Reinforcement learning with human feedback

Generative AI Foundations on AWS

Emily Webber, Principal ML Specialist SA at AWS

Lesson 6 – Level 400

Today's activities

- Not all human feedback is the same
- RLHF in a nutshell
- Many kinds of reward modelling
- Implementation options on AWS
- Hands-on walk-through: reinforcement learning with human feedback on AWS

Reminder – everything we discuss today
is possible on AWS and SageMaker!

Not all human feedback is the same

Objective human feedback

1+1 = 2

Literal translations and classifications

External outcomes

Empirical observations

Subjective human feedback

Nuanced preferences

Gut reactions

Responses to content

Interpreting artwork

Human feedback varies by use case and personality

Objective human feedback

Great for traditional ML tasks

Subjective human feedback

Great for generative ML tasks

From classification to reward modelling

Text: I am not into this house; it's way too expensive and too far from the train line!

$$\begin{tikzcd} \text{Text} \arrow[d] & \\ \boxed{\text{Model}} \arrow[r] & \text{Sentiment: negative} \end{tikzcd}$$

Traditional ML tasks use *objective* human feedback

Text: I am not into this house; it's way too expensive and too far from the train line!

Is this person likely to want to buy this house?

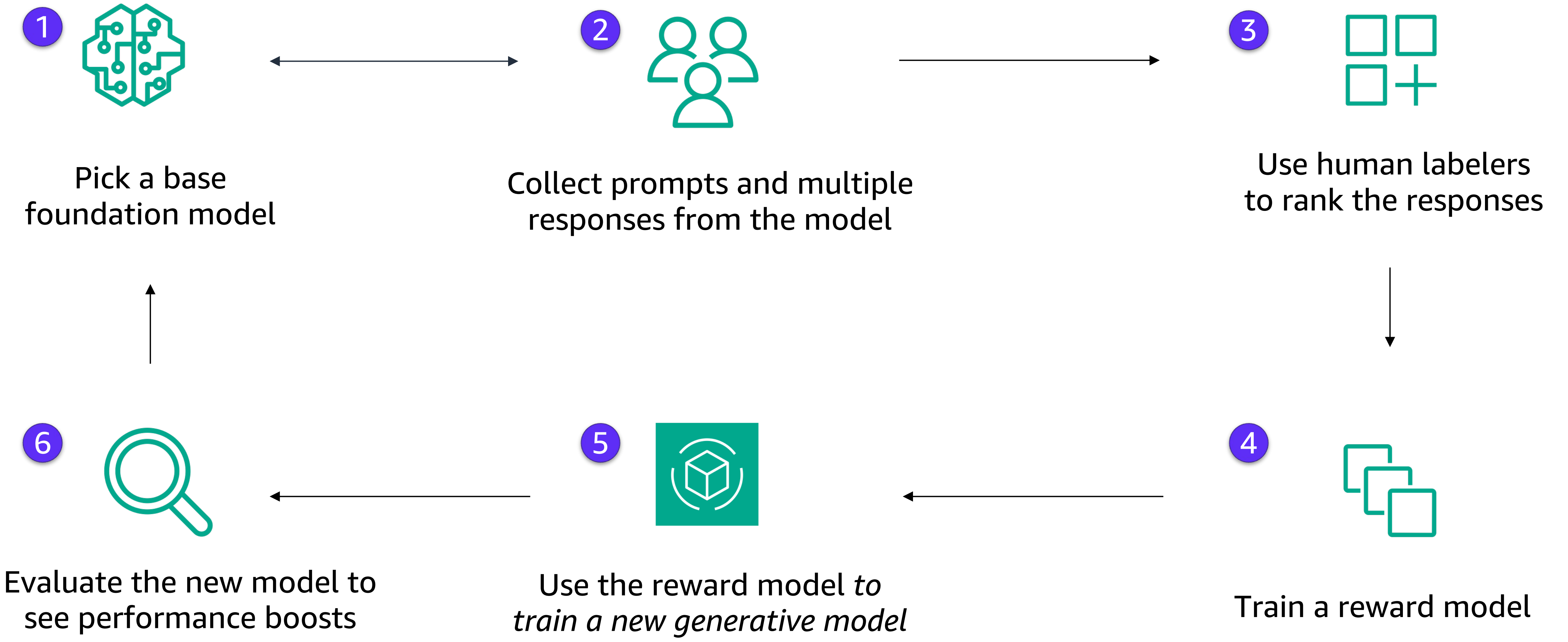
Agent: Very unlikely. Clearly the speaker indicates a strong preference for a less expensive house with a closer proximity to the train line, therefore we can conclude this person will probably not want to buy the house.

Generative ML tasks need *subjective* human responses aggregated at scale

↓

We call this *reward modelling*

Reward modelling aggregates human feedback at scale


1 Pick a base foundation model
2 Collect prompts and multiple responses from the model
3 Use human labelers to rank the responses
4 Train a reward model
5 Use the reward model *to train a new generative model*
6 Evaluate the new model to see performance boosts

Reinforcement learning with human feedback

- Start with a dataset of prompts and responses, with multiple responses for each prompt
- Send these to humans for ranking
- Train a new *reward model* on the human rankings, using reinforcement learning
- Use the reward model to train a new generative model
- The final model should be 2-3x better than the original

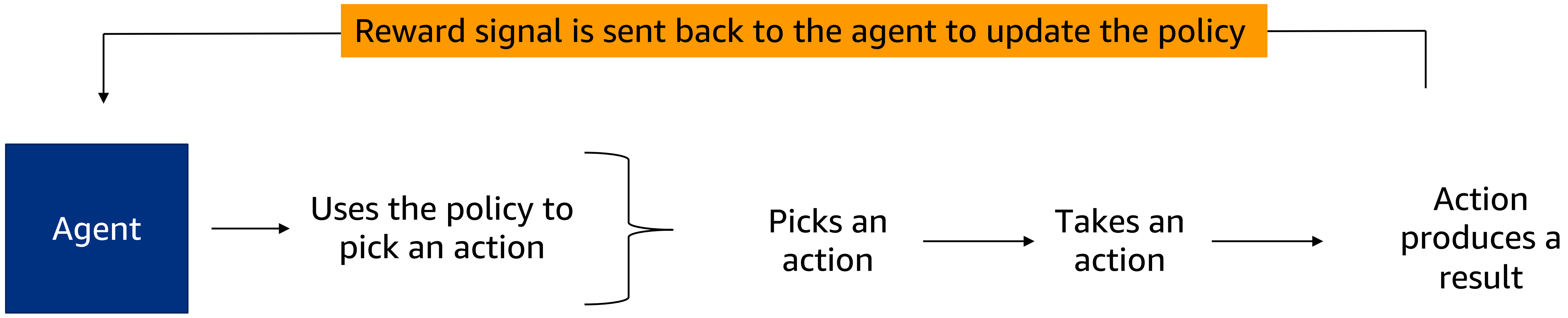
Pro tip:

Reinforcement learning with human feedback is one of the most common ways to perform *reward modelling*

Quick recap of reinforcement learning

Vocabulary

- *Reinforcement learning*: a type of machine learning commonly used to train robotic agents
- *Agent:* an autonomous entity we want to train
- *Policy*: how the agent learns, commonly a neural network
- *Action space*: all possible actions the agent can take
- *Reward function*: a signal provided to the agent to drive its learning


Reward signal is sent back to the agent to update the policy
Agent
Uses the policy to pick an action
Picks an action
Takes an action
Action produces a result

Applying reinforcement learning to update LLMs

- **Policy**: the LLM you want to fine-tune, orchestrated by proxy policy optimization (PPO)
- **Action space**: all possible tokens in the vocabulary
- **Reward model**: a model you train on the human-ranked responses from the LLM
- **Divergence**: a distance function you use to keep the original LLM and the one you are training closer
- **Reward function**: uses a pretrained reward model, combined with the divergence term, to update the agent and its neural network

RLHF mathematically speaking

- x = prompts from the training dataset
- y^* = text generated by the LLM (the PPO) you are training, using the prompts
- y^0 = text generated from the original LLM you used first, also using the prompts

Tells you what humans prefer

$$r_\theta = reward\_model(x + y^*)$$

Prevents out-of-character RL hacks

$$r_{KD} = \text{KLDivergence}(y^*, y^0)$$

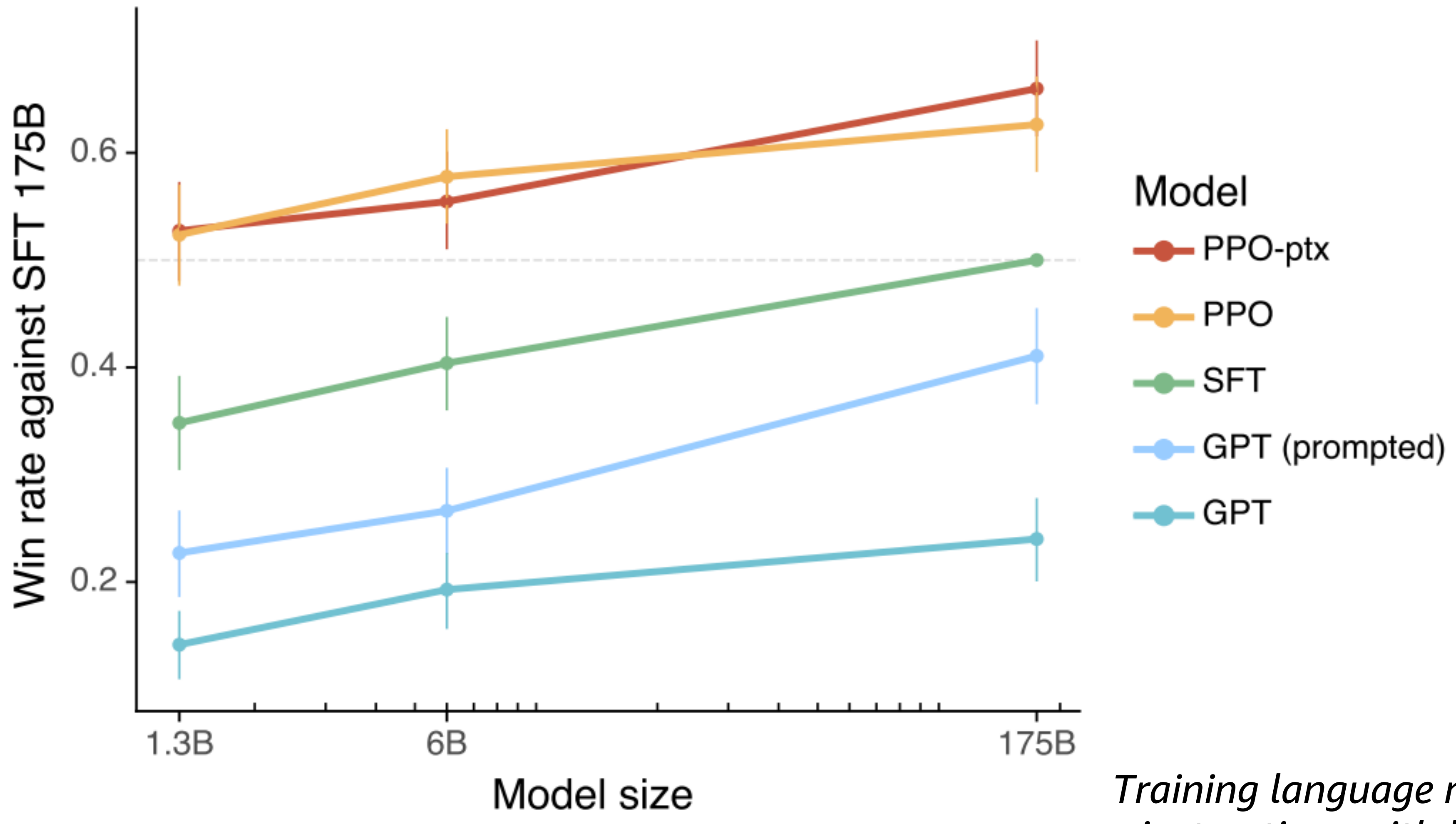
Serves as the signal to update your neural network

$$r_{PPO} = r_\theta - \epsilon * r_{KD} + ?$$

May be useful to add pretraining gradients here

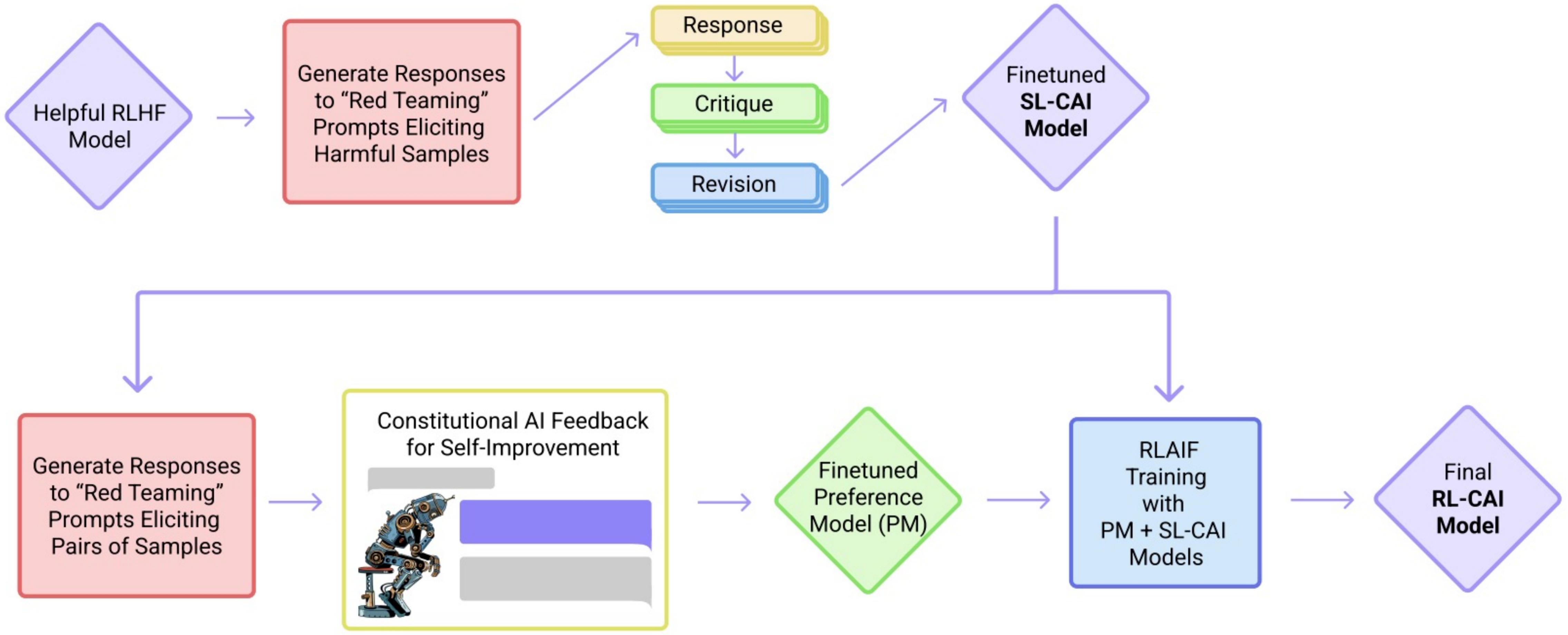
A tunable weighting term

RLHF shows 2-3x boost over base GPT-3



Training language models to follow instructions with human feedback
Ouyang et al, 2022

What if we use AI to evaluate the AI?


Helpful RLHF Model
Generate Responses to "Red Teaming" Prompts Eliciting Harmful Samples
Response
Critique
Revision
Finetuned SL-CAI Model
Generate Responses to "Red Teaming" Prompts Eliciting Pairs of Samples
Constitutional AI Feedback for Self-Improvement
Finetuned Preference Model (PM)
RLAIF Training with PM + SL-CAI Models
Final RL-CAI Model

Constitutional AI: Harmlessness from AI feedback
Yuntao Bai et al, 2022

What you need to train a reward model

1:many dataset with prompts and responses

A GPT-based model that returns a number

Distributed training systems

↑

A regressive large language model

But not that large, ~6B is good enough

Datasets for reward modelling

Prompt → "What's the weather like in Washington, DC?"

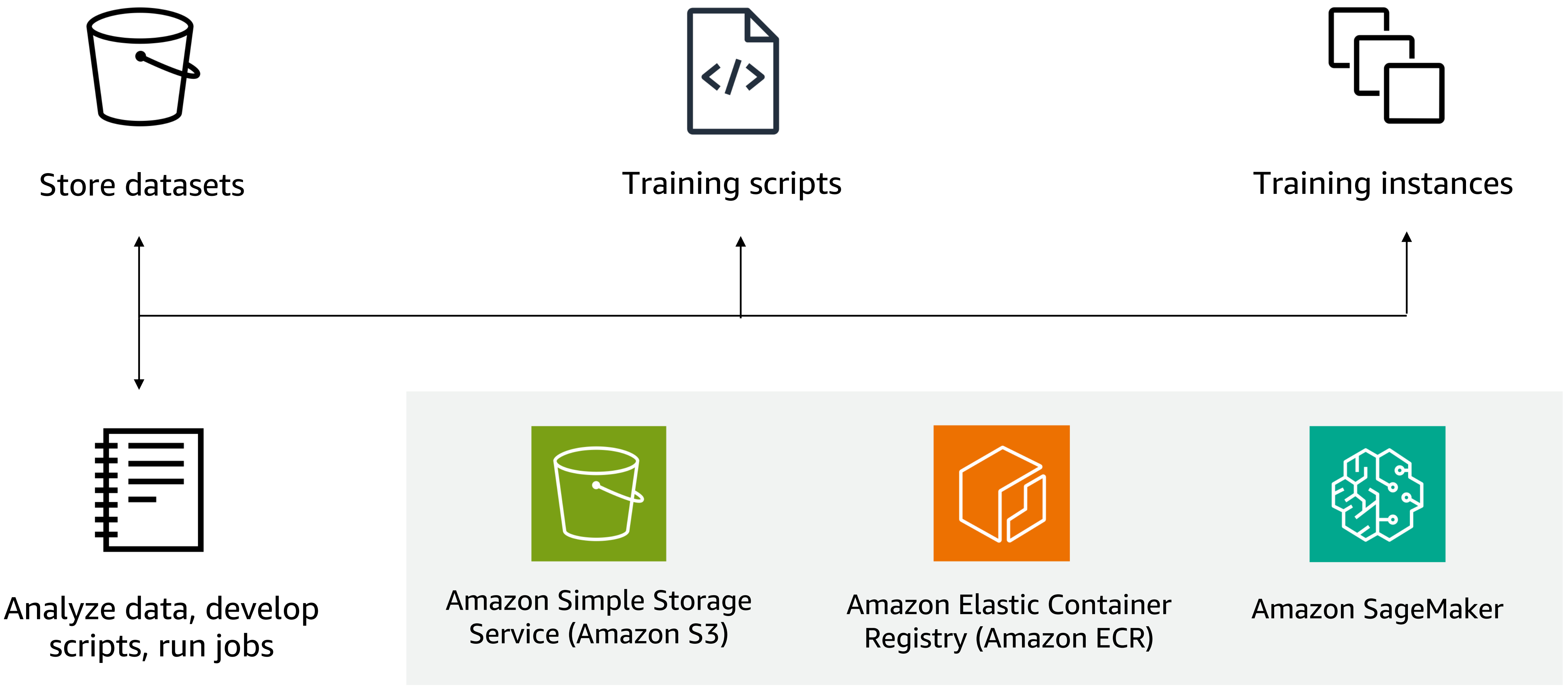
Responses →	The local weather in Washington DC is currently sunny and humid, at a temperature of 82 degrees Fahrenheit.	It's freaking hot!!	Relative to Phoenix, Arizona, Washington DC is a cool 82 degrees.
Preference rankings →	2	1	3

You want some preference number to rank all of the possible responses to each prompt.

You can use humans, AI's, or any kind of digital signal to create these rankings.

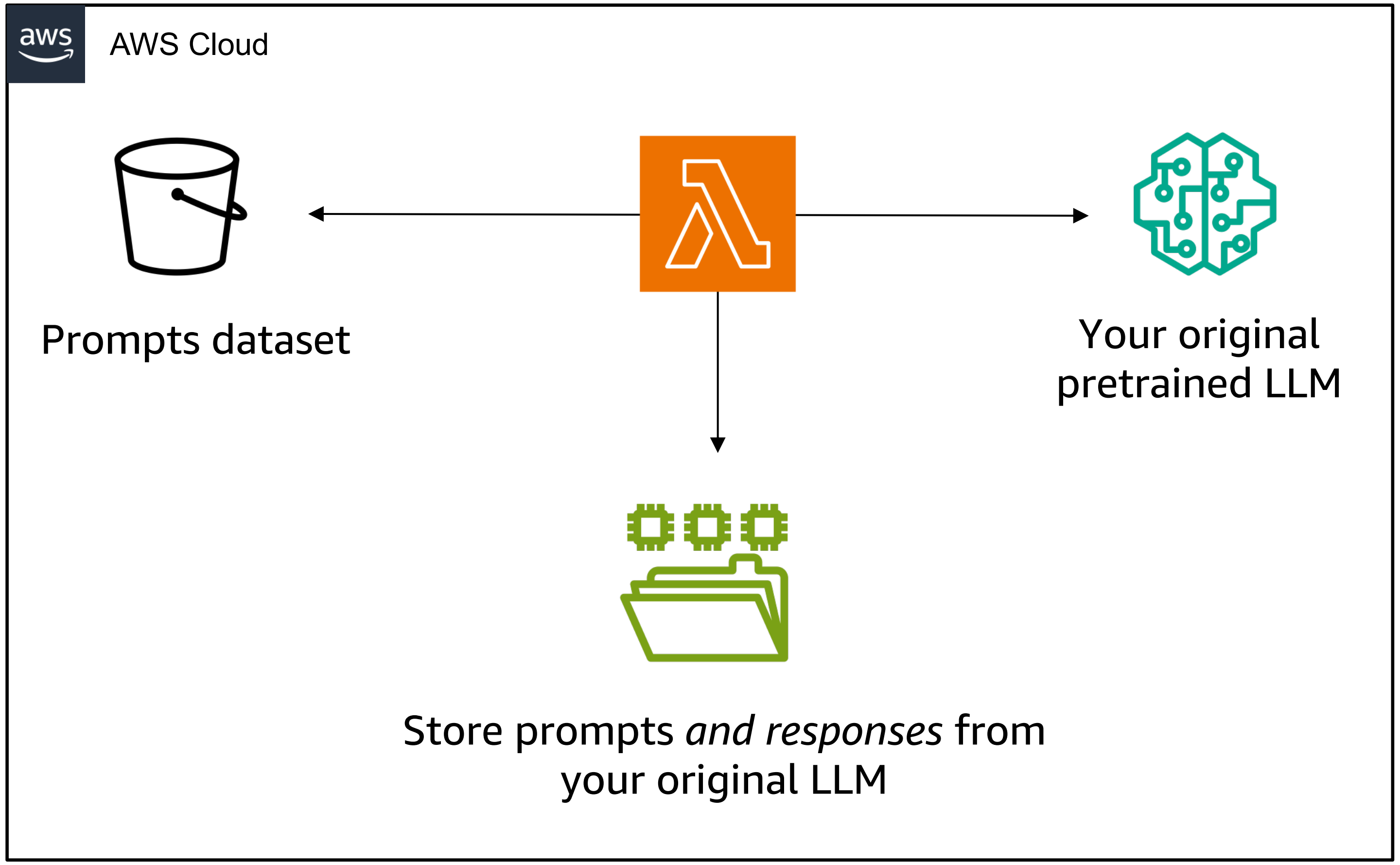
The rankings *become the label to train a supervised reward model.*

How to build and train a reward model on AWS


Store datasets
Training scripts
Training instances
Analyze data, develop scripts, run jobs
Amazon Simple Storage Service (Amazon S3)
Amazon Elastic Container Registry (Amazon ECR)
Amazon SageMaker

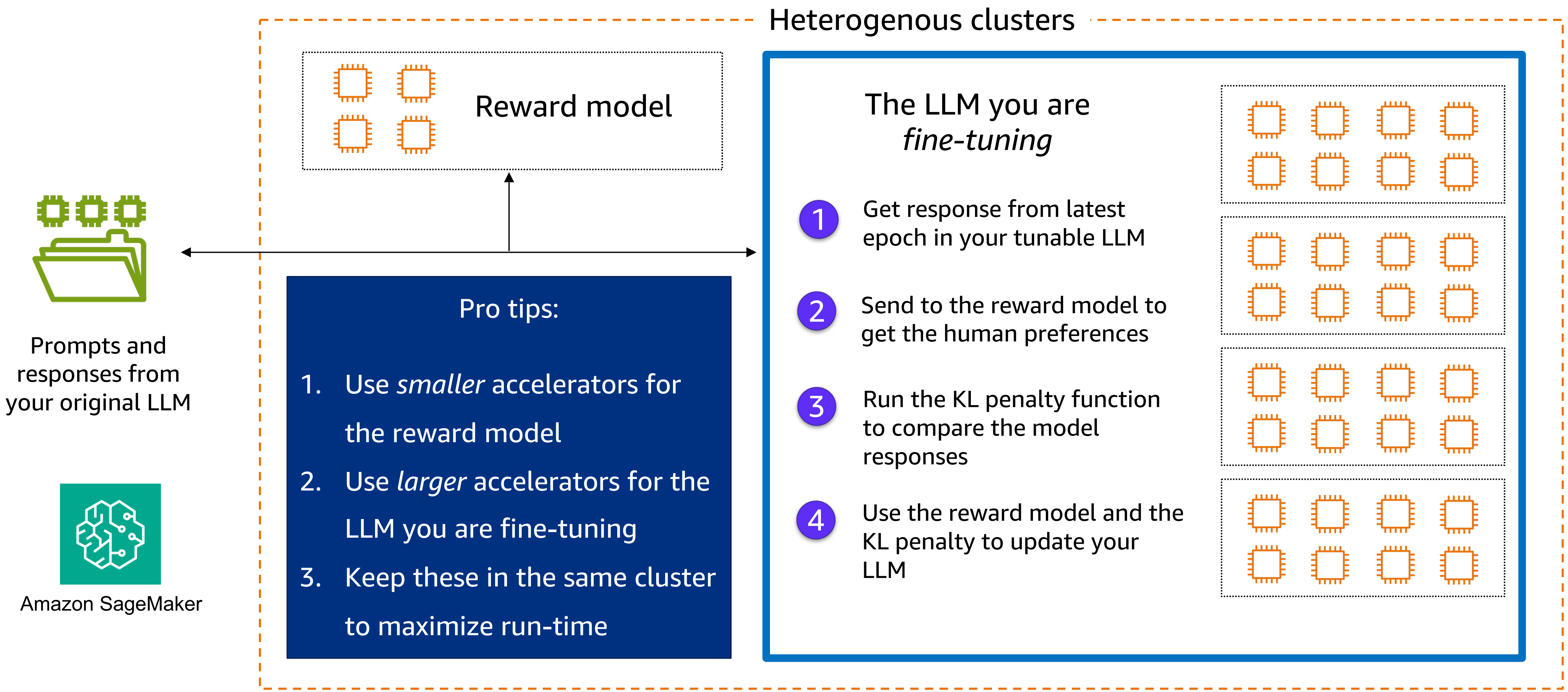
Use your reward model to train a new LLM

Ahead of time, precompute the original model responses


AWS Cloud
Prompts dataset
Your original pretrained LLM
Store prompts *and responses* from your original LLM

- Run a CPU-based and/or serverless job *ahead of time*
- Store both the prompts and the responses from your original LLM
- Prepare the training dataset on a high-performance distributed file system to optimize the training runs
- May already be in your ranking dataset!

Use your reward model to train a new LLM



Prompts and responses from your original LLM

Pro tips:

1. Use *smaller* accelerators for the reward model
2. Use *larger* accelerators for the LLM you are fine-tuning
3. Keep these in the same cluster to maximize run-time

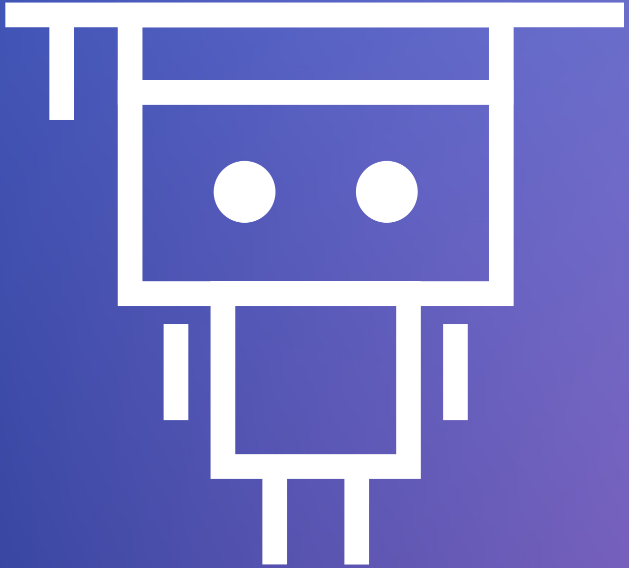
The LLM you are *fine-tuning*

1. Get response from latest epoch in your tunable LLM
2. Send to the reward model to get the human preferences
3. Run the KL penalty function to compare the model responses
4. Use the reward model and the KL penalty to update your LLM

https://bit.ly/sm-nb-6

Hands-on demo

sagemaker-distributed-training-workshop / 9_rlhf / RLHF_locally.ipynb



Thank you!

Type: Corrections, feedback, or other questions?
Contact us at https://support.awsamazon.com/#/contacts/aws-academy.
All trademarks are the property of their owners.