Instruction Finetuning and DPO

InstructGPT: https://arxiv.org/pdf/2203.02155.pdf

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

Reading Group 06/25/23
Notes: Adi

Papers

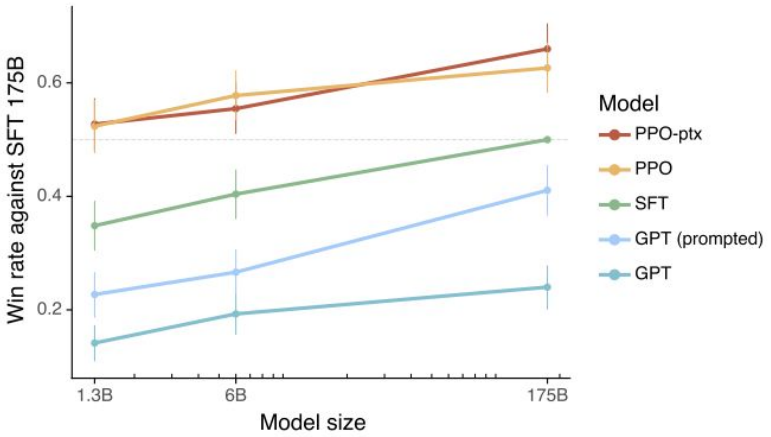
- InstructGPT: https://arxiv.org/pdf/2203.02155.pdf
- DPO: https://arxiv.org/pdf/2305.18290.pdf

InstructGPT

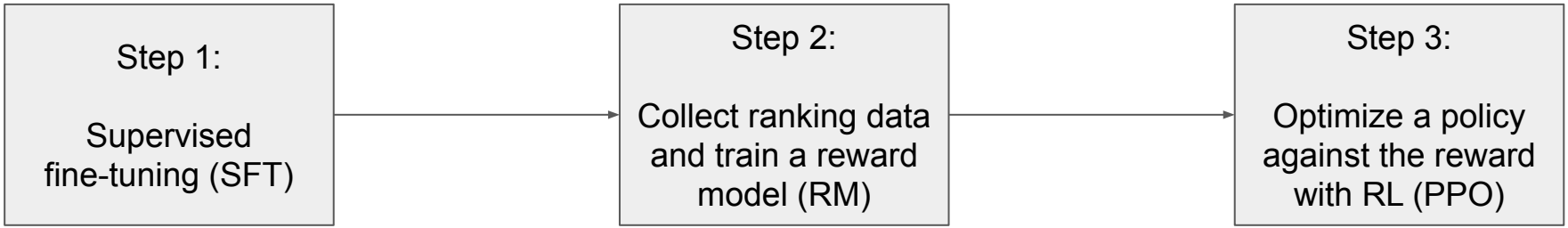
InstructGPT Overview

- Fine-tuning protocol to **align** LLMs with human preferences.
  - Intuition: Predicting the next word as an objective isn't necessarily aligned with what humans **want** from LLMs.

- Eval: Outputs from the 1.3B parameter InstructGPT model are preferred to outputs from the 175B GPT-3 model.



Overview of RLHF



Step 1: Supervised fine-tuning (SFT)

- We sample prompts from our prompt dataset, get sample outputs from labelers, and fine-tune GPT-3 with supervised learning.
  - Used a combination of prompts from past submissions to the OpenAI API and asking labelers to write them.
- Protocol: Train for 16 epochs, cosine learning rate decay, residual dropout of 0.2.

Table 6: Dataset sizes, in terms of number of prompts.

SFT Data			RM Data			PPO Data		
split	source	size	split	source	size	split	source	size
train	labeler	11,295	train	labeler	6,623	train	customer	31,144
train	customer	1,430	train	customer	26,584	valid	customer	16,185
valid	labeler	1,550	valid	labeler	3,488			
valid	customer	103	valid	customer	14,399			

Step 2: Reward-model training (RM)

- Start from the SFT model with the final unembedding layer removed, train a model to take in a prompt and response -> output a scalar reward
- Use 6B-param reward models
  - Found that 175B RM training “could be unstable and thus was less suitable to be used as the value function during RL”
- Loss is below.
  - r_{theta}(x, y) = Scalar output of the reward model for prompt x and completion y.
  - y_w: Preferred completion out of the pair of y_w and y_l.

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta\left(x, y_w\right) - r_\theta\left(x, y_l\right)\right)\right)\right]$$

Step 3: Reinforcement learning (RL)

- Step 3: Fine-tune the SFT model on our environment using **PPO** (proximal policy optimization).
- Environment: Bandit environment which presents a random customer prompt and expects a response to the prompt
- Given the prompt and response, it produces a reward determined by the reward model and ends the episode
- They trained two variants:
- **PPO**
  - Value function initialized from the reward model
  - Per-token KL penalty from the SFT model at each token to mitigate over optimization of the reward model
- **PPO-ptx**
  - Experiment to mix pretraining gradients into the PPO gradients

Breaking down the RL objective

KL penalty to prevent over-optimizing the reward.

Encourages closeness between RL-finetuned model and SFT model.

RL objective

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

Standard log-likelihood loss ("predict the next word")

$r_\theta(x, y)$ Reward model output from prompt x and completion y

π_ϕ^{RL} Learned RL policy

π^{SFT} Supervised trained model

Note: gamma = 0 in the **PPO** models. gamma is nonzero in the **PPO-ptx** models (these models are referred to as the InstructGPT family).

Eval Takeaways (pg. 3-4).

- Labelers significant prefer InstructGPT outputs over GPT-3 outputs, even when comparing 1.3B InstructGPT to 175B GPT-3
- InstructGPT show improvements in truthfulness over GPT-3
- InstructGPT show small improvements in toxicity over GPT-3 but not bias
- We can minimize performance regressions by tuning RLHF procedure

Performance on public datasets

RLHF regresses against GPT on public dataset benchmarks, but humans still prefer RL-finetuned models.

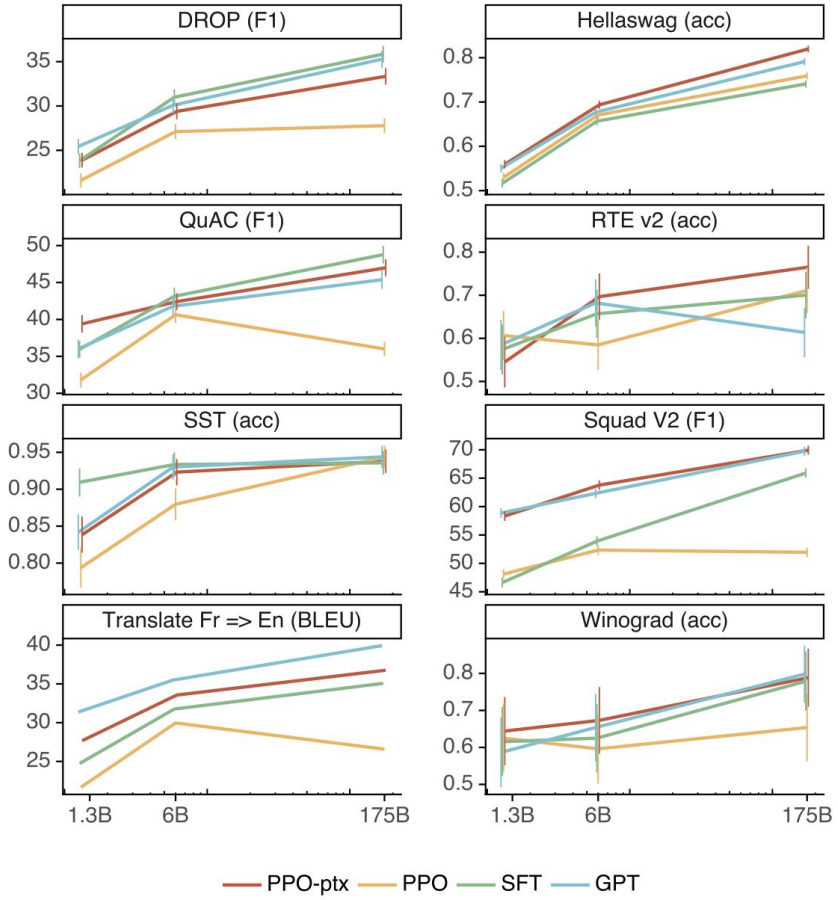


Figure 29: Few-shot performance of our models on various public NLP datasets (compare to zero-shot performance shown in Figure 28

Ablation: Loss weights

Q: Can you fix these regressions by just tuning the loss weight on pretraining / KL terms?

A: You can improve them, but not fix them completely.

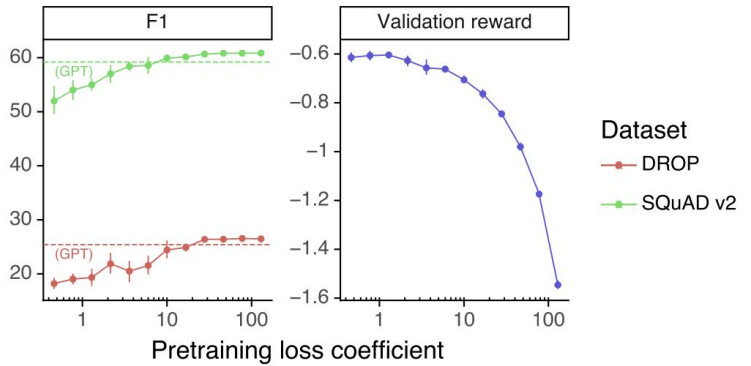


Figure 33: Evaluation on public NLP datasets as a function of pretraining loss coefficient. There is a pretraining coefficient that leads to a significant improvement on DROP and SQuAD and not much regression on validatoin reward.

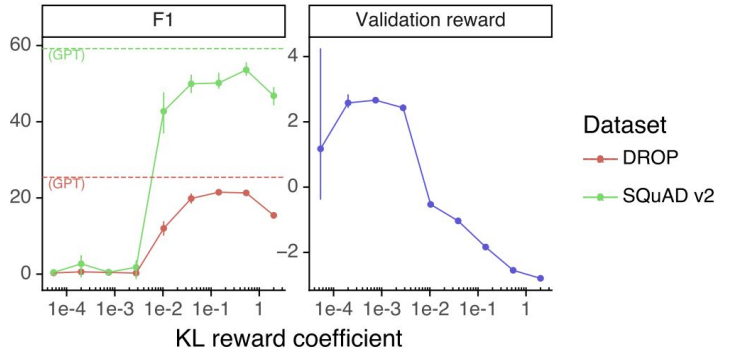
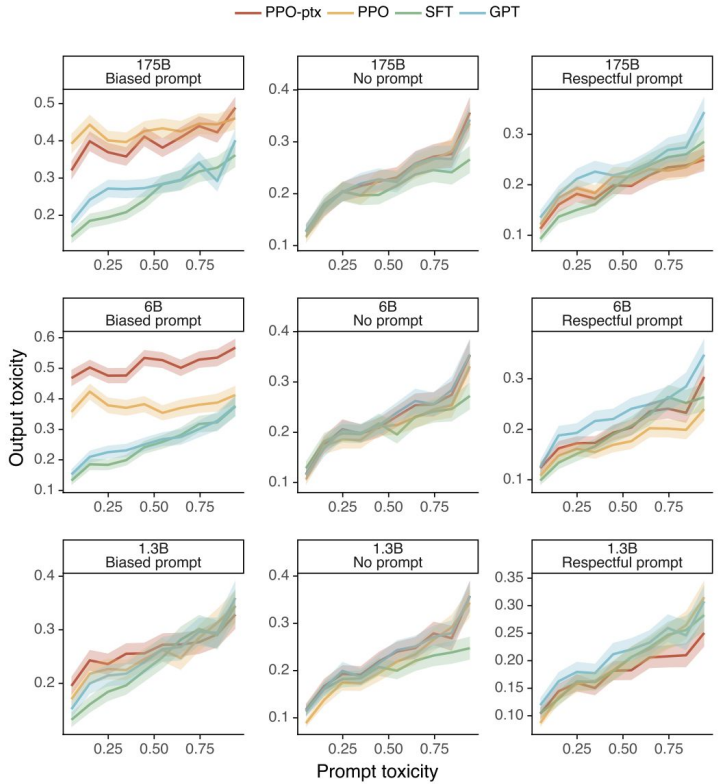


Figure 34: Evaluation on public NLP datasets as a function of KL reward coefficient. Increasing the KL coefficient does not fully mitigate the regressions on DROP and SQuAD.

RealToxicityPrompts Eval

Figure 39: “PPO instruction-following models generally create less toxic output than the non-instruction-following models, but only when instructed to be respectful. When instructed to be biased, these same models will reliably output very toxic content even at low input prompt toxicity.”



Direct Preference Optimization (DPO)

DPO Overview

- RLHF is a complex and often unstable procedure. Can we simplify the pipeline?
- Use a **mapping between reward functions and optimal policy** – this can be optimized exactly with a single stage of policy training.
- Avoids reinforcement learning, and eval results are competitive with RLHF.

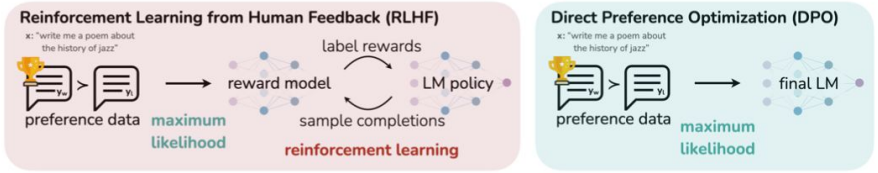


Figure 1: **DPO optimizes for human preferences while avoiding reinforcement learning.** Existing methods for fine-tuning language models with human feedback first fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward. In contrast, DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, without an explicit reward function or RL.

Deriving the DPO Objective

Recall the RLHF objective from before:

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

They say “it is straightforward to show :) that the optimal solution to the above equation takes the form”, where Z(x) is the partition function.

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

Wait, how does that derivation work?

A.1 Deriving the Optimum of the KL-Constrained Reward Maximization Objective

In this appendix, we will derive Eq. 4. Analogously to Eq. 3, we optimize the following objective:

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi}\left[r(x,y)\right] - \beta\mathbb{D}_{\text{KL}}\left[\pi(y|x)||\pi_{\text{ref}}(y|x)\right] \tag{11}$$

under any reward function $r(x, y)$, reference model π_{ref} and a general non-parametric policy class. We now have:

$$\begin{aligned}
&\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi}\left[r(x,y)\right] - \beta\mathbb{D}_{\text{KL}}\left[\pi(y|x) \,||\, \pi_{\text{ref}}(y|x)\right] \\
&\quad= \max_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[r(x,y) - \beta\log\frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)}\right] \\
&\quad= \min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{1}{\beta}r(x,y)\right] \\
&\quad= \min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)} - \log Z(x)\right]
\end{aligned} \tag{12}$$

where we have partition function:

$$Z(x) = \sum_{y} \pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right).$$

Note that the partition function is a function of only x and the reference policy π_{ref}, but does not depend on the policy π. We can now define

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

which is a valid probability distribution as $\pi^*(y|x) \geq 0$ for all y and $\sum_y \pi^*(y|x) = 1$. Since $Z(x)$ is not a function of y, we can then re-organize the final objective in Eq 12 as:

$$\min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\pi^*(y|x)}\right] - \log Z(x)\right] = \tag{13}$$

$$\min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{D}_{\text{KL}}(\pi(y|x) \,||\, \pi^*(y|x)) + Z(x)\right] \tag{14}$$

Now, since $Z(x)$ does not depend on π, the minimum is achieved by the policy that minimizes the first KL term. Gibbs' inequality tells us that the KL-divergence is minimized at 0 if and only if the two distributions are identical. Hence we have the optimal solution:

$$\pi(y|x) = \pi^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right) \tag{15}$$

for all $x \in \mathcal{D}$. This completes the derivation.

Ok, now reparametrize the reward:

From the previous slide, solution to the RLHF objective takes the form:

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

Now reparametrize:

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

Substitute into the Bradley-Terry preference model

Bradley-Terry model says that the probability completion y1 is preferred to y2 can be expressed as the following quantity, where r* is the reward function.

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

Substitute last slide's expression for the reward to get:

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

Call the boxed quantity S. To maximize this probability, we want exp(S) to be 0, so we want S = -infinity.

Writing down the objective

Now, we can write down a **maximum-likelihood** objective – how do we maximize the probability from the previous equation?

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

Boxed quantity is -S. From before, we want S = -infinity, so sigmoid(-S) = 1 and L = -log(sigmoid(-S)) = 0, which is the minimum of log(sigmoid(x)).

Connection: This is analogous to the log-sigmoid loss from the reward model training step in RLHF, just with different quantities:

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta(x,y_w) - r_\theta(x,y_l)\right)\right)\right]$$

Pytorch code

```python
import torch.nn.functional as F

def dpo_loss(pi_logps, ref_logps, yw_idxs, yl_idxs, beta):
    """
    pi_logps: policy logprobs, shape (B,)
    ref_logps: reference model logprobs, shape (B,)
    yw_idxs: preferred completion indices in [0, B-1], shape (T,)
    yl_idxs: dispreferred completion indices in [0, B-1], shape (T,)
    beta: temperature controlling strength of KL penalty
    Each pair of (yw_idxs[i], yl_idxs[i]) represents the
    indices of a single preference pair.
    """
    pi_yw_logps, pi_yl_logps = pi_logps[yw_idxs], pi_logps[yl_idxs]
    ref_yw_logps, ref_yl_logps = ref_logps[yw_idxs], ref_logps[yl_idxs]
    pi_logratios = pi_yw_logps - pi_yl_logps
    ref_logratios = ref_yw_logps - ref_yl_logps
    losses = -F.logsigmoid(beta * (pi_logratios - ref_logratios))
    rewards = beta * (pi_logps - ref_logps).detach()
    return losses, rewards
```

Putting it all together

DPO outline. The general DPO pipeline is as follows: 1) Sample completions $y_1, y_2 \sim \pi_{\text{ref}}(\cdot \mid x)$ for every prompt x, label with human preferences to construct the offline dataset of preferences $\mathcal{D} = \{x^{(i)}, y_w^{(i)}, y_l)^{(i)}\}_{i=1}^N$ and 2) optimize the language model π_θ to minimize $\mathcal{L}_{\text{DPO}}$ for the given π_{ref} and $\mathcal{D}$ and desired β. In practice, one would like to reuse preference datasets publicly available, rather than generating samples and gathering human preferences. Since the preference datasets are sampled using π^{SFT}, we initialize $\pi_{\text{ref}} = \pi^{\text{SFT}}$ whenever available. However, when π^{SFT} is not available, we initialize π_{ref} by maximizing likelihood of preferred completions (x, y_w), that is, $\pi_{\text{ref}} = \arg\max_\pi \mathbb{E}_{x,y_w \sim \mathcal{D}} [\log \pi(y_w \mid x)]$. This procedure helps mitigate the distribution shift between the true reference distribution which is unavailable, and π_{ref} used by DPO. Further details related to the implementation and hyperparameters can be found in Appendix B.

Eval Comparison

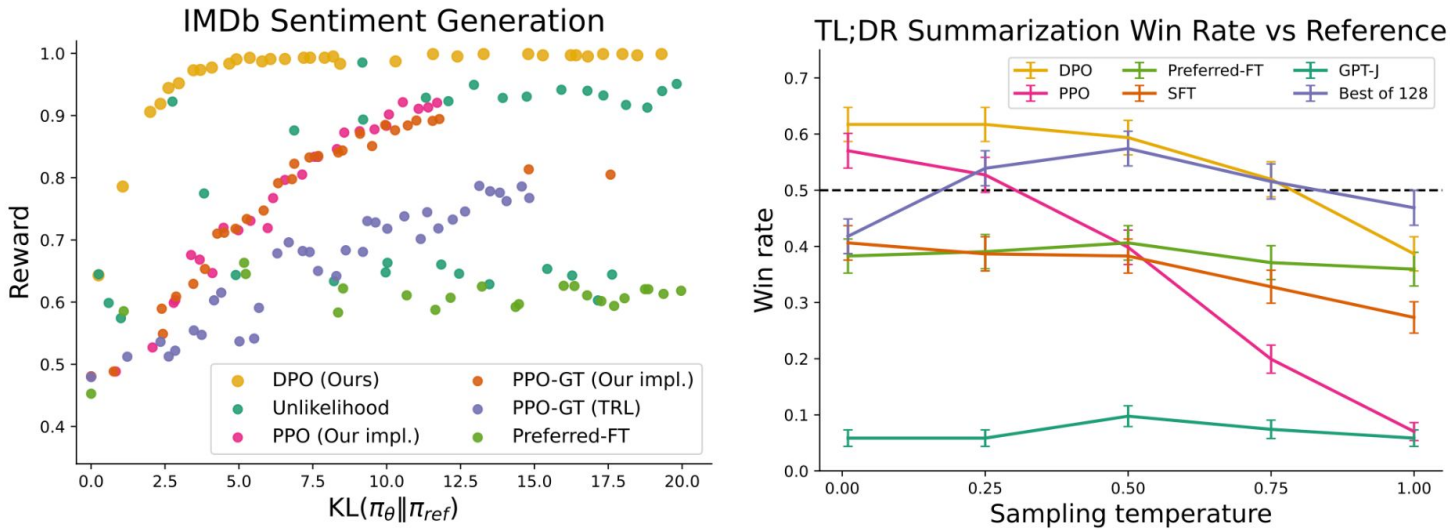


Figure 2: **Left.** The frontier of expected reward vs KL to the reference policy. DPO provides the highest expected reward for all KL values, demonstrating the quality of the optimization. **Right.** TL;DR summarization win rates vs. human-written summaries, using GPT-4 as evaluator. DPO exceeds PPO's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

Appendix: Intro to Policy Gradient / PPO

Introduction to Policy Gradient

- Suppose we have a stochastic parametrized policy, π_θ
- We want to maximize the expected return:

$$J(\pi_\theta) = \underset{\tau \sim \pi_\theta}{\mathrm{E}} [R(\tau)].$$

- We will optimize this by gradient ascent. To do this, we need an expression for the policy gradient that we can numerically compute.

Introduction to Policy Gradient

- The probability of a trajectory (sequence of state / action pairs) is: π_θ

$$P(\tau|\theta) = \rho_0(s_0) \prod_{t=0}^{T} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t).$$

- We will use the log-derivative trick, which follows from d/dx log(x) = 1/x.

$$\nabla_\theta P(\tau|\theta) = P(\tau|\theta)\nabla_\theta \log P(\tau|\theta).$$

- Grad-log-prob of a trajectory

$$\begin{aligned}\nabla_\theta \log P(\tau|\theta) &= \cancel{\nabla_\theta \log \rho_0(s_0)} + \sum_{t=0}^{T} \left(\cancel{\nabla_\theta \log P(s_{t+1}|s_t, a_t)} + \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \\ &= \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t).\end{aligned}$$

Deriving the gradient of J

$$
\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \nabla_\theta \underset{\tau\sim\pi_\theta}{\mathrm{E}}[R(\tau)] \\
&= \nabla_\theta \int_\tau P(\tau|\theta) R(\tau) && \text{Expand expectation} \\
&= \int_\tau \nabla_\theta P(\tau|\theta) R(\tau) && \text{Bring gradient under integral} \\
&= \int_\tau P(\tau|\theta) \nabla_\theta \log P(\tau|\theta) R(\tau) && \text{Log-derivative trick} \\
&= \underset{\tau\sim\pi_\theta}{\mathrm{E}}[\nabla_\theta \log P(\tau|\theta) R(\tau)] && \text{Return to expectation form} \\
\therefore \nabla_\theta J(\pi_\theta) &= \underset{\tau\sim\pi_\theta}{\mathrm{E}}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau)\right] && \text{Expression for grad-log-prob}
\end{aligned}
$$

Overview of Proximal Policy Optimization (PPO)

Advantage function:

Q(s, a) - V(s).

Intuitively, it is the extra reward that could be obtained by the agent taking the action *a*.

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0

for $k = 0, 1, 2, \ldots$ **do**

Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

Compute rewards-to-go $\hat{R}_t$.

Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.

Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

end for

Intuition from simplified PPO objective

Simplified objective

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a),\quad g(\epsilon, A^{\pi_{\theta_k}}(s,a))\right),$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0. \end{cases}$$

Intuition:
Consider cases on the sign of advantage.

Advantage is positive: Suppose the advantage for that state-action pair is positive, in which case its contribution to the objective reduces to

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, (1+\epsilon)\right) A^{\pi_{\theta_k}}(s,a).$$

Because the advantage is positive, the objective will increase if the action becomes more likely—that is, if $\pi_\theta(a|s)$ increases. But the min in this term puts a limit to how *much* the objective can increase. Once $\pi_\theta(a|s) > (1+\epsilon)\pi_{\theta_k}(a|s)$, the min kicks in and this term hits a ceiling of $(1+\epsilon)A^{\pi_{\theta_k}}(s,a)$. Thus: *the new policy does not benefit by going far away from the old policy*.

Advantage is negative: Suppose the advantage for that state-action pair is negative, in which case its contribution to the objective reduces to

$$L(s, a, \theta_k, \theta) = \max\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, (1-\epsilon)\right) A^{\pi_{\theta_k}}(s,a).$$

Because the advantage is negative, the objective will increase if the action becomes less likely—that is, if $\pi_\theta(a|s)$ decreases. But the max in this term puts a limit to how *much* the objective can increase. Once $\pi_\theta(a|s) < (1-\epsilon)\pi_{\theta_k}(a|s)$, the max kicks in and this term hits a ceiling of $(1-\epsilon)A^{\pi_{\theta_k}}(s,a)$. Thus, again: *the new policy does not benefit by going far away from the old policy*.

Appendix: What does the DPO gradient update look like?

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

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