CMPT 413/713: Natural Language Processing

Post-training: Aligning to human preferences

Spring 2026

2026-03-02

Slides adapted from Anoop Sarkar

GPT models (after GPT-3)

OpenAI

InstructGPT and GPT-3.5 [2022]

- Align responses to human feedback
- Instruction fine-tuning
- Reinforcement learning from human feedback
- Used in initial ChatGPT

- Supervised fine-tuning on human conversations
- Data where human will pretend to be user or AI assistant

- Human rank generated output
- Use reinforcement learning to improve generation

GPT-4 [March 2023]

- Multimodal with images and text (GPT-4V)
- Larger, better model

GPT Assistant training pipeline

Stage	Pretraining	Supervised Finetuning	Reward Modeling	Reinforcement Learning
Dataset	**Raw internet** text trillions of words low-quality, large quantity	**Demonstrations** Ideal Assistant responses, ~10-100K (prompt, response) written by contractors low quantity, high quality	**Comparisons** 100K –1M comparisons written by contractors low quantity, high quality	**Prompts** ~10K-100K prompts written by contractors low quantity, high quality
Algorithm	**Language modeling** predict the next token	**Language modeling** predict the next token	**Binary classification** predict rewards consistent w preferences	**Reinforcement Learning** generate tokens that maximize the reward
		init from	init from	init from SFT use RM
Model	**Base model**	**SFT model**	**RM model**	**RL model**
Notes	1000s of GPUs months of training ex: GPT, LLaMA, PaLM **can deploy this model**	1-100 GPUs days of training ex: Vicuna-13B **can deploy this model**	1-100 GPUs days of training	1-100 GPUs days of training ex: ChatGPT, Claude **can deploy this model**

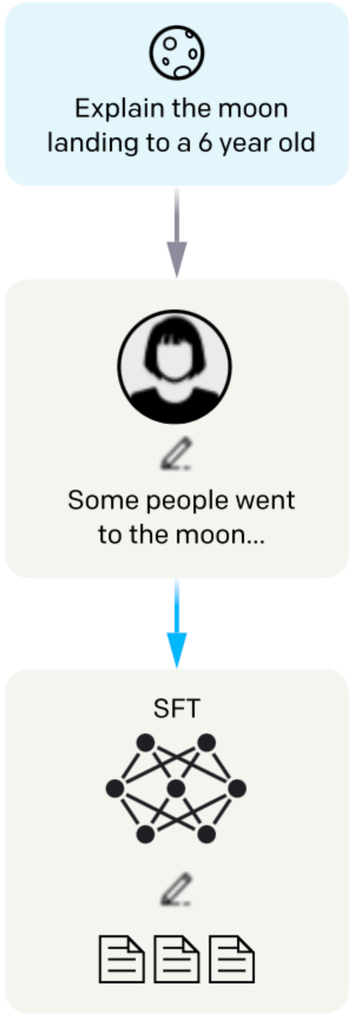
Step 1

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.



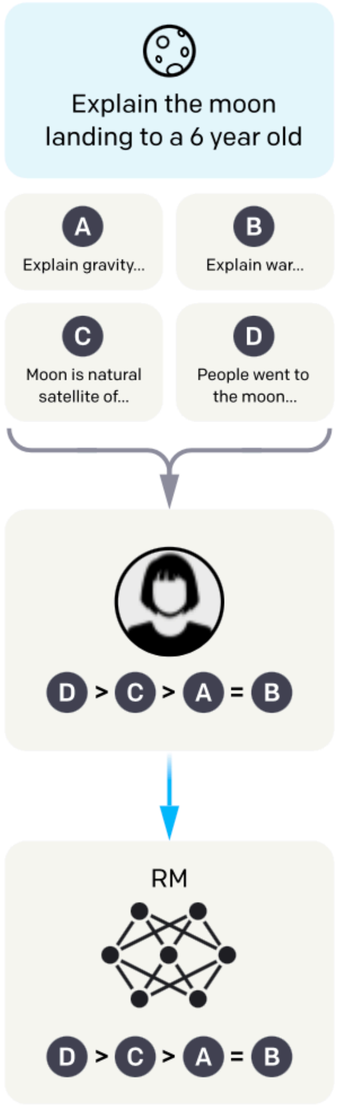
Step 2

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

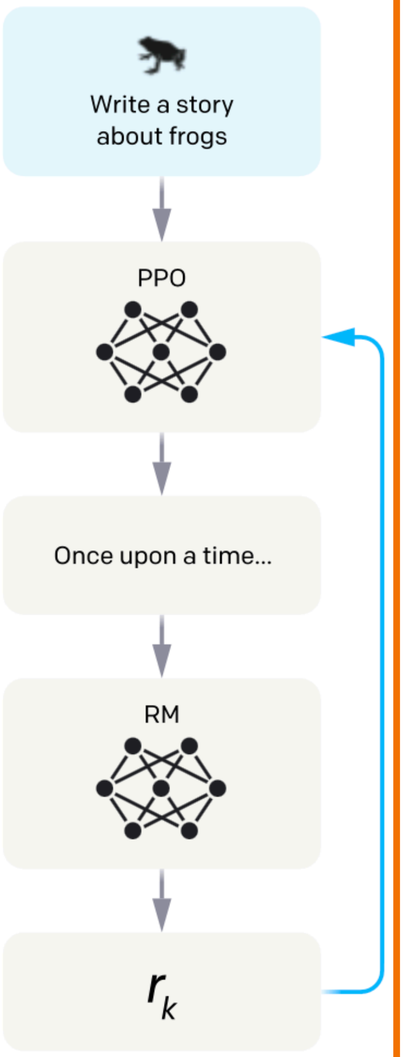
Optimize a policy against the reward model using reinforcement learning.

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.



https://openai.com/research/instruction-following

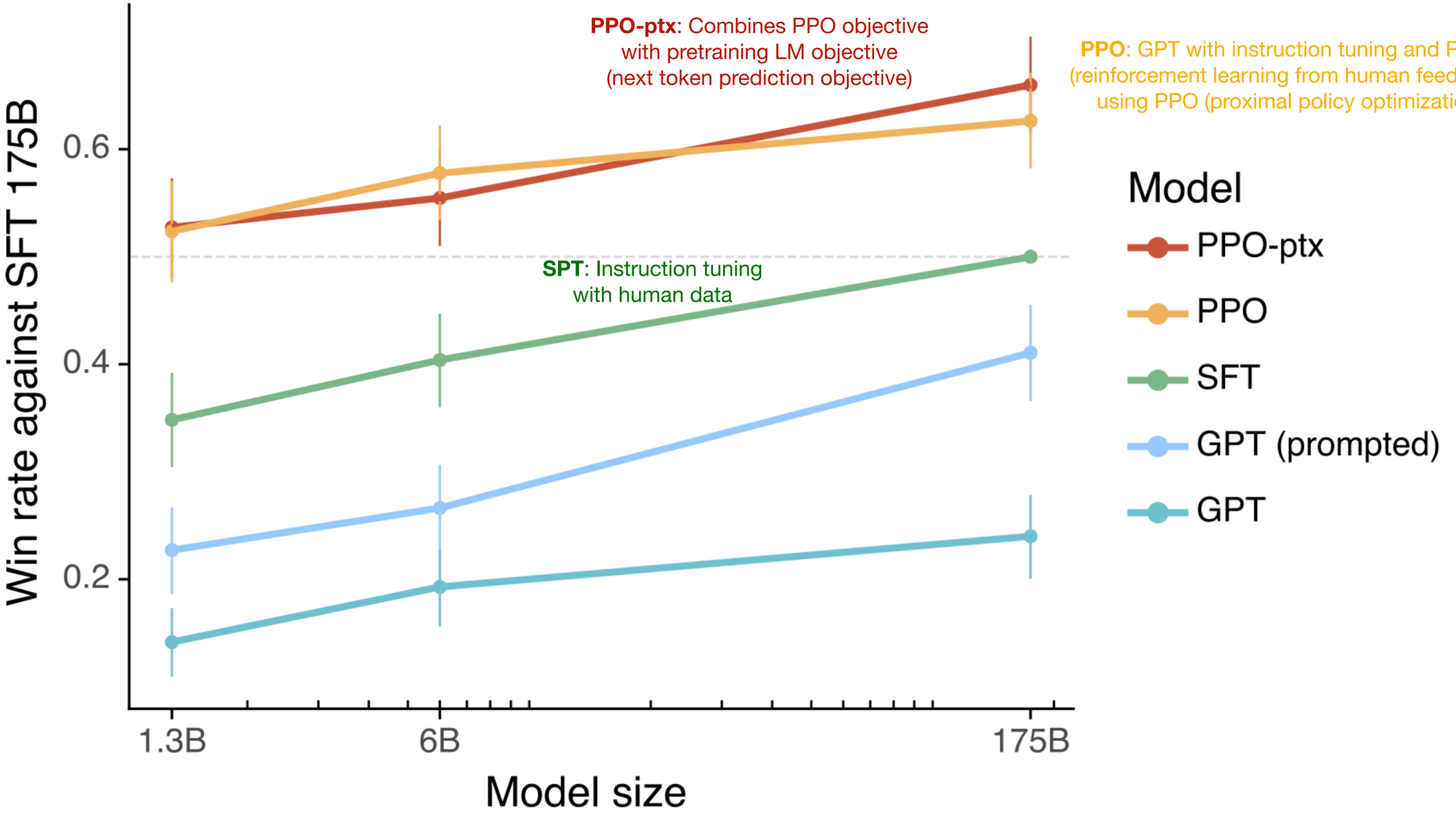
Use human feedback to get better model

- Collect human preferences of generated output
- Build model of human preference (e.g. “reward model”)
- Integrate into model - how to update loss to incorporate reward model?
  - Reinforcement learning (RLHF)
  - Direct preference optimization (DPO)
  - Odds ratio preference optimization (ORPO)

Why RLHF?

Issues with MLE training

- Language model may generate undesirable output due to
  - Not all tokens are equal: mistaken predications worse than others
    - Please send this package to Pittsburgh
    - Please send a package to Pittsburgh
    - Please send this package to Tokyo
    - ****ing send this package to Pittsburgh
  - Training data may contain undesirable outputs (e.g. toxic comments in reddit, disinformation, poor generated data)
  - Exposure bias - model not exposed to mistakes during training, and cannot deal with them at test time


PPO-ptx: Combines PPO objective with pretraining LM objective (next token prediction objective)
PPO: GPT with instruction tuning and RLHF (reinforcement learning from human feedback) using PPO (proximal policy optimization)
SPT: Instruction tuning with human data
Win rate against SFT 175B
0.6
0.4
0.2
1.3B
6B
175B
Model size
Model
PPO-ptx
PPO
SFT
GPT (prompted)
GPT

Dataset

RealToxicity

GPT	0.233
Supervised Fine-Tuning	0.199
InstructGPT	**0.196**

Dataset

TruthfulQA

GPT	0.224
Supervised Fine-Tuning	0.206
InstructGPT	**0.413**

API Dataset

Hallucinations

GPT	0.414
Supervised Fine-Tuning	**0.078**
InstructGPT	0.172

API Dataset

Customer Assistant Appropriate

GPT	0.811
Supervised Fine-Tuning	0.880
InstructGPT	**0.902**

Evaluating InstructGPT for toxicity, truthfulness, and appropriateness. Lower scores are better for toxicity and hallucinations, and higher scores are better for TruthfulQA and appropriateness. Hallucinations and appropriateness are measured on our API prompt distribution. Results are combined across model sizes.

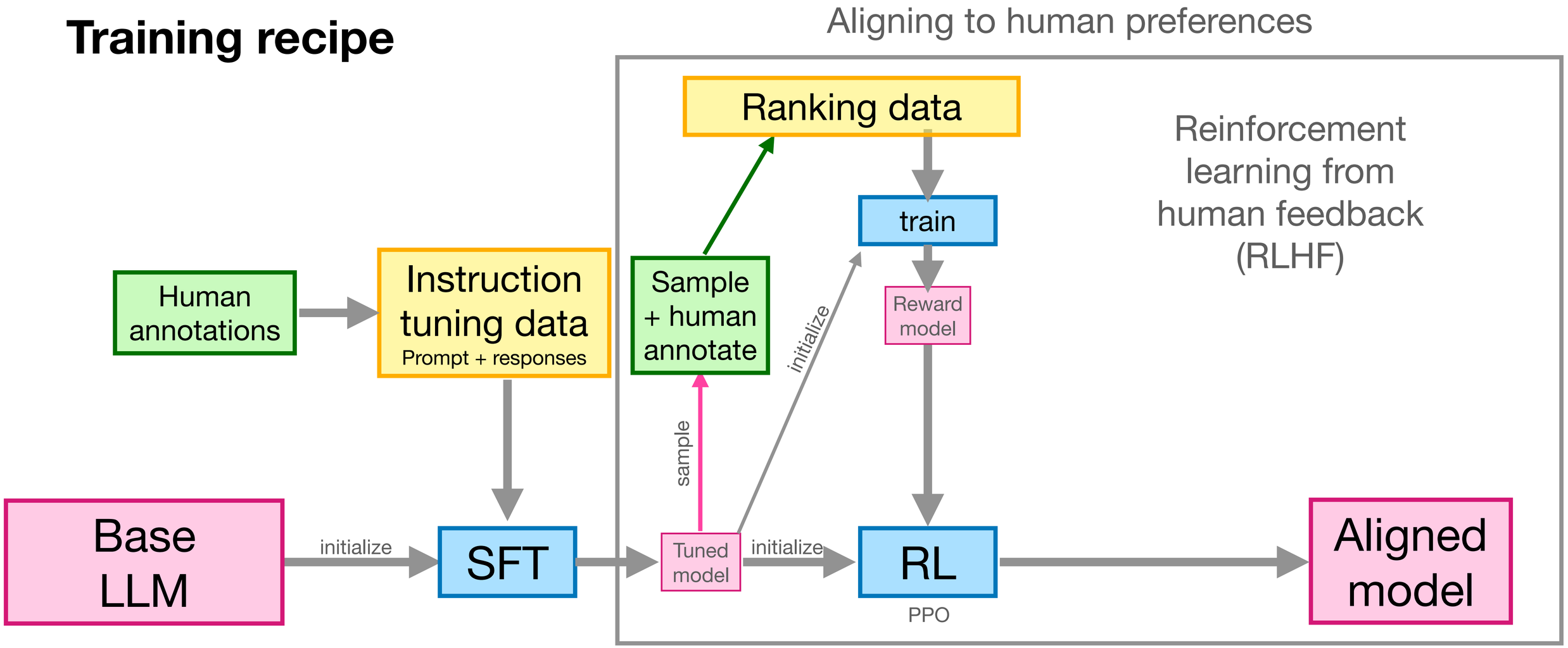
https://openai.com/research/instruction-following

Why RLHF?

- It is often easier to **discriminate** than generate
- Simple example: It is much easier to spot a bad haiku than generate one
- Writing a haiku or writing a summary or writing a story from scratch is a difficult task for humans.
- Humans are better at picking a good example by comparing to other examples.

InstructGPT

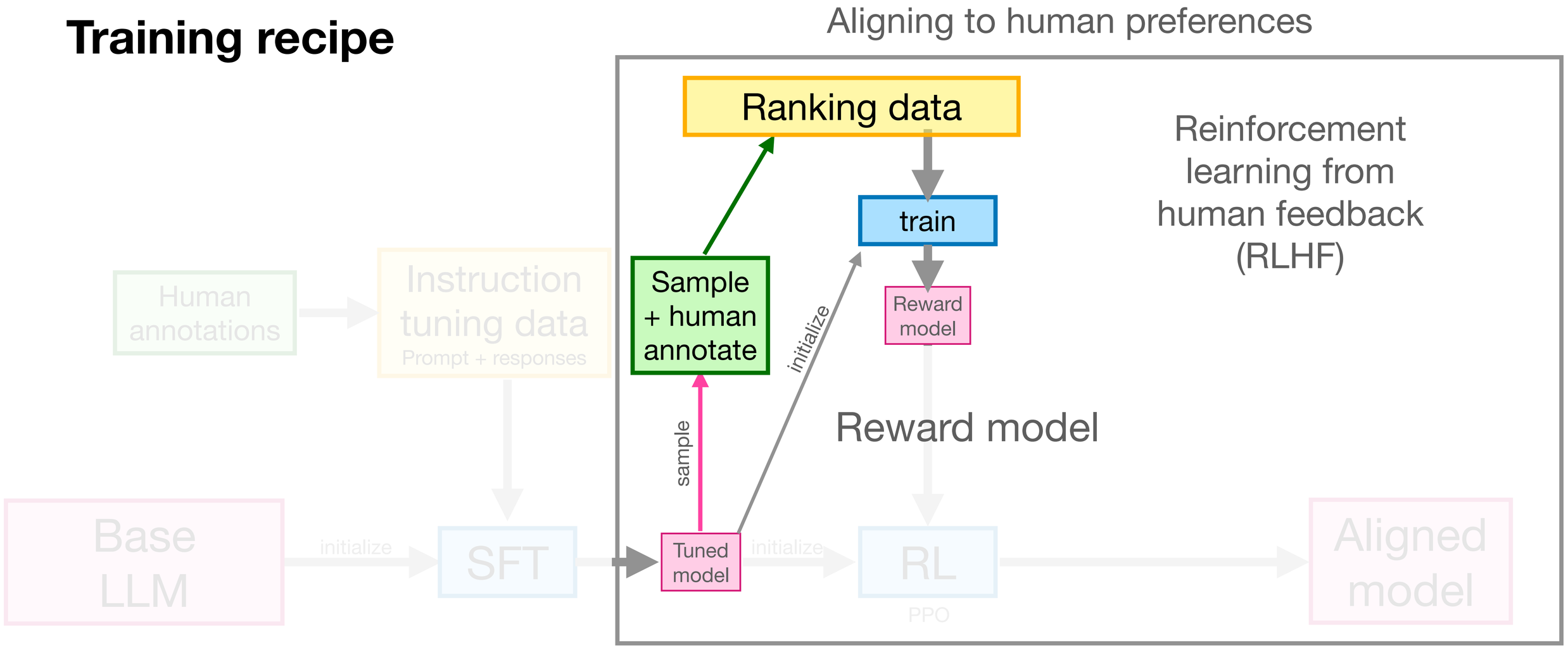
Training recipe



DeepSeek-R1 [DeepSeek 2025] - https://arxiv.org/pdf/2501.12948

InstructGPT

Training recipe



DeepSeek-R1 [DeepSeek 2025] - https://arxiv.org/pdf/2501.12948

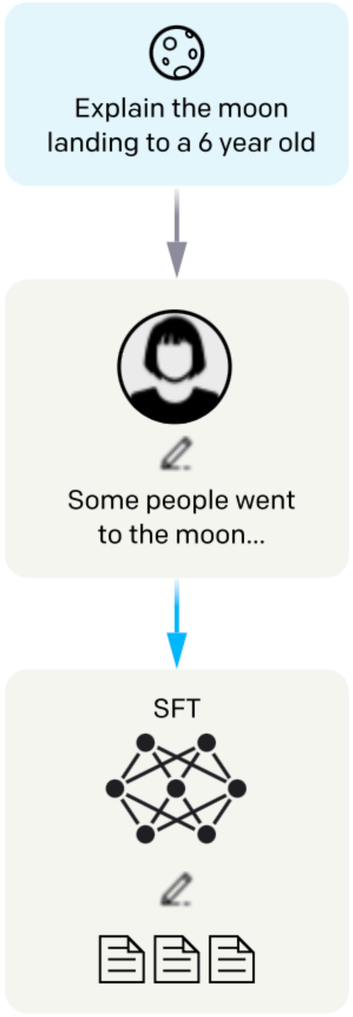
Step 1

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.



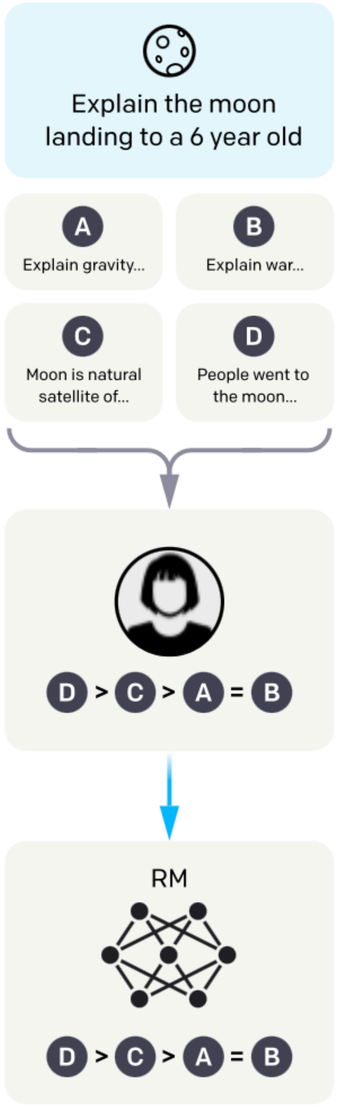
Step 2

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

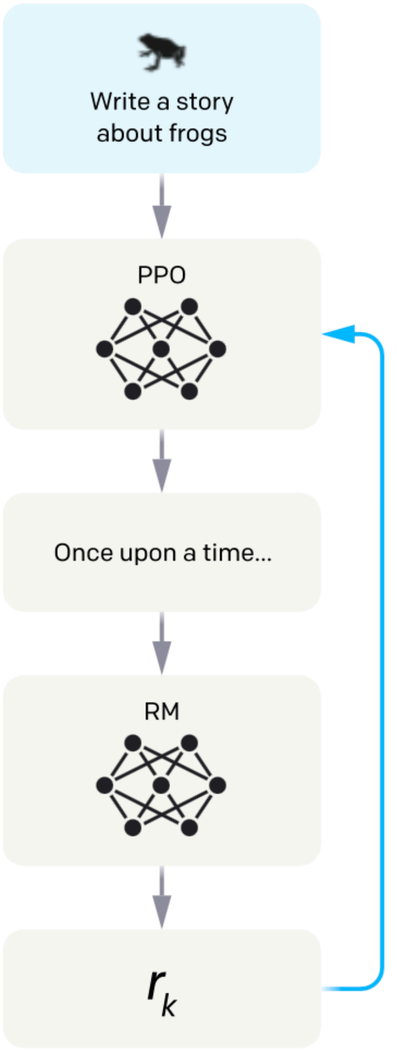
Optimize a policy against the reward model using reinforcement learning.

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.

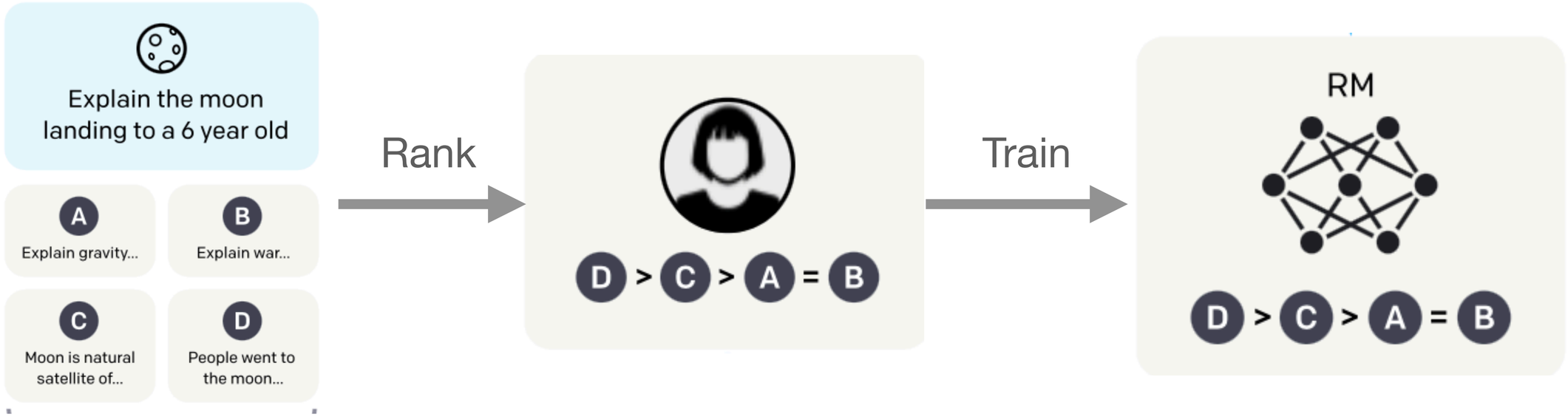


https://openai.com/research/instruction-following

Reward modeling for RLHF

Reward modeling

- Collect dataset of human judgement
- Automatically predict how good an output is based on the collected human preference data
- Use reinforcement learning to train model



https://github.com/openai/following-instructions-human-feedback

Reward model dataset

- Different ways to collect human preferences
  - Directly ask for score (e.g. likert score) overall or for different aspects (e.g. fluency, adequacy, factuality, coherence, etc.)
  - **Ask annotators to compare model outputs and rank**
    - Use ELO or TrueSkill rankings to rank multiple systems
  - Annotate individual errors (fine grained, but time-consuming)

https://github.com/openai/following-instructions-human-feedback

Submit Skip « Page 3 / 11 » Total time: 05:39

Instruction

Include output

Summarize the following news article:

====
{article}
====

Output A

summary1

Rating (1 = worst, 7 = best)

1 2 3 4 5 6 7

Fails to follow the correct instruction / task **?**	Yes	No
Inappropriate for customer assistant **?**	Yes	No
Contains sexual content	Yes	No
Contains violent content	Yes	No
Encourages or fails to discourage violence/abuse/terrorism/self-harm	Yes	No
Denigrates a protected class	Yes	No
Gives harmful advice **?**	Yes	No
Expresses moral judgment	Yes	No

Notes

(Optional) notes

Ranking outputs

To be ranked

B A team of researchers from Yale University and University of California, Davis studied the vocalization patterns of several different types of parrots. They found that parrots like to mimic human speech, and can produce a wide range of sounds, such as whistles, squawks, and other types of vocalizations...

C Parrots have been found to have the ability to understand numbers. Researchers have found that parrots can understand numbers up to six. In a series of experiments, the parrots were able to identify the amount of food items under a number of cups...

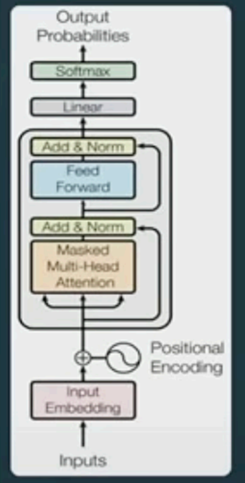
Rank 1 *(best)*	Rank 2	Rank 3	Rank 4	Rank 5 *(worst)*
A A research group in the United States has found that parrots can imitate human speech with ease, and some of them can even do so in the same way as humans. This group studied the sounds that parrots make in their natural habitats and found that they use their tongues and beaks in ways that are strikingly...		E Scientists have found that green-winged parrots can tell the difference between two noises that are the same except for the order in which they are heard. This is important because green-winged parrots are known to imitate sounds. This research shows that they are able to understand the difference between sounds.		
		D Current research suggests that parrots see and hear things in a different way than humans do. While humans see a rainbow of colors, parrots only see shades of red and green. Parrots can also see ultraviolet light, which is invisible to humans. Many birds have this ability to see ultraviolet light, an ability		

Blue are the prompt tokens, identical across rows
Yellow are completion tokens, different in each row
Green is the special <|reward|> token "readout"
Only the outputs at the green cells is used, the rest are ignored

1.2

0.2

-0.5



loss function measures the predicted rewards' consistency with the labeled ordering

T

B

prompt	...	...	completion 1	...	...	<|reward|>			
prompt	...	...	completion 2	...	...	...	...	...	<|reward|>
prompt	...	...	completion 3	...	<|reward|>				

Reward Model Training

- Let θ be the parameters for the <reward> token which is appended at the end of each completion
- Data: Prompt | Completion | <reward>
- K is the number of responses ranked by humans (K={4,9}). D is the dataset of human comparisons
- This produces $\binom{K}{2}$ comparisons for each prompt

Difference in reward between two outputs

(Log-odds that y_w is preferred to y_l)

- Loss function: $loss(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}[\log(\sigma(r_\theta(x,y_w) - r_\theta(x,y_l)))]$
- $r_\theta(x,y)$ is the scalar reward for prompt x and completion y. y_w is preferred to y_l
- Train all $\binom{K}{2}$ comparisons in a single batch.
- Training the 175B model does not work, instead fine-tune a smaller 6B model to predict reward.

Bradley-Terry ranking

- The BT model is a probability model for the outcome of pairwise comparisons.
- Given a pair of individual responses i and j
- The probability of preferring $i > j$ is given by
- $$P(i > j) = \frac{p_i}{p_i + p_j}$$
- The Bradley–Terry model can be used in the forward direction to predict outcomes,
- But is more commonly used in reverse to infer the scores p_i given an observed set of outcomes (preferences from humans)
- More general models exist: e.g. Plackett-Luce models (but not used for RLHF)

Bradley-Terry ranking

- Binary classification problem: given prompt x and responses y_w and y_l, predict the probability that y_w is preferred to y_l
- Let p_w and p_l be scores given to y_w and y_l

$$P(w > l) = \frac{p_w}{p_w + p_l} = \frac{1}{1 + \frac{p_l}{p_w}}$$

$$= \frac{1}{1 + \exp(r_\theta(x, y_l) - r_\theta(x, y_w))}$$

$$= \frac{1}{1 + \exp(-(r_\theta(x, y_w) - r_\theta(x, y_l)))} = \sigma(r_\theta(x, y_w) - r_\theta(x, y_l))$$

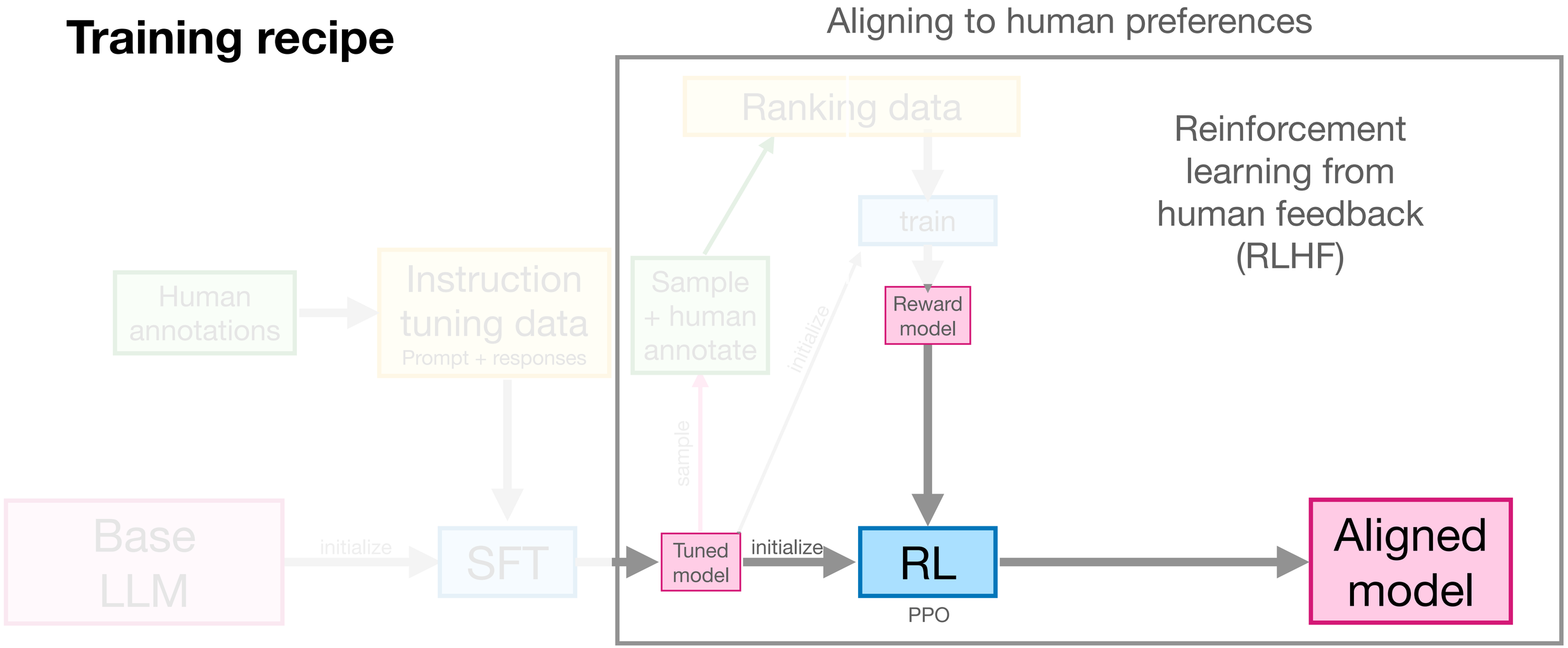
$$p_w = \exp(r_\theta(x, y_w))$$

$$p_l = \exp(r_\theta(x, y_l))$$

Optimizing using reward model

InstructGPT

Training recipe



DeepSeek-R1 [DeepSeek 2025] - https://arxiv.org/pdf/2501.12948

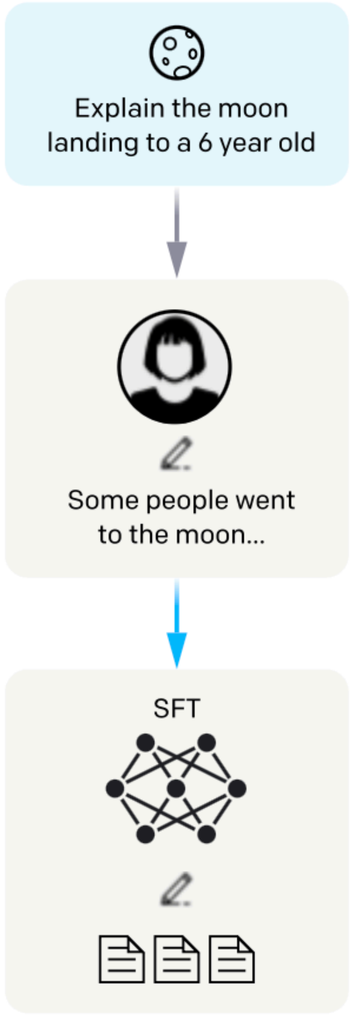
Step 1

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.



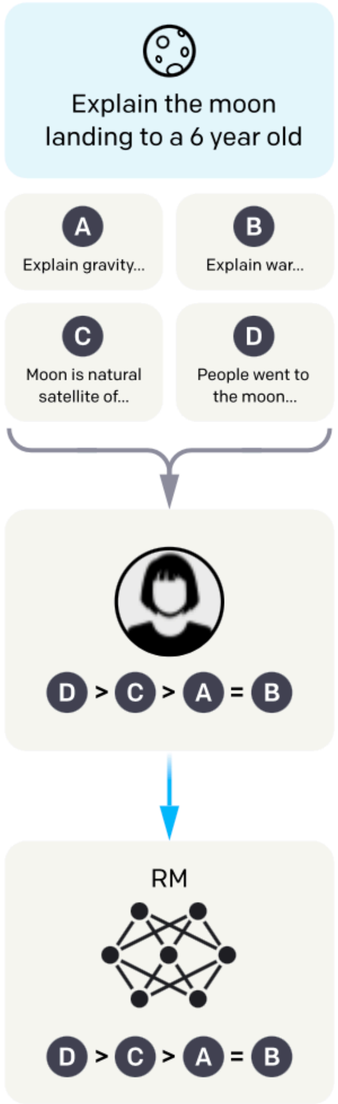
Step 2

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

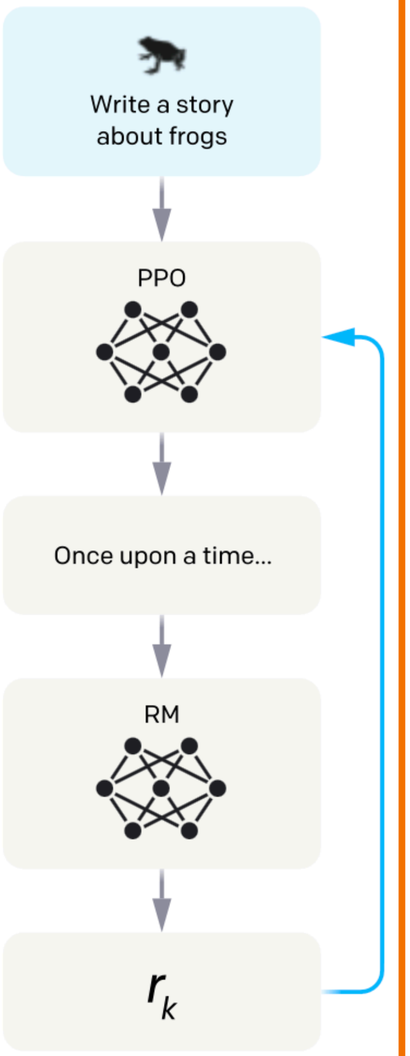
Optimize a policy against the reward model using reinforcement learning.

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.



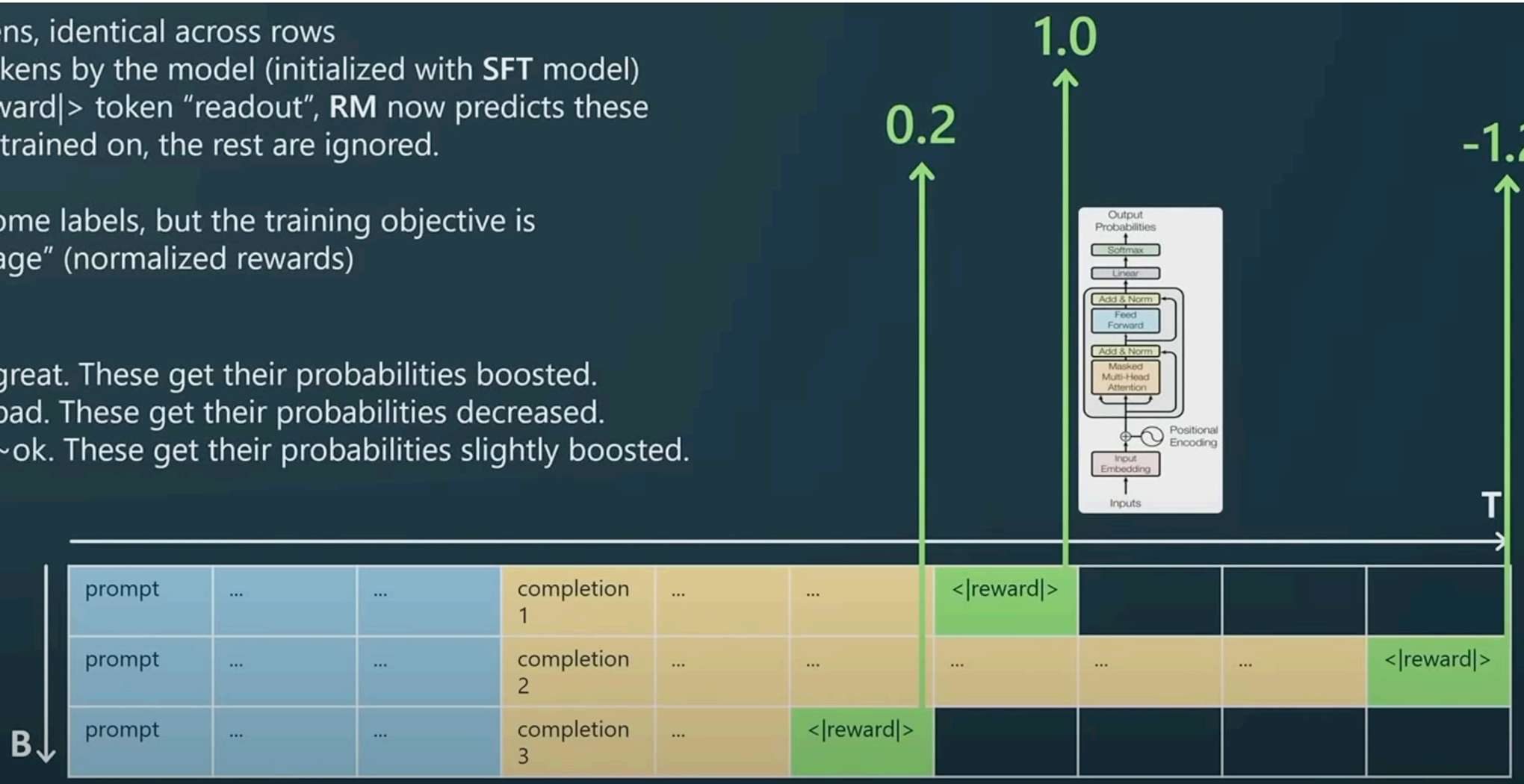
https://openai.com/research/instruction-following

Blue are the prompt tokens, identical across rows
Yellow are completion tokens by the model (initialized with **SFT** model)
Green is the special <|reward|> token "readout", **RM** now predicts these
Only the **yellow** cells are trained on, the rest are ignored.

The sampled tokens become labels, but the training objective is weighted by the "advantage" (normalized rewards)

In this example:

- Row #1 tokens were great. These get their probabilities boosted.
- Row #2 tokens were bad. These get their probabilities decreased.
- Row #3 tokens were ~ok. These get their probabilities slightly boosted.



[Proximal Policy Optimization Algorithms, Schulman et al. 2017]

Initialize RL policy with SFT
Keep RL policy from drifting too far

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

- Let ϕ be the parameters for the language model.
- Parameters for the <reward> token are kept frozen.
- π_ϕ^{RL} is the learned RL policy
- π^{SFT} is the learned supervised fine-tuning model
- β is the KL reward coefficient
- Training for chatGPT (probably) uses an actor-critic algorithm similar to proximal policy optimization (PPO) for training the ϕ parameters

Reinforcement learning

Determine policy to maximize expected accumulated reward.

Typically modelled as POMDP (sequence of states with partial observations)

- **Actions**: What token to output?
- **Policy**: What action(s) to take given sequence of observations and actions?
  - Policy models the probability of action given state
  - For text generation, what sequence of tokens to generate given input tokens: $\pi(a, s) = P(y \mid x)$
- **Reward**: Provided by reward model trained on human preferences

Actor-Critic RL

https://arxiv.org/pdf/1607.07086v2.pdf

- Standard methods to apply RL in LMs involve producing the expected reward of generating a token and generating a per-token loss for each position
- The REINFORCE algorithm is the standard way to do this for language models
- However, REINFORCE only uses a single sample token to compare against (compare y_w with y_l where $p_{y_w} > p_{y_l}$)
- Instead the actor-critic approach uses two models:
  - one is the **critic** and one is the **actor** (both initialized with LM)
- The critic model is trained against the reward model to produce <|reward|> at the end
- The actor model is trained against the critic and produces <|endoftext|> at the end and is trained against the critic output for each time step

RLHF with general policy gradient method

- Initialize the policy π_ϕ^{RL} to π^{SFT}, the policy output from SFT
- Loop for many steps
  - Initialize a new empty dataset $D_{\pi_\phi^{\text{RL}}}$
  - Loop for many steps
    - Sample a random prompt x from D_{RL}
    - Generate a response y from the policy π_ϕ^{RL}
    - Calculate the reward signal $r_\theta(x, y)$ from r_θ
    - Add the triple $(x, y, r_\theta(x, y))$ to $D_{\pi_\phi^{\text{RL}}}$
  - Use policy gradient method to increase objective:

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

RLHF with PPO

Algorithm 1 PPO

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0.
2: **for** $n = 0, 1, 2, \ldots$ **do**
3: Collect a set of trajectories $\mathcal{D}_n = \{\tau_i\}$ by executing policy $\pi(\theta_n)$ within the environment.
4: Compute rewards-to-go $\hat{R}_t$.
5: Compute advantage estimates, $\hat{A}_t$ (using any advantage estimation method) based on the current value function V_{ϕ_n}.
6: Update the policy by maximizing the PPO-penalty/clip/ptx objective:

$$\theta_{n+1} = \arg \max_{\theta} \mathcal{L}_{\text{ppo-clip}}(\theta_n).$$

7: Update the value function by regression on mean-squared error:

$$\phi_{n+1} = \arg \min_{\phi} \mathcal{L}_{\text{critic}}(\phi_n).$$

8: **end for**

https://arxiv.org/pdf/2307.04964

Instruct-GPT objective

https://arxiv.org/pdf/1607.07086v2.pdf

- Full model (PPO-ptx)
- Combines PPO and pretraining objectives using RL training

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

$$+ \quad \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\text{RL}}(x))\right]$$

Direct preference optimization

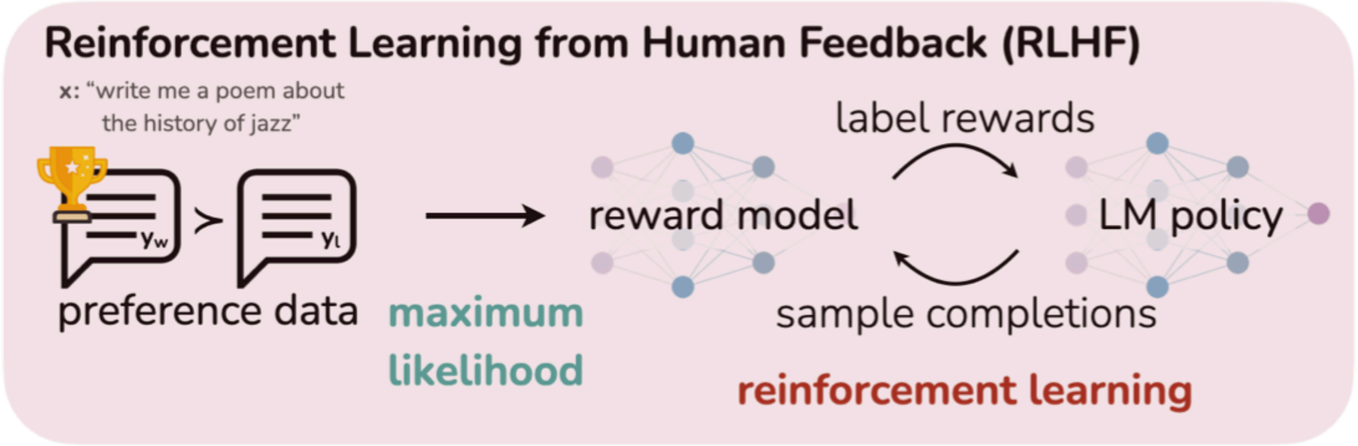
Use human feedback to get better model

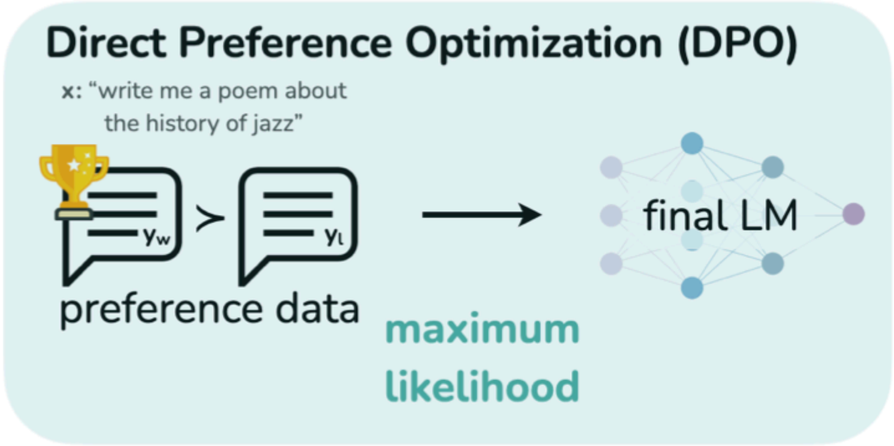
- Is it necessary to train using RL?
- Can directly optimize on preferences using supervised training with preference data.

https://arxiv.org/pdf/2305.18290.pdf

Direct preference optimization

aka, Your Language Model is Secretly a Reward Model





You can use maximum likelihood estimation to directly train for preference optimization

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

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

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

https://arxiv.org/pdf/2305.18290.pdf

Direct preference optimization

aka, Your Language Model is Secretly a Reward Model

Optimal policy is given by $$\pi_r(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

Rewrite to get $$r(x, y) = \beta \log \frac{\pi_r(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x).$$

BT model

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp\left(r^*(x, y_1)\right)}{\exp\left(r^*(x, y_1)\right) + \exp\left(r^*(x, y_2)\right)}$$

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

Maximum likelihood estimate

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

https://arxiv.org/pdf/2305.18290.pdf

Direct preference optimization

aka, Your Language Model is Secretly a Reward Model

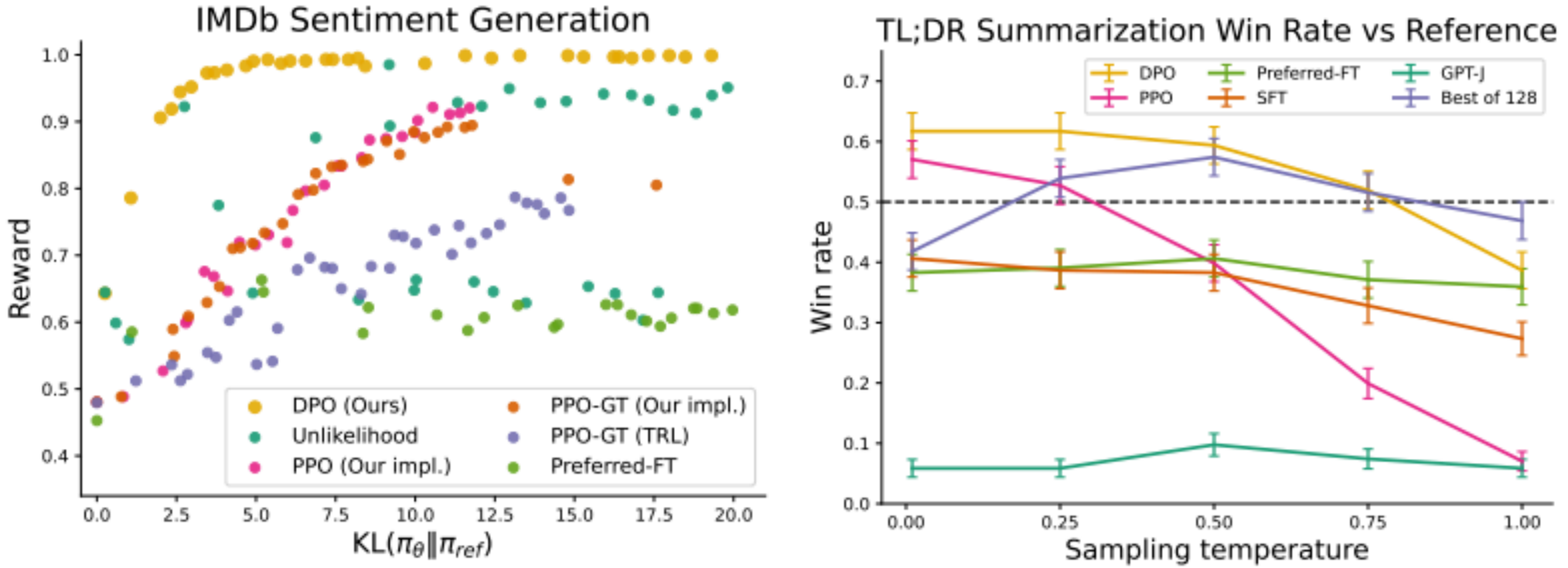
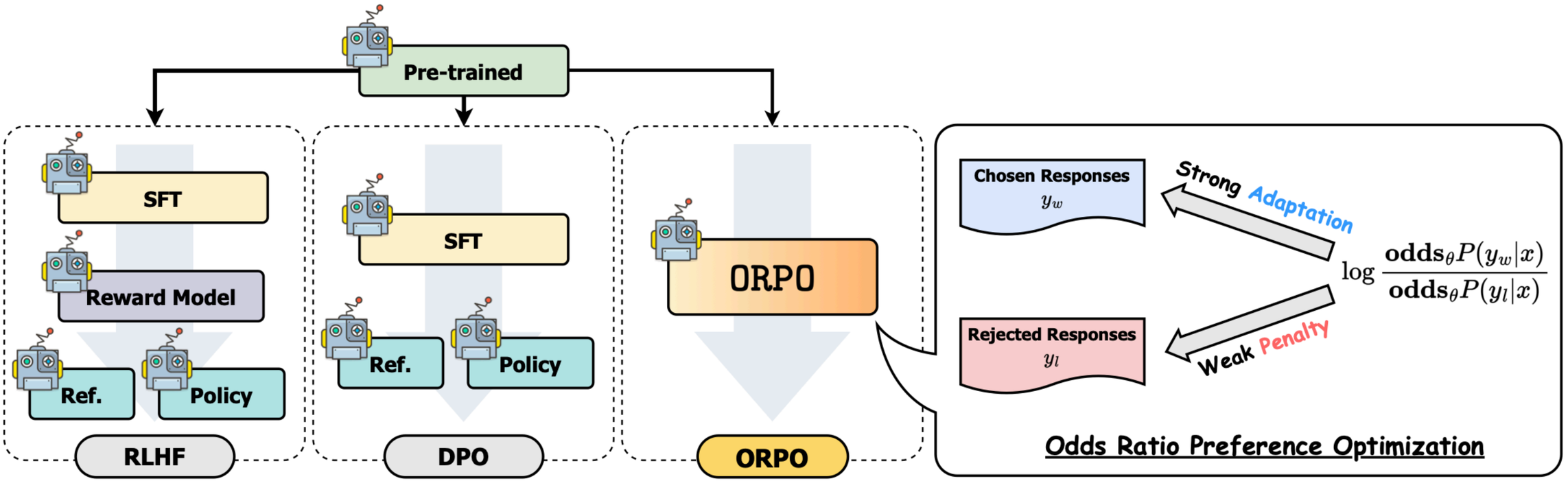


Figure 2: **Left.** The frontier of expected reward vs KL to the reference policy. DPO provides the highest expected reward for all KL values, demonstrating the quality of the optimization. **Right.** TL;DR summarization win rates vs. human-written summaries, using GPT-4 as evaluator. DPO exceeds PPO's best-case performance on summarization, while being more robust to changes in the sampling temperature.

https://arxiv.org/abs/2403.07691

ORPO Preference Optimization without a Reference



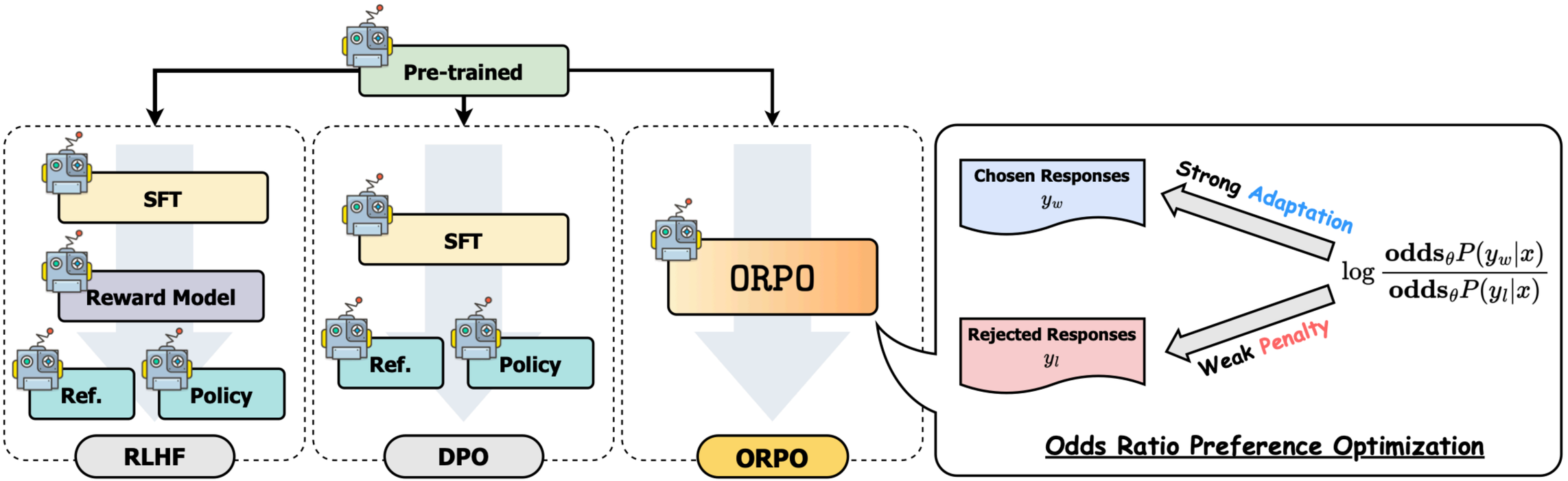
Eliminate need for a separate supervised fine-tuning (SFT) step

Combines normal cross-entropy loss with a odds-ratio loss

$$\mathcal{L}_{ORPO} = \mathbb{E}_{(x,y_w,y_l)}\left[\mathcal{L}_{SFT} + \lambda \cdot \mathcal{L}_{OR}\right]$$

https://arxiv.org/abs/2403.07691

ORPO Preference Optimization without a Reference



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

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

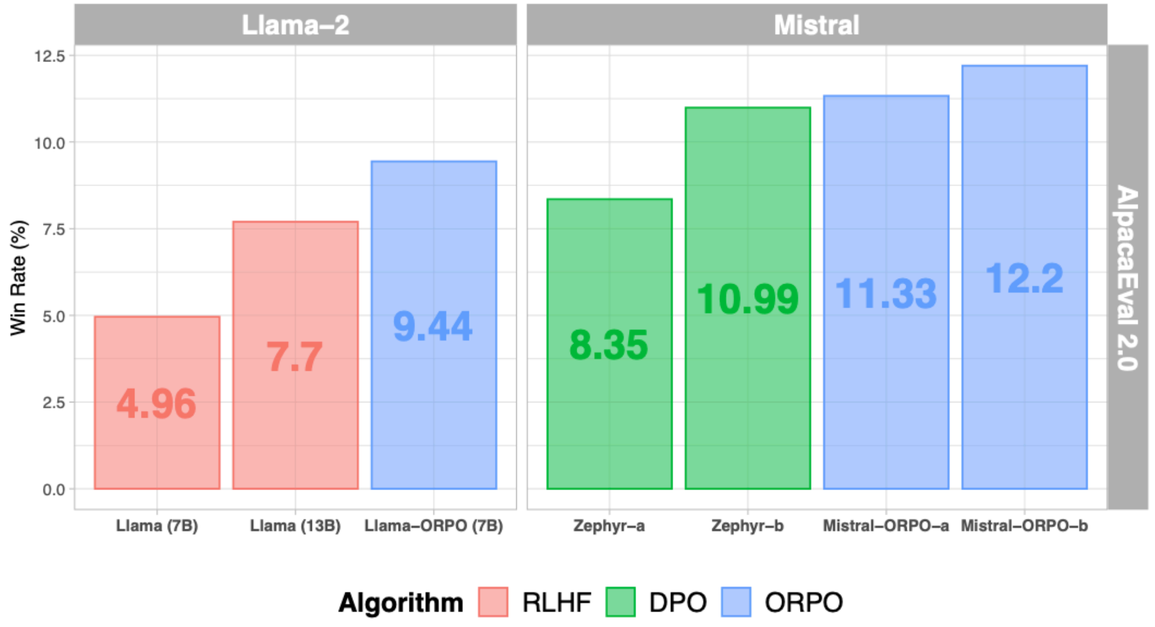
$$\mathbf{OR}_\theta(y_w, y_l) = \frac{\mathbf{odds}_\theta(y_w|x)}{\mathbf{odds}_\theta(y_l|x)}$$

$$\mathcal{L}_{OR} = -\log \sigma \left(\log \frac{\mathbf{odds}_\theta(y_w|x)}{\mathbf{odds}_\theta(y_l|x)} \right)$$

$$\mathcal{L}_{ORPO} = \mathbb{E}_{(x, y_w, y_l)} \left[\mathcal{L}_{SFT} + \lambda \cdot \mathcal{L}_{OR} \right]$$

https://arxiv.org/abs/2403.07691

ORPO Preference Optimization without a Reference



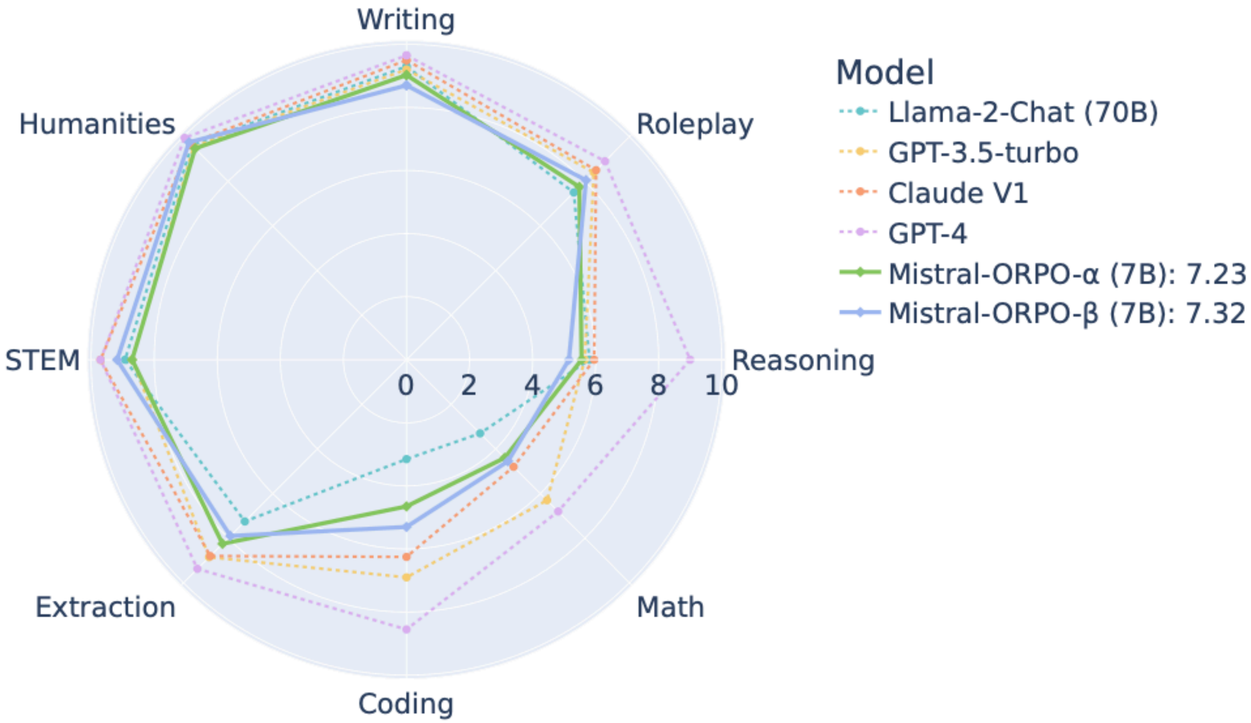
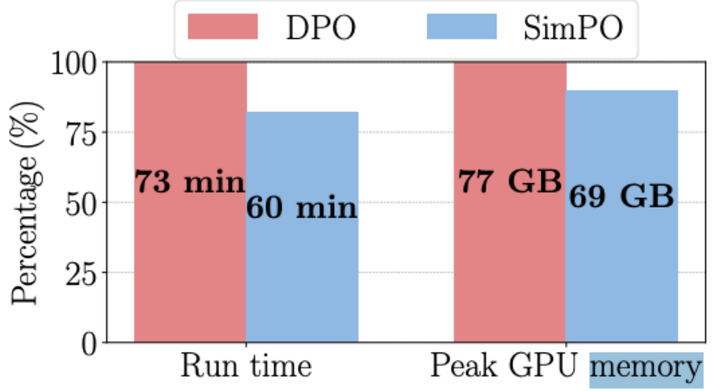


Figure 1: AlpacaEval$_{2.0}$ result of Llama-2 (7B) and Mistral (7B) fine-tuned with ORPO (blue) in comparison to the state-of-the-art models. Notably, Mistral-ORPO-α & β surpasses Zephyr β and Llama-2-Chat (13B) with a single epoch training exclusively on the UltraFeedback.

https://tatsu-lab.github.io/alpaca_eval/ Automated (LLM-based) evaluator for instruction-tuned models

SimPO

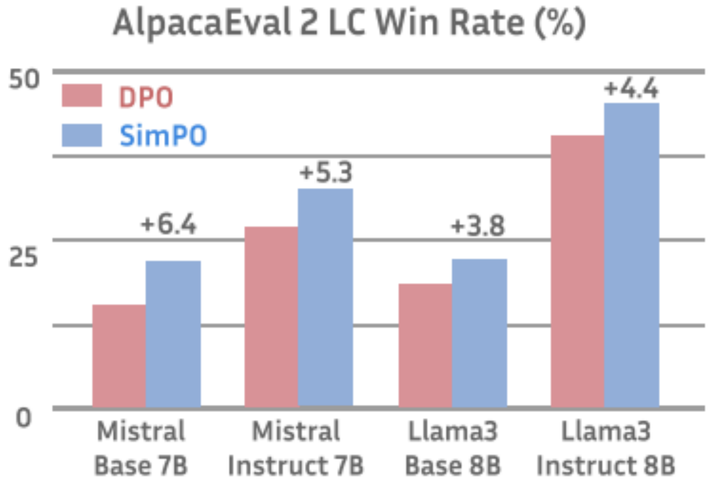
- Length normalization
- No reference policy
- Reward margin γ

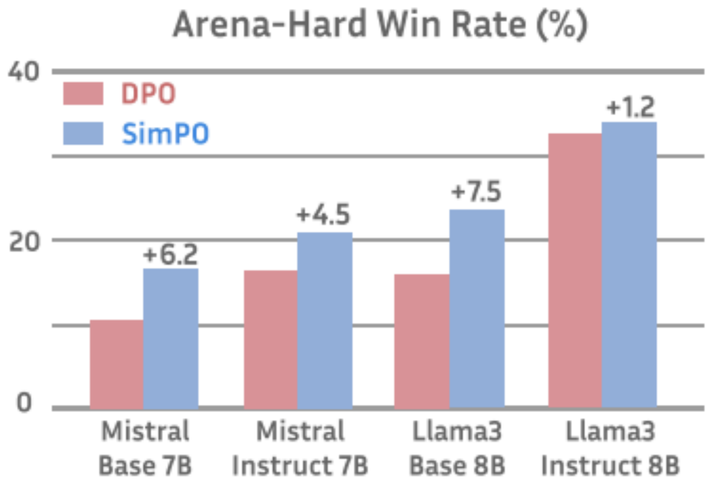


- Lighter weight (20% faster, 10% less memory)
- Better performance

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

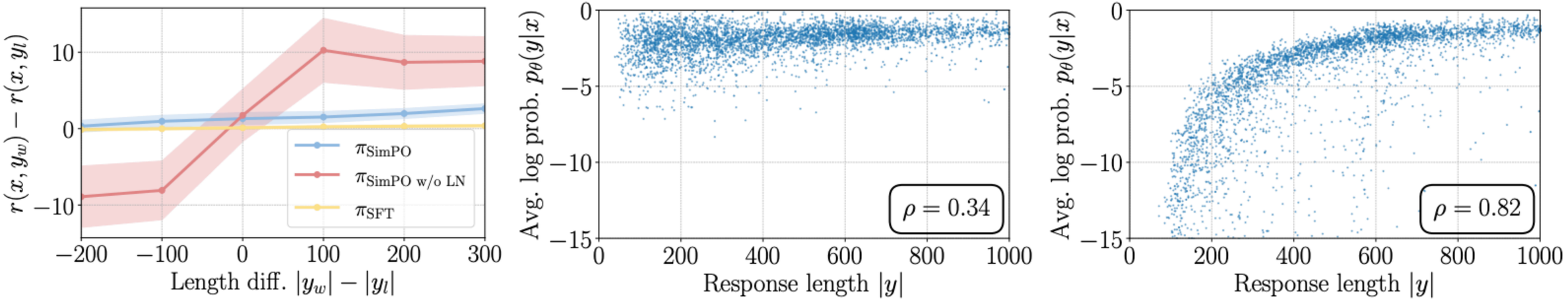
$$\mathcal{L}_{\text{SimPO}}(\pi_\theta) = -\mathbb{E}\left[\log \sigma\left(\frac{\beta}{|y_w|} \log \pi_\theta(y_w \mid x) - \frac{\beta}{|y_l|} \log \pi_\theta(y_l \mid x) - \gamma\right)\right]$$





SimPO: Simple Preference Optimization with a Reference-Free Reward [Meng et al. 2024]
https://arxiv.org/pdf/2405.14734

SimPO



(a) Reward optimization. (b) SimPO. (c) SimPO without LN.

Figure 2: Effect of length normalization (LN). (a) Relationship between reward margin and length difference between winning and losing responses. (b) Spearman correlation between average log probability and response length for SimPO. (c) Spearman correlation for SimPO without LN.

SimPO: Simple Preference Optimization with a Reference-Free Reward [Meng et al. 2024]
https://arxiv.org/pdf/2405.14734

https://princeton-cos597r.github.io/lectures/lec08.pdf

Offline preference optimization

Preference data: (prompt, winning response, losing response) $(x, y_w, y_l) \sim D$

There are many objectives that you can design for directly learning from preference data!

Method	Objective
RRHF [84]	$\max\left(0, -\frac{1}{\lvert y_w\rvert}\log\pi_\theta(y_w\lvert x) + \frac{1}{\lvert y_l\rvert}\log\pi_\theta(y_l\lvert x)\right) - \lambda\log\pi_\theta(y_w\lvert x)$
SLiC-HF [88]	$\max\left(0, \delta - \log\pi_\theta(y_w\lvert x) + \log\pi_\theta(y_l\lvert x)\right) - \lambda\log\pi_\theta(y_w\lvert x)$
DPO [62]	$-\log\sigma\left(\beta\log\frac{\pi_\theta(y_w\lvert x)}{\pi_{\text{ref}}(y_w\lvert x)} - \beta\log\frac{\pi_\theta(y_l\lvert x)}{\pi_{\text{ref}}(y_l\lvert x)}\right)$
IPO [6]	$\left(\log\frac{\pi_\theta(y_w\lvert x)}{\pi_{\text{ref}}(y_w\lvert x)} - \log\frac{\pi_\theta(y_l\lvert x)}{\pi_{\text{ref}}(y_l\lvert x)} - \frac{1}{2\tau}\right)^2$
CPO [81]	$-\log\sigma\left(\beta\log\pi_\theta(y_w\lvert x) - \beta\log\pi_\theta(y_l\lvert x)\right) - \lambda\log\pi_\theta(y_w\lvert x)$
KTO [25]	$-\lambda_w\sigma\left(\beta\log\frac{\pi_\theta(y_w\lvert x)}{\pi_{\text{ref}}(y_w\lvert x)} - z_{\text{ref}}\right) + \lambda_l\sigma\left(z_{\text{ref}} - \beta\log\frac{\pi_\theta(y_l\lvert x)}{\pi_{\text{ref}}(y_l\lvert x)}\right)$, where $z_{\text{ref}} = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\beta\text{KL}\left(\pi_\theta(y\lvert x)\lVert\pi_{\text{ref}}(y\lvert x)\right)\right]$
ORPO [38]	$-\log p_\theta(y_w\lvert x) - \lambda\log\sigma\left(\log\frac{p_\theta(y_w\lvert x)}{1-p_\theta(y_w\lvert x)} - \log\frac{p_\theta(y_l\lvert x)}{1-p_\theta(y_l\lvert x)}\right)$, where $p_\theta(y\lvert x) = \exp\left(\frac{1}{\lvert y\rvert}\log\pi_\theta(y\lvert x)\right)$
R-DPO [60]	$-\log\sigma\left(\beta\log\frac{\pi_\theta(y_w\lvert x)}{\pi_{\text{ref}}(y_w\lvert x)} - \beta\log\frac{\pi_\theta(y_l\lvert x)}{\pi_{\text{ref}}(y_l\lvert x)} - (\alpha\lvert y_w\rvert - \alpha\lvert y_l\rvert)\right)$

WR: winning rate, LC: length-controlled WR

Method	LLama-3-instruct (8B)		
	AlpacaEval 2		Arena-Hard
	LC (%)	WR (%)	WR (%)
SFT	26.0	25.3	22.3
RRHF [84]	37.9	31.6	28.8
SLiC-HF [88]	33.9	32.5	29.3
DPO [62]	48.2	**47.5**	35.2
IPO [6]	46.8	42.4	**36.6**
CPO [81]	34.1	36.4	30.9
KTO [25]	34.1	32.1	27.3
ORPO [38]	38.1	33.8	28.2
R-DPO [60]	48.0	45.8	35.1
SimPO	**53.7**	**47.5**	36.5

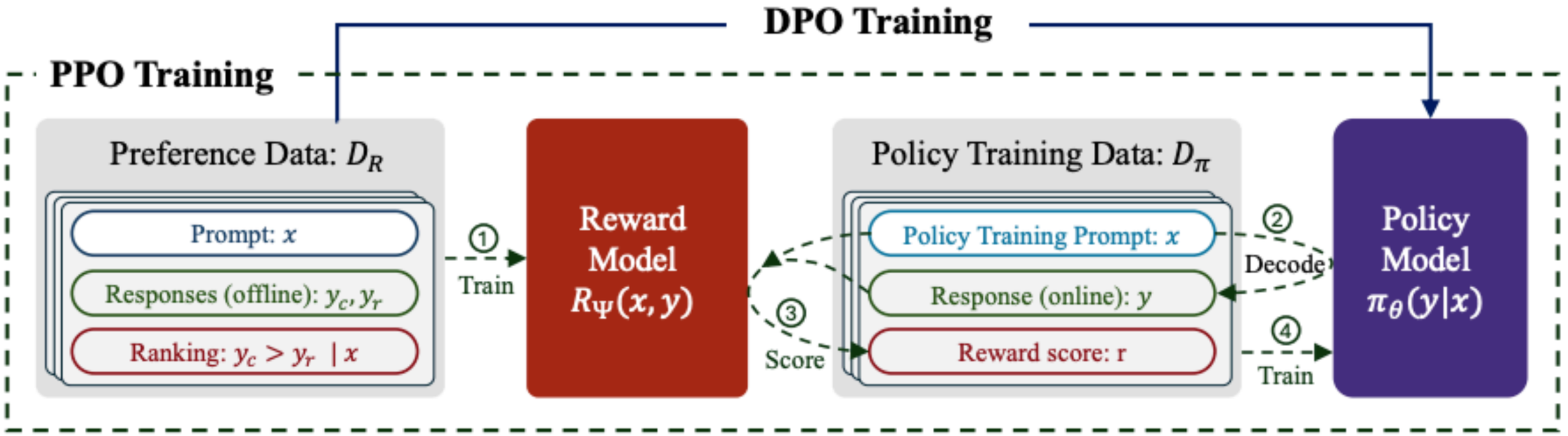
SimPO: Simple Preference Optimization with a Reference-Free Reward [Meng et al. 2024]
https://arxiv.org/pdf/2405.14734

Offline vs online preference optimization

- Many different ways to use preference data
- Online RLHF
  - New preference data generation using reward model
  - Critic: learning of value function with new preference data
  - Actor: learning of policy learning with updated critic and new preference data
- Offline RLHF
  - Can be more efficient, but limited to labeled preference data

DPO vs PPO

- PPO - Online Actor-Critic model - new model generated response during training
- DPO - Offline model - only have access to human labeled preference data

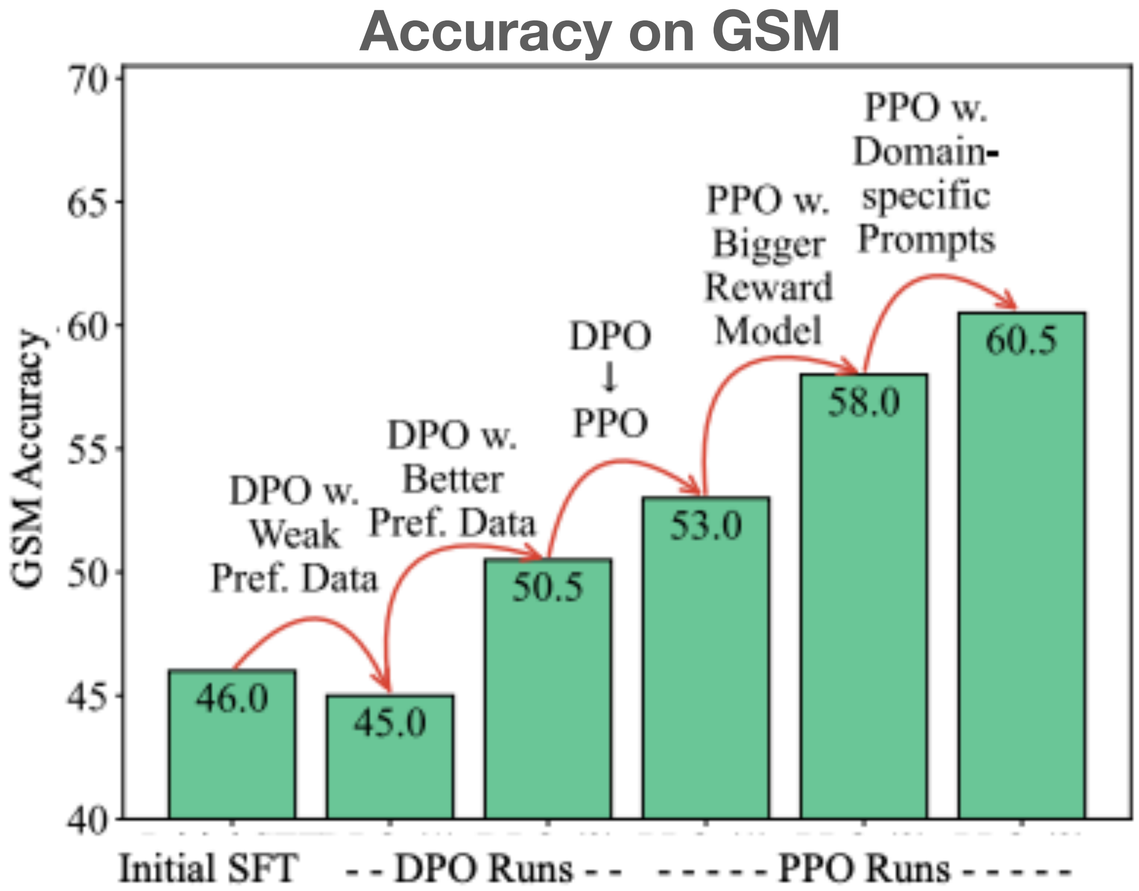


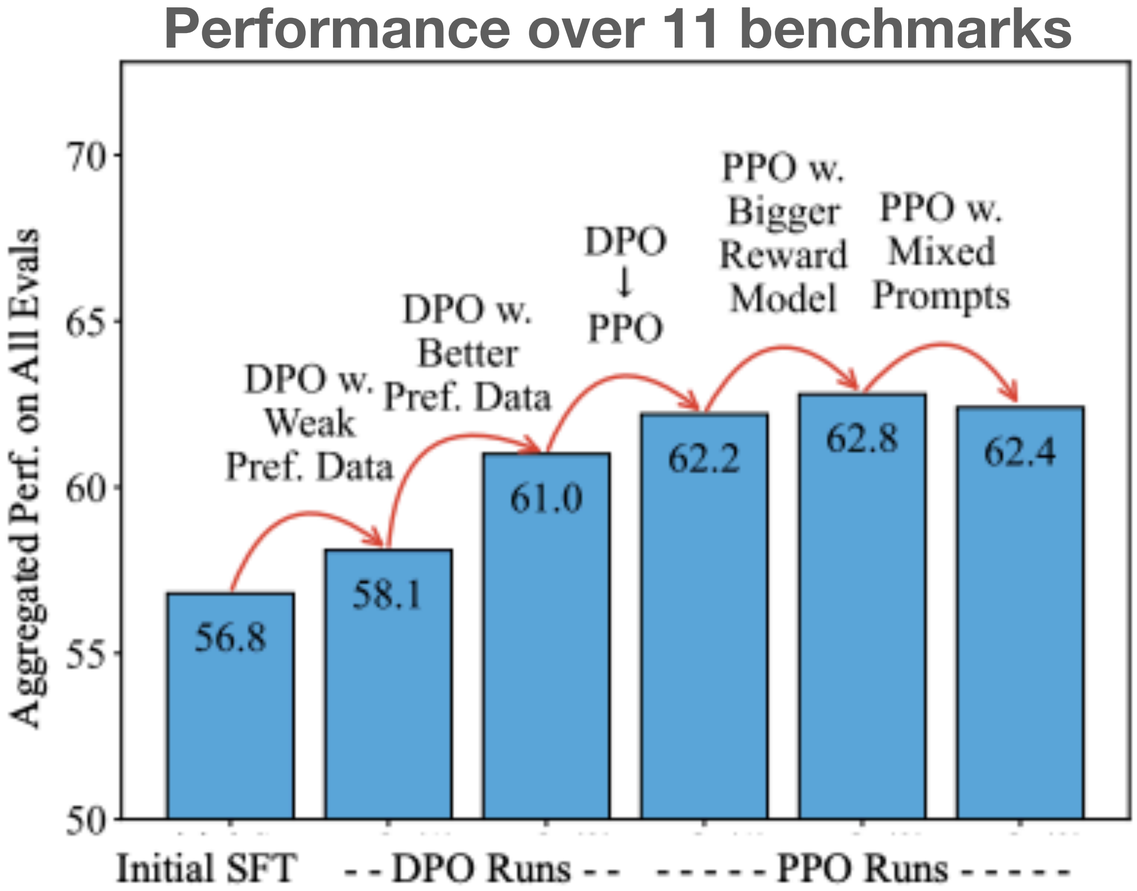
Unpacking DPO and PPO: Disentangling Best Practices for Learning from Preference Feedback [Ivison et al. 2024]
https://arxiv.org/pdf/2406.09279

DPO vs PPO

Performance training of TULU 2 13B
(Series of Llama 2 model fine-tuned across different sizes)

(Covers factuality, reasoning, coding, truthfulness, safety, instruction following)





Unpacking DPO and PPO: Disentangling Best Practices for Learning from Preference Feedback [Ivison et al. 2024]
https://arxiv.org/pdf/2406.09279

Problems with aligning to human preferences

- Mode Collapse
- Fine-tuned models lose entropy compared to original LLM (base model)
- RLHF models confidently output very few variations
- Base models can be better at tasks that require diverse outputs

https://www.lesswrong.com/posts/t9svvNPNmFf5Qa3TA/mysteries-of-mode-collapse

SYSTEM

Q. Are bugs real?

A.

ASSISTANT	Yes, bugs are real.
ASSISTANT	Yes, bugs are real.
ASSISTANT	Yes, bugs are real. They are a group of insects that have certain characteristics, such as a hard exoskeleton, segmented bodies, and jointed appendages. Bugs include insects like beetles, butterflies, ants, and bees.
ASSISTANT	Yes, bugs are real. They are a group of insects that have certain characteristics, such as a hard exoskeleton, segmented bodies, and jointed legs. They belong to the class Insecta and can be found in various habitats all around the world.
ASSISTANT	Yes, bugs are real. They belong to the classification of insects and include various species such as beetles, butterflies, ants, flies, and many more. Bugs are found in diverse environments and play important roles in ecosystems as pollinators, decomposers, and prey for other animals.
ASSISTANT	Yes, bugs are real. Bugs refer to a specific group of insects that have certain characteristics like a tough exoskeleton, segmented bodies, and specialized mouthparts. They play important roles in ecosystems as pollinators, decomposers, and as a food source for other animals.

RLHF vs. Base LM

- Labelers significantly prefer InstructGPT outputs over outputs from GPT-3
- InstructGPT models show improvements in truthfulness over GPT-3 (on the Truthful QA task)
- InstructGPT shows small improvements in toxicity over GPT-3, but not bias (on the RealToxicityPrompts dataset)
- Can minimize performance regressions on public NLP datasets by modifying our RLHF fine-tuning procedure (by mixing in the pretrained distribution)

Training language models to follow instructions with human feedback [Ouyang et al. 2022]

RLHF vs. Base LM

- GPT3.5 models generalize to the preferences of “held-out” labelers that did not produce any training data
- Public NLP datasets are not reflective of how language models are used
- InstructGPT models show promising generalization to instructions outside of the RLHF fine- tuning distribution
- InstructGPT still makes simple mistakes

Training language models to follow instructions with human feedback [Ouyang et al. 2022]