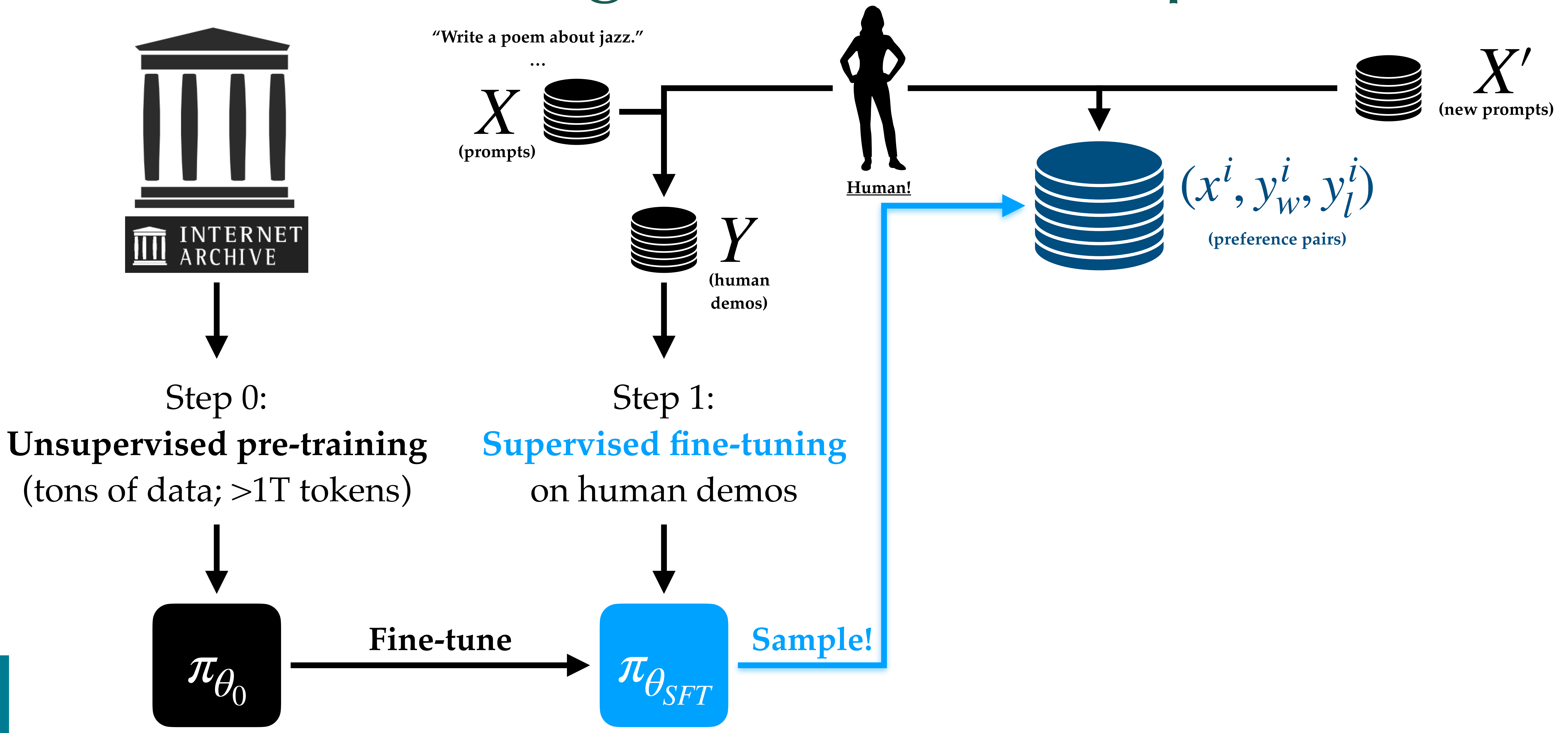
Train the reward model by **minimizing negative log likelihood:**

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

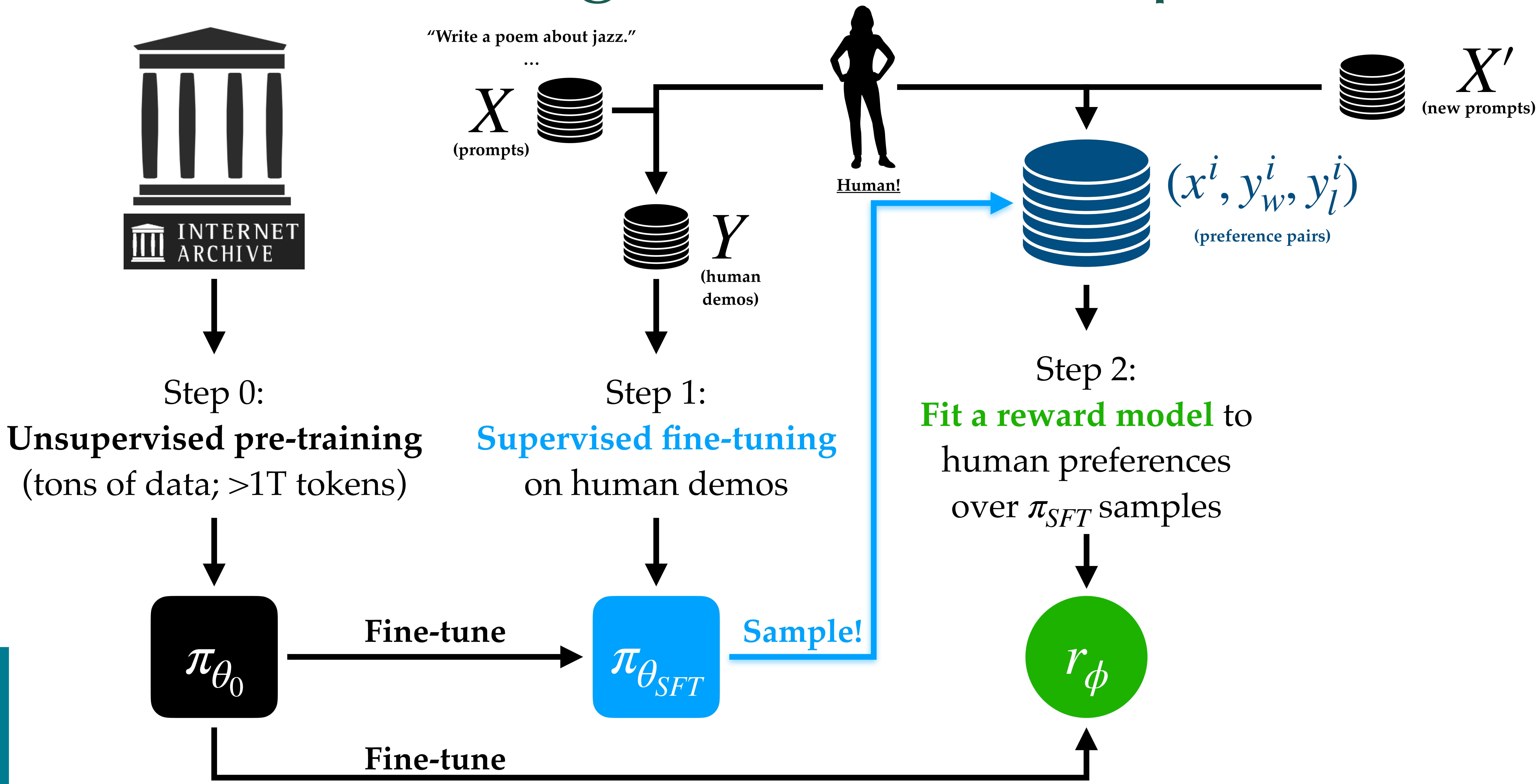
RLHF: Learning rewards from preferences



RLHF: Learning rewards from preferences



RLHF: Learning rewards from preferences



RLHF: Learning a policy that optimizes reward

Now we have a **reward model** r_ϕ representing **goodness according to humans (allegedly)**

RLHF: Learning a policy that optimizes reward

Now we have a **reward model** r_ϕ representing **goodness according to humans (allegedly)**

So we learn a policy π_θ achieving **high reward**

$$\max_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(y|x)} \left[r_\phi(x, y) \right]$$

Sample from policy

Want high reward …

RLHF: Learning a policy that optimizes reward

Now we have a **reward model** r_ϕ representing **goodness according to humans (allegedly)**

So we learn a policy π_θ achieving **high reward** while **staying close** to original model π_{ref}

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

Sample from policy

Want high reward …

… but keep KL to original model small!

RLHF: Learning a policy that optimizes reward

Now we have a **reward model** r_ϕ representing **goodness according to humans (allegedly)**

So we learn a policy π_θ achieving **high reward** while **staying close** to original model π_{ref}

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

Sample from policy

Want high reward …

… but keep KL to original model small!

Avoid outputs where our **reward model is inaccurate** (it was trained on π_{ref} outputs!)

RLHF: Learning a policy that optimizes reward

Now we have a **reward model** r_ϕ representing **goodness according to humans (allegedly)**

So we learn a policy π_θ achieving **high reward** while **staying close** to original model π_{ref}

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

Sample from policy

Want high reward …

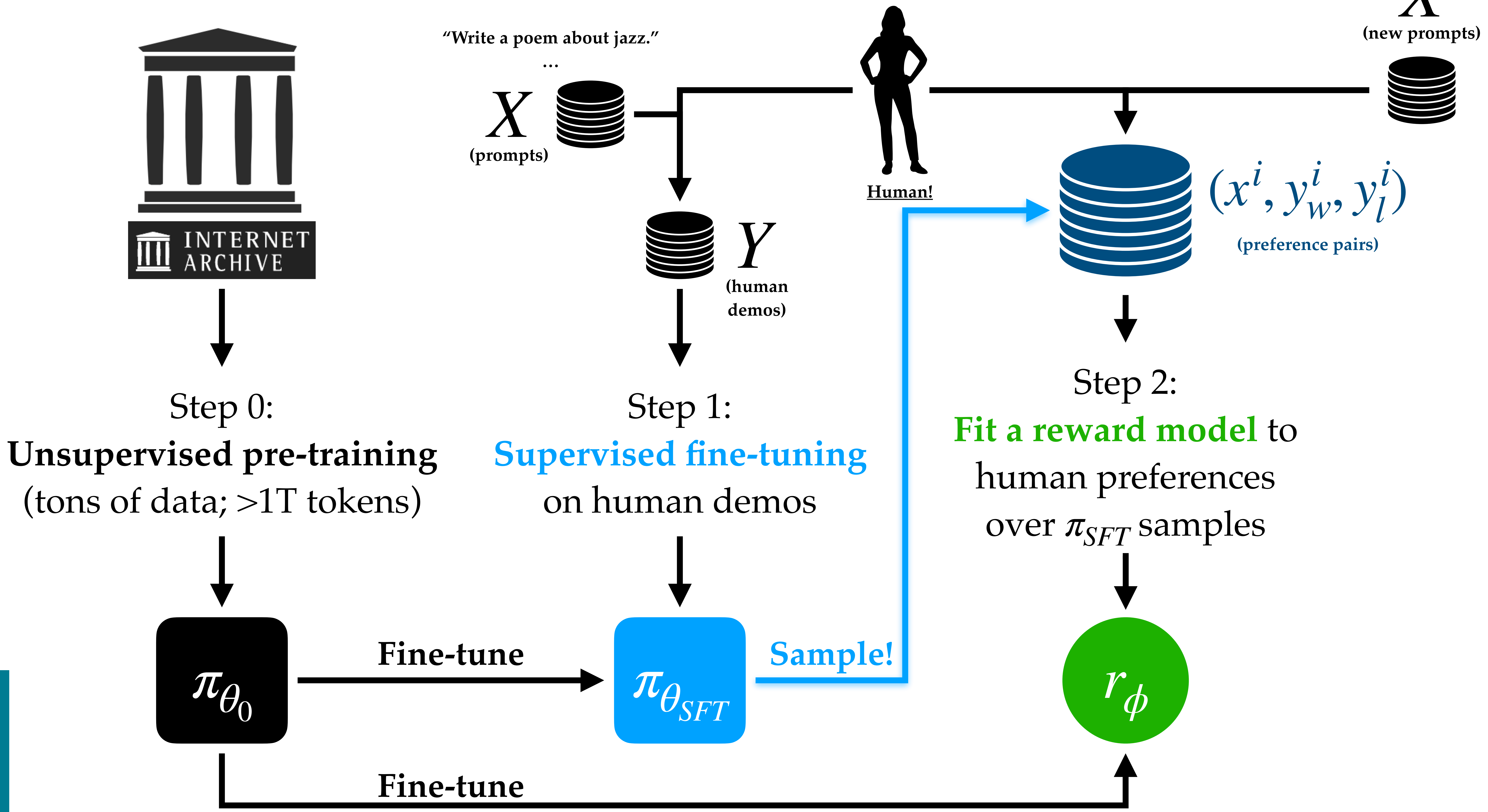
… but keep KL to original model small!

Avoid outputs where our **reward model is inaccurate** (it was trained on π_{ref} outputs!)

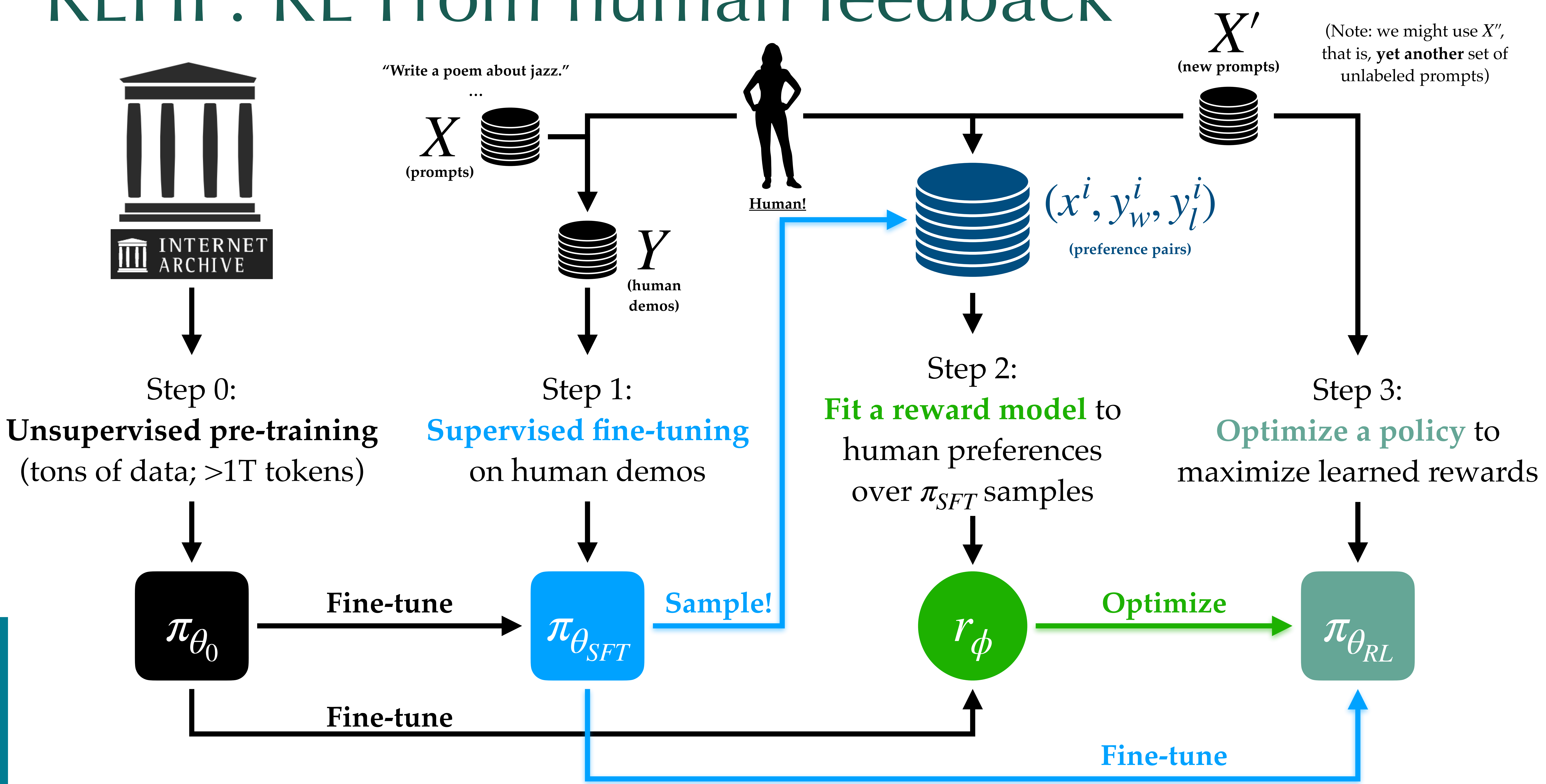
Optimize the whole thing with PPO (off-the-shelf RL algorithm)

[**Proximal Policy Optimization** Algorithms, Schulman, Wolski, Dhariwal, Radford, Klimov, 2017]

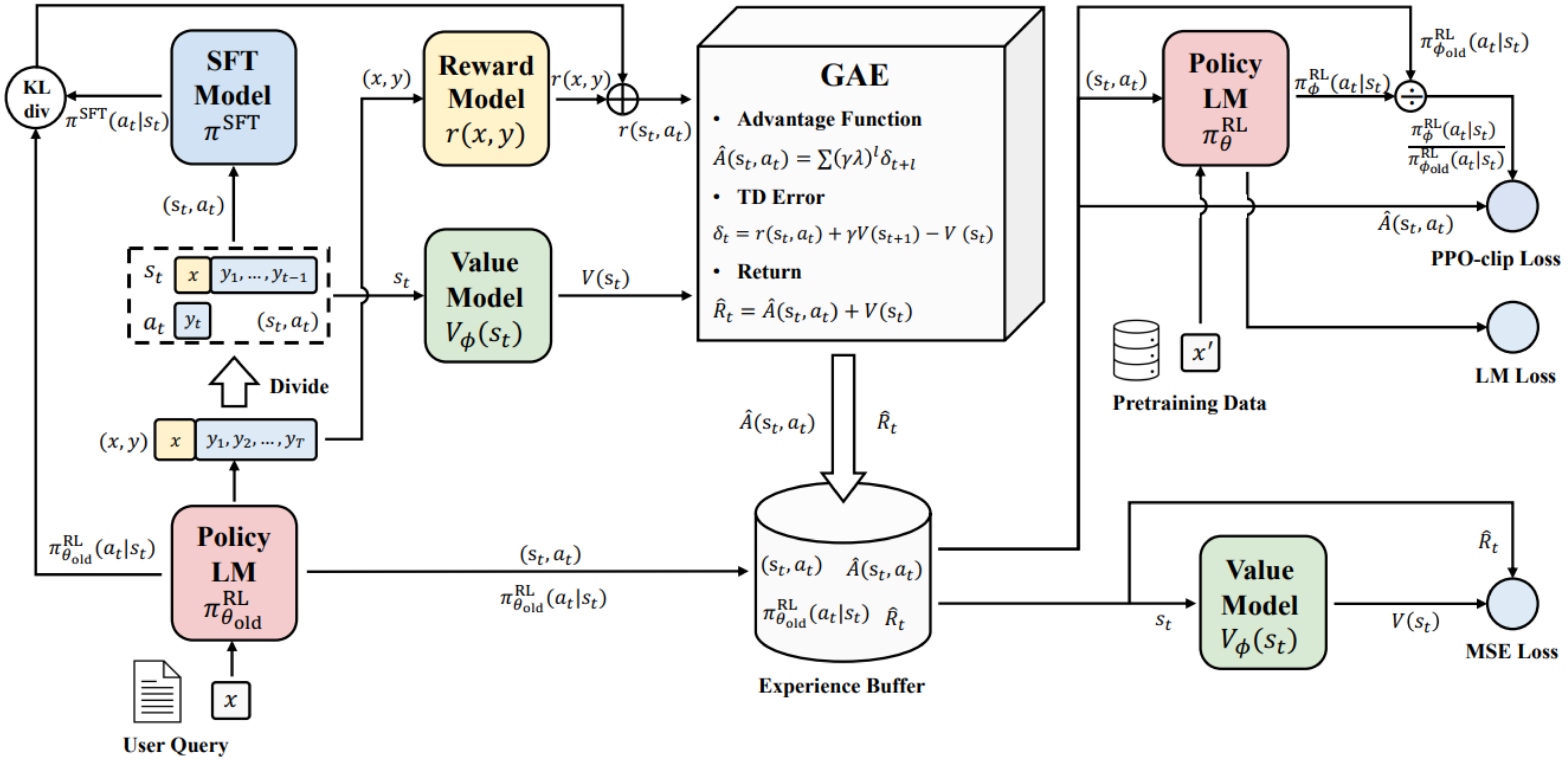
RLHF: RL From human feedback



RLHF: RL From human feedback



Traditional RLHF is *complex*



[Secrets of RLHF in Large Language Models Part I: PPO, Zheng, et al. 2023]

Direct Preference Optimization

Direct Preference Optimization

High-level punchline

If we parameterize our reward model correctly…

Direct Preference Optimization

High-level punchline

If we parameterize our reward model correctly…

…we can extract the optimal policy for our learned reward model **in closed form, with no additional training**

Direct Preference Optimization

High-level punchline

If we parameterize our reward model correctly…

…we can extract the optimal policy for our learned reward model **in closed form, with no additional training**

The trick: use a direct correspondence between optimal policy and reward model!

$$\pi(y|x) \Leftrightarrow r(x, y)$$

Direct Preference Optimization: Putting it together

Intractable *closed-form* optimal RLHF policy

$$\pi_r^*(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

*Every **reward function** r induces an **optimal policy** π_r^**

Another view of this identity

$$r_\pi^*(x, y) = \beta \log \frac{\pi(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x)$$

But we can't compute this (sums over all sequences)!

*Every **policy** π is the optimal policy for some **induced reward function** r_π^**

Key idea of DPO: train the policy π so that r_π fits the human preference data!

Direct Preference Optimization: Putting it together

Fortunately, the reward modeling loss only depends on **differences** in rewards:

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

For two different responses, the **induced reward difference is:**

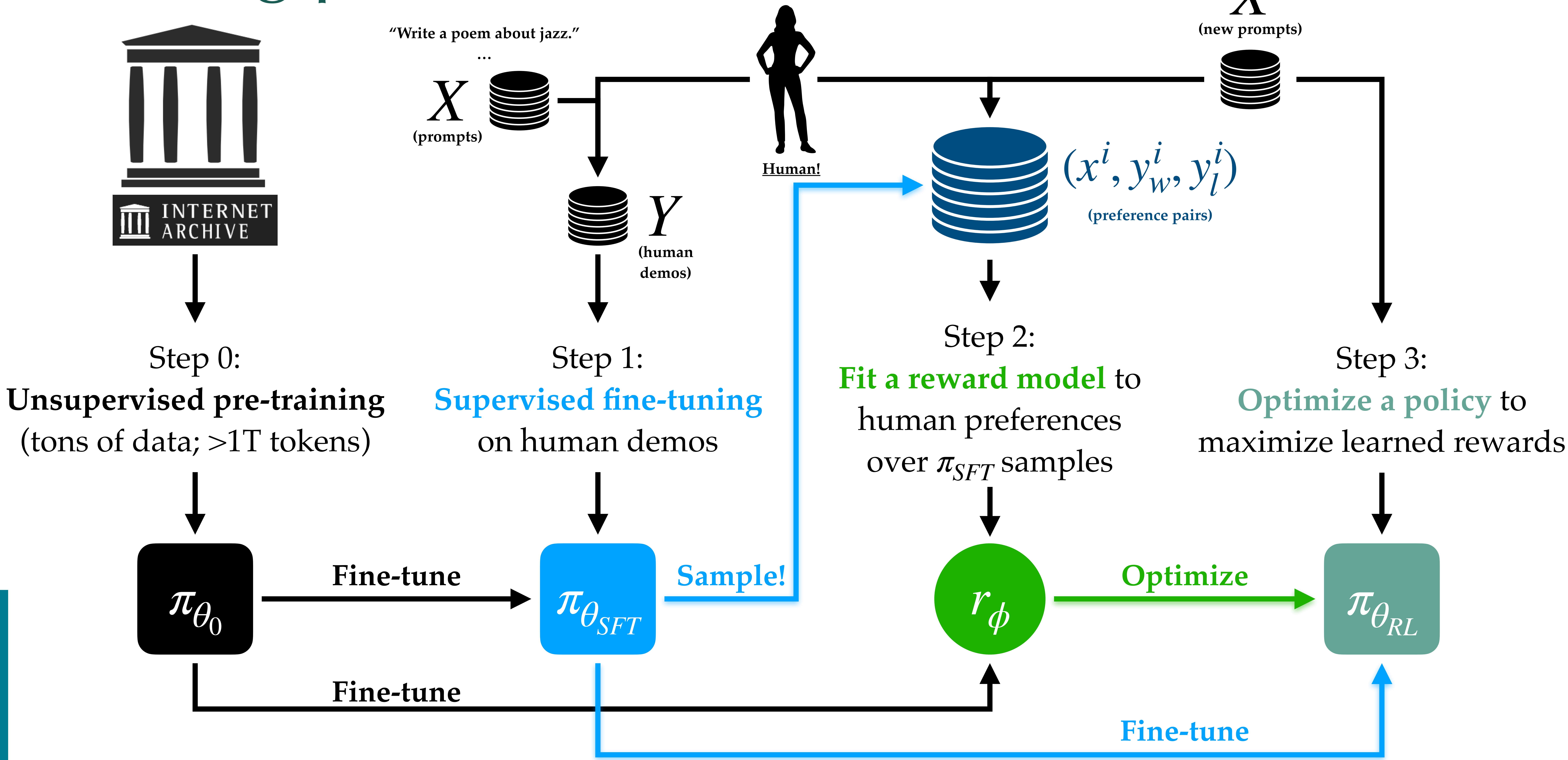
$$r_{\pi_\theta}(x, y_w) - r_{\pi_\theta}(x, y_l) = \underbrace{\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)}}_{\text{induced reward for } y_w} - \underbrace{\beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}}_{\text{induced reward for } y_l}$$

The **intractable partition function cancels out** when we take the difference (i.e., it only depends on the prompt)!

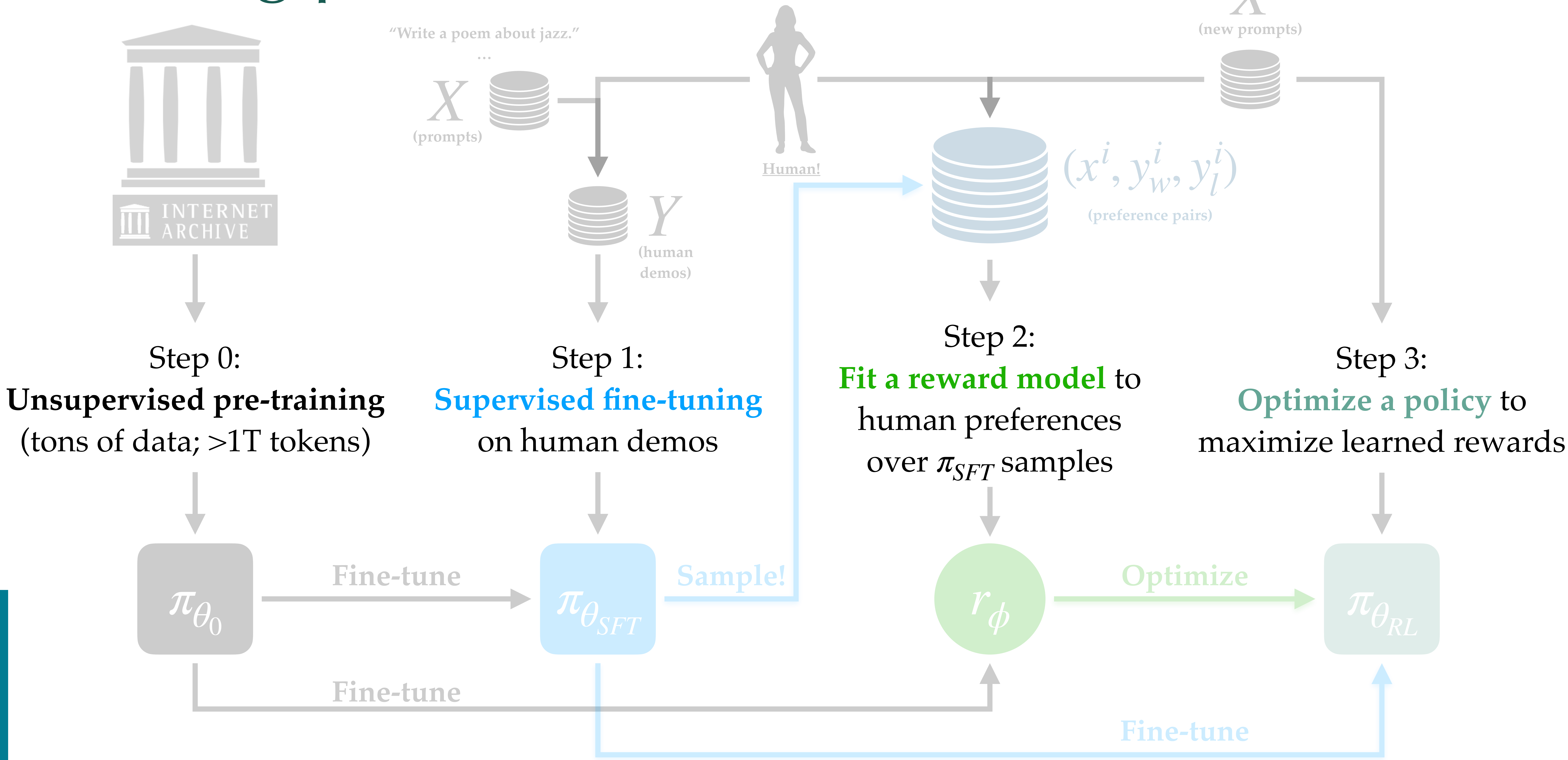
$$\mathcal{L}_{\text{DPO}}(\pi_\theta, \mathcal{D}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(r_{\pi_\theta}(x, y_w) - r_{\pi_\theta}(x, y_l) \right) \right]$$

DPO: a simple **classification loss** for optimizing the RLHF objective!

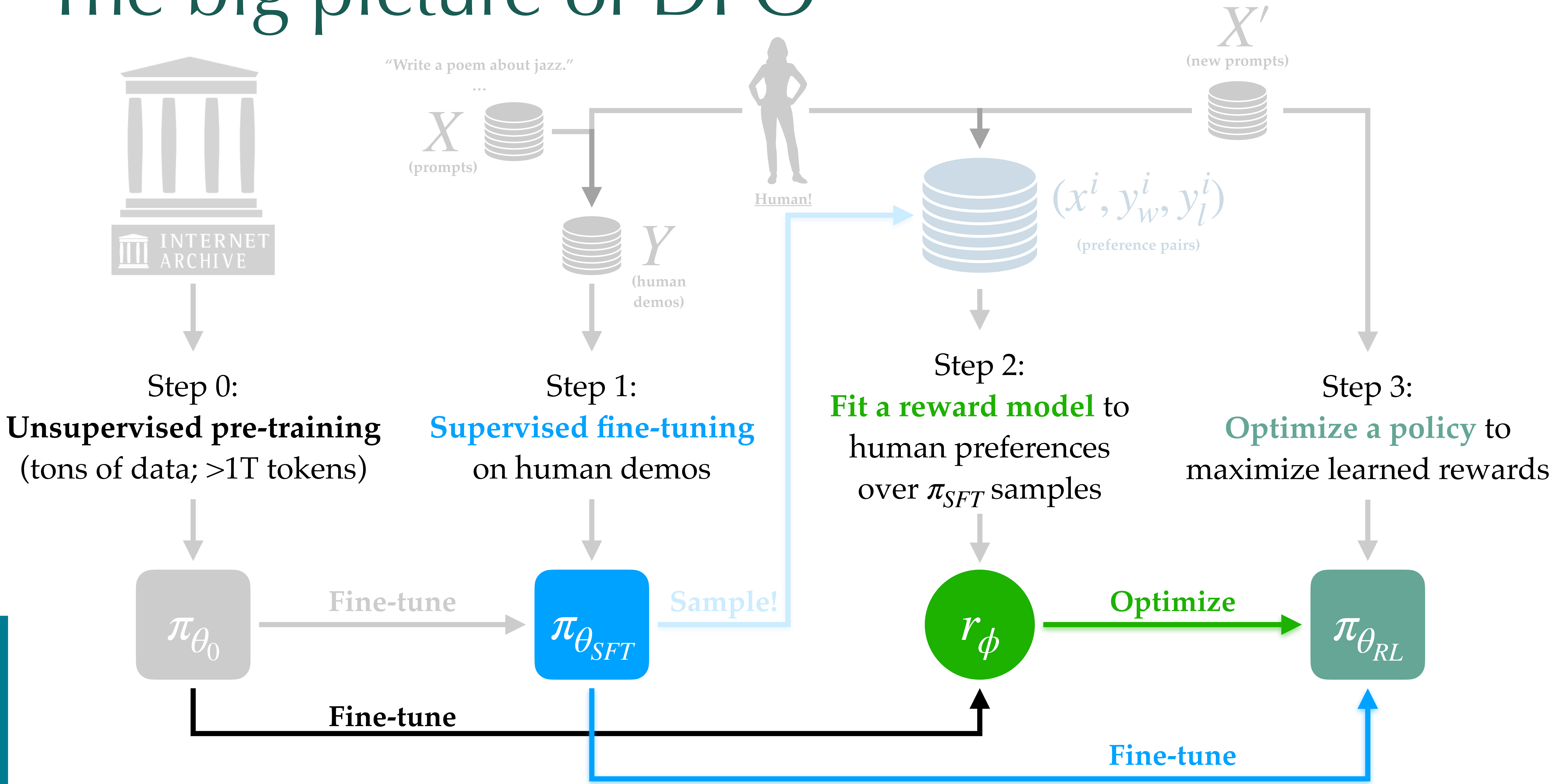
The big picture of DPO



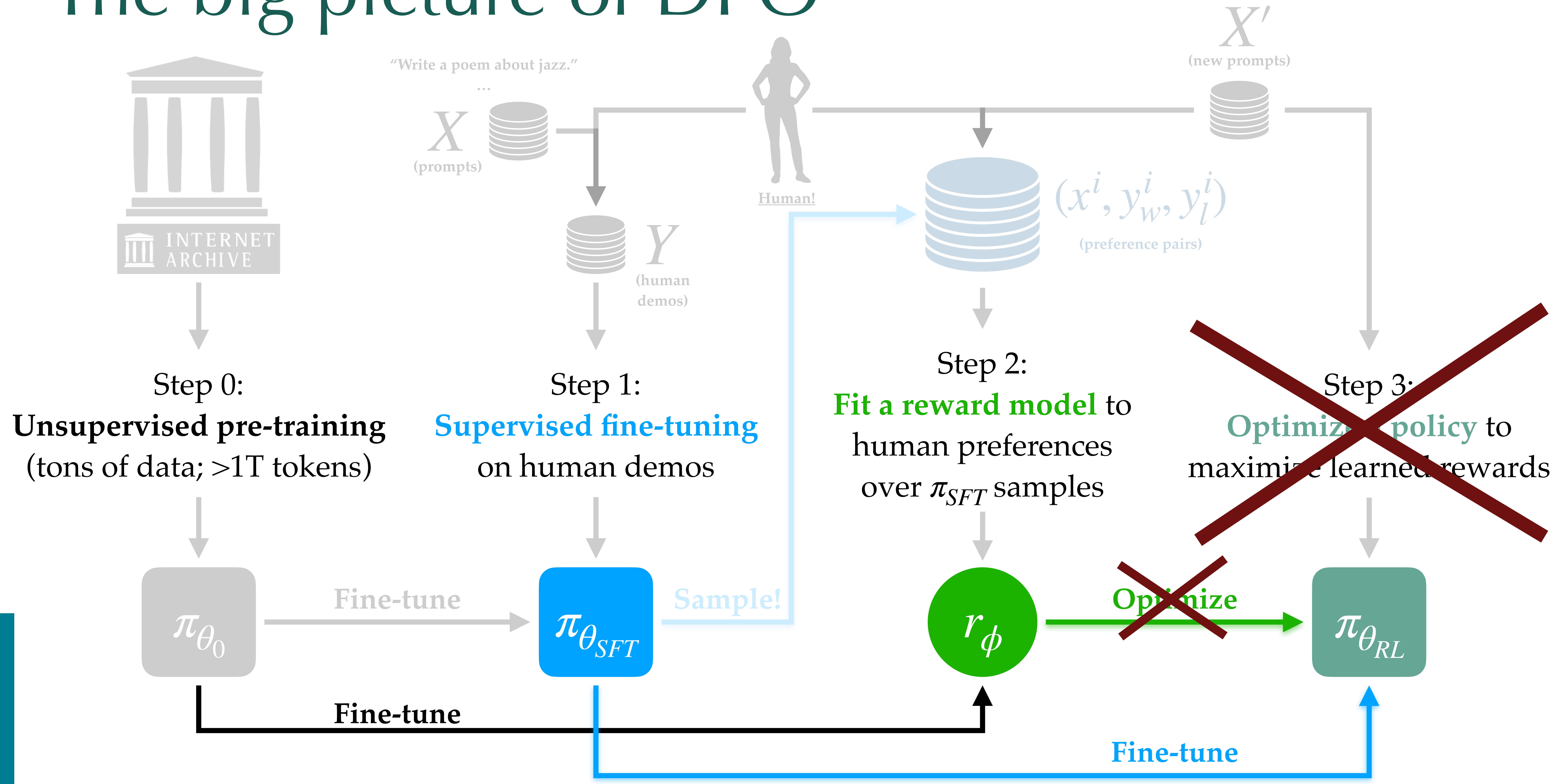
The big picture of DPO



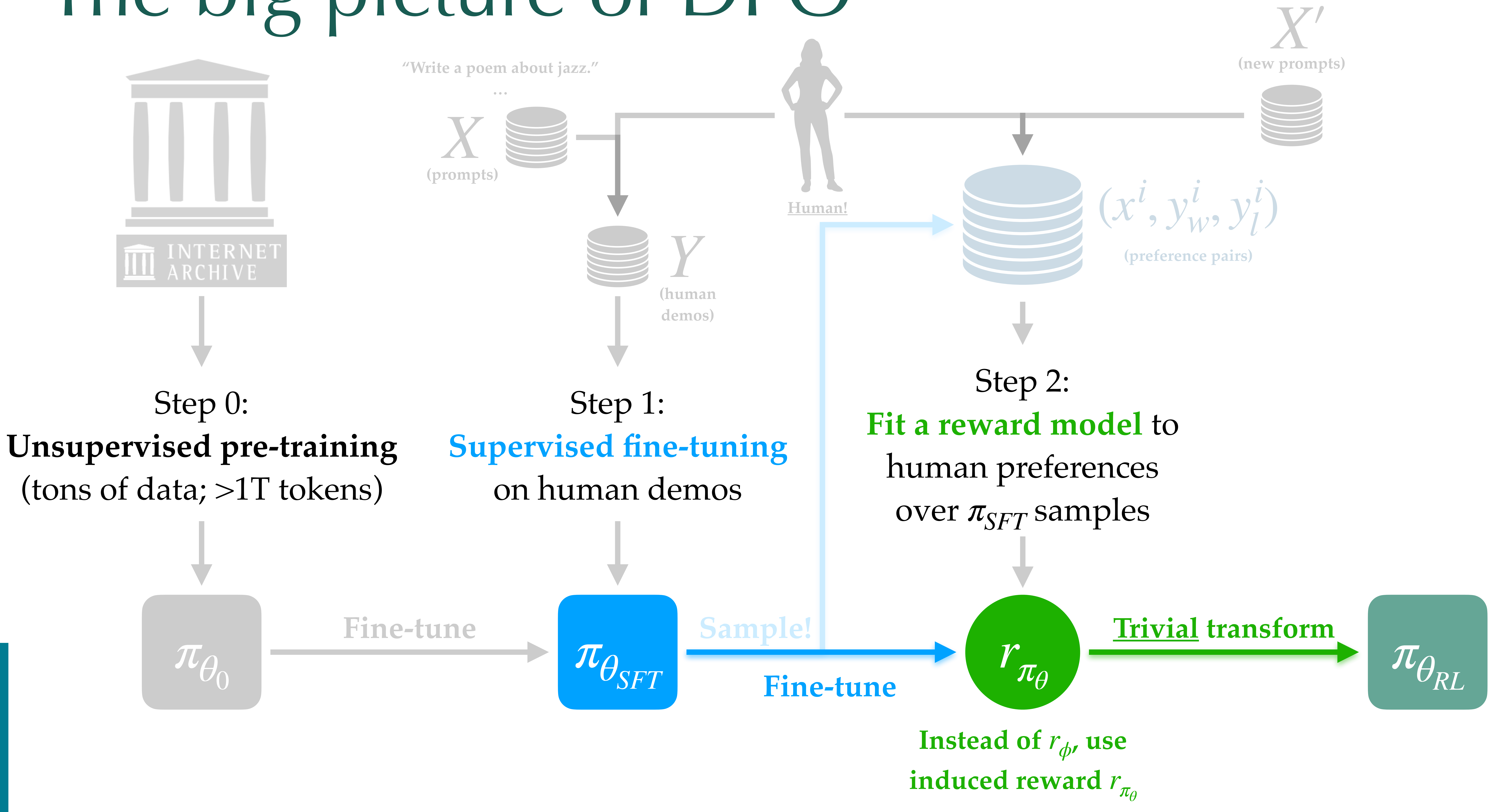
The big picture of DPO



The big picture of DPO



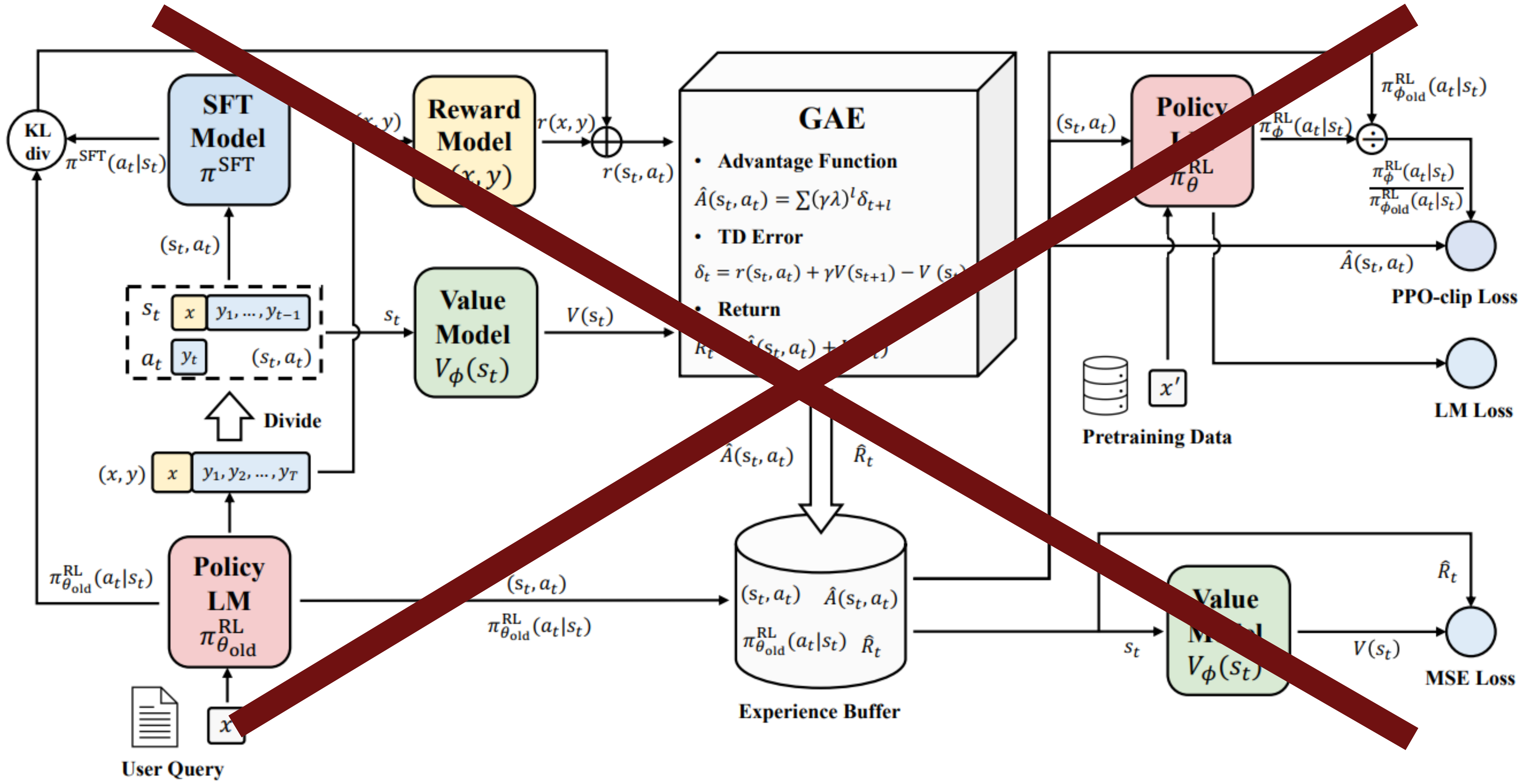
The big picture of DPO



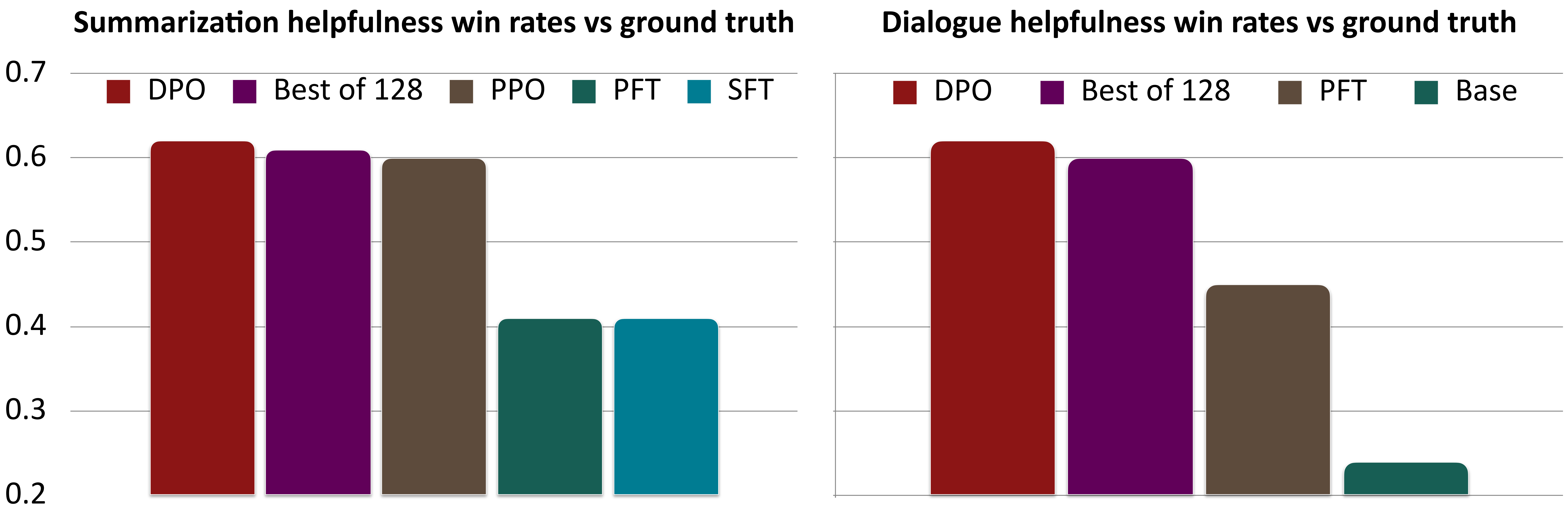
The big picture of DPO

In other words, **skip the complexity of:**

- Fitting value function
- Sampling from policy during training
- Storing replay buffer of trajectories
- …

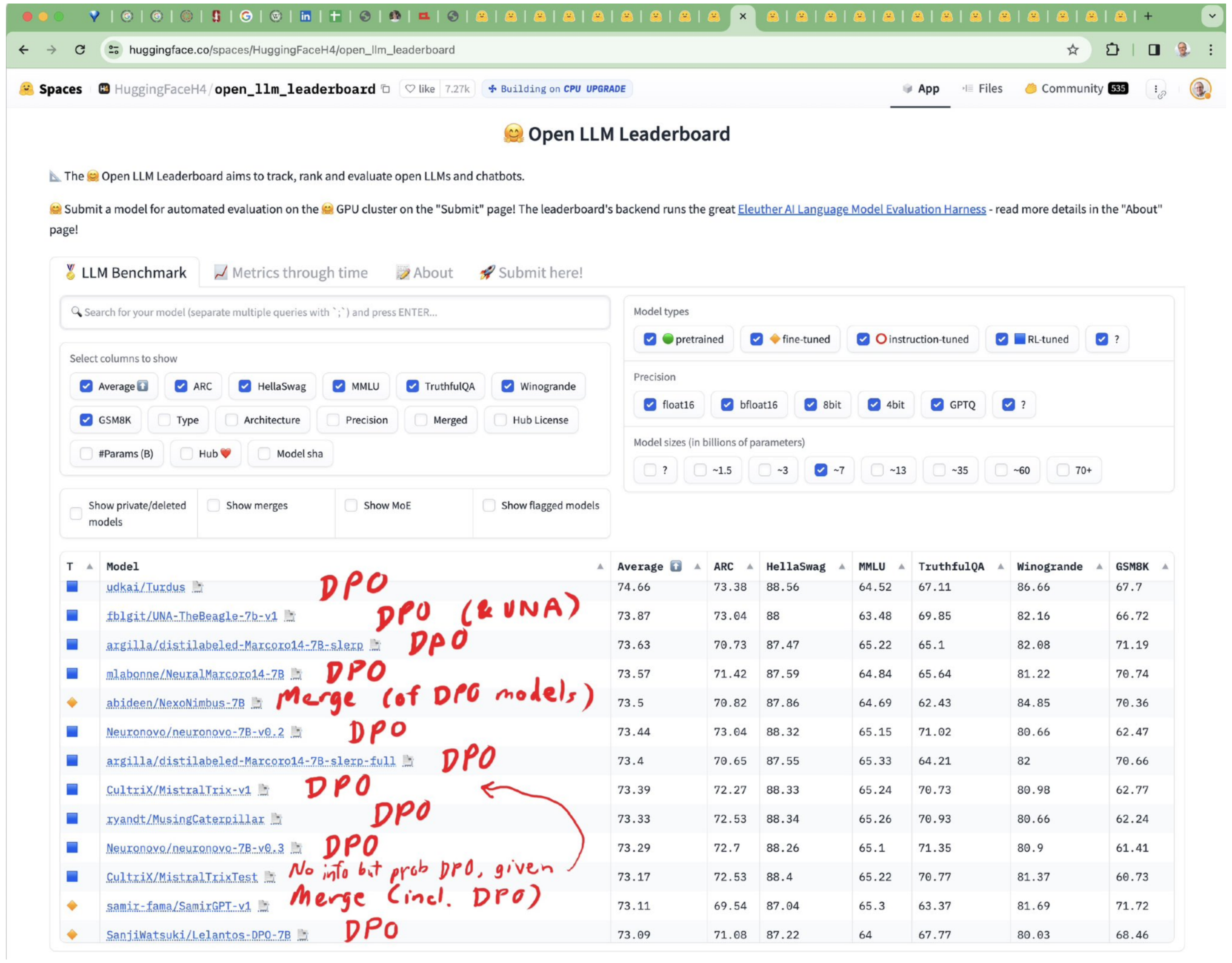


Results: Overview



DPO performs similarly to other RL-based baselines, while being **substantially simpler, computationally cheaper, and stabler**

Strong models trained with DPO



huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

Open LLM Leaderboard

The Open LLM Leaderboard aims to track, rank and evaluate open LLMs and chatbots.

Submit a model for automated evaluation on the GPU cluster on the "Submit" page! The leaderboard's backend runs the great Eleuther AI Language Model Evaluation Harness - read more details in the "About" page!

LLM Benchmark | Metrics through time | About | Submit here!

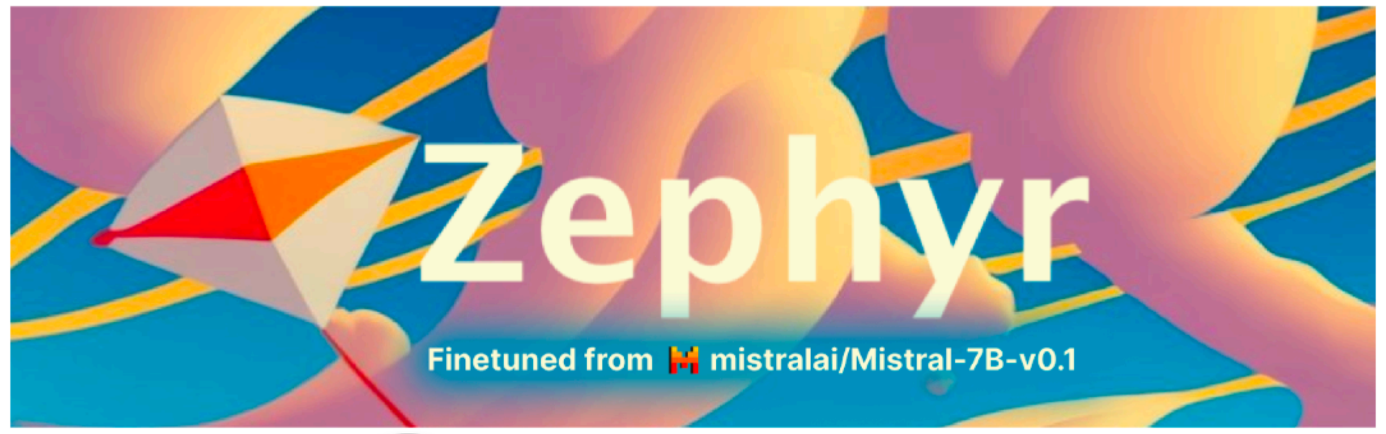
T	Model		Average	ARC	HellaSwag	MMLU	TruthfulQA	Winogrande	GSM8K
	udkai/Turdus	DPO	74.66	73.38	88.56	64.52	67.11	86.66	67.7
	fblgit/UNA-TheBeagle-7b-v1	DPO (& UNA)	73.87	73.04	88	63.48	69.85	82.16	66.72
	argilla/distilabeled-Marcoro14-7B-slerp	DPO	73.63	70.73	87.47	65.22	65.1	82.08	71.19
	mlabonne/NeuralMarcoro14-7B	DPO	73.57	71.42	87.59	64.84	65.64	81.22	70.74
	abideen/NexoNimbus-7B	Merge (of DPO models)	73.5	70.82	87.86	64.69	62.43	84.85	70.36
	Neuronovo/neuronovo-7B-v0.2	DPO	73.44	73.04	88.32	65.15	71.02	80.66	62.47
	argilla/distilabeled-Marcoro14-7B-slerp-full	DPO	73.4	70.65	87.55	65.33	64.21	82	70.66
	CultriX/MistralTrix-v1	DPO	73.39	72.27	88.33	65.24	70.73	80.98	62.77
	ryandt/MusingCaterpillar	DPO	73.33	72.53	88.34	65.26	70.93	80.66	62.24
	Neuronovo/neuronovo-7B-v0.3	DPO	73.29	72.7	88.26	65.1	71.35	80.9	61.41
	CultriX/MistralTrixTest	No info but prob DPO, given	73.17	72.53	88.4	65.22	70.77	81.37	60.73
	samir-fama/SamirGPT-v1	Merge (incl. DPO)	73.11	69.54	87.04	65.3	63.37	81.69	71.72
	SanjiWatsuki/Lelantos-DPO-7B	DPO	73.09	71.08	87.22	64	67.77	80.03	68.46

Almost all the top models on the OpenLLM Leaderboard use DPO!

MISTRAL AI_ 8X7

	GPT - 3.5	Mistral Small	Mistral Medium
MT Bench (for Instruct models)	8.32	8.30	**8.61**

https://mistral.ai/news/mixtral-of-experts/



Hugging Face

Zephyr: Direct Distillation of LLM Alignment. Tunstall et, al., 2023.

Tülu v2

Open instruction & RLHF models

AI2

Camels in a Changing Climate: Enhancing LM Adaptation with Tulu 2. Ivison, et al., 2023

Part 2: Contrastive Preference Learning

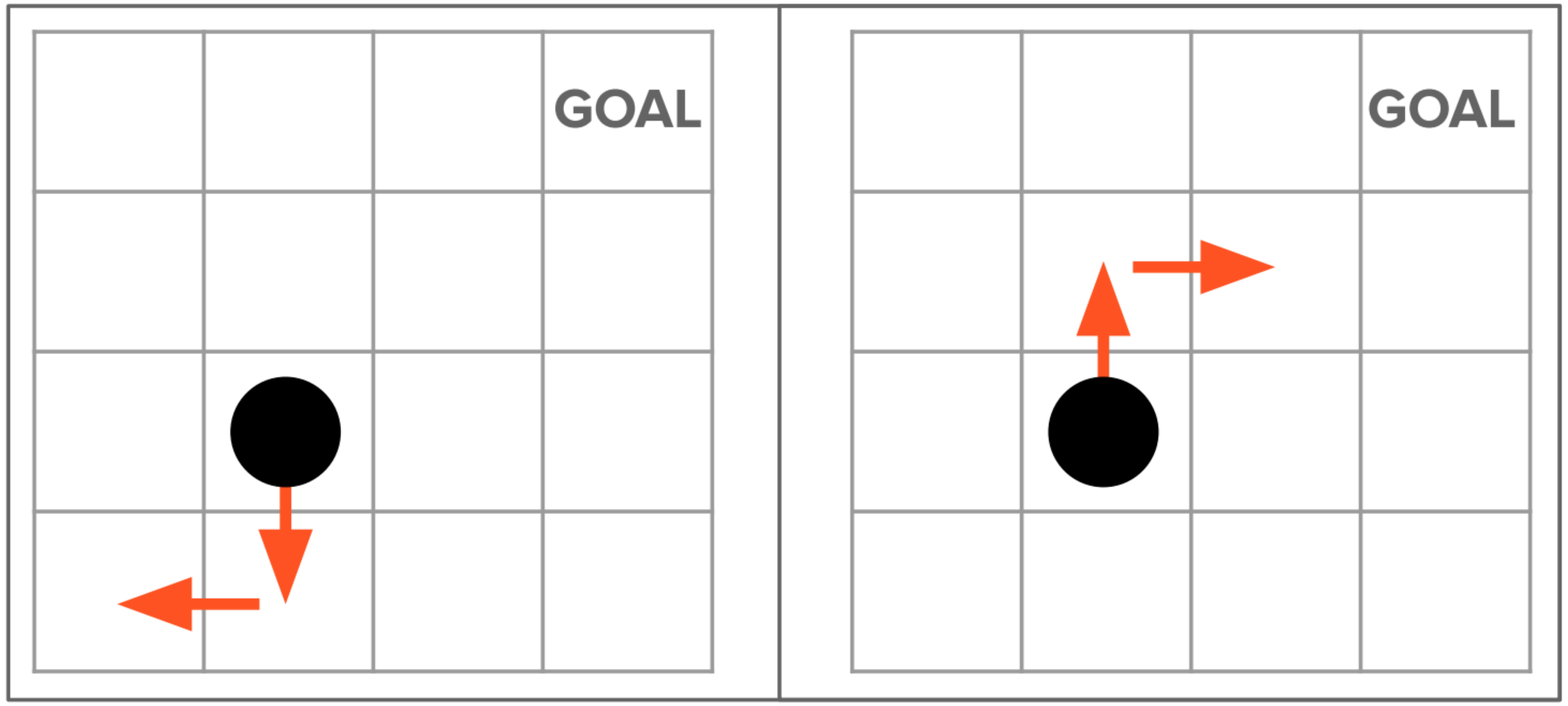
Can the DPO trick work for sequential settings?

Slide credits: Joey Hejna

Models of Human Preference for Learning Reward Functions

W. Bradley Knox, Stephane Hatgis-Kessell, Serena Booth, Scott Niekum, Peter Stone, Alessandro Allievi

Suboptimal segment **Optimal segment**



$$P_r(\sigma^+ \succ \sigma^-) = \frac{\exp \sum_t r(s_t^+, a_t^+)}{\exp \sum_t r(s_t^+, a_t^+) + \exp \sum_t r(s_t^-, a_t^-)}$$

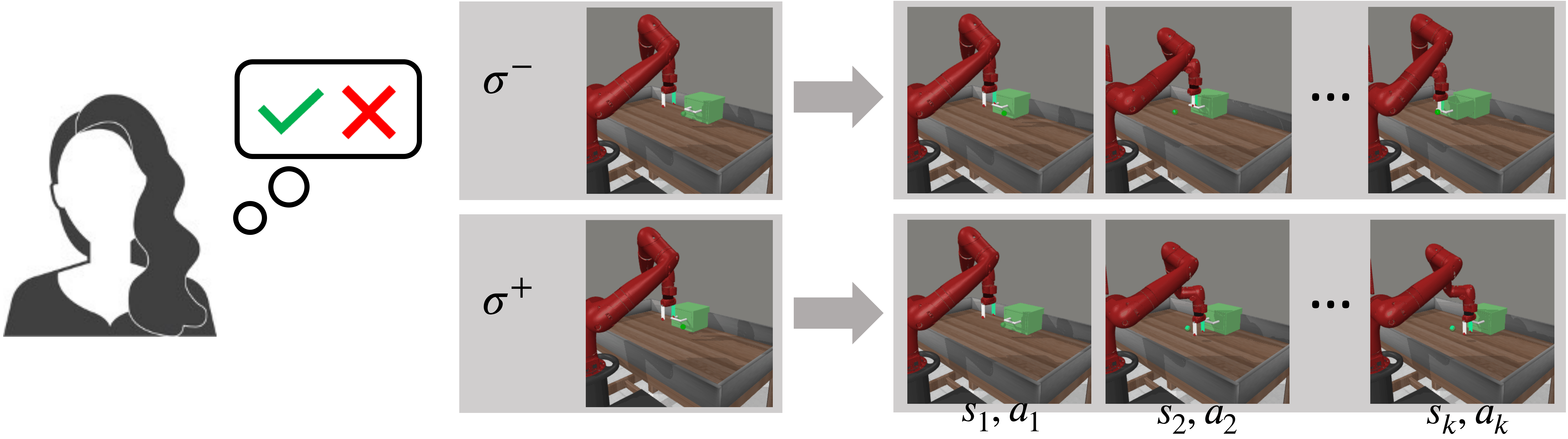
Models of Human Preference for Learning Reward Functions

W. Bradley Knox, Stephane Hatgis-Kessell, Serena Booth, Scott Niekum, Peter Stone, Alessandro Allievi

Regret-based Model of Preferences:

$$P_{A^*}(\sigma^+ \succ \sigma^-) = \frac{\exp \sum_t A^*(s_t^+, a_t^+)}{\exp \sum_t A^*(s_t^+, a_t^+) + \exp \sum_t A^*(s_t^-, a_t^-)}$$

…but no efficient algorithm to learn from it!



$$P_{A^*}(\sigma^+ \succ \sigma^-) = \frac{\exp \sum_t A^*(s_t^+, a_t^+)}{\exp \sum_t A^*(s_t^+, a_t^+) + \exp \sum_t A^*(s_t^-, a_t^-)}$$

A Naïve Approach

$$\min_A -\mathbb{E}_D\left[\log P_A\left(\sigma^+ \succ \sigma^-\right)\right]$$

1. Advantage Learning

$$\min_\pi -\mathbb{E}_D\left[e^{A(s,a)}\log\pi(a\,|\,s)\right]$$

2. Policy Extraction

Problem: (Ziebart 2010)

$$\pi^*(a\,|\,s) = e^{A^*(s,a)} \implies \int e^{A^*(s,a)}da = 1$$

To be optimal, our learned advantage must be normalized.

Solution: Just learn π

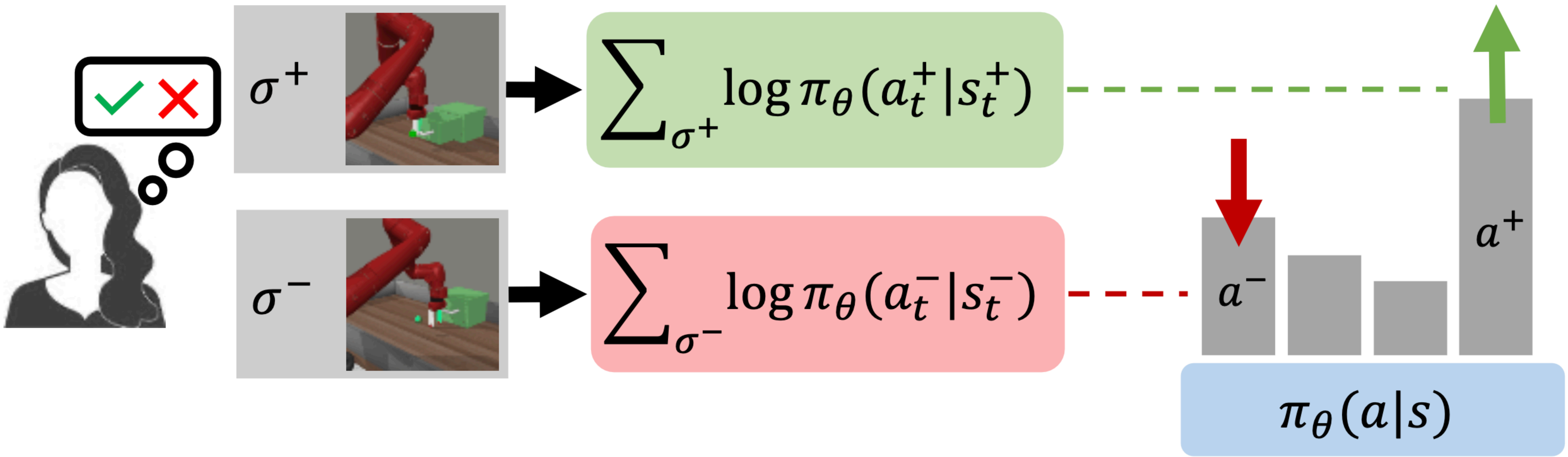
$$\pi^*(a \mid s) = e^{A^*(s,a)} \implies \log \pi^*(a \mid s) = A^*(s,a)$$

$$P_{\pi^*}\left(\sigma^+ \succ \sigma^-\right) = \frac{\exp \sum_t \log \pi^*(a_t^+ \mid s_t^+)}{\exp \sum_t \log \pi^*(a_t^+ \mid s_t^+) + \exp \sum_t \log \pi^*(a_t^- \mid s_t^-)}$$

Contrastive Preference Learning

$$\min_\pi -\mathbb{E}_D \left[\log \frac{\exp \sum_t \log \pi(a_t^+ \mid s_t^+)}{\exp \sum_t \log \pi(a_t^+ \mid s_t^+) + \exp \sum_t \log \pi(a_t^- \mid s_t^-)} \right]$$

Contrastive Preference Learning



Regret-based Preferences

$$P_{A^*}[\sigma^+ \succ \sigma^-] = \frac{e^{\sum_{\sigma^+} A^*(s_t^+, a_t^+)}}{e^{\sum_{\sigma^+} A^*(s_t^+, a_t^+)} + e^{\sum_{\sigma^-} A^*(s_t^-, a_t^-)}}$$

Contrastive Learning

$$L_{CPL} = -\mathbb{E}\left[\log P_{\log \pi_\theta}[\sigma^+ \succ \sigma^-]\right]$$

Theoretical Properties

Prop 1. CPL always learns the optimal policy for some reward function.

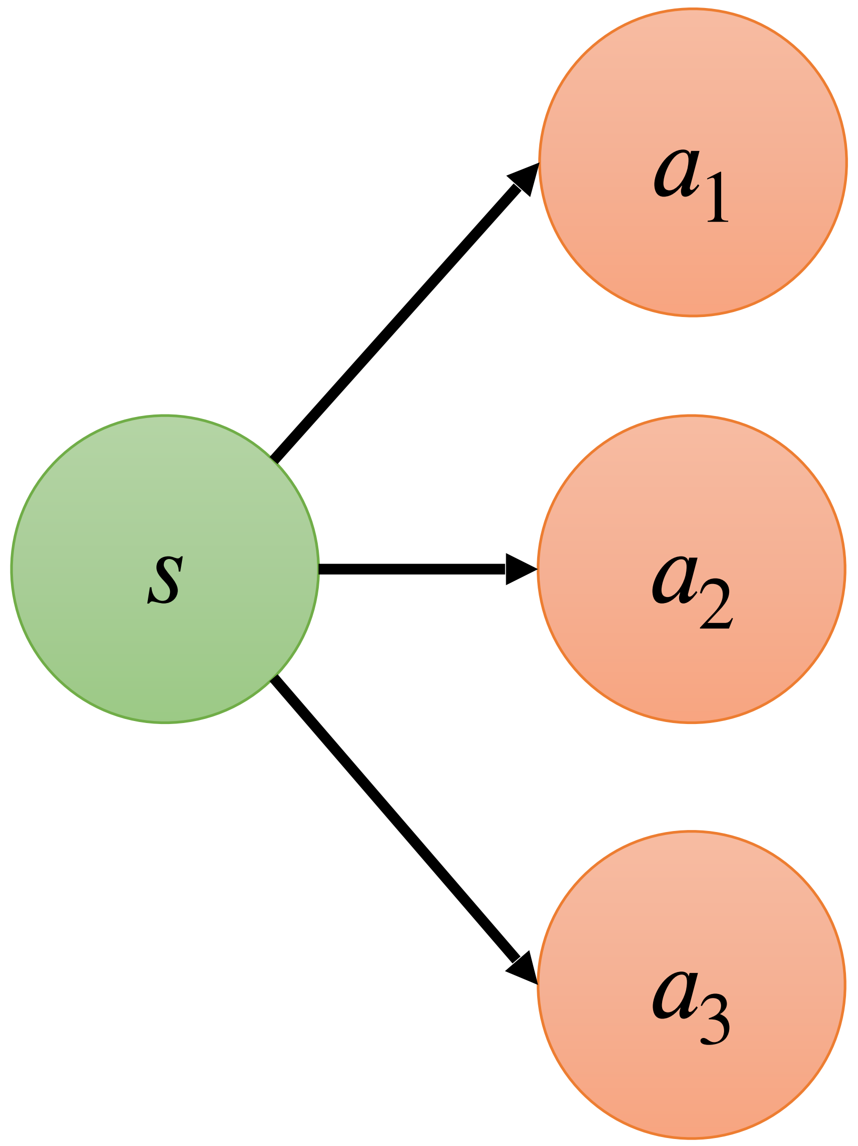
Idea: show that using the normalized advantage function as the reward function results in the same policy.

Theorem 1. Given unbounded regret-based comparison data, CPL converges to the optimal policy.

Idea: Given identifiability of regret-based preferences, CPL loss can equal zero. This implies the advantage functions are the same.

Regularization

Problem: CPL can place high-likelihood on OOD actions.



Let $D = \{(a_1 \succ a_2), (a_2 \succ a_1)\}$

Then, minimum of CPL loss is underspecified:

$$\text{logistic}\Big(\log\pi(a_1 | s) - \log\pi(a_2 | s)\Big) = \frac{1}{2}$$

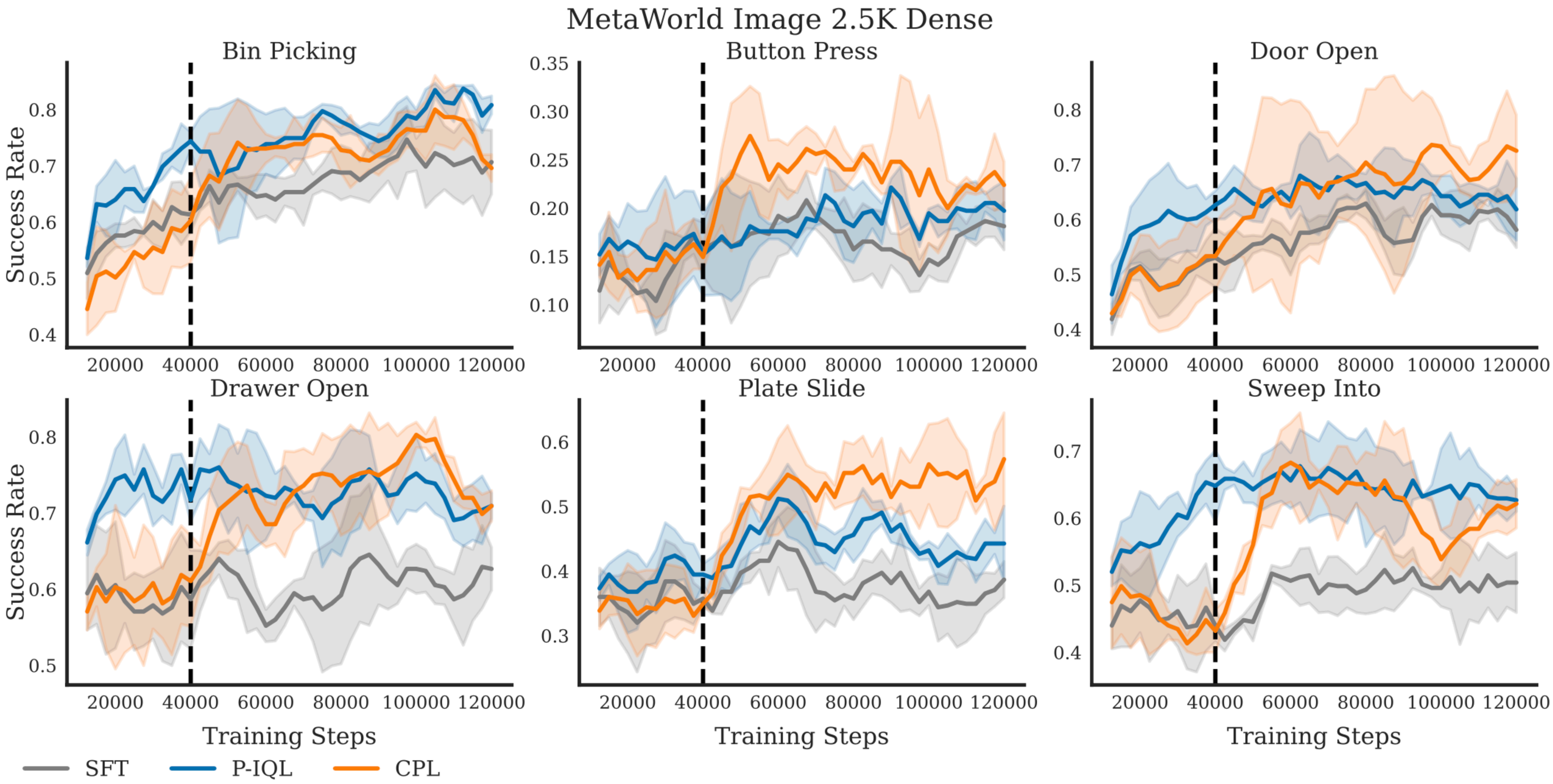
$[0.5,\ 0.5,\ 0]$ and $[0.01,\ 0.01,\ 0.98]$ both minimize the CPL Loss. Our null space is too big!

Regularized Contrastive Preference Learning

$$\min_{\pi} -\mathbb{E}_D \left[\log \frac{\exp \sum_t \log \pi(a_t^+ \mid s_t^+)}{\exp \sum_t \log \pi(a_t^+ \mid s_t^+) + \exp \lambda \sum_t \log \pi(a_t^- \mid s_t^-)} \right]$$

Prop 2. $0 < \lambda < 1$ makes the regularized CPL loss lower when a higher weight is put on in-distribution actions.

Does CPL work as well as traditional RLHF?



CPL vs. DPO

DPO is a special case of CPL, where we learn a contextual bandit policy in the KL-constrained setting