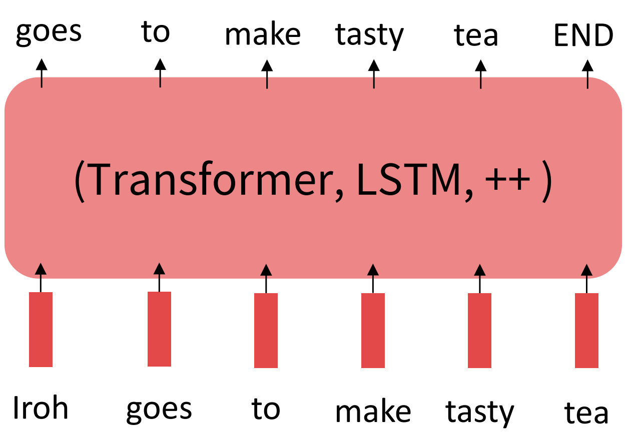
Lecture 16: More on Reinforcement Learning from Human Feedback (RLHF)

Recall: Pretraining / finetuning paradigm

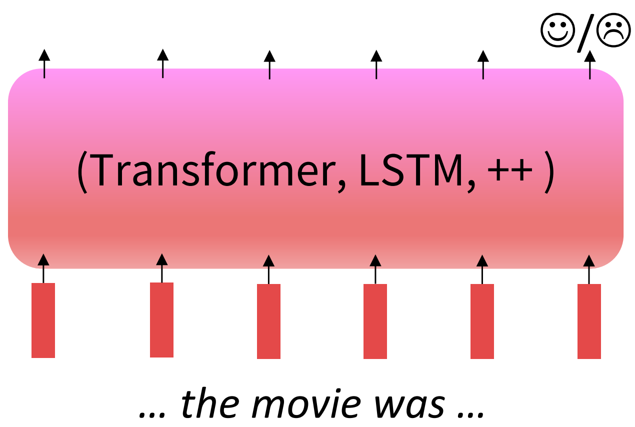
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



Step 2: Finetune (on your task)

Not many labels; adapt to the task!



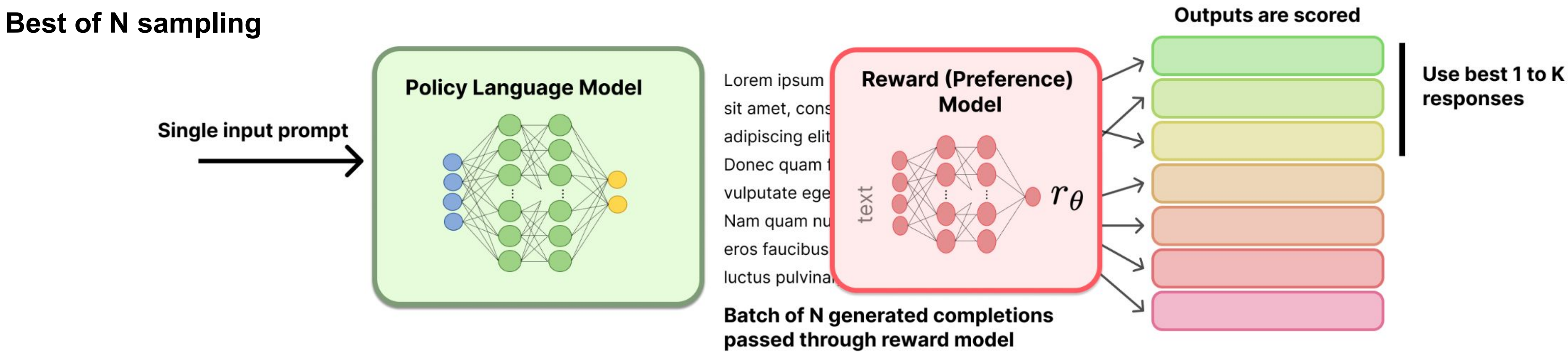
Supervised fine-tuning (SFT) - aka imitation learning!

- We have (s, a) pairs, where s is a prompt and a is a generation corresponding to that prompt (consisting of several tokes
- These are taken from already existing data (eg internet docs, QA, solved problems...)
- Train a policy $\pi_{\textsf{sft}}$ that maximizes the likelihood of the observed data:

$$J_{\textsf{sft}}(\theta) = \mathbb{E}_{(s,a)\sim P_{\textsf{sft}}}\left[\frac{1}{|a|}\sum_{k=1}^{|a|}\log \pi_\theta(a_k|s, a_{i<k})\right]$$

  Training is done by gradient ascent
- Aka teacher forcing

Better version: Rejection sampling (aka Best-of-N)



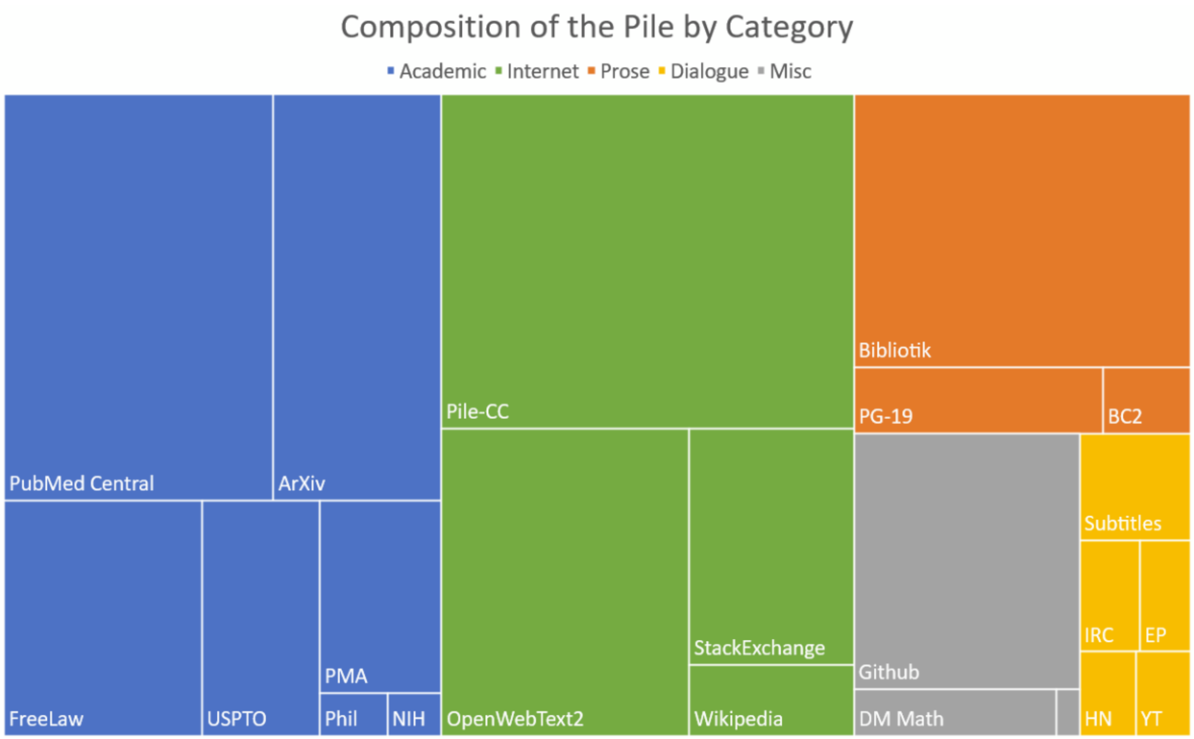
- Generate N answers from the model, reinforce the correct/top one(s)
- Train a policy π_{sft} that maximizes the likelihood of the *top data*:

$$J_{\text{rft}}(\theta) = \mathbb{E}_{s \sim P_{\text{sft}}, a \sim \pi_{\text{sft}}(\cdot|s)} \left[\frac{1}{|a|} \mathbf{1}_{a \text{ is at the top}} \sum_{k=1}^{|a|} \log \pi_\theta(a_k | s, a_{i<k}) \right]$$

Training is done by gradient ascent

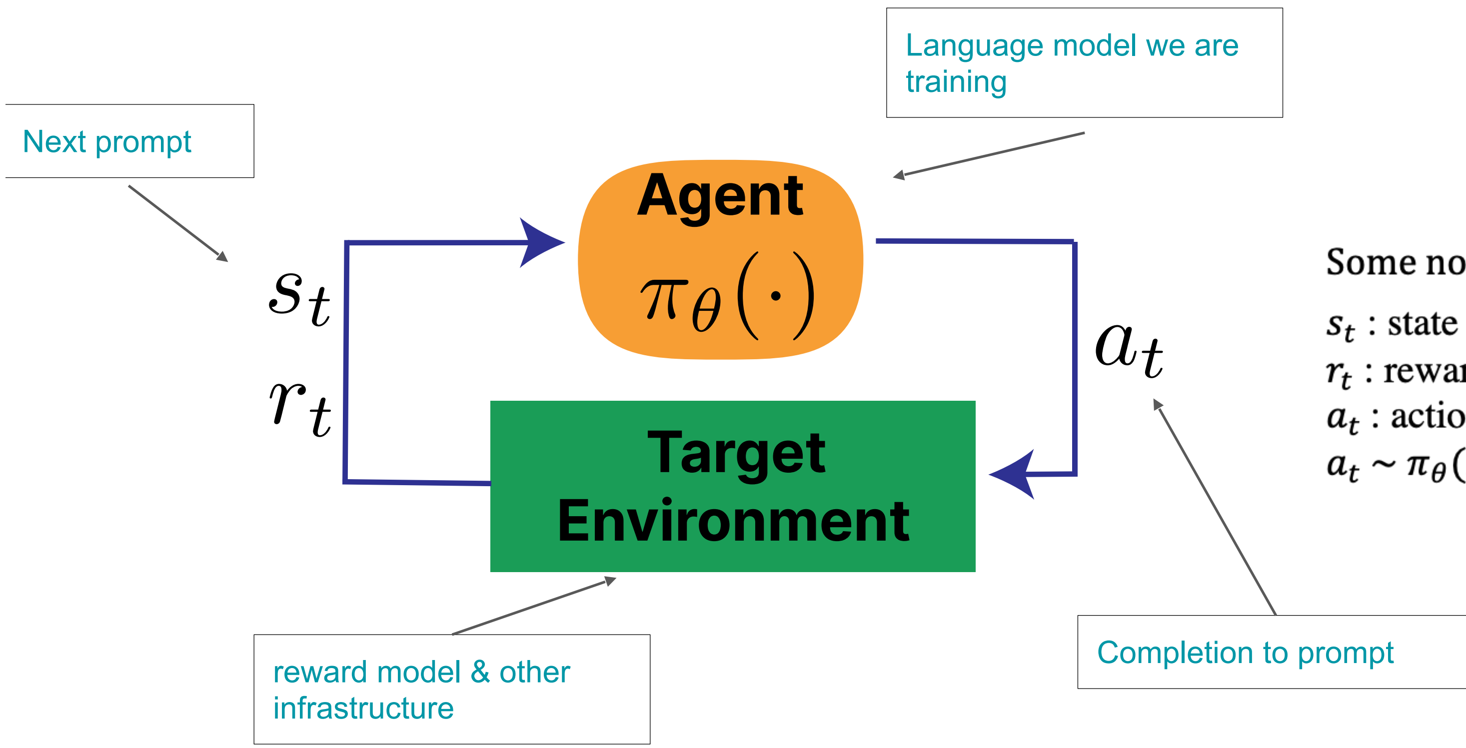
- *Online* rejection sampling finetuning: $a \sim \pi_\theta$ instead of $a \sim \pi_{\text{sft}}$

Recall: Data is critical!!!



Model	Training Data
BERT	BookCorpus, English Wikipedia
GPT-1	BookCorpus
GPT-3	CommonCrawl, WebText, English Wikipedia, and 2 book databases (“Books 1” and “Books 2”)
GPT-3.5+	Undisclosed

RL comes typically after SFT

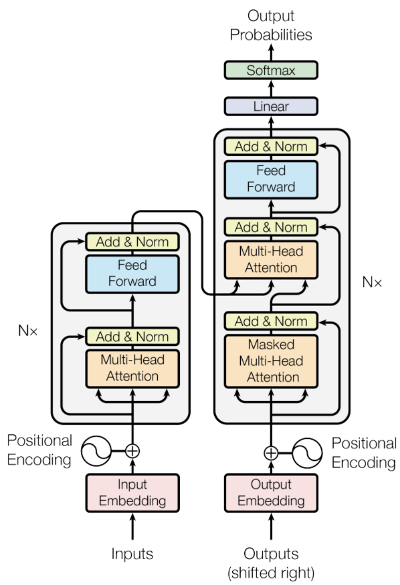


Some notation:

s_t : state
r_t : reward
a_t : action
$a_t \sim \pi_\theta(s_t)$: policy

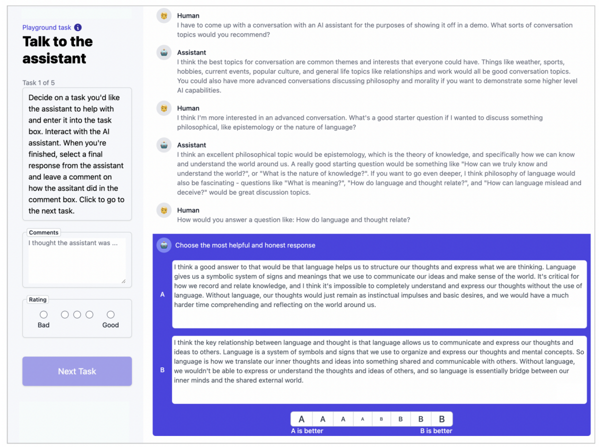
RLHF training phases

base model (instruction, helpful, chatty etc.)

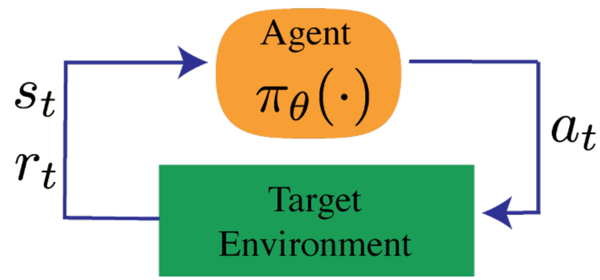


Vaswani et al. 2017

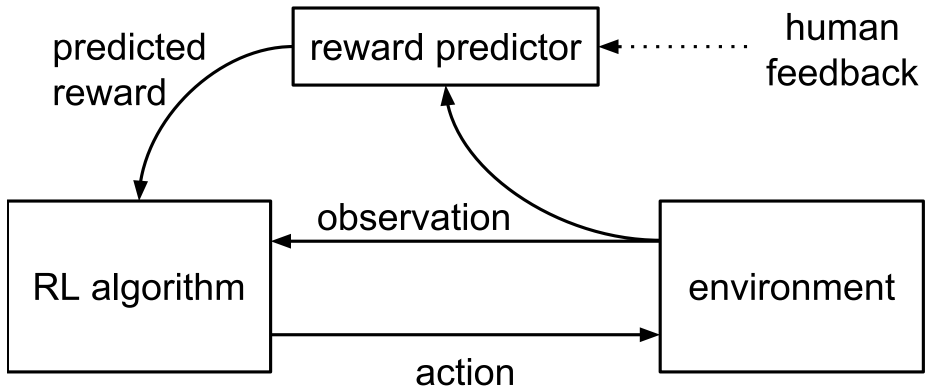
preference collection & training



RL optimization



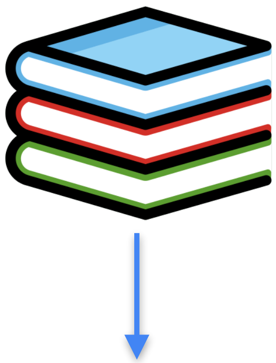
Recall: Deep RL from Human Feedback (Christiano et al, 2017)



- People provide a *preference* among two choices
- Assuming there is a latent variable explaining the choice, reward is fit using maximum likelihood (Bradley-Terry model)
- Cf. https://arxiv.org/pdf/1706.03741.pdf

RLHF early attempts

Summarization



“Three pigs defend themselves from a mean wolf”

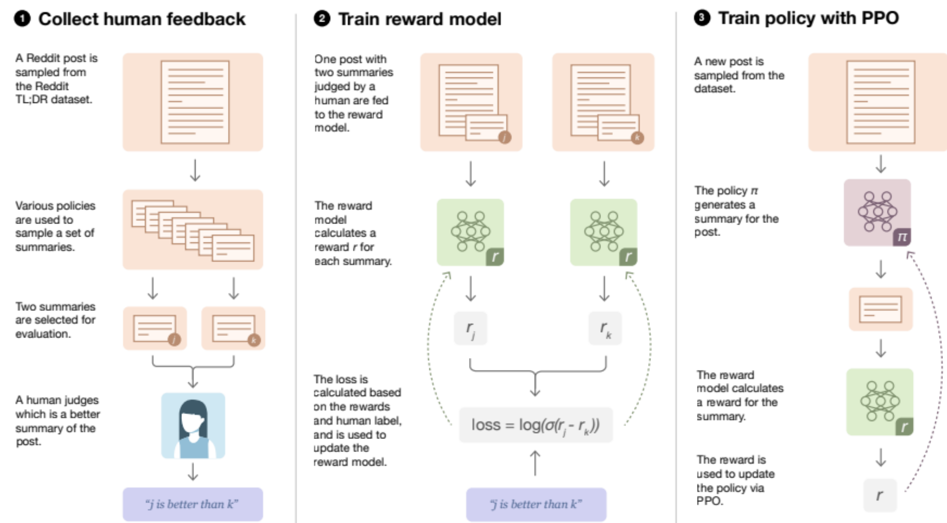


Figure 2: Diagram of our human feedback, reward model training, and policy training procedure.

Stiennon, Nisan, et al. "Learning to summarize with human feedback." *2020.*

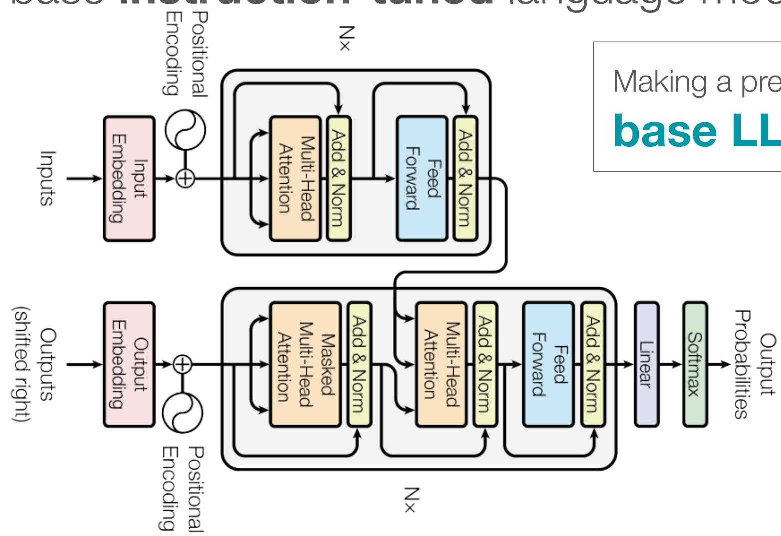
Model structure

starting point: a base **instruction-tuned** language model

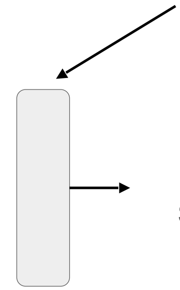
Making a preference model:
base LLM with new final layer

input:

prompt+completion



The Transformer - Vaswani et al. 2017



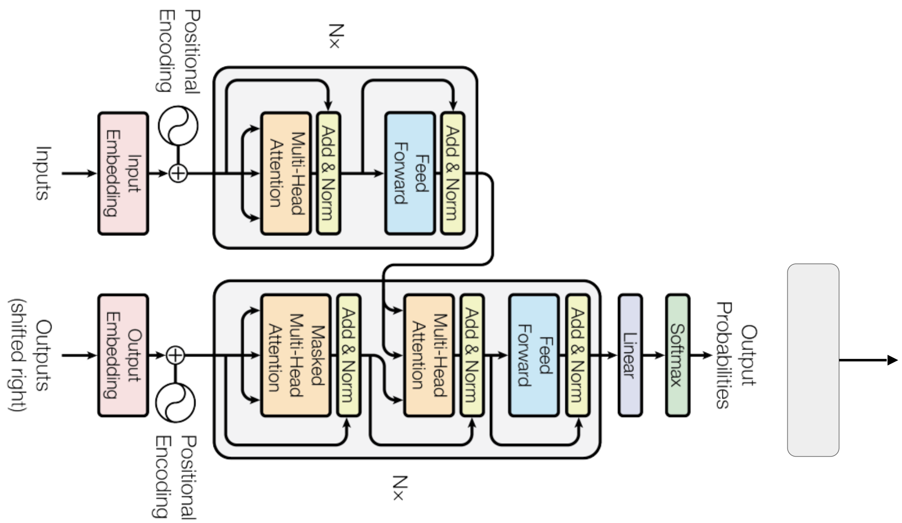
output:
scalar rewards

Training a reward model

input pair:

selected prompt +completion

rejected prompt +completion



The Transformer - Vaswani et al. 2017

output:
scalar rewards

loss: increase difference of predicted reward

Bradely-Terry reward model

- Collect data from human raters (pairs of y_w, y_l responses to a prompt x)
- Optimize the expected value of:

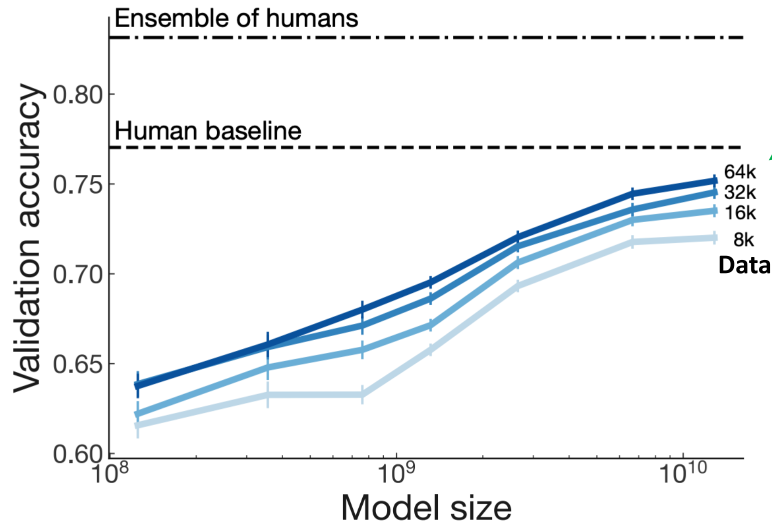
$$-\log(\sigma(r_\theta(x, y_w) - r_\theta(x, y_l)))$$

  wrt reward parameter vector θ

- Cf. Ouyang et al, InstructGPT
- Corresponds to maximum likelihood fitting of binomial preference function if reward is linear over the variables

Evaluating the reward model

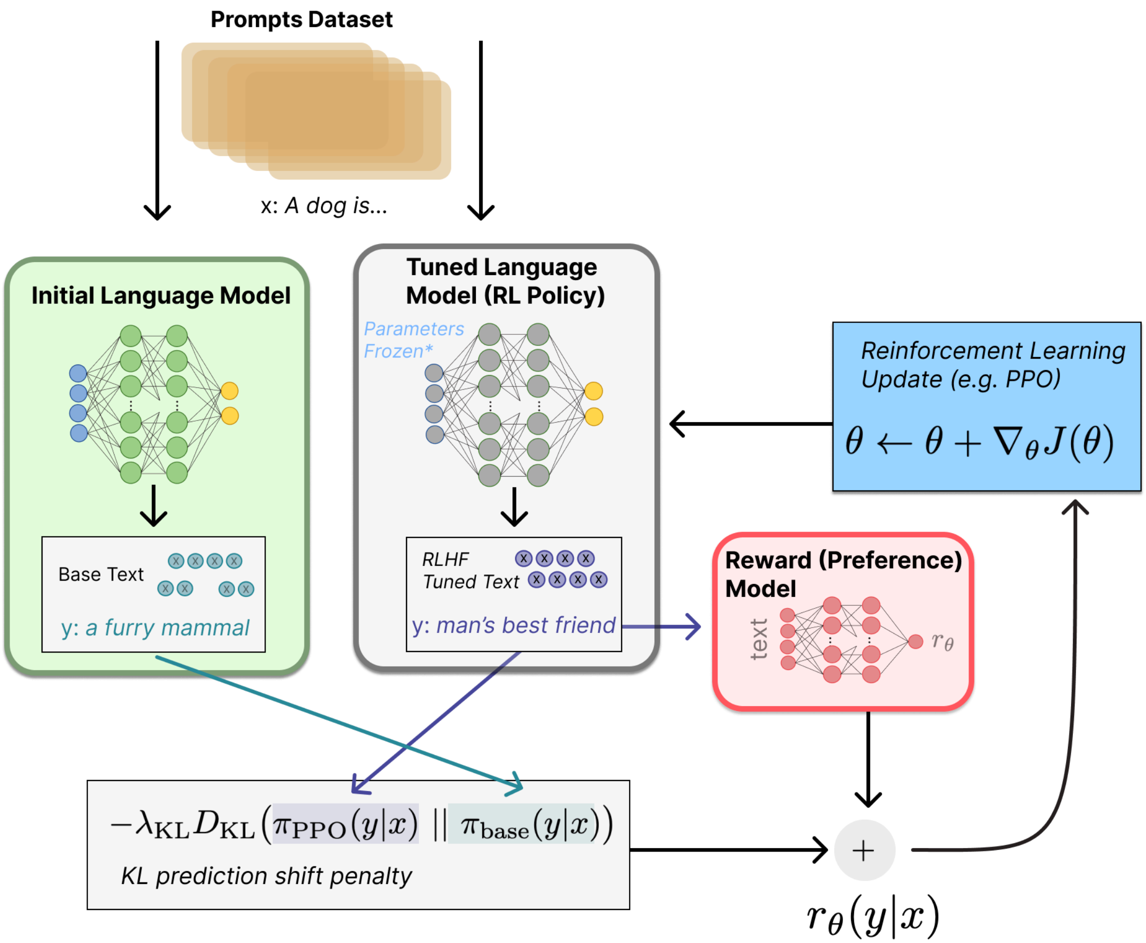
Evaluate RM on predicting outcome of held-out human judgments



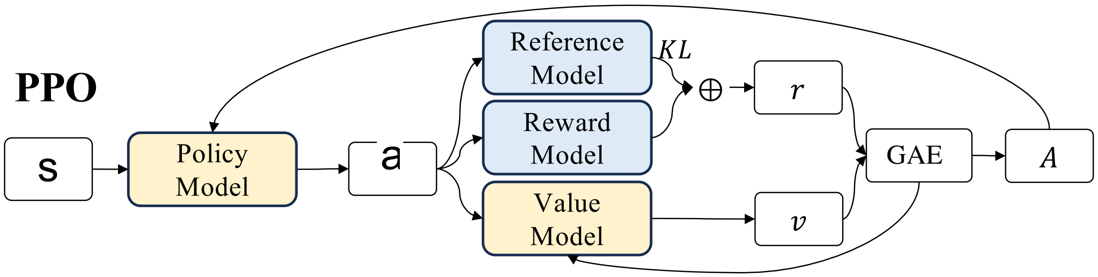
Large enough RM trained on enough data approaching single human perf

[Stiennon et al., 2020]

RLHF finetuning



PPO for RLHF

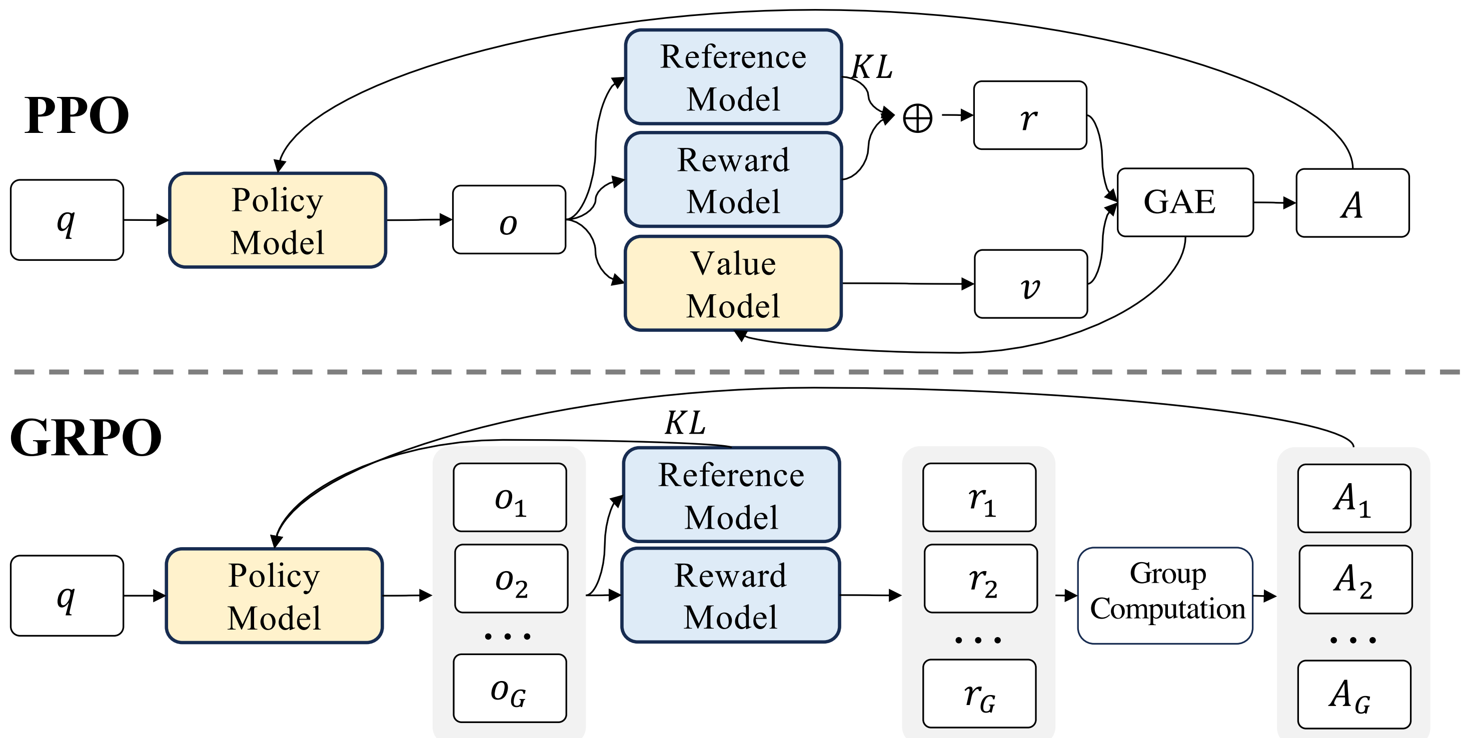


- Train a policy π_θ that maximizes advantage:

$$J_{\mathsf{PPO}}(\theta) = \mathbb{E}_{s\sim P_{\mathsf{sft}}, a\sim\pi_{\theta_{\mathsf{old}}}(\cdot|s)}\left[\frac{1}{|a|}\sum_{k=1}^{|a|}\frac{\pi_\theta(a_k|s, a_{i<k})}{\pi_{\theta_{\mathsf{old}}}(a_k|s, a_{i<k})}A_i\right]$$

  where A_i is the advantage function
- Reward function uses a penalty per token for straying from reference policy: $r_t = r_\phi(s, a_{<t}) - \beta\log\frac{\pi_\theta(a_t|s,a_{<t})}{\pi_{\mathsf{sft}}(a_t|s,a_{<t})}$
- Value function/advantage needs to be estimated!

GRPO (DeepSeek, 2025)



Instead of estimating value, use a group (non-parametric approach)

Notation: $q = s, o = a$

GRPO Objective (DeepSeek, 2025)

- Generate G answers and estimate their reward (no regularization towards reference policy)
- Compute a normalized advantage based on the mean $\bar{r}_t$ and standard deviation of the rewards:

$$\hat{A}_{i,t} = \frac{r_{i,t} - \bar{r}_t}{std(r_{1,t}, \ldots r_{G,t})}$$

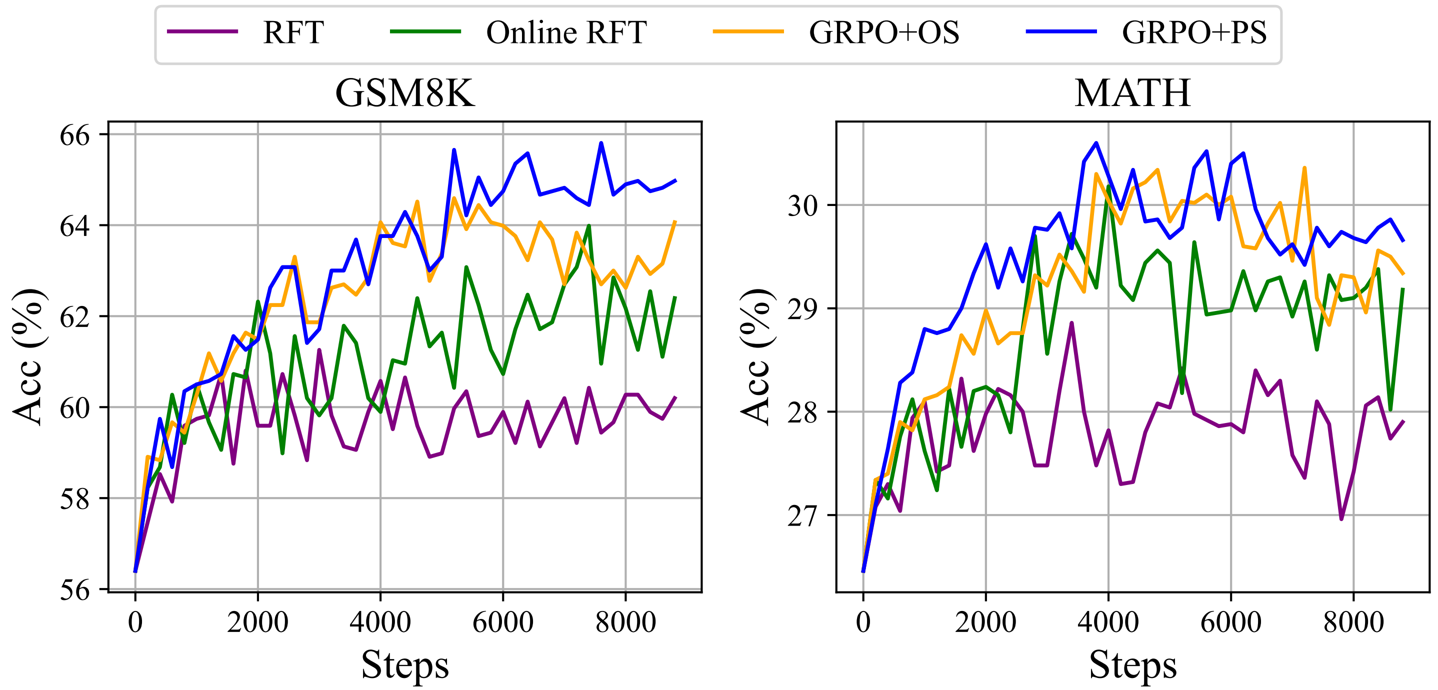
- GRPO objective - very similar to PPO!

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{s \sim P_{\text{sft}}, a_i \sim \pi_{\theta_{\text{old}}}(\cdot|s), i=1,\ldots G} \left[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|a_i|} \sum_{k=1}^{|a_i|} \frac{\pi_\theta(a_k|s, a_{i<k})}{\pi_{\theta_{\text{old}}}(a_k|s, a_{i<k})} \hat{A}_{i,t} - \beta D_{KL}(\pi_\theta, \pi_{\theta_{\text{old}}}) \right]$$

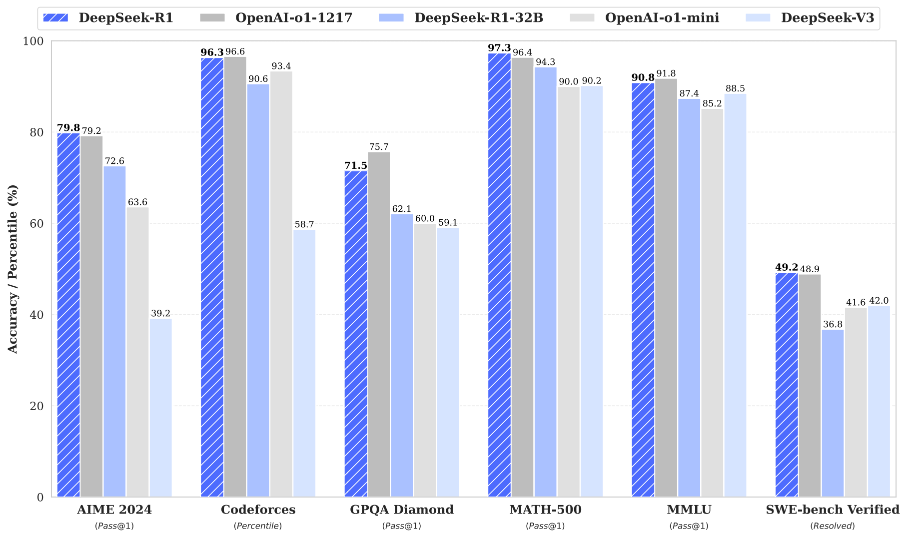
- D_{KL} is also estimated a bit differently (cf Shulman et al, 2020):

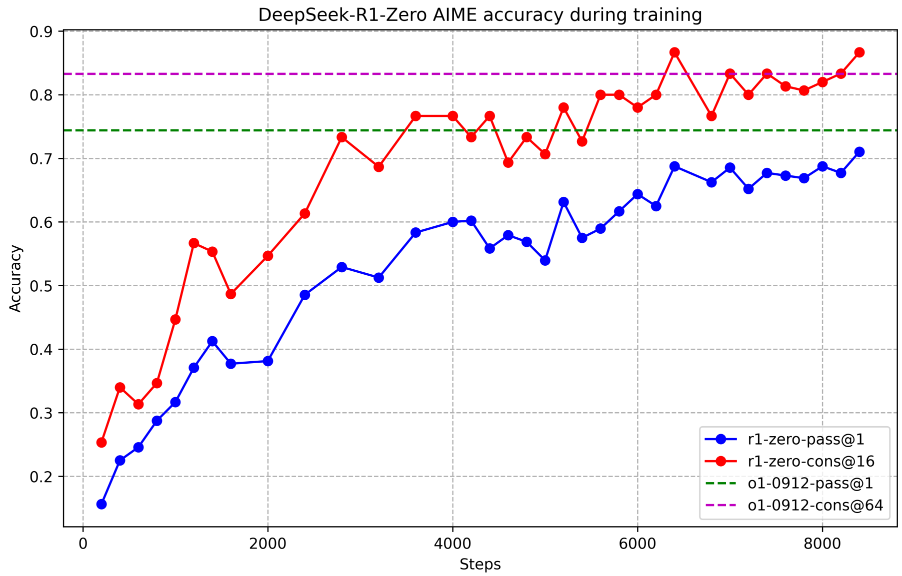
$$D_{KL}(\pi_\theta, \pi_{\theta_{\text{old}}}) = \frac{\pi_{\theta_{\text{old}}}(a_k|s, a_{i<k})}{\pi_\theta(a_k|s, a_{i<k})} - \log \frac{\pi_{\theta_{\text{old}}}(a_k|s, a_{i<k})}{\pi_\theta(a_k|s, a_{i<k})} - 1$$

The Advantage of RL over SFT (DeepSeek, 2025)



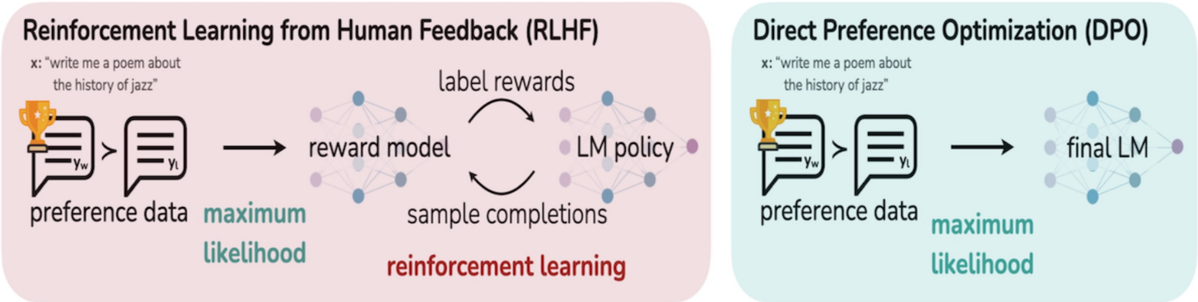
DeepSeek Overall Results (DeepSeek, 2025)





SOTA results back in January 2025

Direct Preference Optimization



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

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

$$\hat{r}_\theta(x,y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$$

- You can replace the complex RL part with a very simple weighted MLE objective
- Other variants (KTO, IPO) now emerging too

[Rafailov+ 2023]

Learning with non-transitive preferences: NashLLM

- Objective:find a policy π^* which is preferred over any other policy

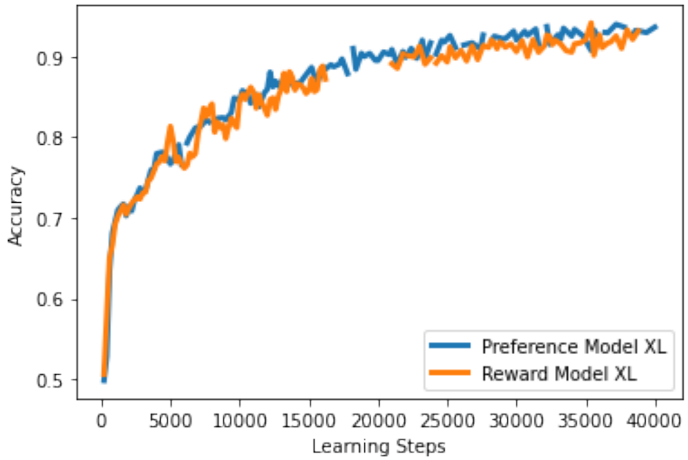
$$\pi^* = \arg\max_{\pi} \min_{\pi'} \mathbb{P}(\pi' \preceq \pi)$$

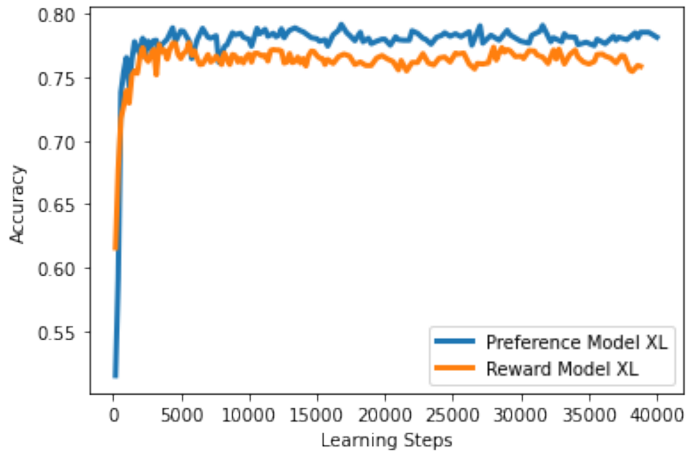
- Think of this as a game: one player picks π the other picks π'
- When both players use π^* this is a *Nash equilibrium* for the game
- For this game an equilibrium exists (even if eg preferences are not transitive)
- Cf. Munos et al, 2024 (https://arxiv.org/pdf/2312.00886.pdf)

NashLLM-style algorithms

- Fit a *two-argument preference function* by supervised learning
- Decide what is the *set of opponent policies*
- Ideally, the max player should play against a mixture of past policies
- *Optimize* using eg online mirror descent, convex-concave optimization...
- A lot of algorithmic variations to explore!

NashLLM results





Using preferences instead of rewards leads to less overfitting

General blueprint of RLHF training

Finally, we have everything we need:

- A pretrained (possibly instruction-finetuned) LM $p^{PT}(s)$
- A reward model $RM_\phi(s)$ that produces scalar rewards for LM outputs, trained on a dataset of human comparisons
- A method for optimizing LM parameters towards an arbitrary reward function.

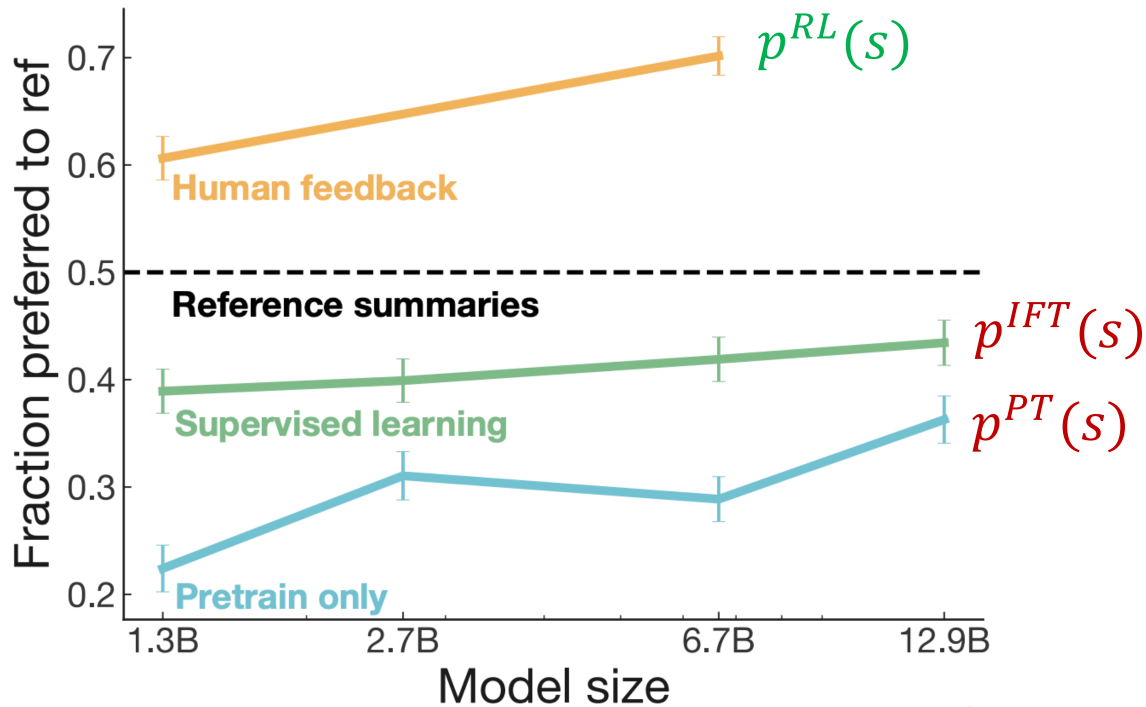
Now to do RLHF:

- Initialize a copy of the model $p_\theta^{RL}(s)$, with parameters θ we would like to optimize
- Optimize the following reward with RL:

$$R(s) = RM_\phi(s) - \underbrace{\beta \log \left(\frac{p_\theta^{RL}(s)}{p^{PT}(s)} \right)}_{\substack{\text{This is a penalty which prevents us from diverging too far from} \\ \text{the pretrained model. In expectation, it is known as the} \\ \textbf{Kullback-Leibler}\text{ (}\textbf{KL}\text{) divergence between } p_\theta^{RL}(s) \text{ and } p^{PT}(s).}}$$

Pay a price when $p_\theta^{RL}(s) > p^{PT}(s)$

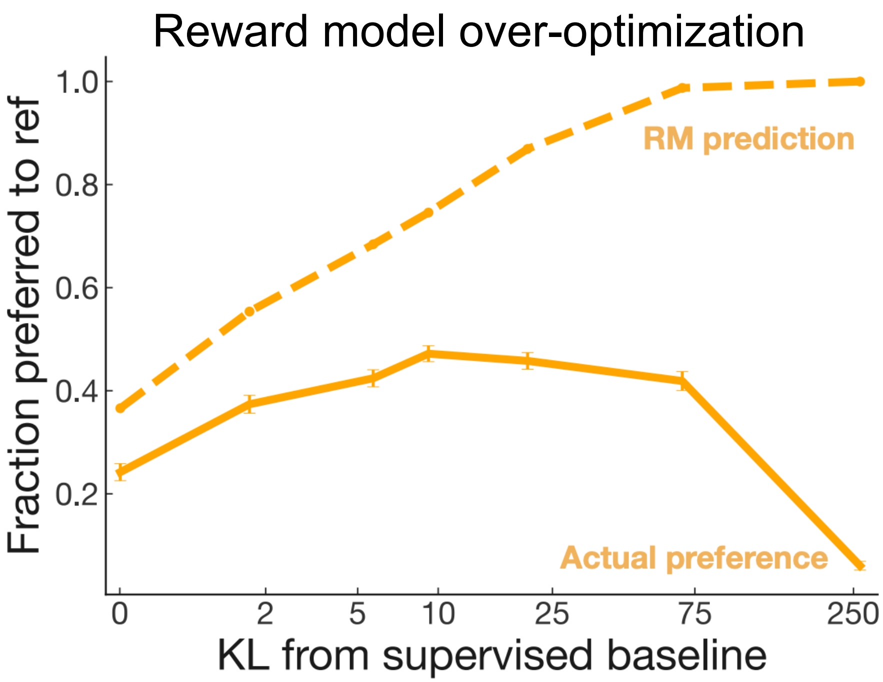
RLHF results



[Stiennon et al., 2020]

Problem: reward hacking

- Human preferences are unreliable!
  - ”Reward hacking” is a common problem in RL
  - Chatbots are rewarded to produce responses that *seem* authoritative and helpful, *regardless of truth*
  - This can result in making up facts + hallucinations
- **Models** of human preferences are *even more* unreliable!



$$R(s) = RM_{\phi}(s) - \beta \log\left(\frac{p_{\theta}^{RL}(s)}{p^{PT}(s)}\right)$$

More important methods

- Self-improvement
- Chain-of-thought prompting
- Distillation from large models to small
- Utilizing more inference time using search (cool new work)

More open directions

- Multi-turm
- Exploration
- .....