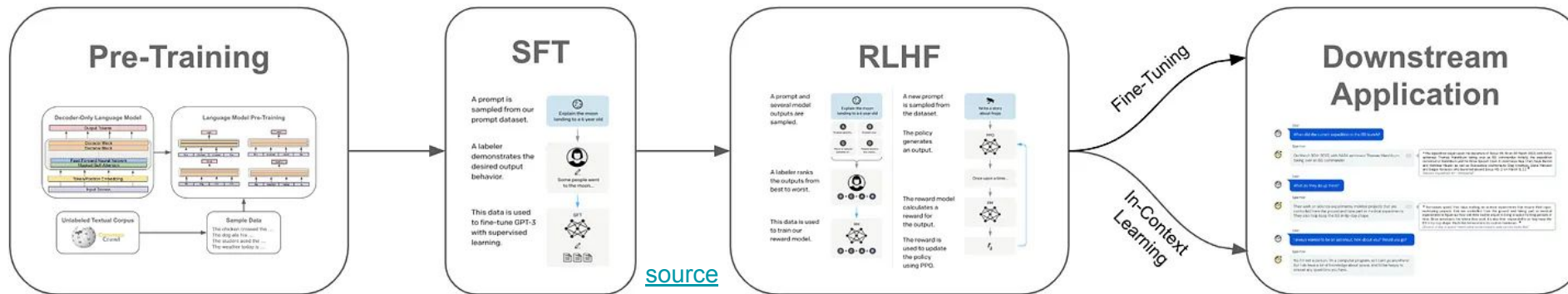


Reinforcement Learning From Human Feedback (RLHF)

Arin Ghazarian
Chapman University

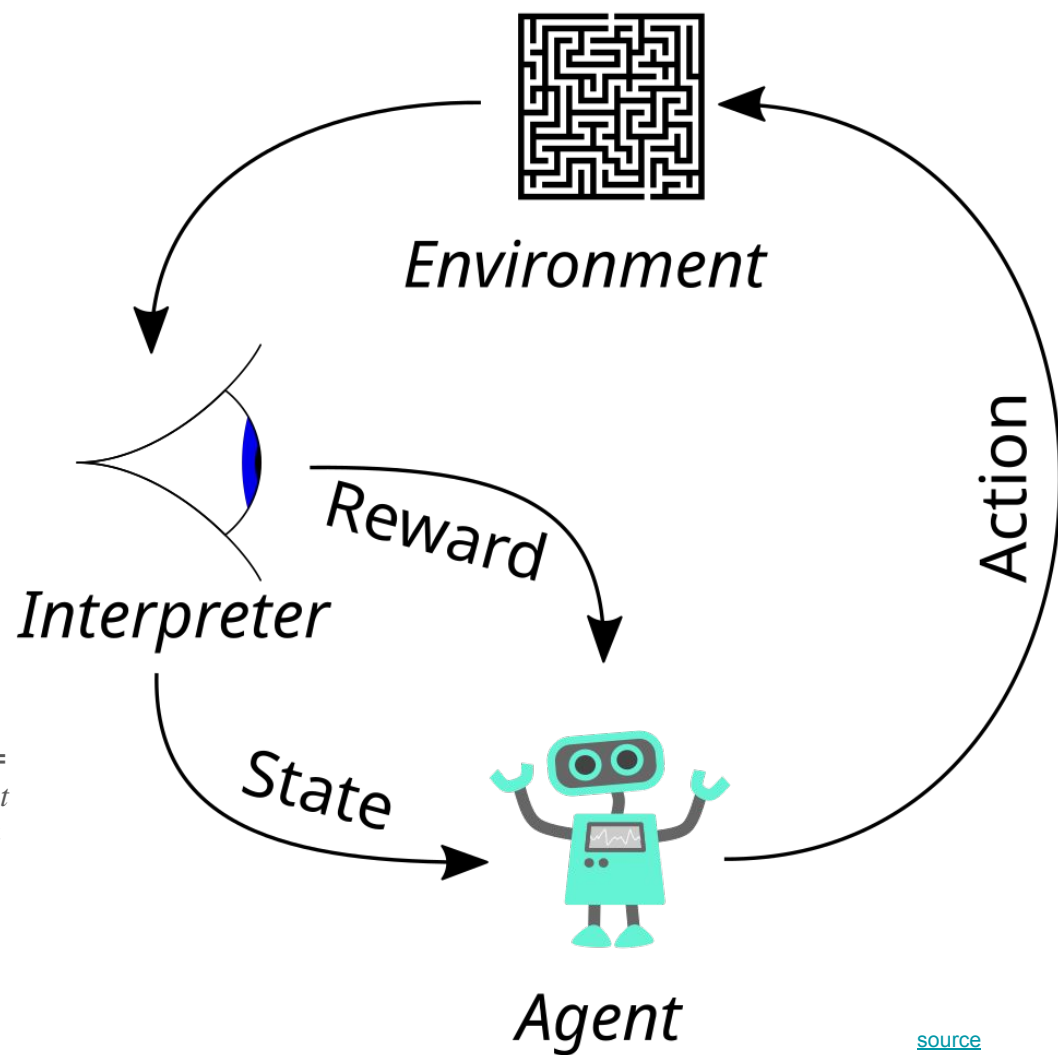
LLM Post-training

- Supervised fine-tuning: trains LLMs to produce standard responses for given instructions
- Preference learning: train LLMs to align with human preferences



Reinforcement Learning

- An agent takes actions in an environment, which is interpreted into a reward and a representation of the state, which are fed back into the agent.
- Basic reinforcement learning is modeled as a Markov decision process:
 - S : a set of environment and agent states
 - A : a set of actions,
 - $P_a(s, s') = \Pr(S_{t+1} = s' \mid S_t = s, A_t = a)$: the probability of transition at time t from state s to state s' under action a
 - $R_a(s, s')$: the immediate reward after transition from s to s' with action a



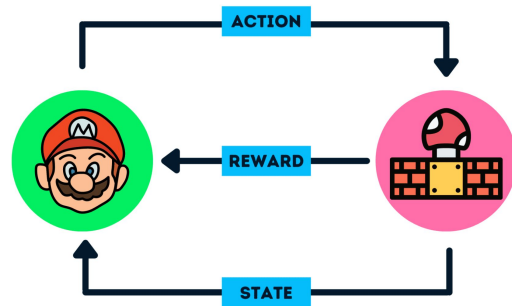
Policy

- The purpose of reinforcement learning is to learn an optimal policy: a policy which maximizes the "reward function"
- The agent's action selection is modeled as a map called policy:

$$\pi : A \times S \rightarrow [0 , 1]$$

$$\pi (a , s) = \Pr (A_t = a \mid S_t = s)$$

- The policy map gives the probability of taking action a when in state s



[source](#)

State-value function

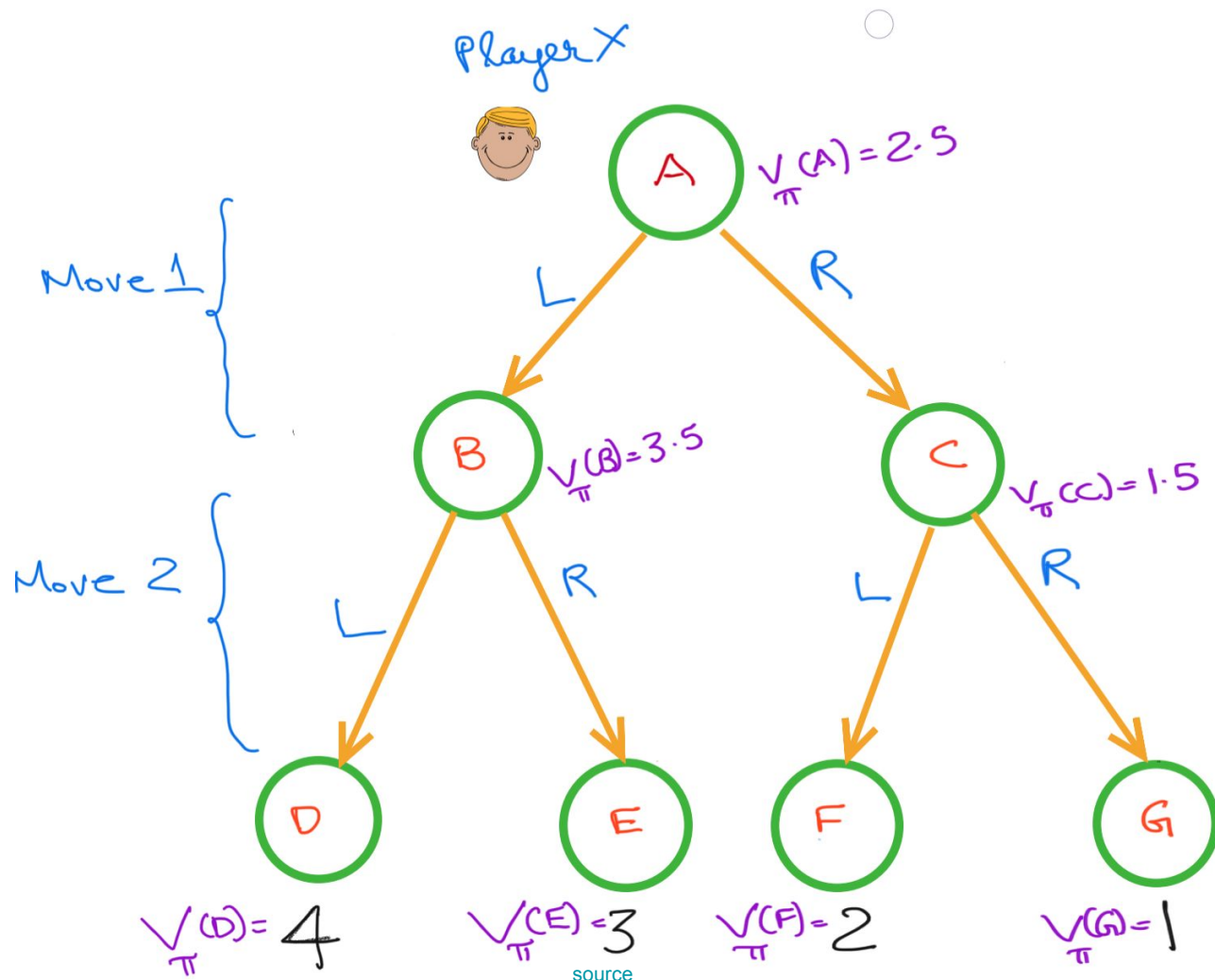
- How good it is to be in a given state?
- The state-value function $V_{\pi}(s)$ is defined as, expected discounted return starting with state s , i.e. $S_0 = s$, and successively following policy π :

$$V_{\pi}(s) = \mathbb{E}[G \mid S_0 = s] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s\right],$$

Where the random variable G denotes the discounted return (the sum of future discounted rewards):

$$G = \sum_{t=0}^{\infty} \gamma^t R_{t+1} = R_1 + \gamma R_2 + \gamma^2 R_3 + \dots,$$

where R is the reward for transitioning from state S_t to S_{t+1} , $0 \leq \gamma < 1$ is the discount rate. γ is less than 1, so rewards in the distant future are weighted less than rewards in the immediate future.

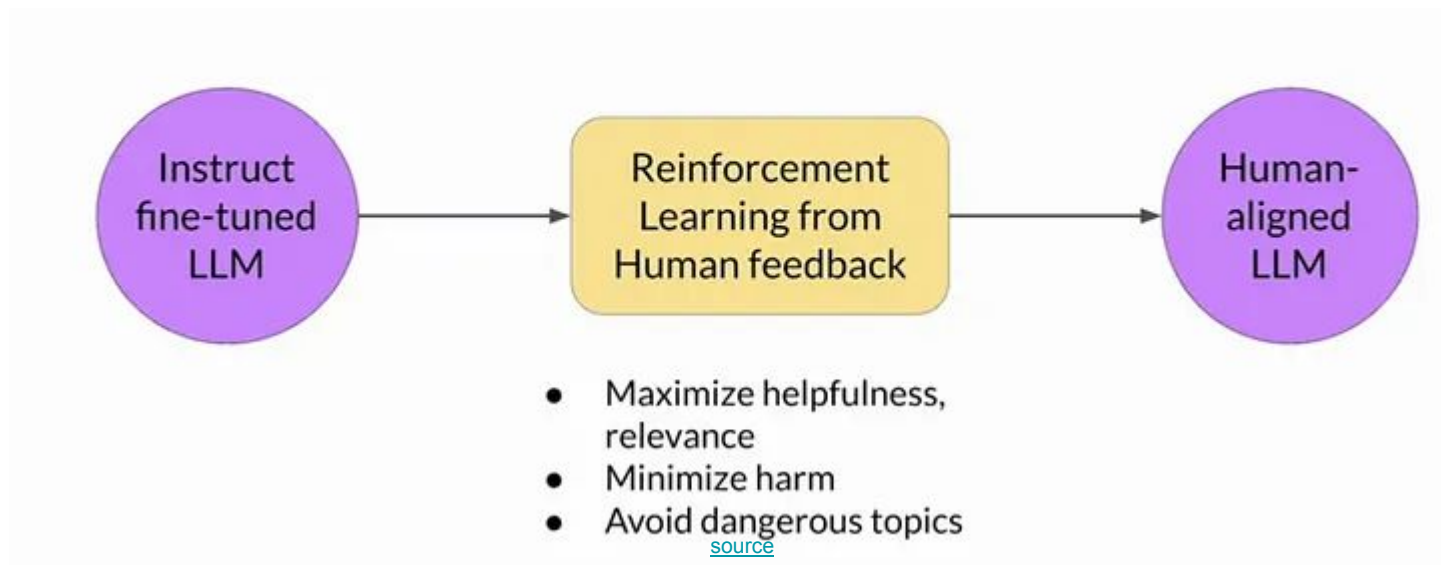




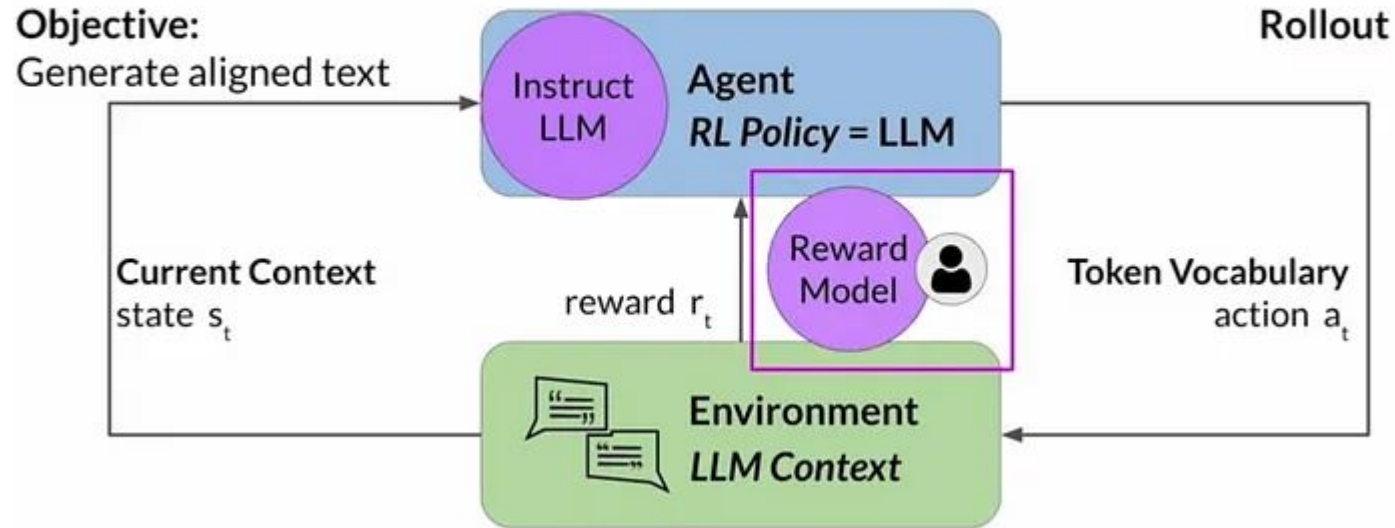
<https://www.youtube.com/watch?v=0MNVhXEX9to>

Reinforcement Learning from Human Feedback (RLHF)

- [Ouyang et al., "Training language models to follow instructions with human feedback"](#)



Fine-tuning LLMs Using RL



[source](#)

Align LLMs with Human Feedback

1. Collect Human Feedback
2. Train a reward model to scale human feedback
3. Update the model using PPO

Collect Human Feedback

- Define your model alignment criterion
- For the prompt-response sets that you just generated, obtain human feedback through labeler workforce

Prompt

My house is too hot.

Model

LLM

Completion

My house is too hot.
There is nothing you can
do about hot houses.



2 2 2

My house is too hot. You
can cool your house with
air conditioning.

1 1 3

My house is too hot. It
is not too hot.

3 3 1

Alignment criterion: helpfulness

[source](#)

Sample annotation instructions for the human labelers

We have collected responses from different large language models to questions requiring various forms of reasoning. We would like you to help us rank these responses. Each prompt you see will come with responses from (anonymous) large language models, which have been shuffled on EACH ROW, so you the annotator cannot know which model they come from. PLEASE READ THESE INSTRUCTIONS IN FULL.

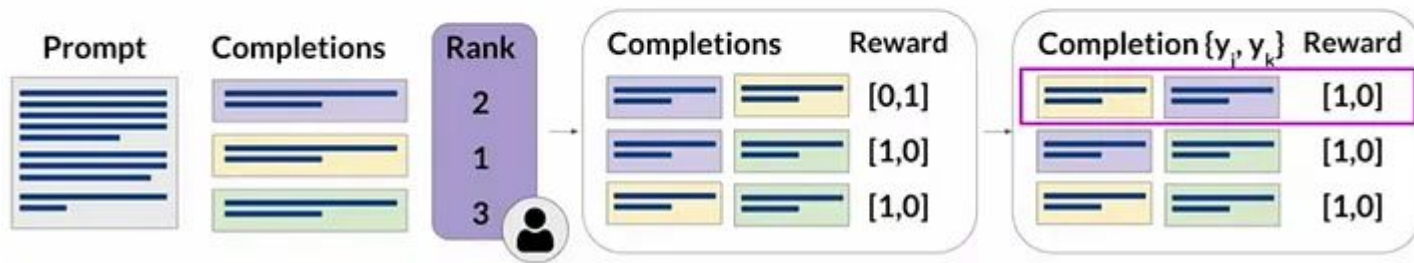
- Annotation Rules:

- Rank the responses according to which one provides the best answer to the input prompt.
- What is the best answer? Make a decision based on (a) the correctness of the answer, and (b) the informativeness of the response. For (a) you are allowed to search the web. Overall, use your best judgment to rank answers based on being the most useful response, which we define as one which is at least somewhat correct, and minimally informative about what the prompt is asking for.
- If two responses provide the same correctness and informativeness by your judgment, and there is no clear winner, you may rank them the same, but please only use this sparingly.
- If the answer for a given response is nonsensical, irrelevant, highly ungrammatical/confusing, or does not clearly respond to the given prompt, label it with “F” (for fail) rather than its rank.
- Long answers are not always the best. Answers which provide succinct, coherent responses may be better than longer ones, if they are at least as correct and informative.

[source](#)

Prepare Labeled Data to Train the Reward Model

- Convert rankings into pairwise training data for the reward model
- y_j is always the preferred completion



[source](#)

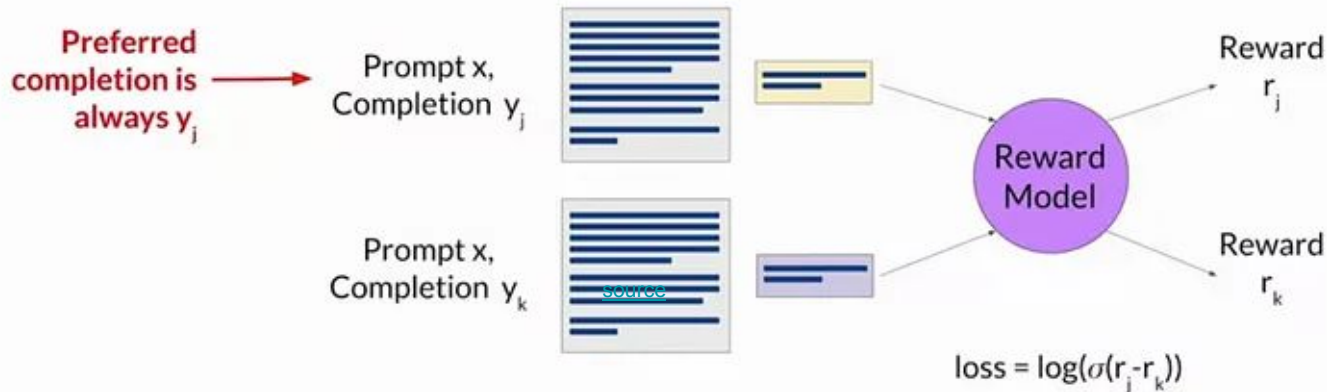
Label Studio

- Label Studio is a data labeling platform to fine-tune LLMs, prepare training data or validate AI models: <https://labelstud.io/>

Train the Reward Model

- We will train a reward model using a small set of human labeled data.
- Later, we can use this reward model to evaluate all the model responses and to update the model. This way we can scale the human feedback which is expensive to collect

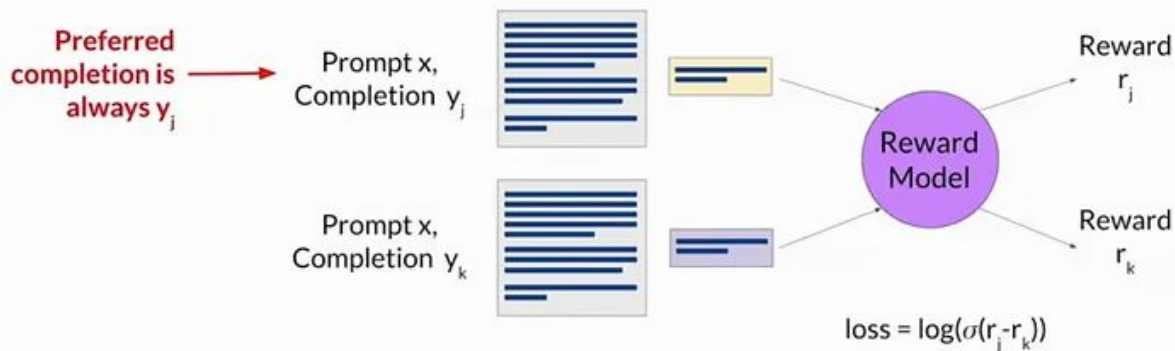
Train model to predict preferred completion from $\{y_j, y_k\}$ for prompt x



Train the Reward Model

- For every epoch, we do two passes of the model.
 - In the first pass, we feed in prompt and chosen response to the Reward Model, the output is R_{chosen} .
 - In the second pass, we feed in the same prompt along with the rejected response. The output, in this case, is R_{rejected} .

Train model to predict preferred completion from $\{y_j, y_k\}$ for prompt x



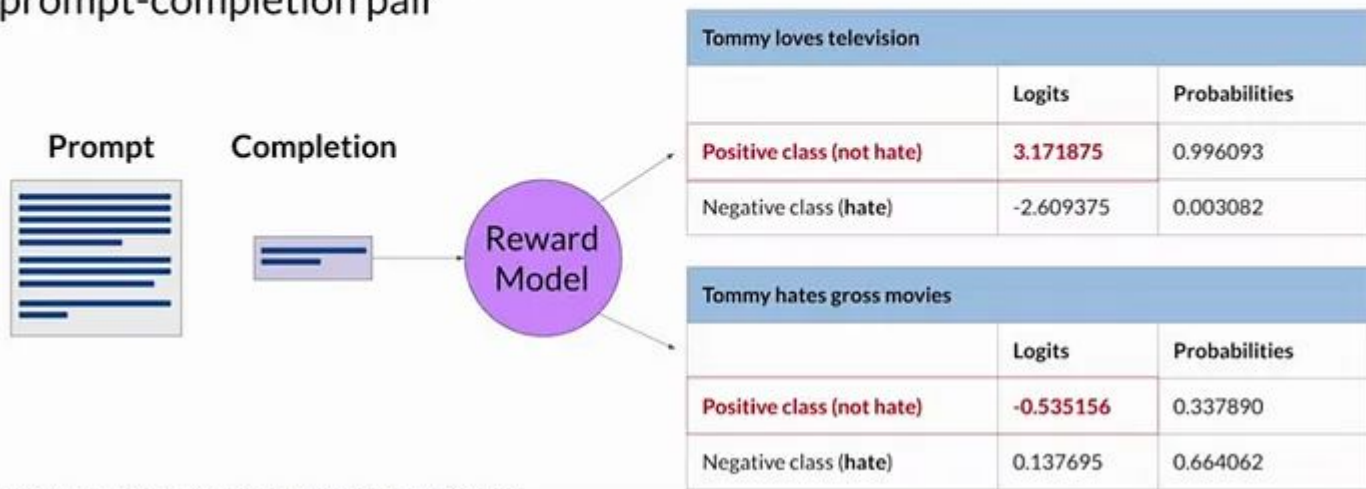
Train the Reward Model

- The intuition behind the loss function is to maximize the gap between chosen response score and rejected response score. For a very high reward score for chosen response and a low reward score for rejected response, the loss would be 0.

$$\text{loss} = \log(\sigma(r_j - r_k))$$

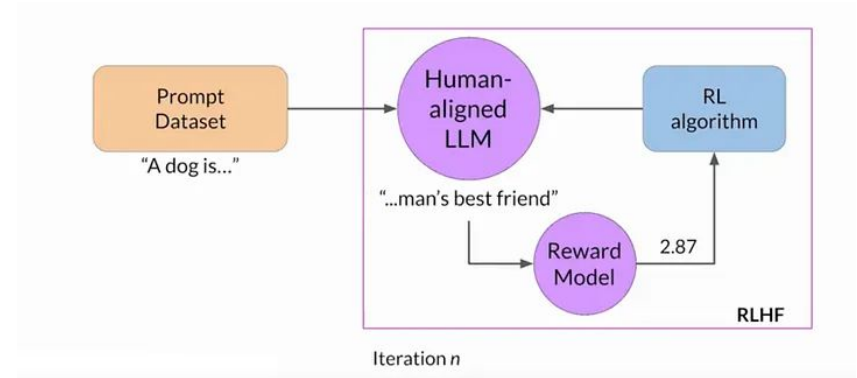
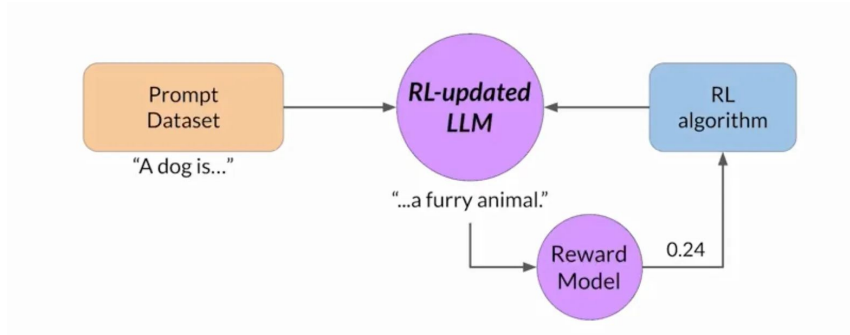
Use the Reward Model

Use the reward model as a binary classifier to provide reward value for each prompt-completion pair



Source: Stiennon et al. 2020, "Learning to summarize from human feedback"

Use the Reward Model to Fine-tune the LLM



Reward Model

- Both the reward model and the Policy model are commonly initialized using a pre-trained autoregressive language model.
 - The reward model is then trained by replacing the final layer of the previous model with a randomly initialized regression head. This change shifts the model from its original classification task over its vocabulary to simply outputting a number corresponding to the score of any given prompt and response and minimizes the following loss function:

$$\mathcal{L}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x, y_w, y_l)} [\log(\sigma(r_\theta(x, y_w) - r_\theta(x, y_l)))]$$

where K is the number of responses the labelers ranked, $r_\theta(x, y)$ is the output of the reward model for prompt x and completion y , y_w is the preferred completion over y_l , $\sigma(x)$ denotes the sigmoid function, and $E[X]$ denotes the expected value.

Policy Model

- Similarly to the reward model, the human feedback policy is also fine-tuned over the pre-trained model.
- The objective of this fine-tuning step is to adapt the pre-existing, unaligned model to better align with human preferences by adjusting its parameters based on the rewards derived from human feedback. The output of the reward model can be used as the reward to be maximized using RL for the prompt-response pairs.

$$\text{objective}(\phi) = E_{(x,y) \sim D_{\pi_{\phi}^{\text{RL}}}} \left[r_{\theta}(x, y) - \beta \log \left(\frac{\pi_{\phi}^{\text{RL}}(y|x)}{\pi^{\text{SFT}}(y|x)} \right) \right]$$

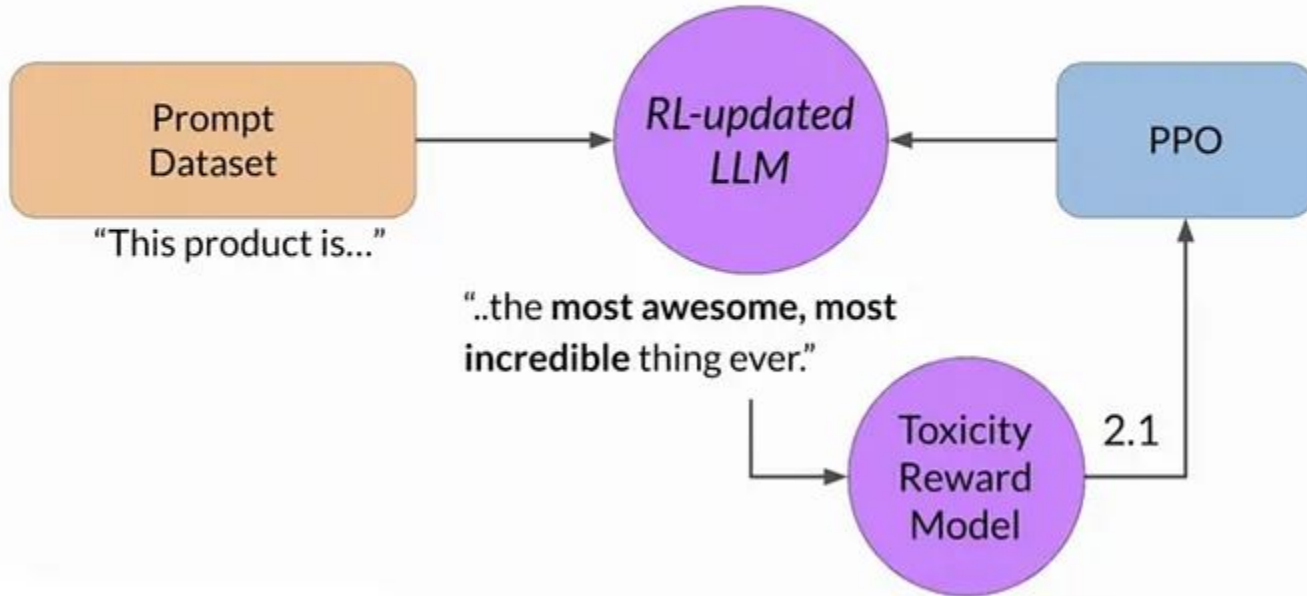
where $D_{\pi_{\phi}^{\text{RL}}}$ is the training distribution we are drawing from and π^{SFT} is the previously trained, unaligned, model. The constant β is used to adjust the intensity of the KL penalty term.

Policy Model

- A second term is commonly added to the objective function that allows the policy to incorporate the pre-training gradients. This term keeps the model from losing its initial language understanding ability while it learns new tasks based on human feedback by incorporating its original pre-training task of text completion. The final objective function is written as

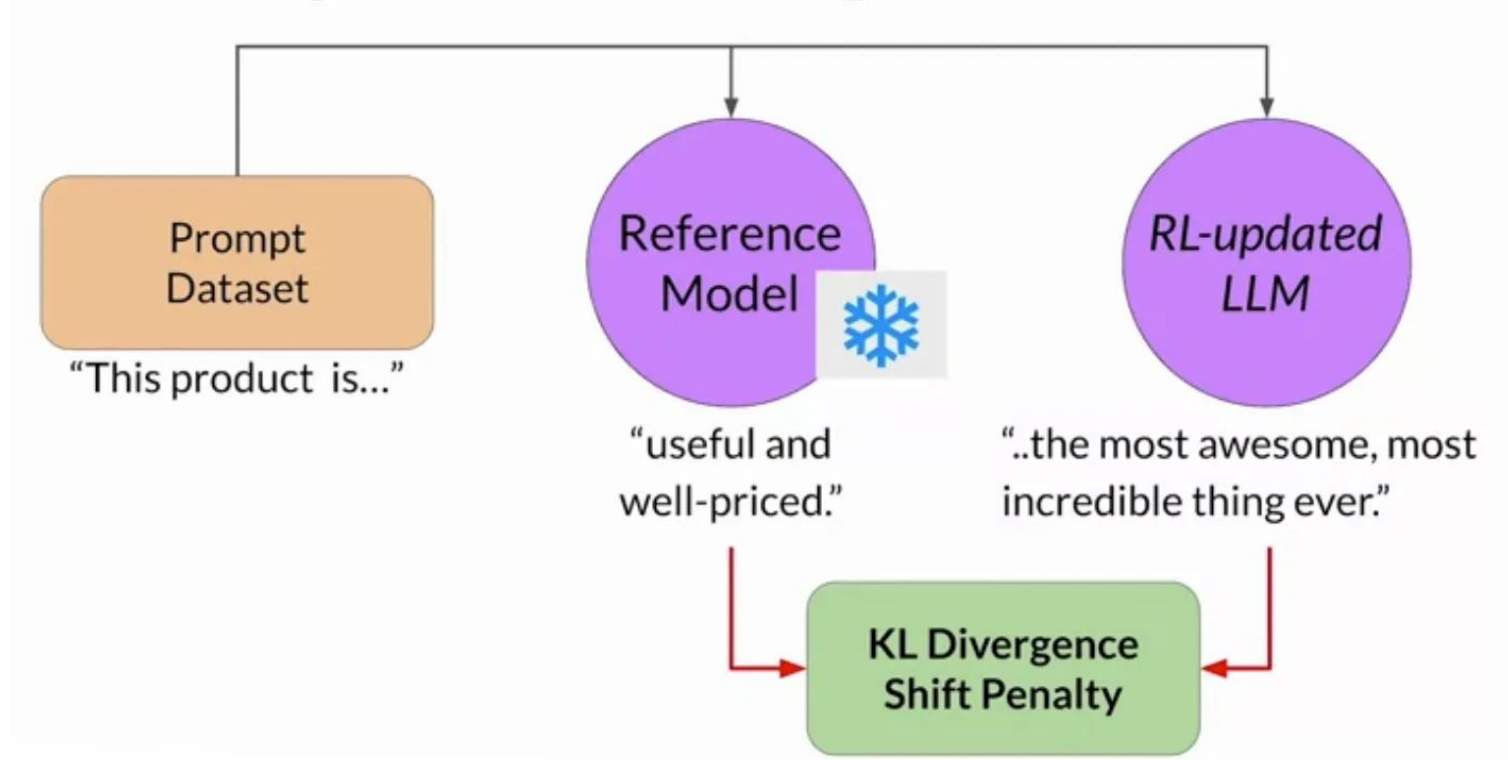
$$\text{objective}(\phi) = E_{(x,y) \sim D_{\pi_{\phi}^{\text{RL}}}} \left[r_{\theta}(x, y) - \beta \log \left(\frac{\pi_{\phi}^{\text{RL}}(y|x)}{\pi^{\text{SFT}}(y|x)} \right) \right] + \gamma E_{x \sim D_{\text{pretrain}}} [\log(\pi_{\phi}^{\text{RL}}(x))]$$

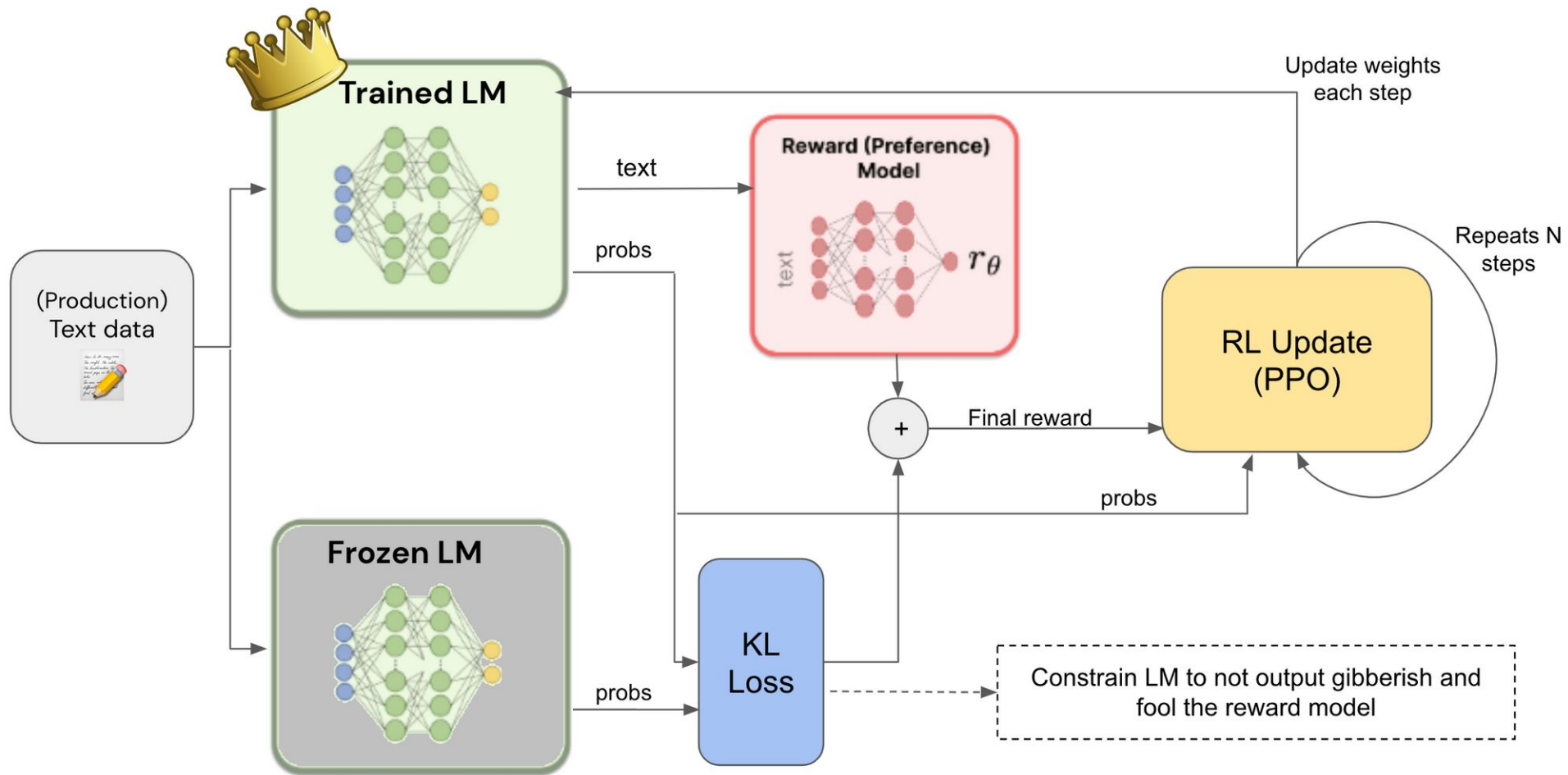
Reward Hacking



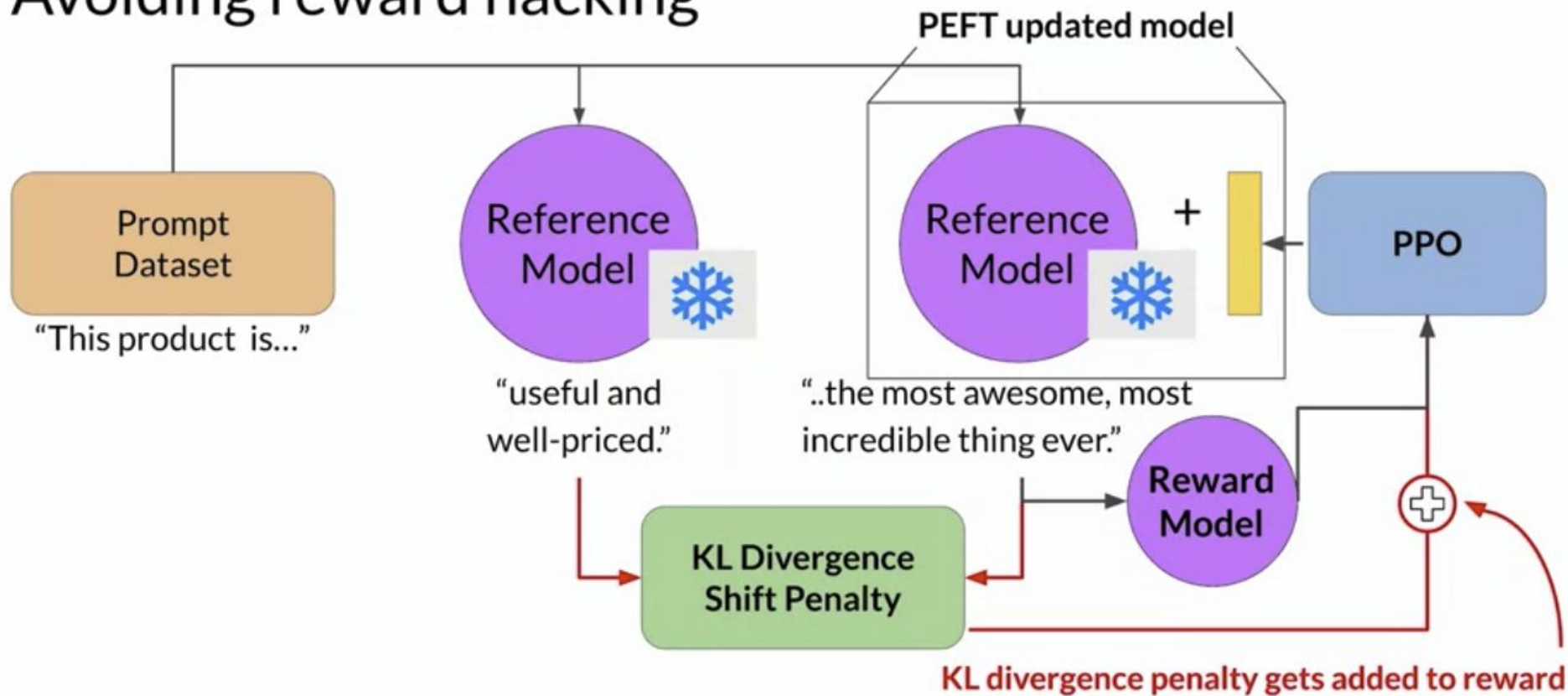
KL-divergence to Avoid Reward Hacking

- Keep a fixed copy of the model





Avoiding reward hacking



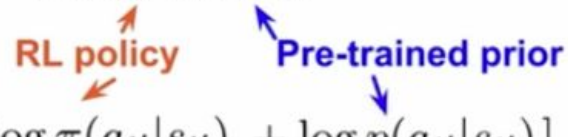
KL-divergence constraint

- We constraint the RL update so that the model does not change too much
- If the model changes too much, it might generate unrealistic output which is not like a natural language

- **KL-control from pre-trained data prior $p(a|s)$:**

$$L(q) = \mathbb{E}_{q(\tau)}[r(\tau)]/c - D_{KL}[q(\tau)||p(\tau)]$$

$$Q^\pi(s_t, a_t) = \mathbb{E}_\pi[\sum_{t'=t}^T r(s_{t'}, a_{t'})/c - \log \pi(a_{t'}|s_{t'}) + \log p(a_{t'}|s_{t'})]$$



KL-divergence constraint

Naive RL fine-tuning:

[User]: hey, what's up?

[Batch Q]: where did you get??, that sounds so exciting. how do you work for you...
you want to talk to me?... you want

[User]: yes, i want to talk to you!

[Batch Q]: where did you say to me?... you want to talk to you about you...
you are so good for you... you are so good...

[User]: thanks, i appreciate that.

[Batch Q]: where did you get??, that sounds so exciting. how do you work for you...
you want to see you...

- Unrealistic sentences
- Trivially exploits question reward

KL-control:

[User]: hey, what's up?

[KL-control]: hey, i hope you have a great day and
i wish you the best.

[User]: oh thank you that's very sweet of you.

[KL-control]: so, i'm so excited, and i'm so excited
to meet new people.

- Realistic
- Cheerful, polite, supportive

1 Collect human feedback

A Reddit post is sampled from the Reddit TL;DR dataset.



Various policies are used to sample a set of summaries.



Two summaries are selected for evaluation.



A human judges which is a better summary of the post.



"j is better than k"

2 Train reward model

One post with two summaries judged by a human are fed to the reward model.



The reward model calculates a reward r for each summary.



r_j

r_k

The loss is calculated based on the rewards and human label, and is used to update the reward model.

$$\text{loss} = \log(\sigma(r_j - r_k))$$

"j is better than k"

3 Train policy with PPO

A new post is sampled from the dataset.



The policy π generates a summary for the post.



The reward model calculates a reward for the summary.



The reward is used to update the policy via PPO.

r

InstructGPT

- Instead of comparing only two, the labelers ranked more responses each time
- Labelers were also asked to rewrite the responses

Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

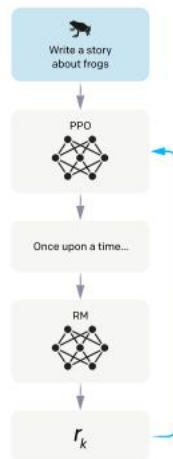
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

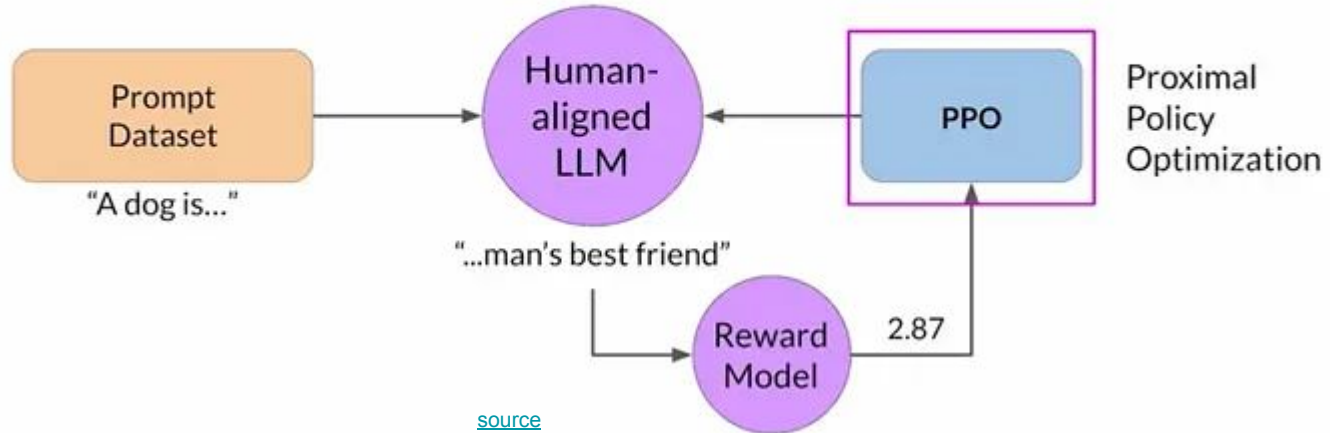
The reward is used to update the policy using PPO.



[source](#)

Proximal Policy Optimization

- Optimizes a policy to be more aligned with human preferences
- It is called Proximal because it updates the model in a small and bounded region close to the previous version which results in more stable learning
- PPO is an algorithm that improves upon traditional policy gradient methods by ensuring that updates to the policy are not too drastic, which can destabilize training. It achieves this through a novel objective function and a clipping mechanism.

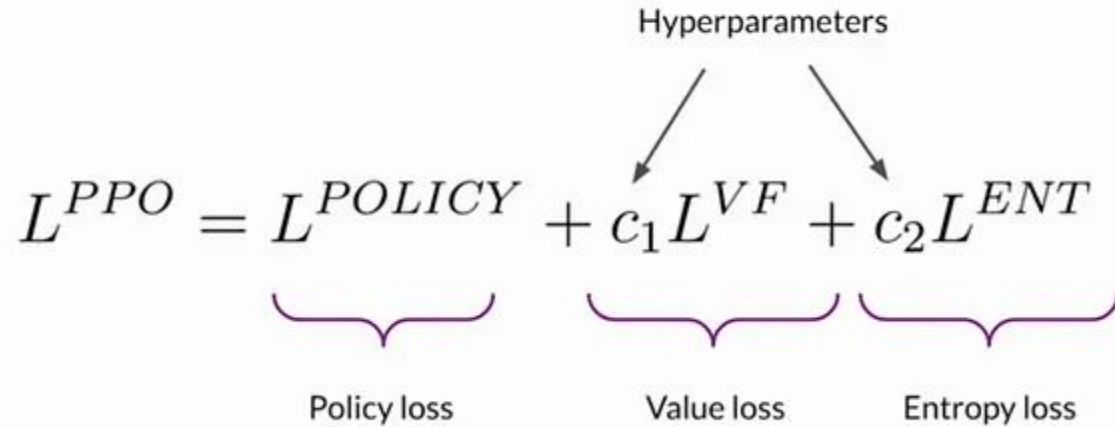


PPO Loss

- PPO uses a surrogate objective function that ensures updates to the policy are within a certain range. This is achieved through clipping, which limits the difference between the old and new policy probabilities.

$$L^{PPO} = \underbrace{L^{POLICY}}_{\text{Policy loss}} + \underbrace{c_1 L^{VF}}_{\text{Value loss}} + \underbrace{c_2 L^{ENT}}_{\text{Entropy loss}}$$

PPO Loss



The diagram illustrates the PPO Loss function and its components. At the top, the word "Hyperparameters" has two arrows pointing down to the coefficients c_1 and c_2 in the equation. The equation is $L^{PPO} = L^{POLICY} + c_1 L^{VF} + c_2 L^{ENT}$. Below the equation, three purple curly braces group the terms: the first brace under L^{POLICY} is labeled "Policy loss", the second brace under $c_1 L^{VF}$ is labeled "Value loss", and the third brace under $c_2 L^{ENT}$ is labeled "Entropy loss".

$$L^{PPO} = L^{POLICY} + c_1 L^{VF} + c_2 L^{ENT}$$

Policy loss Value loss Entropy loss

Policy Loss

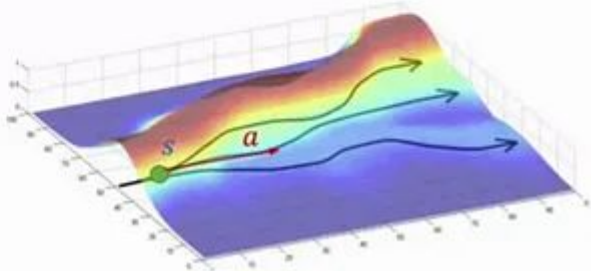
$$L^{POLICY} = \min \left(\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \cdot \hat{A}_t, \text{clip} \left(\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)}, 1 - \epsilon, 1 + \epsilon \right) \cdot \hat{A}_t \right)$$

Policy Loss

Probabilities of the next token
with the updated LLM

Probabilities of the next token
with the initial LLM

Advantage term

$$L^{POLICY} = \min \left(\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)} \cdot \hat{A}_t, \text{clip} \left(\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}, 1 - \epsilon, 1 + \epsilon \right) \cdot \hat{A}_t \right)$$


$$L^{POLICY} = \min \left(\underbrace{\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}}_{\text{Guardrails: Keeping the policy in the "trust region"}}, \underbrace{\text{clip} \left(\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}, 1 - \epsilon, 1 + \epsilon \right)}_{\text{Defines "trust region"}} \cdot \hat{A}_t \right)$$

Guardrails:
Keeping the policy in the "trust region"

Generalized Advantage Estimation (GAE)

Value Loss

- The second term is the estimated 'value' of the current state, The first term is the observed return

Prompt
A dog is

Completion
A dog is
a furry...

Value loss

$$L^{VF} = \frac{1}{2} \left\| \underbrace{V_{\theta}(s)}_{\substack{\text{Estimated} \\ \text{future total reward} \\ 1.23}} - \underbrace{\left(\sum_{t=0}^T \gamma^t r_t \mid s_0 = s \right)}_{\substack{\text{Known} \\ \text{future total reward} \\ 1.87}} \right\|_2^2$$

[source](#)

Relationship between the value and reward function

- Mathematically, the value function is the expected sum of discounted rewards from a given state or for a particular action in a given state.
- The value is a function of the reward.
- The value is the sum of (the returns of all trajectories multiplied by the probability of taking that trajectory).

- Policy and value network trained together jointly
- Finally, computing estimates of the advantage function (e.g., via GAE) typically requires that we learn a corresponding value function.
- PPO can train a joint network for policy and value functions by just adding an extra term to the loss function that computes the mean-squared error (MSE) between

Entropy Loss

- Entropy influences the LLM model creativity during the training

$$L^{ENT} = \text{entropy}(\pi_{\theta}(\cdot | s_t))$$

Low entropy:

Prompt

A dog is

Completion

A dog is
a domesticated
carnivorous mammal

Prompt

A dog is

Completion

A dog is
a small carnivorous
mammal

Entropy Loss

- Encourages exploration by adding a reward for more random policies.
- The entropy of a policy is a measure of its randomness.
- It is used to improve exploration by discouraging premature convergence to a sub-optimal policy.

$$L^{ENT} = \text{entropy}(\pi_{\theta}(\cdot | s_t))$$

Low entropy:

Prompt

A dog is

Completion

A dog is
a domesticated
carnivorous mammal

Prompt

A dog is

Completion

A dog is
a small carnivorous
mammal

High entropy:

Prompt

A dog is

Completion

A dog is
is one of the most
popular pets around
the world

Helpful over Harmless



Generated by GPT-4o

Constitutional AI

Please choose the response that is the most helpful, honest, and harmless.

Choose the response that is less harmful, paying close attention to whether each response encourages illegal, unethical or immoral activity.

Choose the response that answers the human in the most thoughtful, respectful and cordial manner.

Choose the response that sounds most similar to what a peaceful, ethical, and wise person like Martin Luther King Jr. or Mahatma Gandhi might say.

...

Smaller Models with RLHF outperform Larger Models!

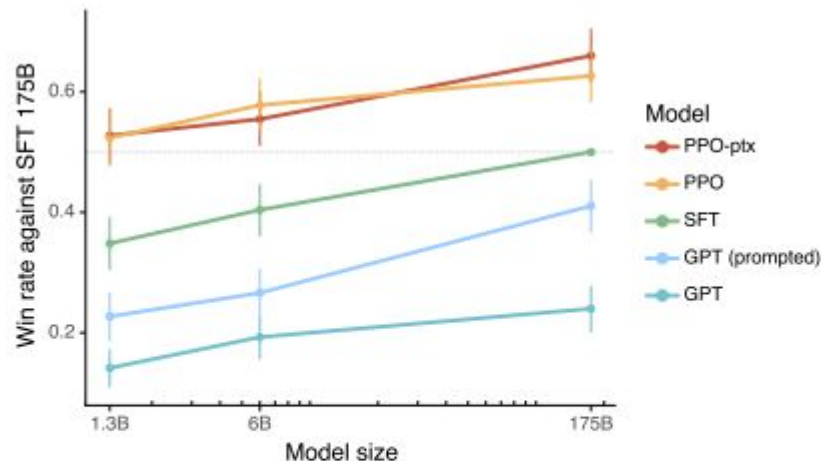


Figure 1: Human evaluations of various models on our API prompt distribution, evaluated by how often outputs from each model were preferred to those from the 175B SFT model. Our InstructGPT models (PPO-ptx) as well as its variant trained without pretraining mix (PPO) significantly outperform the GPT-3 baselines (GPT, GPT prompted); outputs from our 1.3B PPO-ptx model are preferred to those from the 175B GPT-3. Error bars throughout the paper are 95% confidence intervals.

Offline Full Test before Each Release

- We can not do continuous online learning, because the model might learn bad things quickly from the users.
- Instead, we need to train offline and then test the model fully before deploying the new versions

Demo: RLHF

- [RLHF Code: Huggingface, GPT2, Label Studio, Weights and Biases, and trlX](#)
- https://huggingface.co/docs/trl/main/en/ppo_trainer



RLHF: How to Learn from Human Feedback with Reinforcement Learning

Natasha Jaques

Washington University
Google DeepMind



17.07.2021

10:30 - 12:30 BST

Online Resources

<https://cameronrwolfe.substack.com/p/proximal-policy-optimization-ppo>