南方科技大学

MAT8034: Machine Learning

RL in LLMs

Fang Kong

https://fangkongx.github.io/Teaching/MAT8034/Spring2026/index.html

Slide credits: Emma Brunskill, Stanford CS234

Outline

- Basic: preference model
- RLHF
- DPO

Basic: preference model

Human Input to Train RL Agents

- If we want RL agents that can match human performance and/or human values
- How to provide helpful human feedback?

Pairwise Comparisons

- Often easier for people to make than hand writing a reward function
- Often easier than providing scalar reward (how much do you like this answer?)

Bradley-Terry Model (1952)

- First consider simpler setting of k-armed bandits: K actions $b_1, b_2, \ldots b_k$. No state transition
- Assume a human makes noisy pairwise comparisons, where the probability she prefers $b_i \succ b_j$ is

$$P(b_i \succ b_j) = \frac{\exp(r(b_i))}{\exp(r(b_i)) + \exp(r(b_j))} = p_{ij}$$

$$= \sigma\left(r(b_i) - r(b_j)\right)$$

- Transitive: p_{ik} is determined from p_{ij} and p_{jk}

Which one is the best?

- **Condorcet Winner**

  An item b_i is a **Condorcet winner** if

  $$P(b_i \succ b_j) \geq 0.5, \quad \forall\, j \in [K],\ j \neq i.$$

Copeland Winner

A **Copeland winner** is defined as

$$\arg\max_{i \in [K]} \sum_{j \neq i} \mathbf{1}\{P(b_i \succ b_j) > 0.5\}.$$

Borda Winner

A **Borda winner** is defined as

$$\arg\max_{i \in [K]} \sum_{j \neq i} P(b_i \succ b_j).$$

Under the BT model, three winners are consistent

Train the Bradley-Terry Model: The bandit setting

- Assume have N tuples of form (b_i, b_j, μ) where $\mu = 1$ if the human marked $b_i \succ b_j$, $\mu = 0.5$ if the human marked $b_i = b_j$, else 0 if $b_i \prec b_j$
- Maximize likelihood with cross entropy

$$loss = -\sum_{(b_i, b_j, \mu) \in \mathcal{D}} \mu \quad \log P(b_i \succ b_j) + (1 - \mu \quad) \log P(b_j \succ b_i)$$

Recall logistic regression

Let's write the Likelihood function. Recall:

$$P(y = 1 \mid x; \theta) = h_\theta(x)$$
$$P(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

Then,

$$L(\theta) = P(y \mid X; \theta) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta)$$

$$= \prod_{i=1}^{n} h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1 - y^{(i)}}$$ exponents encode "if-then"

Taking logs to compute the log likelihood $\ell(\theta)$ we have:

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))$$

Train the Bradley-Terry Model: The RL setting

- Can also do this for trajectories

  Consider two trajectories, $\tau^1(s_0, a_7, s_{14}, \ldots)$ and $\tau^2(s_0, a_6, s_{12}, \ldots)$

  Let $R^1 = \sum_{i=0}^{T-1} r_i^1$ be the (latent, unobserved) sum of rewards for trajectory τ^1 and similarly for R^2.

  Define the probability that a human prefers $\tau^1 \succ \tau^2$ as

$$\hat{P}\left[\tau^1 \succ \tau^2\right] = \frac{\exp \sum_{i=0}^{t-1} r_i^1}{\exp \sum_{i=0}^{t-1} r_i^1 + \exp \sum_{i=0}^{t-1} r_i^2},$$

- Use learned reward model, and do PPO with this model

RLHF

RLHF pipeline

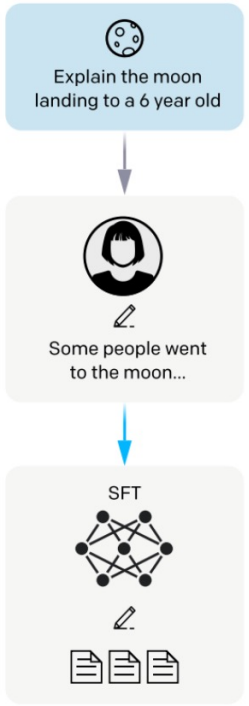
Step 1

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.



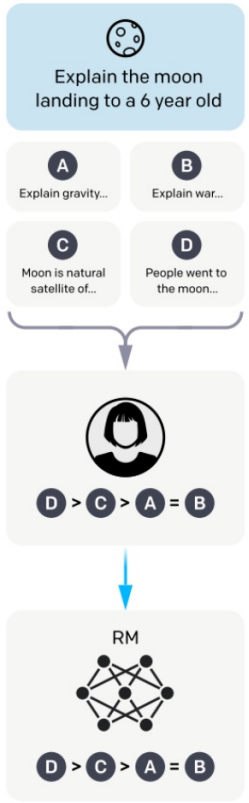
Step 2

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

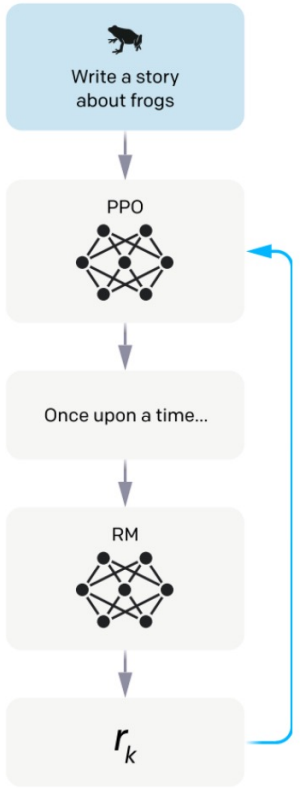
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

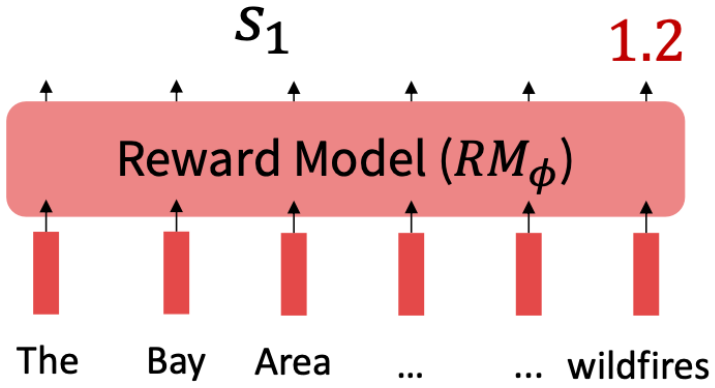
The reward is used to update the policy using PPO.



How to model human preferences?

- Human judgments are noisy and miscalibrated!
- Solution: instead of asking for direct ratings, ask for pairwise comparisons, which can be more reliable

An earthquake hit San Francisco. There was minor property damage, but no injuries. > A 4.2 magnitude earthquake hit San Francisco, resulting in massive damage. > The Bay Area has good weather but is prone to earthquakes and wildfires.



s_3 s_2

Bradley-Terry [1952] paired comparison model

$$J_{RM}(\phi) = -\mathbb{E}_{(s^w, s^l)\sim D}\left[\log \sigma(RM_\phi(s^w) - RM_\phi(s^l))\right]$$

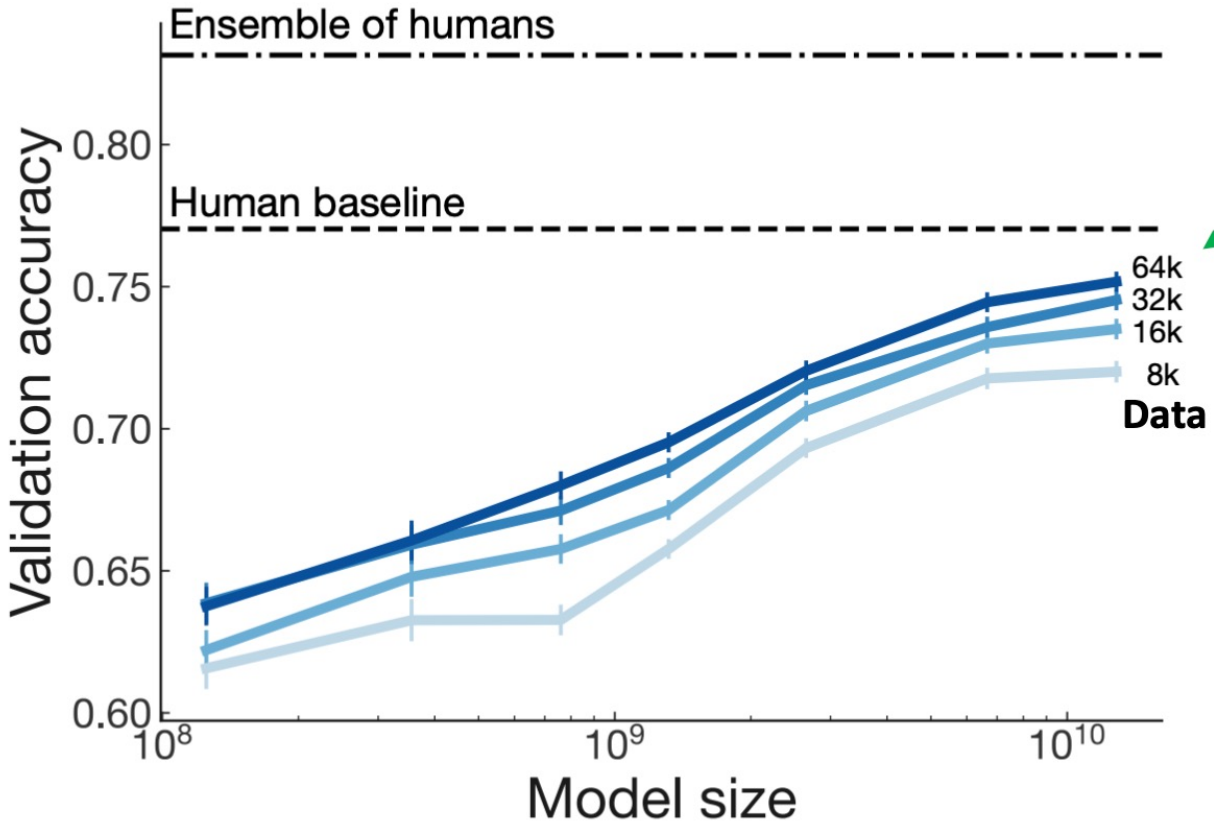
"winning" sample

"losing" sample

s^w should score higher than s^l

Reward model works

- Evaluate RM on predicting outcome of held-out human judgments



Large enough RM trained on enough data approaching single human perf

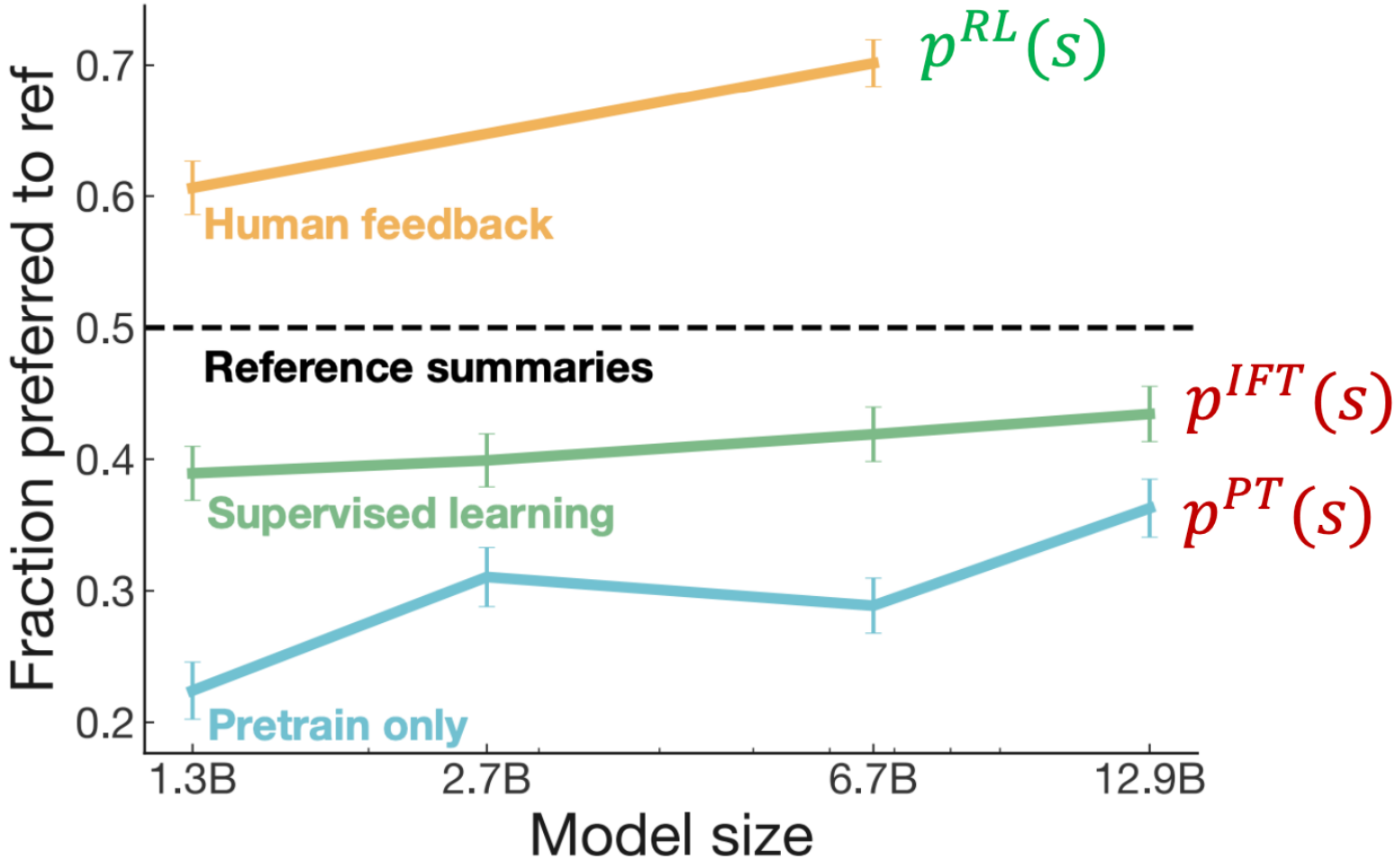
[Stiennon et al., 2020]

RLHF: putting it all together

- Finally, we have:
  - A pretrained (possibly instruction-finetuned) reference model π_{ref}
  - A reward model r_ϕ that produces scalar rewards for LM outputs
  - A method for optimizing LM parameters towards reward functions
- Now to do RLHF:
  - Initialize a copy of the model π_θ with parameters θ to optimize
  - Optimize the following reward with RL

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

RLHF provides gains over pretraining + finetuning



Controlled comparisons of “RLHF” style algorithms

Method	Simulated win-rate (%)	Human win-rate (%)
GPT-4	79.0 ± 1.4	69.8 ± 1.6
ChatGPT	61.4 ± 1.7	52.9 ± 1.7
PPO	46.8 ± 1.8	55.1 ± 1.7
Best-of-n	45.0 ± 1.7	50.7 ± 1.8
Expert Iteration	41.9 ± 1.7	45.7 ± 1.7
SFT 52k (Alpaca 7B)	39.2 ± 1.7	40.7 ± 1.7
SFT 10k	36.7 ± 1.7	44.3 ± 1.7
Binary FeedME	36.6 ± 1.7	37.9 ± 1.7
Quark	35.6 ± 1.7	-
Binary Reward Conditioning	32.4 ± 1.6	-
Davinci001	24.4 ± 1.5	32.5 ± 1.6
LLaMA 7B	11.3 ± 1.1	6.5 ± 0.9

- Many works study RLHF behaviors using GPT-4 feedback (**Simulated**) as a surrogate for **Human** feedback.
- PPO (method in InstructGPT) does work
- Simple baselines (Best-of-n, Training on ‘good’ outputs) works well too

[Dubois et al 2023]

DPO

RLHF: preference->reward->policy

- RLHF
  - First train the reward model by minimizing negative log likelihood:

$$J_{RM}(\phi) = -\mathbb{E}_{(s^w, s^l)\sim D}\left[\log \sigma(RM_\phi(s^w) - RM_\phi(s^l))\right]$$

  - Then learn a policy that maximizes the reward (with small distance to the pretrained model)

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

- Can we remove the reward step?

Direct Preference Optimization

RLHF Objective

(get **high reward**, stay **close** to reference model)

any reward function

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi(y|x)} \left[r(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}(\pi(\cdot \mid x)\|\pi_{\mathrm{ref}}(\cdot \mid x))$$

Closed-form Optimal Policy

(write **optimal policy** as function of **reward function**; from prior work)

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

with $Z(x) = \sum_y \pi_{\mathrm{ref}}(y \mid x)\exp\left(\frac{1}{\beta}r(x,y)\right)$

Note **intractable sum** over possible responses; can't immediately use this

Rearrange

(write **any reward function** as function of **optimal policy**)

Ratio is **positive** if policy likes response more than reference model, **negative** if policy likes response less than ref. model

$$r(x,y) = \underbrace{\beta\log\frac{\pi^*(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)} + \beta\log Z(x)}_{\text{some parameterization of a reward function}}$$

DPO: Putting it together

A loss function on reward functions

+

A transformation between reward functions and policies

=

A loss function on policies

Derived from the Bradley-Terry model of human preferences:

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

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

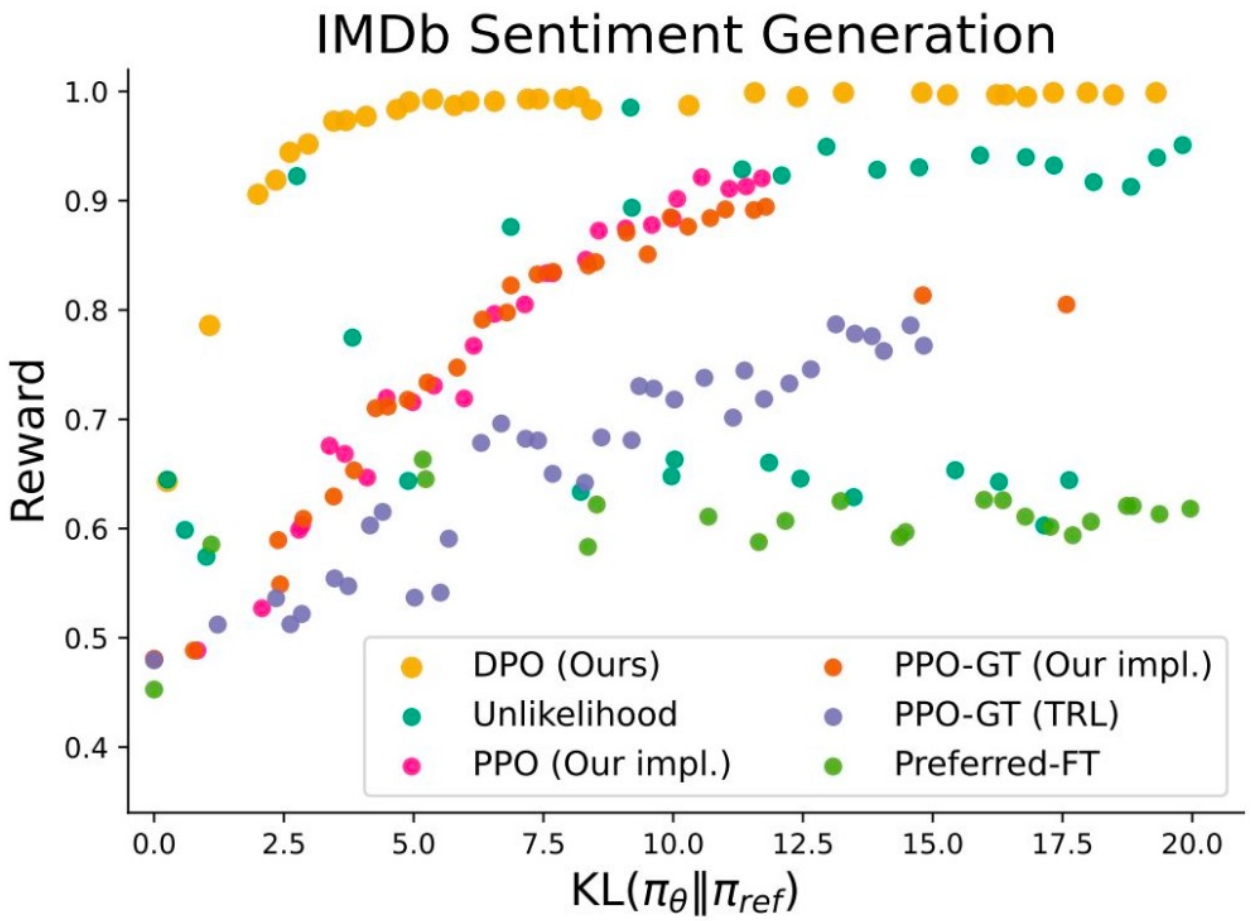
When substituting, the **log Z term cancels**, because the loss only cares about **difference** in rewards

Reward of **preferred** response

Reward of **dispreferred** response

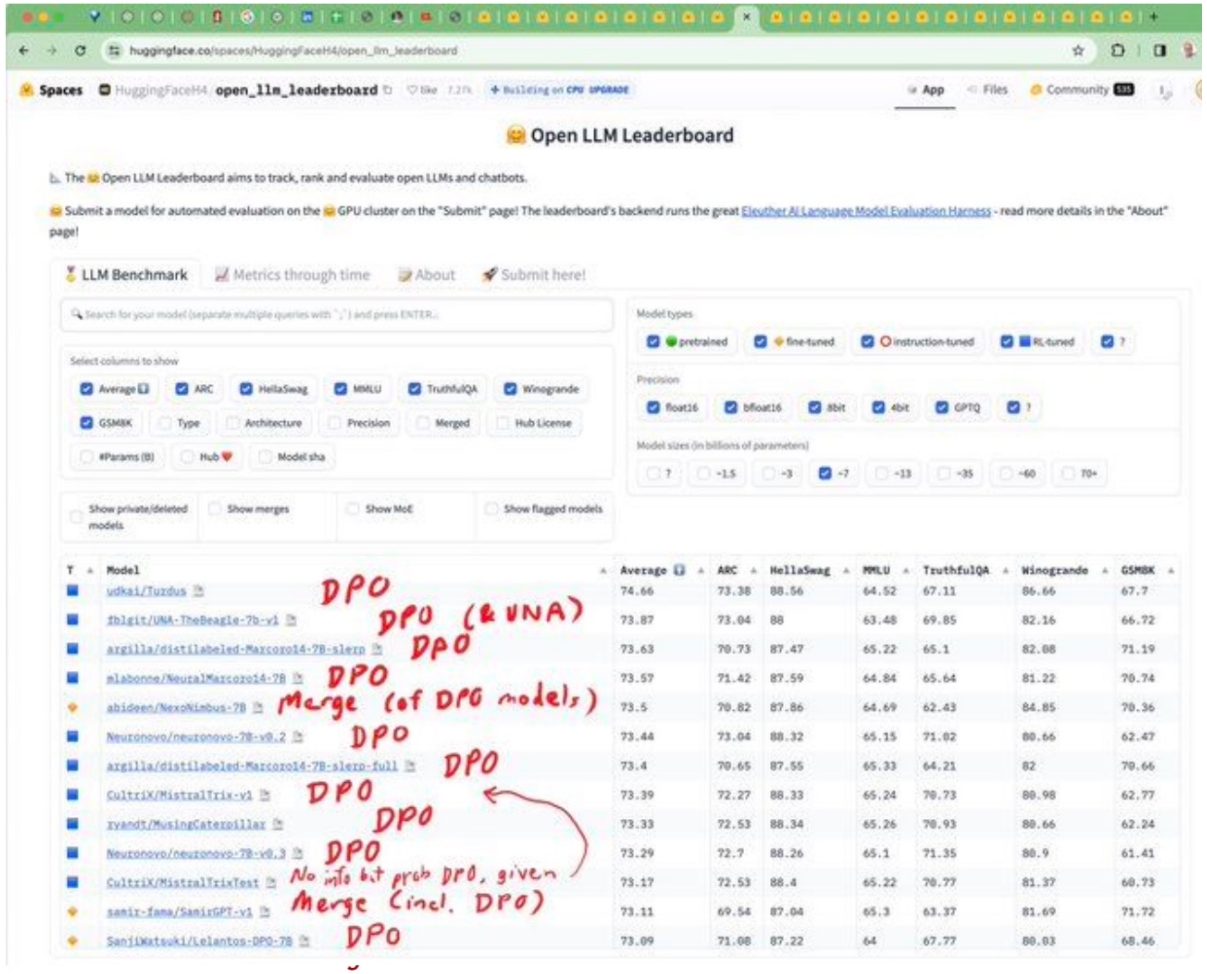
$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]$$

Trade off Reward & KL



1. Generate positive IMDB reviews from GPT2-XL
2. Use pre-trained sentiment classifier as Gold RM
3. Create preferences based on Gold RM
4. Optimize with PPO and DPO

Models Trained With DPO



Instruction fine-tuning

To fully unlock the potential of our pretrained models in chat use cases, we innovated on our approach to instruction-tuning as well. Our approach to post-training is a combination of supervised fine-tuning (SFT), rejection sampling, proximal policy optimization (PPO), and direct preference optimization (DPO). The quality of the prompts that are used in SFT and the preference rankings that are used in PPO and DPO has an outsized influence on the performance of aligned models. Some of our biggest improvements in model quality came from carefully curating this data and performing multiple rounds of quality assurance on annotations provided by human annotators.

LLaMa3

Learning from preference rankings via PPO and DPO also greatly improved the performance of Llama 3 on reasoning and coding tasks. We found that if you ask a model a reasoning question that it struggles to answer, the model will sometimes produce the right reasoning trace: The model knows how to produce the right answer, but it does not know how to select it. Training on preference rankings enables the model to learn how to select it.

4 Instruction Fine-tuning

We train Mixtral – Instruct using supervised fine-tuning (SFT) on an instruction dataset followed by Direct Preference Optimization (DPO) [25] on a paired feedback dataset. Mixtral – Instruct reaches a score of 8.30 on MT-Bench [33] (see Table 2), making it the best open-weights model as of December 2023. Independent human evaluation conducted by LMSys is reported in Figure 6[3] and shows that Mixtral – Instruct outperforms GPT-3.5-Turbo, Gemini Pro, Claude-2.1, and Llama 2 70B chat.

Mistral

Model	Arena Elo rating	MT-bench (score)	License
GPT-4-Turbo	1243	9.32	Proprietary
GPT-4-0314	1192	8.96	Proprietary
GPT-4-0613	1158	9.18	Proprietary
Claude-1	1149	7.9	Proprietary
Claude-2.0	1131	8.06	Proprietary
Mixtral-8x7b-Instruct-v0.1	1121	8.3	Apache 2.0
Claude-2.1	1117	8.18	Proprietary
GPT-3.5-Turbo-0613	1117	8.39	Proprietary
Gemini Pro	1111		Proprietary
Claude-Instant-1	1110	7.85	Proprietary
Tulu-2-DPO-70B	1110	7.89	AI2 ImpACT Low-risk
Yi-34B-Chat	1110		Yi License
GPT-3.5-Turbo-0314	1105	7.94	Proprietary
Llama-2-70b-chat	1077	6.86	Llama 2 Community

Figure 6: LMSys Leaderboard. (Screenshot from Dec 22, 2023) Mixtral 8x7B Instruct v0.1 achieves an Arena Elo rating of 1121 outperforming Claude-2.1 (1117), all versions of GPT-3.5-Turbo (1117 best), Gemini Pro (1111), and Llama-2-70b-chat (1077). Mixtral is currently the best open-weights model by a large margin.

GRPO

GRPO in deepseek

Group Relative Policy Optimization In order to save the training costs of RL, we adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which foregoes the critic model that is typically the same size as the policy model, and estimates the baseline from group scores instead. Specifically, for each question q, GRPO samples a group of outputs $\{o_1, o_2, \cdots, o_G\}$ from the old policy $\pi_{\theta_{old}}$ and then optimizes the policy model π_θ by maximizing the following objective:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right), \tag{1}$$

$$\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1, \tag{2}$$

where ε and β are hyper-parameters, and A_i is the advantage, computed using a group of rewards $\{r_1, r_2, \ldots, r_G\}$ corresponding to the outputs within each group:

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{\text{std}(\{r_1, r_2, \cdots, r_G\})}. \tag{3}$$