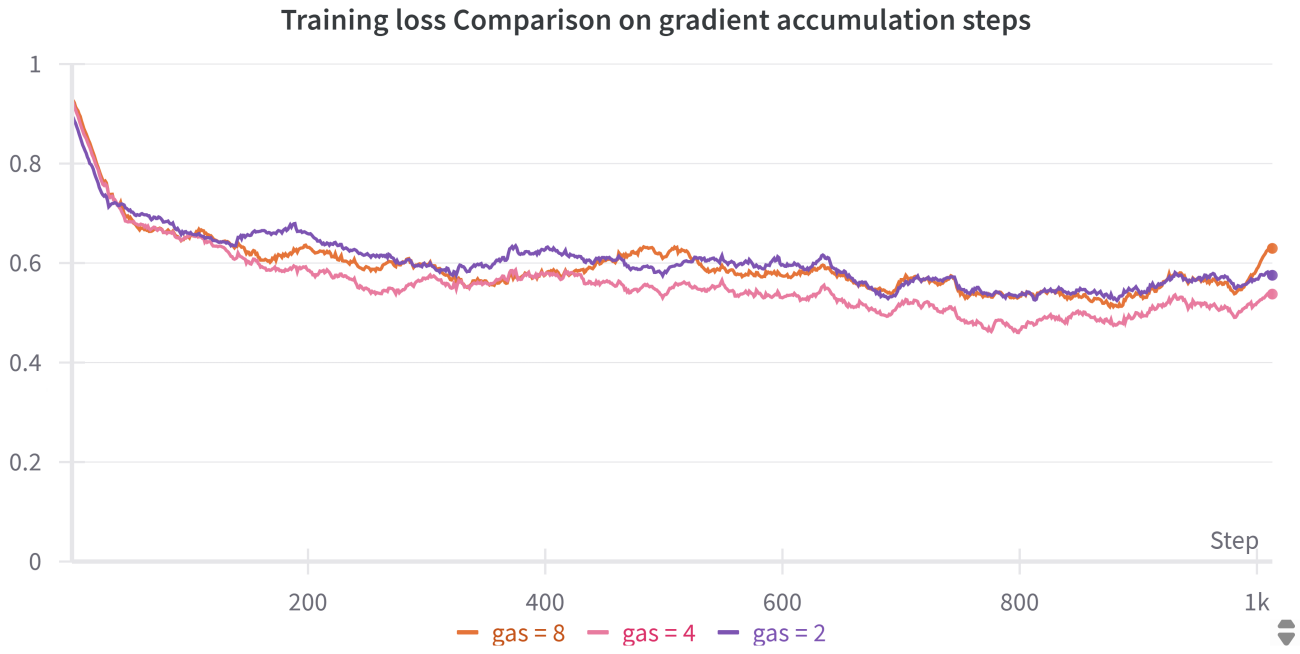
锺易文 2200094605

1.1.3

以下为对超参数进行对比实验：

根据上图实验表示，在Training Loss对比图里，learning rate为2e-5时，training loss最低



根据上图实验表示，在Training Loss对比图里，gradient accumulation steps为4时，training loss最低

因此，在1.1.2的模型训练超参数设为：

- learning rate = 2e-5
- gradient accumulation steps = 4

上图为rm model的 Training Loss

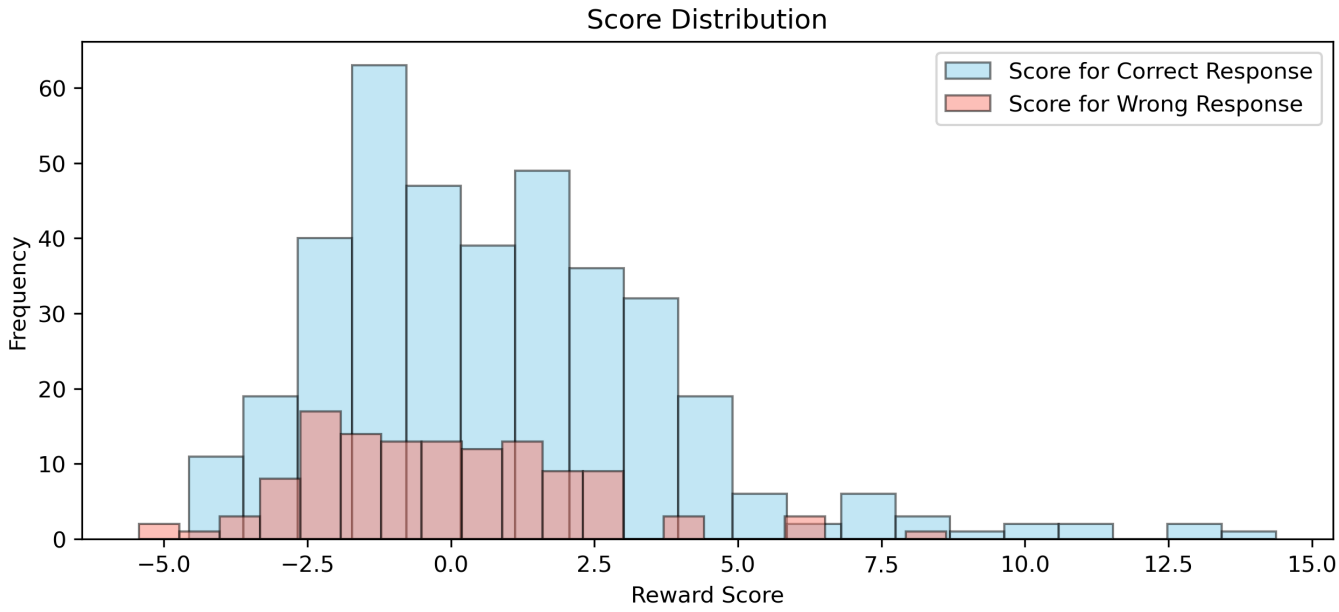
奖励模型 validation 结果为：

- accuracy = 0.758483
- reward_mean = -0.117616
- reward_std = 2.741278

结合Training Loss和 Validation accuracy，模型没有过拟合的现象。

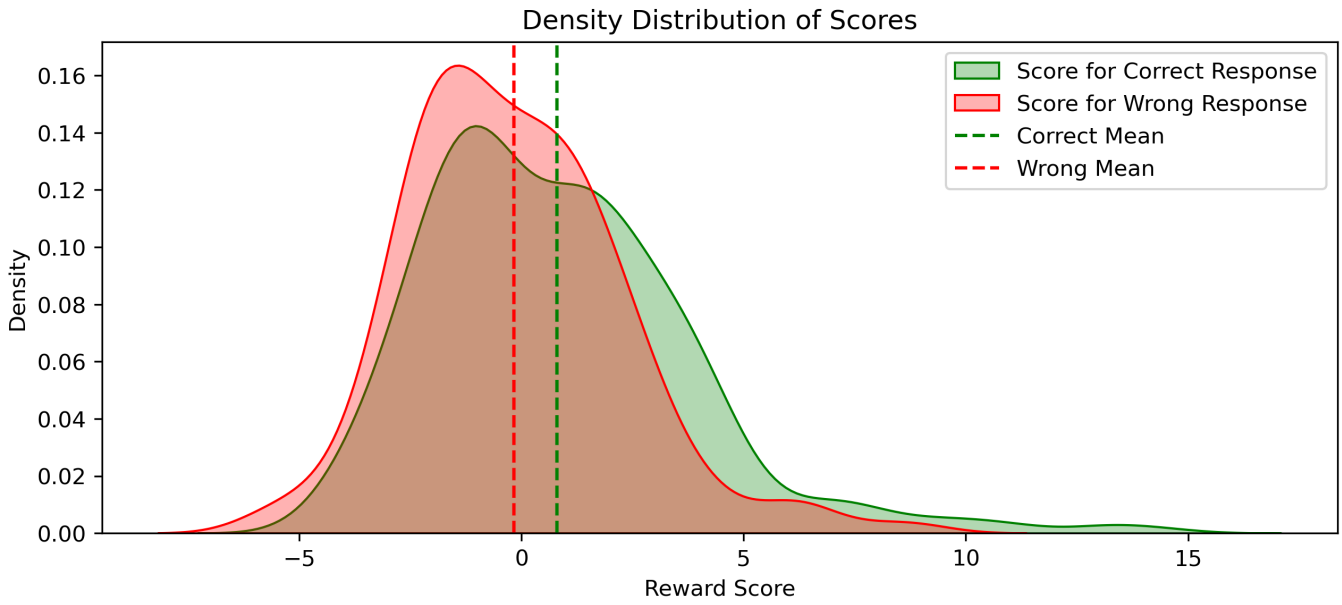
1.1.4

以下数据是对于每一个在验证集里的question，都记录高分者的分数，即Score for Correct Response 表示当偏好的response 得分更高时的分数，相反 Score for Wrong Response表示非偏好的response得分更高时的分数，这样以下的对比图就可以凸显当奖励分别在偏好回答和非偏好回答里评分的分布。



上图为使用训练好的奖励模型进行validation后，可视化问答对的评分频率分布，根据分析可得

- 答对的频率是大于答错的
- 答对的评分分布比起答错的评分分布更偏右，也就是答对时分数更高



上图为分数的KDE图，根据分析可得

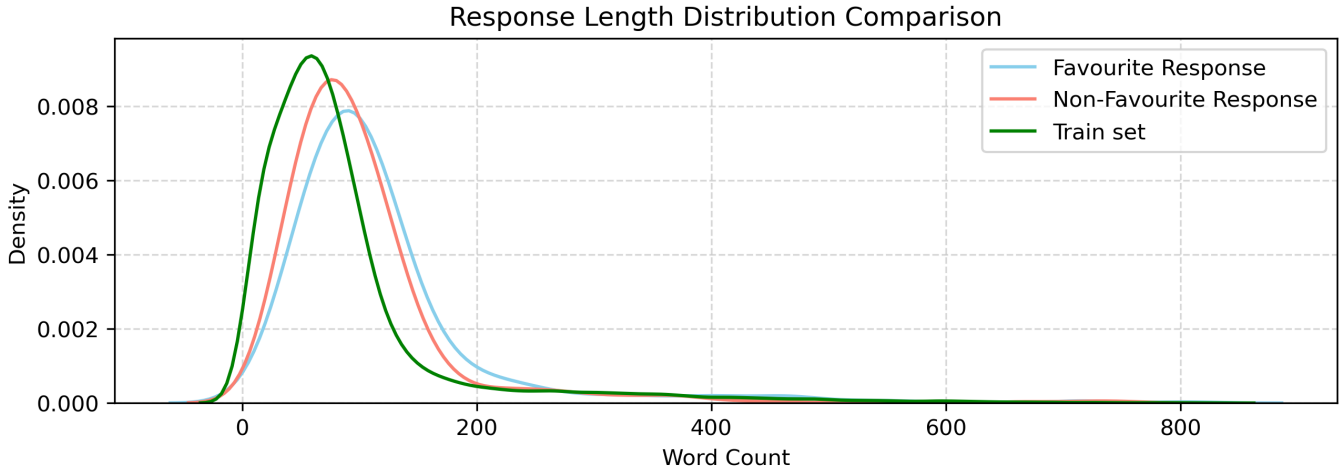
- 两峰分离明显，模型能清晰区分偏好集

使用T-test检验:

- T统计量 = 3.297，这也表示正确回答的分数显著高于错误回答的分数
- p值 = 0.00105，远小于显著性阈值 0.05，说明两组分数的差异具有高度统计学意义。

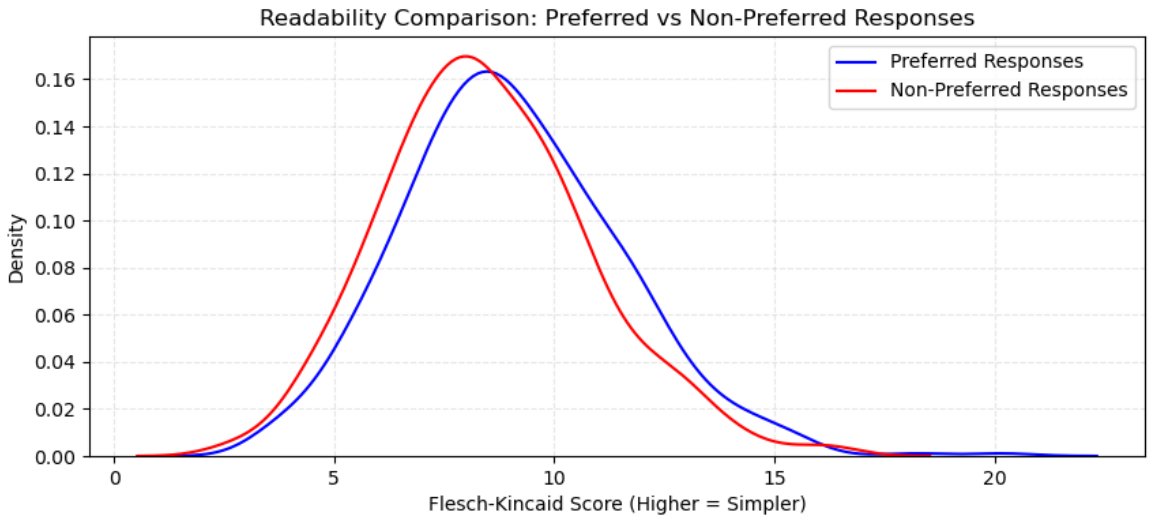
小结：

- 训练好的奖励模型能够区分正确与错误回答，且正确回答的奖励分数显著更高
- p值极低，表明差异不是随机误差导致的，而是模型真实学到了偏好信号。



上图为观察奖励模型是否可能偏好较长回答，即查看在val_1k.parquet里，偏好回答和非偏好回答的单词数作为衡量进行对比，根据KDE图显示：

- 奖励模型打分高的回答的WordCount整体偏右，因此显示模型可能倾向较长的回答



上图为对奖励模型在val_1k.parquet上偏好回答和不偏好回答计算的Flesch-Kincaid可读性，根据图可得：

- 尽管模型可能偏好长回答，但是模型在句式可读性上不是一味地追求复杂长句式而已

使用TF-IDF对比train_30k里的高频字与奖励模型偏好response的高频字：

Top Keywords in Favored Responses: ['additionally' 'alibaba' 'assistant' 'best' 'cloud' 'created' 'data' 'help' 'helpful' 'like' 'make' 'need' 'people' 'person' 'qwen' 'time' 'use' 'user' 'using' 'way']

Top Keywords in Train Set Responses: ['additionally' 'best' 'create' 'data' 'help' 'important' 'include' 'information' 'like' 'make' 'need' 'people' 'person' 'provide' 'sure' 'time' 'use' 'used' 'using' 'way']

可以发现与偏好子集相近

1.2

1. 奖励是由环境给的一种标量的反馈信号，这种信号可显示智能体在某一步采取某个策略的表现如何，因此，奖励建模可以作为引导让智能体学习即称为强化学习。其中，奖励建模的应用可以大致概括为一个搜索模型，打分模型，以及反馈模型的集合。

   在围棋游戏里，搜索模型可以看成是当前棋盘状态下后续可能发生状态的预测，而打分模型可以为这些状态进行打分，然后智能体可以通过这些分数采用不同的策略如minimax，k-步最大奖励等进行步骤的决策。最终，智能体完成行动后再由反馈模型给出执行后的反馈。在这个围棋游戏里，奖励建模就是在为这三个方面进行建模或训练。通过奖励建模再对状态转移矩阵进行修改，在实际运用时就能根据状态转移方程做出决定。

   从LLM的方面来看，LLM一直做的是预测下一个token，因此运用奖励模型就可以让LLM在训练期间以一个分数为标准去判断无标签训练集或无法达到机器学习要求的连续类任务。强化学习里的奖励建模让智能体在环境里最大化它的期望累积奖励，以便能完成博弈，长连贯性序列等任务。

2. 根据参考资料，鲁棒性主要解决的问题是提高策略在面对不确定性或者对抗性攻击场景时的最差性能，也就是在当训练场景所模拟的情况与真实使用时的场景有差距时，模型是否能做出最正确的预测。因此，形式化的表示考虑了鲁棒性的奖励建模应该为求解以下 max-min 问题:

$$\max_{\pi} \min_{U \in F_u} J_M(\pi, U)$$

- π 是策略
- U 是扰动变量且属于集合 F_u
- J_M 是目标函数

照成不确定性的原因有：

- 在真实场景下，对一些关键因素的采集精度或准确度不能达标，导致触发了模型概率模型里错误反应的概率更高
- 真实世界实时充满着各自噪音和不确定性，可能导致模型面对突如其来的意外束手无策
- 模型可能在真实世界下使用工具时产生了偏差无法及时通过反馈进行调整照成错误判断

3. RLHF是alignment的关键方法，但是数据集总是会伴随一个问题，也就是在语言表达方面，长句带来的信息量会比短句更多。因此，在RLHF里一般长句得分会比短句高，也就是说会导致奖励模型常常被误导人类喜欢更长的反应，所以参考文献提出了缓解这个情况的方法，即

   首先，将集群专家分为主要专家（main experts）和偏好专家（bias-only experts），两种专家将同时进行强化学习。接着，在选定为偏好专家的模型上，训练数据将被加入噪音打乱，因此模型无法对训练集本身提取学习特征，也就只能获得长度等外形特征，换句话说也就是"专门学习长度偏差"。此时，两种专家仍然都还对长回答有偏好。因此，接下来只要将主要专家的预测减掉偏好专家的预测概率，就能在不影响基础任务的情况下，去除掉长度偏差。其对应的奖励建模为最大化似然函数为

$$-E_{(x,y)\ D}[log(\sigma(\hat{r}(x,y_i)-\hat{r}(x,y_{1-i})))]$$

   其中：

$$\hat{r}(x,y)=Softmax(log(r_\phi(x,y))+log(r_\psi(x,y)))$$

   r_{phi} 是主要专家的输出，而r_{psi}是偏好专家的输出

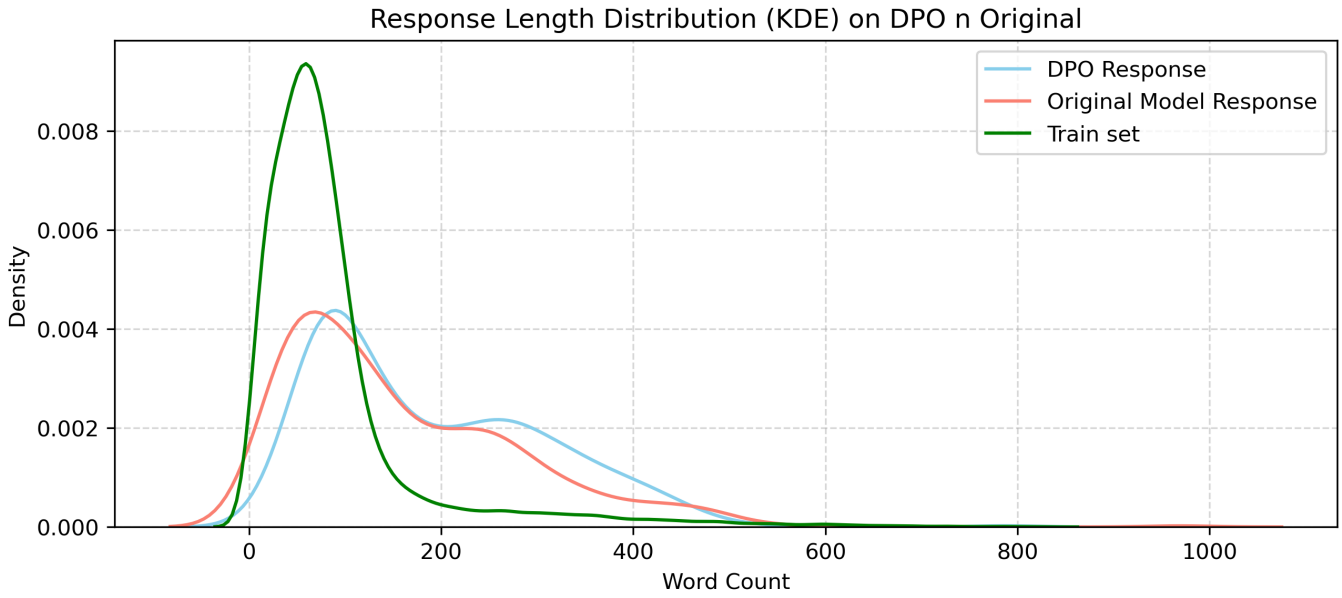
2.1.2

以上为DPO微调的 Trainin Loss

DPO微调模型 validation 结果为：

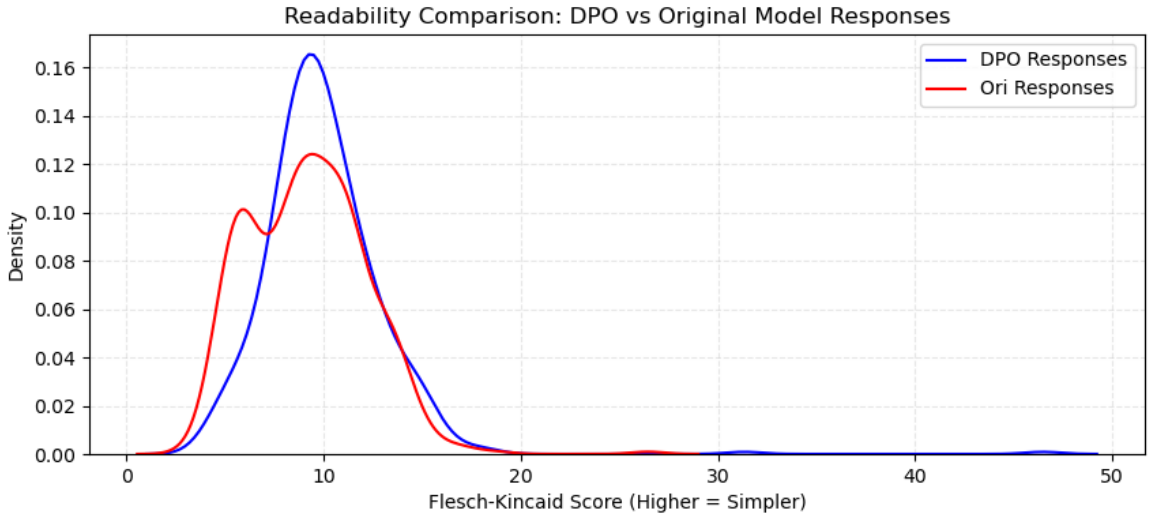
- accuracy = 0.540918
- reward_mean = 0.111244
- reward_std = 1.44942 模型并无明显过拟合行为

1. 下图分析DPO是否改变了初始模型的回答即长度和可读性，并观察是否与偏好数据集一致。即挑选在val_1k.parquet里的500条question，使用大模型的生成脚本让两个模型分别进行回答后，对回答文本的分析



上图为DPO微调模型和初始模型的回答长度对比，根据图可得:

- DPO的回答整体比初始模型长



上图为对比DPO微调模型和初始模型的回答所计算Flesch-Kincaid可读性，根据图可得：

- DPO微调的回答可读性集中，并且不会比初始模型复杂

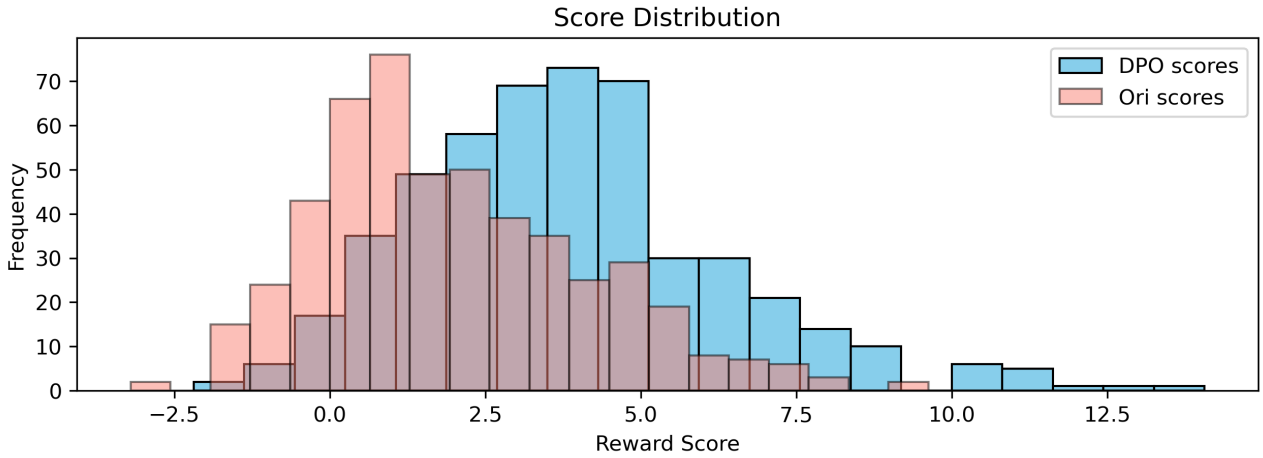
以下使用TF-IDF检验高频词

- Top Keywords in DPO Responses: ['alibaba' 'assistant' 'cloud' 'created' 'help' 'helpful' 'important' 'including' 'information' 'language' 'like' 'need' 'people' 'professional' 'provide' 'qwen' 'remember'

'sorry' 'use' 'user']

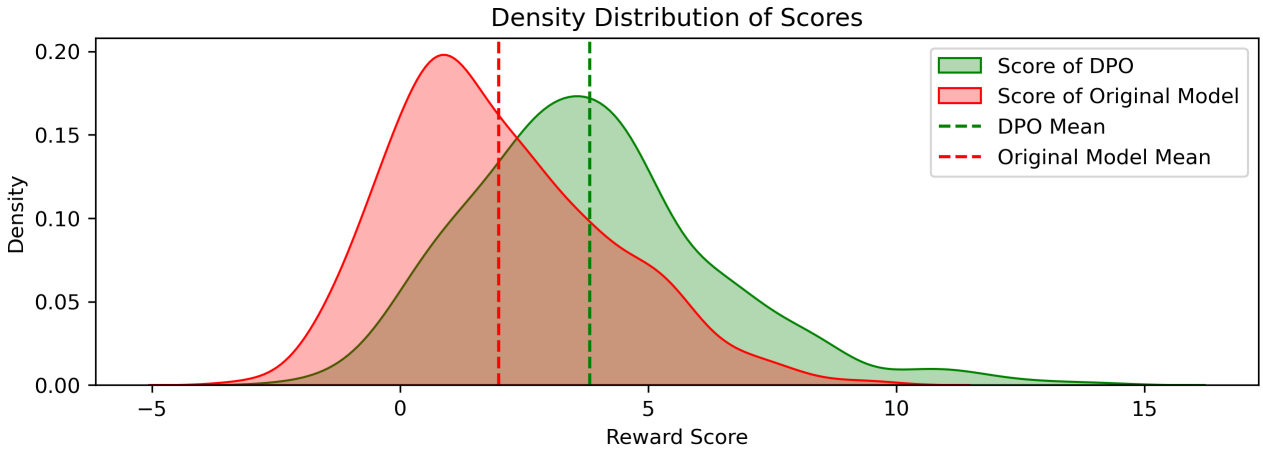
- Top Keywords in Original Model Responses: ['ai' 'alibaba' 'assist' 'assistant' 'cloud' 'created' 'help' 'helpful' 'important' 'information' 'language' 'like' 'mature' 'model' 'people' 'provide' 'qwen' 'sorry' 'use' 'user']
- Top Keywords in Train Set: ['additionally' 'best' 'create' 'data' 'help' 'important' 'include' 'information' 'like' 'make' 'need' 'people' 'person' 'provide' 'sure' 'time' 'use' 'used' 'using' 'way']

2. 以下数据为使用奖励模型为DPO微调模型和初始模型生成的回答分别打分后的结果



上图为分别让DPO和原始模型对问题生成相应回答后，利用奖励模型对这两个回答进行评分的分布比较

- 奖励模型偏好DPO的回答



上图为分数的KDE图，根据分析可得

- 绿色曲线的高峰在4.5分附近 → 多数DPO的回答获4.5分左右
- 红色曲线的高峰在-0.1分附近 → 原始模型的回答集中在-0.1分
- 两峰分离明显，表示两个模型有很大的区别

使用T-test检验:

- T-统计量=12.412，这也表示DPO的分数显著高于初始模型的分数
- p值=5.41e-33，远小于显著性阈值 0.05，说明两组分数的差异具有高度统计学意义。
- 观察两图都可以得到DPO的评分分布整体偏右，也就是表示奖励模型偏好DPO的回答

3. 测试集表现对比

指标	初始模型	DPO微调模型
Accuracy	0.5469	0.5509
Reward Mean	0.7339	-0.2924
Reward Std	1.4068	1.5045

分析DPO微调模型在validation的结果：

- 准确率并无显著提升，
- 奖励均分从正变负，且绝对值大幅下降，DPO把整体打分的高度压低了
- 奖励标准差无显著区分

2.2

1. 从强化学习的角度来看，DPO是一个off-policy方法，因为DPO通过优化一个静态的偏好数据集来训练策略，无需与环境实时交互。此外，DPO的训练过程不需要与环境或人类实时交互，而是直接利用已有的静态数据集，因此，它也是offline方法。

2. DPO的关键见解为DPO将RL问题转化为监督学习，直接通过静态偏好数据训练策略模型，省去显式的奖励建模和RL训练循环。。因此，DPO主要针对传统RLHF的优化有：

对比	传统 RLHF	DPO
奖励模型	需训练显式奖励模型（r_\phi）	无需奖励模型，隐式通过策略优化实现
优化方法	依赖 PPO 等 RL 算法	仅需监督学习损失函数
计算效率	需多次策略-环境交互	单次前向传播计算损失
训练稳定性	可能因 RL 训练不稳定而发散	监督学习方式更稳定

3. DPO和传统RLHF相比的局限性为：

   - DPO 依赖离线偏好数据，无法实时利用新反馈，导致数据覆盖不足时泛化能力受限
   - DPO 奖励函数由策略模型与参考模型的概率比隐式定义，受限于参考模型质量和策略容量
   - DPO 难以直接建模冲突目标，需通过数据混合或后处理实现
   - DPO 的参考模型的偏差会直接影响策略优化方向
   - DPO 依赖逐段偏好标注，对长序列任务的全局一致性建模较弱

4. **KTO对齐**

   - 特点为不直接使用最大期望奖励的决策
   - KTO对齐策略的训练数据是（x，y，if_can_accept）三元组形式，其中x为输入，y为输出，if_can_accept为y是否可以被接受作为输入x的输出，也就是不是直接告诉模型higher_marks和lower_marks。
   - KTO的优势在于对前景理论的引入，即可能是对真实世界更好的逼近，因此可以作为模型初步对世界的建模，也就是说，KTO可能会被用于快速对模型进行预对齐训练。此外，KTO的使用在实际场景中是切实可行的，因为它只要求一个简单的二进制信号来指示输出是否是理想的。

SimPO对齐

- 特点为直接优化偏好对的概率比值，省去参考模型的对数概率计算，简化DPO的损失函数，减少计算开销。
- SimPO 使用长度归一化的奖励，模型不会偏向生成更长的响应
- SimPO 使用显式边界，避免奖励坍缩，确保策略区分度
- SimPo 使用策略模型计算响应中所有标记的平均对数概率来计算，即省去参考模型的对数概率计算，仅需策略模型单次前向传播
- SimPO的优势在于在对话生成中，模型更关注内容相关性而非堆砌冗余文本。此外，去除参考模型计算，训练速度大大提升

ORPO

- 特点为将对齐过程直接融入监督微调（SFT）阶段，从而减少传统RLHF（如PPO）或对比学习（如DPO）所需的额外训练成本
- ORPO 引入胜率比，显式区分“期望输出”和“不期望输出”，解决了传统SFT在偏好对齐中的缺陷，即传统SFT隐式提高所有相关输出的概率以抑制不期望输出，而ORPO通过胜率比显式惩罚负例，避免模型生成错误但概率高的内容。

备注： 在results里保存了

- Rm_model_mark_on_val_1k.json：这是奖励模型对val_1k里回答的打分
- Response_of_DPO_n_Ori_val_with_rm.json: 这是使用DPO微调模型和初始模型对挑选了val_1k里500个question进行回答后再让奖励模型打分的数据集