Direct Preference Optimization

Your Language Model is Secretly a Reward Model

Rafailov, Sharma, Mitchell, Ermon, Manning, Finn
Stanford University — NeurIPS 2023 (arXiv:2305.18290)



Paper presentation and critical evaluation

Wong Wei Heng

Background: why preference learning?

- Pretraining on web text gives broad knowledge *and* the full range of human quality: we want the model to imitate the *good* part
- Demonstrations of ideal answers are expensive to write
- **Comparisons are cheap**: "answer A is better than answer B"

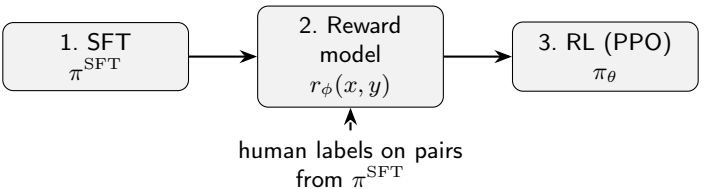
Data: preference pairs

$$\mathcal{D} = \{(x^{(i)}, y_w^{(i)}, y_l^{(i)})\}_{i=1}^{N}$$

prompt x, preferred ("winning") response y_w, dispreferred ("losing") response y_l.

Goal: turn $\mathcal{D}$ into a model that produces responses people prefer.

Background: the RLHF pipeline



- **SFT:** fine-tune on good demonstrations $\Rightarrow \pi^{\mathrm{SFT}}$, used as the reference model π_{ref}
- **Reward model:** learn a scalar score from the human comparisons
- **RL:** sample from the policy, score with r_ϕ, update the policy to score higher

Background: Bradley–Terry preference model

How do pairwise labels relate to a hidden reward r^*? Assume each response has a latent strength; the better one wins more often:

$$p^*(y_1 \succ y_2 \mid x) = \frac{e^{r^*(x,y_1)}}{e^{r^*(x,y_1)} + e^{r^*(x,y_2)}} = \sigma\big(r^*(x,y_1) - r^*(x,y_2)\big)$$

- Depends only on the **difference** of rewards for the *same prompt*: adding any $f(x)$ to all rewards changes nothing
- Fit the reward model by maximum likelihood (logistic regression):

$$\mathcal{L}_R(r_\phi) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \log \sigma\big(r_\phi(x,y_w) - r_\phi(x,y_l)\big)$$

Caveat to carry forward: this is a scalar, transitive, context-free model of human preference. Everything DPO proves is conditional on it.

Background: the RLHF objective

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

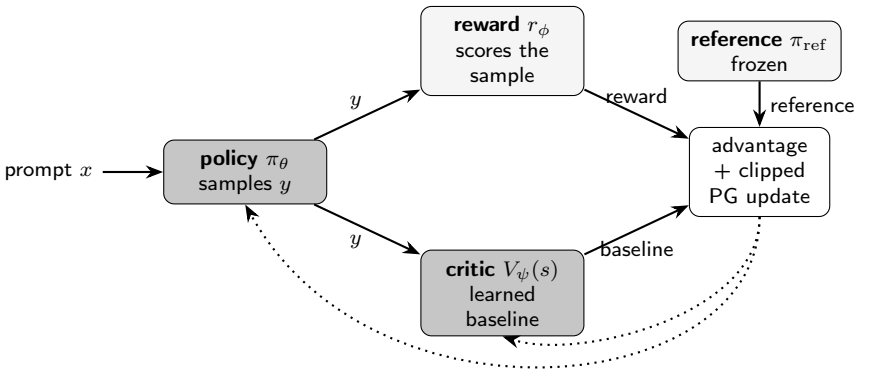
Why not just backprop r_ϕ through y?

- y is a sequence of *sampled* discrete tokens: sampling is not differentiable, so there is no gradient path from the reward to θ
- The standard fix is the policy-gradient identity: differentiate $\log \pi_\theta(y|x)$ instead, and use the reward as a *weight* on that update

Why is RL hard in practice?

- the expectation is over $y \sim \pi_\theta$, so we must *sample* from the current policy at every step (hence generation in the loop)
- the policy-gradient estimate $r \nabla \log \pi_\theta$ is high-variance
- the standard solution is a **baseline** $b(x)$: subtract it from r; unbiased, lower variance

Background: PPO step



- **Critic:** predicts the expected return from a partial response, $V_\psi(s)$. Subtracting it from the reward gives an *advantage* — a control variate that lowers the variance of the policy gradient without biasing it.

Four LLM-sized networks: policy and critic trained, reward model and reference frozen. Per step: 1 generation + 3 forward passes + 2 backward passes.

What that costs us

- **Complex:** three sequential stages, four networks resident, two of them training, interlocking hyperparameters
- **Expensive:** generation inside every training step, plus a separately trained reward model
- **Unstable:** high-variance policy gradients; the critic exists to reduce that variance, but adds its own optimisation problem

The paper's question

*Can we optimise the **same** objective directly from preferences, without an explicit reward model and without RL?*

Step 1: solve the KL-regularised problem

For a fixed prompt x and any reward r, the optimal policy has a closed form:

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

where

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

- Start with the reference distribution.
- Reweight each response by its exponentiated reward.
- Divide by $Z_r(x)$ so the probabilities sum to one.

This closed-form result is prior work used by DPO; the paper's novelty is what it does with the mapping.

Step 2: read the reward back from the policy

Take logs of the optimal-policy equation and rearrange:

$$\boxed{r(x, y) = \beta \log \frac{\pi_r(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z_r(x)}$$

An optimal policy and a reference policy determine the reward, up to the normalising term.

- The log-ratio measures how much the optimal policy changes the likelihood of a response relative to the reference.
- But $Z_r(x)$ sums over *all possible completions*; it is generally intractable to compute.

Can we remove the unknown $\log Z_r(x)$ without changing the preference model?

Step 3: absolute reward is not identifiable

BT only depends on the difference between two responses to the *same* prompt:

$$P(y_w \succ y_l|x) = (r(x, y_w) - r(x, y_l)) .$$

Now shift every reward for prompt x by the same amount $c(x)$:

$$r'(x, y) = r(x, y) + c(x).$$

Then

$$r'(x, y_w) - r'(x, y_l) = r(x, y_w) - r(x, y_l).$$

- The preference probabilities are unchanged.
- The KL-optimal policy is also unchanged: the factor $e^{c(x)/\beta}$ cancels in the partition function.

A preference dataset cannot identify the prompt-wise reward offset. Only the reward-equivalence class matters.

Theorem 1: choose a canonical reward in each class

Theorem 1. Under mild assumptions, all reward classes consistent with the Plackett-Luce (and Bradley-Terry in particular) models can be represented with the reparameterization

$$\boxed{r(x, y) = \beta \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)}}$$

for some policy π and a given reference policy π_{ref}.

Proof idea.

- We subtracted only a function of x, so we stayed in the same preference-equivalence class.
- Therefore the policy-ratio form loses no generality at the level of reward classes.

What Theorem 1 really buys us: normalisation

For the canonical reward

$$\tilde{r}(x, y) = \beta \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)},$$

its partition function is

$$\begin{aligned} Z_{\tilde{r}}(x) &= \sum_y \pi_{\text{ref}}(y|x) \exp\left(\frac{\tilde{r}(x, y)}{\beta}\right) \\ &= \sum_y \pi_{\text{ref}}(y|x) \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} \\ &= \sum_y \pi(y|x) = 1. \end{aligned}$$

$$Z_{\tilde{r}}(x) = 1 \qquad \Rightarrow \qquad \log Z_{\tilde{r}}(x) = 0$$

- DPO selects the normalised representative of each reward class.

Instability of Actor–Critic Algorithms

The paper next rewrites optimisation as minimising $D_{\mathrm{KL}}(\pi_\theta \| \pi^*)$. This gives (Eq. 10):

$$\max_{\pi_\theta} \mathbb{E}_{y\sim\pi_\theta}\left[\underbrace{r_\phi(x,y) - \beta \log Z_\phi(x)}_{\text{reward with prompt-level baseline}} \quad -\beta \log \frac{\pi_\theta(y|x)}{\pi_{\mathrm{ref}}(y|x)}\right].$$

- $\beta \log Z_\phi(x)$ is the soft-value / normalisation term: it depends on the prompt, not the sampled response.
- It does not change the exact optimum, but omitting it from a *sampled policy-gradient estimator* can increase variance.

DPO's normalised implicit reward has $Z = 1$, so this normalisation baseline does not need to be estimated.

Step 4: define the implicit reward of the trainable policy

Theorem 1 motivates parameterising the reward directly by the policy:

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

Interpretation:

- If π_θ raises the likelihood of y relative to the reference, its implicit reward is positive.
- If it lowers the likelihood, its implicit reward is negative.
- β sets the scale and controls how strongly the policy is regularised relative to the reference.

We don't need to add a reward head. The policy/reference log-ratio *is* the reward representation. During training, π_θ is adjusted so that this implicit reward explains the observed preference comparisons.

Step 5: turn the preference model into the DPO loss

Start from Bradley–Terry:

$$P(y_w \succ y_l|x) = (r(x, y_w) - r(x, y_l)) .$$

Use the implicit reward for each response. The prompt-only normaliser cancels, leaving

$$P_\theta(y_w \succ y_l|x) = (\hat{r}_\theta(x, y_w) - \hat{r}_\theta(x, y_l)) .$$

Maximum likelihood on the preference dataset gives

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

This is the original reward-model logistic loss, after a change of variables from reward parameters to policy parameters.

What does the DPO gradient do?

$$\nabla_\theta \mathcal{L}_{\mathrm{DPO}} = -\beta\, \mathbb{E}\Big[\underbrace{\sigma\big(\hat{r}_\theta(y_l) - \hat{r}_\theta(y_w)\big)}_{\text{weight}} \Big(\underbrace{\nabla \log \pi_\theta(y_w|x)}_{\text{inc likelihood of } y_w} - \underbrace{\nabla \log \pi_\theta(y_l|x)}_{\text{dec likelihood of } y_l} \Big) \Big]$$

The bracket: raise the likelihood of y_w, lower the likelihood of y_l.
$\nabla \log \pi_\theta(y|x)$ is exactly the direction SFT would move in, so DPO is *SFT on the winner minus SFT on the loser*

The weight: $\sigma(\hat{r}_\theta(y_l) - \hat{r}_\theta(y_w))$ is the model's current probability of getting the pair *wrong*. It is the prediction error of the implicit reward model

π_{ref} does not depend on θ, so it contributes no gradient — it only anchors the *size* of the implicit rewards.

Full derivation in Appendix A.4.

Why the weight matters

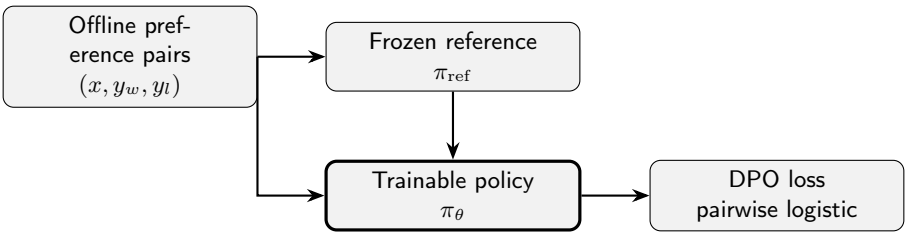
Intuition. The weight is large when the pair is ranked wrongly and small when it is already correct. Training therefore concentrates on the mistakes and *stops pushing* once a pair is solved — there is no incentive to keep inflating the margin forever.

$\Delta = \hat{r}_\theta(y_w) - \hat{r}_\theta(y_l)$	weight	what training does
0	0.50	undecided, half push
$+2$	0.12	mostly right, small push
$+5$	0.007	solved, essentially stops
-2	0.88	confidently wrong, big push

The authors reports the naive version degenerates. They defined the "unlikelihood" baseline by removing the weight (and the reference model) and just push $\log p(y_w)$ up and $\log p(y_l)$ down. It degenerates: Appendix Table 3 shows the samples are gibberish.

Two consequences: (i) the loss constrains the *margin*, not $p(y_w)$ itself, so both probabilities can fall together (likelihood displacement); (ii) pairs with the same margin get the same push regardless of how subtle the preference is.

The practical algorithm



1. For each pair, compute chosen/rejected log probabilities under π_θ and frozen π_{ref}.
2. Form the log-ratio margin and apply the DPO logistic loss.
3. Backpropagate through π_θ; keep π_{ref} fixed.

No reward-model training, no critic, and no fresh generations inside the optimisation loop.

There is still a frozen reference model and a preference dataset; DPO removes the separate reward-model and online-RL stages, not every auxiliary computation.

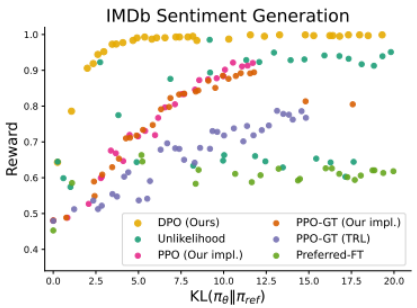
Experiments: setup

Task	Data	Model	Metric
Sentiment	IMDb prefixes; pairs from a sentiment classifier (known r)	GPT-2-large	reward vs KL frontier
Summarise	Reddit TL;DR + human prefs	GPT-J (6B)	GPT-4 win rate
Dialogue	Anthropic HH, 170k	Pythia-2.8B	GPT-4 win rate

Baselines

- SFT, Preferred-FT (SFT on y_w), Unlikelihood
- **PPO** (learned reward), **PPO-GT** (oracle reward, sentiment only)
- **Best-of-N**: sample N, return highest reward (strong but costly at test time)

Results: sentiment reward–KL frontier



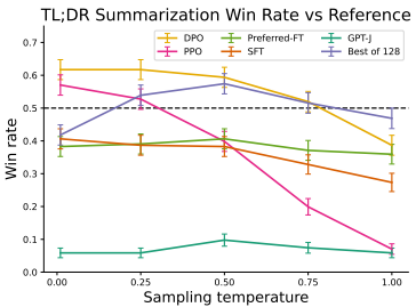
IMDb sentiment: reward vs. KL

- **Controlled setting:** ground-truth reward is known, so the frontier is measured against the true objective
- 22 runs sweeping each method's knob (β, target KL, α)
- **DPO is highest at every KL level** — strictly dominates PPO
- Beats PPO *even with the oracle reward* (PPO-GT)

Same objective, so the gap is *optimisation efficiency*, not a different target. This is the paper's cleanest experiment.

The authors modified their own PPO (the off-the-shelf TRL version did much worse). PPO used extra unlabeled prompts that DPO didn't (Sec. 6.3), which cuts in DPO's favour.

Results: TL;DR summarisation



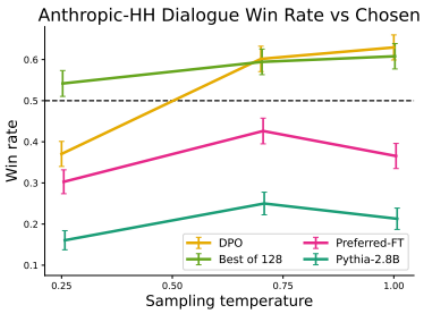
Reddit TL;DR: GPT-4 win rate vs temperature

- **Win rate vs reference:** DPO ≈61%, PPO ≈57%
- **Human study:** DPO(0.25) preferred 58% of the time
- **Robustness:** PPO degrades toward the base model as temperature rises; DPO stays comparatively flat

Out-of-distribution CNN/DM also favours DPO: 0.36 vs. 0.26 at $T{=}0$, 0.31 vs. 0.23 at $T{=}0.25$.

Human study was 25 volunteers giving about 272 judgments on DPO vs PPO. Humans agreed with each other 65% of the time.

Results: Anthropic-HH dialogue



Anthropic HH: win rate vs temperature

- **170k human preference pairs**, Pythia-2.8B
- The only *efficient* method to beat the dataset's preferred responses
- $\approx$ Best-of-128 quality at a fraction of the test-time cost

Caveat: here the stand-in for PPO is Best-of-128, not a trained PPO policy — so this is DPO vs. a sampling baseline, not DPO vs. PPO.

What the results establish

- **Sentiment:** DPO's reward–KL frontier strictly dominates PPO and PPO-GT — the objective is optimised better, not just differently
- **Summarisation:** $\approx$61% GPT-4 win rate vs $\approx$57% for PPO; the win rate holds up as sampling temperature rises
- **Dialogue:** the only efficient method to beat the dataset's preferred responses, matching Best-of-128 at a fraction of the test-time cost
- **Consistency:** the same loss, with essentially one knob (β), works across all three tasks without per-task architecture changes

What the results do *not* establish

- The PPO baseline is the authors' own implementation, tuned within their compute budget
- Scale is capped at 6B parameters; evaluation is single-turn and GPT-4-judged

What the paper concludes

- **On all three tasks, DPO matches or beats the authors' PPO baseline** — and in the controlled sentiment setting it beats even PPO-GT at every KL level
- **The gain is optimisation efficiency, not a different objective:** both methods target the same KL-constrained reward, but DPO's loss is easier to optimise and more robust to sampling temperature
- **The pipeline is simpler:** one supervised loss, no reward model, no critic, no generation in the loop

Evaluation: were the aims achieved, and is it replicable?

Aim: RLHF-quality alignment, no reward model, no RL.

- **Largely met:** exact under BT; one loss; no sampling in the loop; one main knob
- **But:** "beats PPO" compares against the authors' own PPO; in dialogue the comparison is against Best-of-128

Replicability.

- **Good:** public datasets and open models, loss plus PyTorch snippet, hyperparameters, complete derivations in the appendix
- **Weaker:** closed non-deterministic GPT-4 judge whose verdict depends on the prompt; Uncertainty reporting and reproducibility details could be clearer; hand-tuned PPO baselines; $n = 25$ human study

Evaluation: behind the theoretical guarantee

assumption	**What it enables**	**What it does not guarantee**
Bradley–Terry	Scalar reward explains pairwise preferences; prompt-wise shifts are unidentifiable	Real preferences may be context-sensitive, intransitive or heterogeneous across annotators
π_{ref} is a good reference	Policy/reference log-ratio is defined for responses of interest.	DPO is sensitive to the reference; a poor π_{ref} caps what "better" can mean
Data coverage	Offline comparisons provide a learning signal on observed pairs	No direct constraint on outputs far outside the dataset's support
RLHF-optimal π^* prefers y_w	the gradient raises y_w	if not, the DPO loss can fall while the preference flips — later work shows the equivalence is *conditional*

Theorem 1 is a representational guarantee inside the assumptions. It does not prove that human preferences obey BT exactly, that finite data covers all relevant outputs, or that training is immune to distribution shift.

Evaluation: why a well-tuned PPO can win

- **The training distribution moves with the policy.** PPO samples from the current π_θ; DPO keeps lowering the loss on a frozen dataset, including answers the policy may no longer produce
- **The objective is margin-only.** Nothing in $\mathcal{L}_{\mathrm{DPO}}$ bounds $p(y_w)$, whereas PPO's KL penalty is applied to the *sampled* distribution

How to read the original comparison

- "PPO" in the paper is the authors' own implementation, tuned within their compute budget. The paper did not run a systematically re-tuned PPO

Mechanisms from later analyses (likelihood displacement; feasible-set containment; offline distribution shift; conditional equivalence).

Evaluation: offline: a trade, not an improvement

What offline buys

- no generation in the loop: cheap, reproducible, easy to scale out
- no reward model to hack and no critic to fit

What it costs

- no exploration beyond $\mathcal{D}$; quality is capped by data coverage
- no mechanism to notice when the policy has left the data's support

Offline is *why* DPO became the default for open-model post-training, and *why* iterative/online variants exist.

Evaluation: the contribution

- **Conceptually:** the RL stage is not essential for most of the gain under BT and offline data; reward and policy are two views of one object
- **Practically:** lowered the barrier to preference tuning; adopted for open-model post-training

The DPO family is the clearest evidence of impact

variant	**what it changes**
IPO	near-deterministic labels over-fit the logistic loss $\Rightarrow$ add a margin / regulariser
KTO	pairwise comparisons are expensive $\Rightarrow$ binary good/bad signal only
SimPO	the reference model is a cost $\Rightarrow$ length-normalised margin, no π_{ref}
ORPO	two stages are a cost $\Rightarrow$ SFT plus odds-ratio penalty in one stage
iterative / online DPO	the offline assumption $\Rightarrow$ refresh $\mathcal{D}$ from the current policy

Overall assessment and takeaways

1. **Theory:** KL-regularised RL gives a closed-form policy; pairwise preferences ignore prompt-wise reward offsets; Theorem 1 lets us choose a normalised policy-ratio reward without losing a reward-equivalence class.
2. **Algorithm:** the implicit reward turns Bradley–Terry maximum likelihood into the DPO loss, trained directly on fixed preference pairs.
3. **Evidence:** strong controlled sentiment result, encouraging summarisation and dialogue results, with important differences in evaluation and baselines.
4. **Caveat:** the guarantee is conditional on the preference model; offline coverage, reference choice, proxy evaluation and scale remain limitations.

DPO's key insight is not just a substitution. It uses reward non-identifiability to choose a policy parameterisation whose normalisation is built in.

Your language model is secretly a reward model.