DPO

DPO

RLHF-PPO训练时要加载4个模型，面临着显存占用大的问题，并且需要利用偏好数据宇轩训练一个奖励模型（或者指定基于规则的奖励模型），如果奖励模型存在偏差，那也会与人类偏好存在偏差。DPO消除了奖励模型，直接使用人类标注的偏好数据，并且不再用强化学习的方法，而是通过数学推理，将原始的偏好对齐优化目标步步简化，显存占用和训练速度都更优。最后通过类似于sft的方式，用更简化的步骤训练出对齐模型。

举个例子，比如我要做美食：

- RLHF-PPO：先训练一个评委（奖励模型），我按照策略模型做出一道菜，评委给出打分（奖励），用评估模型去接近评委的喜好，用参照模型避免我的做菜方法法发生很大变化，用以上三者去对我的做菜策略（策略模型）进行更新，再更新评估模型。
- DPO：直接用正确配方和错误配方对来更新策略模型，用参照模型避免我的做菜方法法发生很大变化。

总结DPO与RLHF-PPO：**DPO把"对齐人类偏好"这个过程，从需要专业评委打分的竞技比赛，变成了直接对比参考答案的学生自习。**

RLHF-PPO回顾

SFT：通过对预训练的LLM模型进行有监督微调，得到模型 π^{SFT}

奖励模型：使用SFT模型，根据prompt x 生成答案对 $(y1, y2) \sim \pi^{SFT}(y|x)$，人类对标注答案进行标注，得到prefer回复 y_w 和dispref回复 y_l。为了拟合人类的偏好，常用的偏好模型包括**Bradley-Terry模型**，人类偏好分布p*可以写为：

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp\left(r^*(x, y_1)\right)}{\exp\left(r^*(x, y_1)\right) + \exp\left(r^*(x, y_2)\right)}. \tag{1}$$

其中r*是奖励模型，根据从p*采样的训练数据集 $\mathcal{D} = \{x^i, y_w^i, y_l^i\}_{i=1}^N$，用来训练奖励模型，损失函数为：

$$\mathcal{L}_R(r_\phi, \mathcal{D}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))\right] \tag{2}$$

σ 是sigmoid函数。在现实应用中，奖励模型r用SFT模型+一个线性层初始化。为了确保奖励函数具有较低的方差，对奖励进行归一化

$$\mathbb{E}_{x,y\sim\mathcal{D}}\left[r_\phi(x, y)\right] = 0 \text{ for all } x.$$

策略模型：基于奖励模型对策略模型 π_θ 进行更新：

$$\max_{\pi} E_{x\sim\mathcal{D},y\sim\pi}[R(x,y) - \beta D_{KL}(\pi(y|x)||\pi_{ref}(y|x))]$$

其中 π_{ref} 表示参考策略，一般使用SFT模型。π_θ 也用SFT模型初始化，第二项的KL散度是为了防止 π_θ 偏离奖励模型准确的分布，并保证模型模型生成的多样性。DPO的策略模型更新目标从上式开始推导。

DPO

区别于前边需要学习奖励模型，并通过其对强化学习策略进行更新，DPO利用了一种奖励模型参数化方法，可以以闭合形式提取其最优策略，而无需 RL 训练循环。利用从奖励函数到最优策略的映射，将奖励函数的损失函数转换为策略的损失函数。

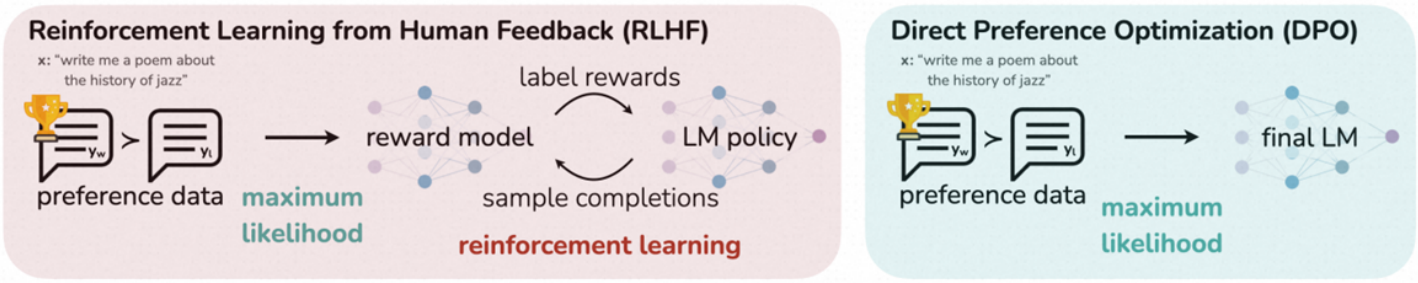


Figure 1: **DPO optimizes for human preferences while avoiding reinforcement learning.** Existing methods for fine-tuning language models with human feedback first fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward. In contrast, DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, fitting an *implicit* reward model whose corresponding optimal policy can be extracted in closed form.

基于奖励模型求解最优策略模型

首先对策略模型训练目标进行改写，

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

- 1-2行：$D_{KL}(\pi(y|x)||\pi_{ref}(y|x)) = E_{x\sim\mathcal{D},y\sim\pi}[log\frac{\pi(y|x)}{\pi_{ref}(y|x)}]$
- 2-3行：除以 β，并取反，max变成min
- 3-4行：由于 π 是一个概率分布，现在想把分母 $\frac{1}{Z(x)}\pi_{ref}(y|x) * exp(\frac{1}{\beta}r(x,y))$ 也变成概率分布的形式，这样就可以把第四行左侧视为KL散度的形式，因此将 $Z(x)$ 构造为：

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

Z(x)是一个归一化参数，确保分母是概率分布，$\pi_{ref}(y|x) * exp(\frac{1}{\beta} r(x,y))$ 所有可能输出y的概率之和为1。将 $Z(x)$ 带入分母中就得到：

$$\frac{\pi_{ref}(y|x) * exp(\frac{1}{\beta} r(x,y))}{\sum_y \pi_{ref}(y|x) * exp(\frac{1}{\beta} r(x,y))}$$

上式分子表示在一个（x,y）下的期望奖励，分母表示给定x下所有y的期望之和。通过乘以 $exp(\frac{1}{\beta} r(x,y))$，将奖励函数r转换为概率的权重，再通过归一化调整这些权重，使得高奖励的输出在新的策略 $\pi(y|x)$ 中获得更高的概率。 $Z(x)$ **只与x有关而与策略模型** π **无关，这很重要**。

现在已知分母也是概率分布，将分母改写为 $\pi^*(y|x) = \frac{1}{Z(x)} \pi_{ref}(y|x) * exp(\frac{1}{\beta} r(x,y))$。这样第四行就变成了：

$$\min_\pi E_{x\sim\mathcal{D}}[E_{y\sim\pi(y|x)}[log\frac{\pi(y|x)}{\pi^*(y|x)}] - logZ(x)] = \min_\pi E_{x\sim\mathcal{D}}[D_{KL}(\pi(y|x)||\pi^*(y|x)) - logZ(x)]$$

上式中 $Z(x)$ 与策略模型 π 无关，KL散度当两个分布完全相等时最小，因此策略模型 π 的最优解为：

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

通过改变变量将奖励优化问题直接转化为策略优化问题，已经在奖励函数 r 与策略 π 之间建立一个显式的函数关系。在RLHF-PPO的训练中，奖励模型 r 是基于大量数据训练出的最优奖励模型 r^*，然后再训练出最优策略模型 π^*，二者之间的关系为：

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

跳过奖励模型直接求解最优策略

将最优奖励模型提取出来：

$$r^*(x,y) = \beta log\frac{\pi^*(y|x)}{\pi_{ref}(y|x)} + \beta log Z(x)$$

先说结论，将其带入到**Bradley-Terry模型**中，可以得到用最优策略 π^* 和参考策略 π_{ref} 表示的人类偏好：

$$p^*(y_1 \succ y_2 \mid x) = \frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} - \beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)} \tag{6}$$

与公式2类似，就可以用最大似然来估计 π_θ：

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]. \tag{7}$$

σ 是sigmoid函数。这样就替换掉了奖励模型和评估模型，直接对策略模型进行优化。

🌟 **如何理解DPO的损失函数：**

- 首先明确，π_{ref} 在训练过程中保持不变
- 如果对于prefer的回复 y_w 预测概率增大，对disprefer的回复 y_l 概率减小，也就是 $\pi_\theta(y_w|x)$增大，$\pi_\theta(y_l|x)$减小，那对应着 $\beta log\frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} - \beta log\frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)}$ 增大，对应着 $\sigma(\beta log\frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)})$ 更接近于1，损失越接近于0.
- DPO的损失函数最大化prefer的回复与disprefer回复之间概率的差值，但是可能面临prefer和disprefer的概率都上升胡总和下降的情况。

对 θ 求梯度得到：

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

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

这是原论文中的推导，本质就是求导的**链式法则**，不过红色部分的w和l应该是标反了，

In this section we derive the gradient of the DPO objective:

$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\nabla_\theta \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} - \beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} \right) \right] \tag{21}$$

We can rewrite the RHS of Equation 21 as

$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\frac{\sigma'(u)}{\sigma(u)} \nabla_\theta (u) \right], \tag{22}$$

where $u = \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} - \beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)}$.

Using the properties of sigmoid function $\sigma'(x) = \sigma(x)(1-\sigma(x))$ and $\sigma(-x) = 1-\sigma(x)$, we obtain the final gradient

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

After using the reward substitution of $\hat{r}_\theta(x,y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ we obtain the final form of the gradient from Section 4.

接下来是理论推导：

奖励模型可以用偏好排序来训练得到，偏好排序存在两种方式：

- **每个用户查询只生成两个回答** `<prompt x, chosen y1, reject y2>`：希望奖励模型对chosen回答的给分尽量高于对reject回答的给分。
- **每个用户查询生成多于2个的回答** `<prompt x, y1, ..., yK>`：假设人工标注后的偏好排序组合为 τ （比如人工人为偏好从大到小应该为y2 > y3 > y1 >... > yK，则这个排列就为 τ ），那么我们希望奖励模型对 τ 这个排序的总得分要大于其余任何可能的偏好排序。(在chatGPT的训练中，会将生成的多个回复拆成两两的pair对，本质和只生成两个回答时的目标函数一致。而在更一般的场景中，希望将**每一种可能的回答偏好排序当成一个整体数据进行处理，**也就是希望 τ 的得分最高)。

每个用户查询只生成两个回答

y_w 和 y_l 分别表示chosen和reject回答，由**Bradley-Terry模型：**

$$p^*(y_w \succ y_l|x) = \frac{exp(r^*(x,y_w))}{exp(r^*(x,y_w)) + exp(r^*(x,y_l))}$$

训练希望生成 y_w 的概率大于 y_l ，奖励函数的优化目标为：

$$L(r_\phi, \mathcal{D}) = -E_{(x,y_w,y_l)\sim\mathcal{D}}[logp^*(y_w \succ y_l|x)]$$
$$= -E_{(x,y_w,y_l)\sim\mathcal{D}}[\frac{exp(r^*(x,y_w))}{exp(r^*(x,y_w)) + exp(r^*(x,y_l))}]$$
$$= -E_{(x,y_w,y_l)\sim\mathcal{D}}[\frac{1}{1+exp^{-r(x,y_w)-r(x,y_l)}}]$$
$$= -E_{(x,y_w,y_l)\sim\mathcal{D}}[log\sigma[r(x,y_w) - r(x,y_l)]]$$

将最优奖励模型 $r^*(x,y)$ 带入，优化目标变为：

$$L(r_\phi, \mathcal{D}) = -E_{(x,y_w,y_l)\sim\mathcal{D}}[log\sigma(\beta log\frac{\pi^*(y_w|x)}{\pi_{ref}(y_w|x)} - \beta log\frac{\pi^*(y_l|x)}{\pi_{ref}(y_l|x)})]$$

然后将最优策略模型 π^* 替换为待训练的策略模型 π_θ ，就将奖励模型的目标函数转化成了只与策略模型 π_θ 相关，绕过了奖励模型的训练：

$$L_{DPO}(\pi_\theta, \pi_{ref}) = -E_{(x,y_w,y_l)\sim\mathcal{D}}[log\sigma(\beta log\frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} - \beta log\frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)})]$$

每个用户查询生成多于2个的回答

基于**Plackett-Luce**模型， τ 优于其他任何排序的概率为：

$$p^*(\tau|y_1,...,y_k,x) = \Pi_{k=1}^K \frac{exp(r^*(x,y_{\tau(k)}))}{\sum_{j=k}^K exp(r^*(x,y_{\tau(j)}))}$$

其中， τ_k 表示人类标注的偏好序列 τ 中的第k个数据，序列 τ 中的K个回答已经按照偏好从高到低进行排序。将 r^* 带入，就能得到损失函数：

$$L_{DPO}(\pi_\theta, \pi_{ref}) = -E_{(\tau,y_1,...,y_K,x\sim\mathcal{D})}[log\Pi_{k=1}^K \frac{exp(\beta log\frac{\pi_\theta*(y_{\tau(k)}|x)}{\pi_{ref}*(y_{\tau(k)}|x)})}{\sum_{j=k}^K exp(\beta log\frac{\pi_\theta*(y_{\tau(j)}|x)}{\pi_{ref}*(y_{\tau(j)}|x)})}]$$

实验结果：

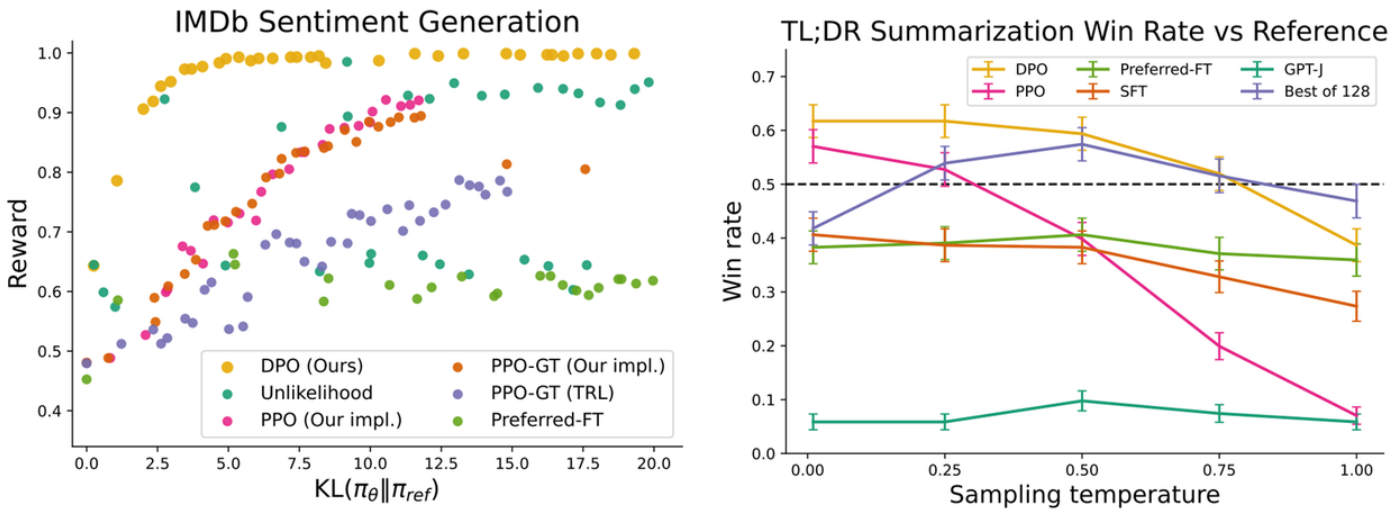


Figure 2: **Left.** The frontier of expected reward vs KL to the reference policy. DPO provides the highest expected reward for all KL values, demonstrating the quality of the optimization. **Right.** TL;DR summarization win rates vs. human-written summaries, using GPT-4 as evaluator. DPO exceeds PPO's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

🌟 **如何构建DPO训练数据集**

> 数据格式：{
> "prompt": "你好，请介绍一下transformers库？",
> "chosen": "Transformers是一个强大的自然语言处理库，支持…",
> "rejected": "不清楚"
> }

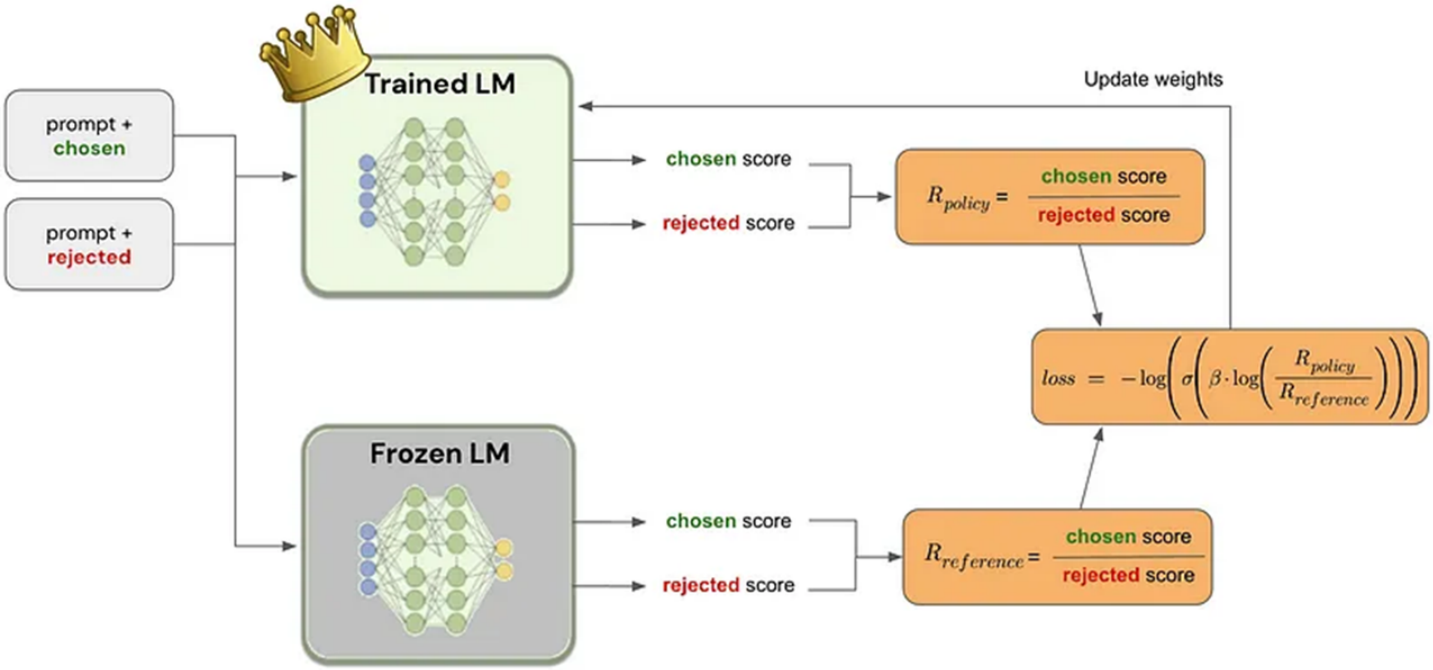
1. 采用开源的偏好数据集，从huggingface或modelscope中搜索。

2. **拒绝采样**：提出一个问题，让模型提出N个候选回复，用奖励模型或者人工对回复进行排序。
3. **多模型对比采样**：引入不同的模型分别生成回复，再利用其他大模型或者奖励模型进行排序。
4. **人工构造**：首先构建领域（数学、代码）、任务类型（C++、python）、难度等级（easy、medium、hard）三层体系，为每个分类构建20条以上高质量样本。然后根据prompt生成模板（例如："请用{语言}实现{功能}，要求{约束条件}"）生成大量的用户问题，然后让模型生成回复，最后再对模型回复进行排序。

有哪些评估DPO训练的指标

- **文本质量相关**：
  - **困惑度**：衡量模型对测试文本的预测能力
  - **多样性**：利用**n-gram、词汇多样性**等指标
- **人类偏好对齐相关**：
  - **偏好数据集测试**：在测试偏好数据集上的准确率，模型选择chosen数据的概率是否显著高于reject数据。
  - **胜率**：与未训练模型进行对比，让人类或者相关的奖励模型进行打分，验证DPO模型是否输出更高奖励值的回答。
- **安全性与可靠性相关**：
  - **有害内容检测**：使用**Perspective API**、**RealToxicityPrompts**等方法检测有害内容。
  - **可靠性检测**：验证模型是否过度自信（如通过Expected Calibration Error, ECE），以及检验模型输出中的数据、观点、计算结果等是否正确。
- **具体任务相关**：
  - 比如翻译任务的**BLEU**，摘要生成任务的**ROUGE**，语音识别任务的**WER**，代码生成任务的**代码执行成功率**，分类任务的**准确率、F1 score**

DPO的改进：ORPO



在训练过程中，始终要保留训练模型和参照模型。训练过程中去掉参照模型，例如ORPO，ORPO观察到生成了非常多的dispref的样本，其直接对序列的生成概率进行优化（DPO中还是在对奖励函数进行优化，只不过是通过最大似然替换掉了奖励模型，通过最大化奖励值使得生成prefer的样本的概率增大），优化目标就是最大化生成prefer概率。考虑到存在很多没有标注偏好的SFT生成的数据，为了利用这些数据加上 L_{SFT} 。

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

Define ratio between odd: $$\mathbf{OR}_\theta(y_w, y_l) = \frac{\mathbf{odds}_\theta(y_w|x)}{\mathbf{odds}_\theta(y_l|x)}$$

Define loss: $$\mathcal{L}_{ORPO} = \mathbb{E}_{(x,y_w,y_l)}\left[\mathcal{L}_{SFT} + \lambda \cdot \mathcal{L}_{OR}\right] \qquad \mathcal{L}_{OR} = -\log\sigma\left(\log\frac{\mathbf{odds}_\theta(y_w|x)}{\mathbf{odds}_\theta(y_l|x)}\right)$$

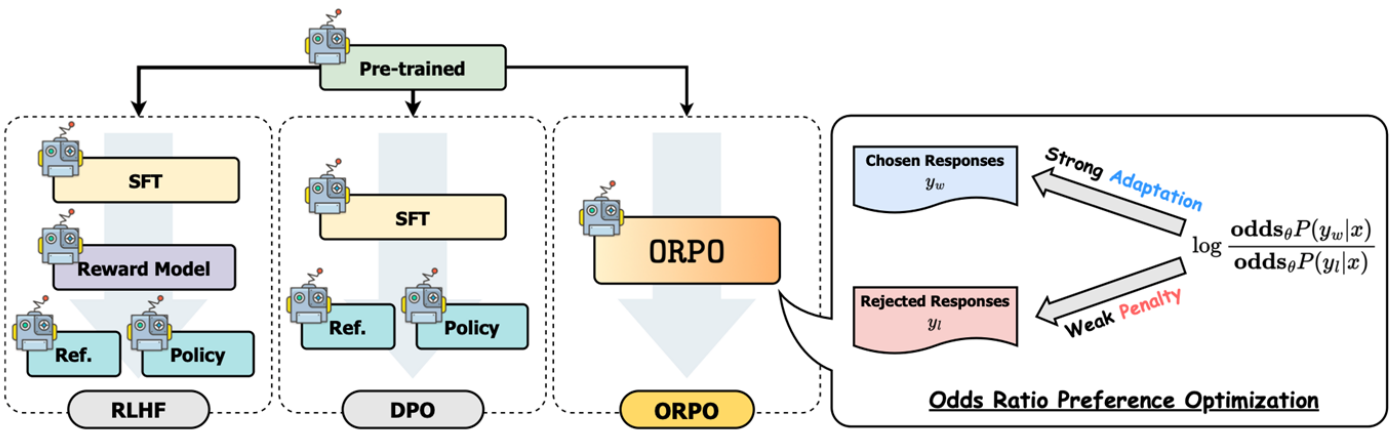


Figure 2: Comparison of model alignment techniques. ORPO aligns the language model *without a reference model* in a single-step manner by assigning a weak penalty to the rejected responses and a strong adaptation signal to the chosen responses with a simple log odds ratio term appended to the negative log-likelihood loss.