RLHF

RLHF的定义

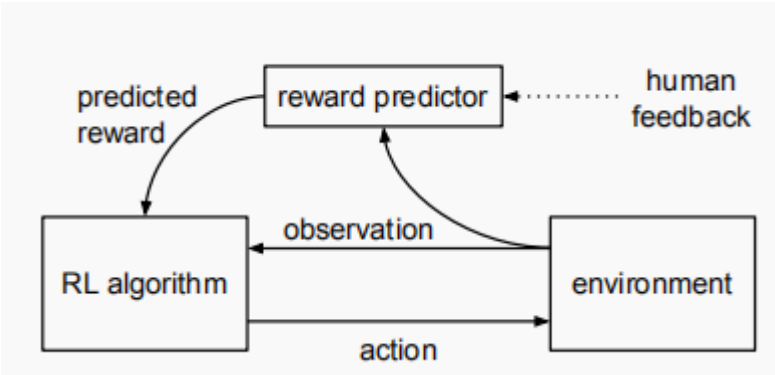
RLHF的定义首先由《Deep Reinforcement Learning from Human Preferences》提出，解决了强化学习中奖励函数怎么设计的问题。

传统RL的问题：依赖于人类设计的奖励函数，这在实际应用中往往非常困难且不够灵活，因为面临涉及复杂、定义不明确或难以指定的目标的任务时，很难定义奖励函数。而不正确或者有偏的奖励函数会导致**reward hacking**问题，导致训练出的模型不符合预期。而让人类实时的对模型当前行为进行反馈的成本又过高。

RLHF提出从人类给出的反馈中学习奖励函数，该奖励函数满足以下需求：

1. 能够解决人类 **只能识别所需行为** ，但不一定能提供演示的任务
2. 允许 **非专家用户** 进行示教
3. 能扩展到大规模问题
4. 用户给出反馈的成本不高

如下图所示，利用人类偏好来拟合奖励函数，同时利用RL算法优化当前预测的奖励函数。人类比较两个agent的行为轨迹片段哪个更好，而不是提供绝对数值分数。在某些领域中，人类更擅长比较agent的性能而不是给出绝对的评分，这种比较也能学习到人类偏好。比较agent轨迹片段与比较单个状态几乎一样快，比较轨迹片段明显更有帮助。**在线地收集反馈可以提高系统的性能，并防止agent利用学到的奖励函数的弱点刷分。**



训练目标

考虑agent与环境的交互为：在每个时刻 t，代理从环境中接收到一个观测 $o_t \in \mathcal{O}$ (这份o跟状态s是一个东西，这里采用原论文的denotation)，然后向环境发送一个动作 $a_t \in \mathcal{A}$。环境不产生奖励，而是人类给出对轨迹片段之间的偏好。

轨迹片段：观察和动作组成的序列 $\sigma = \big((o_0, a_0), (o_1, a_1), ..., (o_{k-1}, a_{k-1})\big) \in (\mathcal{O} \times \mathcal{A})^k$ 。用符号 $\sigma^1 \succ \sigma^2$ 表示人类相比轨迹段 σ^2 更偏好轨迹段 σ^1 。agent的目标是生成人类喜欢的轨迹，同时向人

类发出尽可能少的查询.

训练流程

训练一个策略模型 $\pi : \mathcal{O} \rightarrow \mathcal{A}$ 和一个奖励模型 $\hat{r} : \mathcal{O} \times \mathcal{A} \rightarrow \mathbb{R}$，这两个模型都是神经网络。训练分为三个步骤，这三个步骤交替进行，所以也是一种在线RLHF（Online RLHF）：

1. 策略 π 与环境相互作用产生一组轨迹 $\{\tau^1, \tau^2, ..., \tau^i\}$。 π 的参数通过传统的 RL 算法进行更新，以最大化预测奖励 $r_t = \hat{r}(o_t, a_t)$ 的总和
2. 我们从第一步产生的轨迹 $\{\tau^1, \tau^2, ..., \tau^i\}$ 中选择成对的轨迹片段 (σ^1, σ^2)，并将其发送给人类进行比较
3. 奖励模型 $\hat{r}$ 的参数通过监督学习进行优化，以适应迄今为止收集的人类比较

策略优化

将 $\hat{r}$ 产生的奖励分布 normalize 为零均值和常数标准差，奖励函数 $\hat{r}$ 可能是**非平稳**的（就是可能随时间变化），所以使用对奖励函数的变化鲁棒的方法，采用了策略梯度算法（Actor-critic 和 TRPO）。

奖励拟合

将奖励函数视为解释人类判断的潜在因素，人类偏好片段 σ^i 的概率与潜在奖励在该片段长度上的总和呈指数相关，将奖励函数估计值 $\hat{r}$ 解释为偏好预测器。基于Bradley-Terry 模型，人类偏好片段 σ^1 超过 σ^2 的概率:

$$\hat{P}[\sigma 1 \succ \sigma 2] = \frac{exp \sum \hat{r}(o_t^1, a_t^1)}{exp \sum \hat{r}(o_t^1, a_t^1) + exp \sum \hat{r}(o_t^2, a_t^2)}$$

训练损失为这些预测和实际人类偏好之间的**交叉熵损失**, $(\sigma^1, \sigma^2, \mu)$ 表示人类的偏好，$\mu = \{0, 1, 0.5\}\ if\ \{\sigma^1 \succ \sigma^2, \sigma^2 \succ \sigma^1, \sigma^1 = \sigma^2\}$。

$$loss(\hat{r}) = - \sum_{(\sigma^1, \sigma^2, \mu) \in \mathcal{D}} \mu(\sigma^1 \succ \sigma^2) log \hat{P}[\sigma^1 \succ \sigma^2] + \mu(\sigma^2 \succ \sigma^1) log \hat{P}[\sigma^2 \succ \sigma^1]$$

计正样本和负样本为 σ^+, σ^-，上述的损失可以写为：

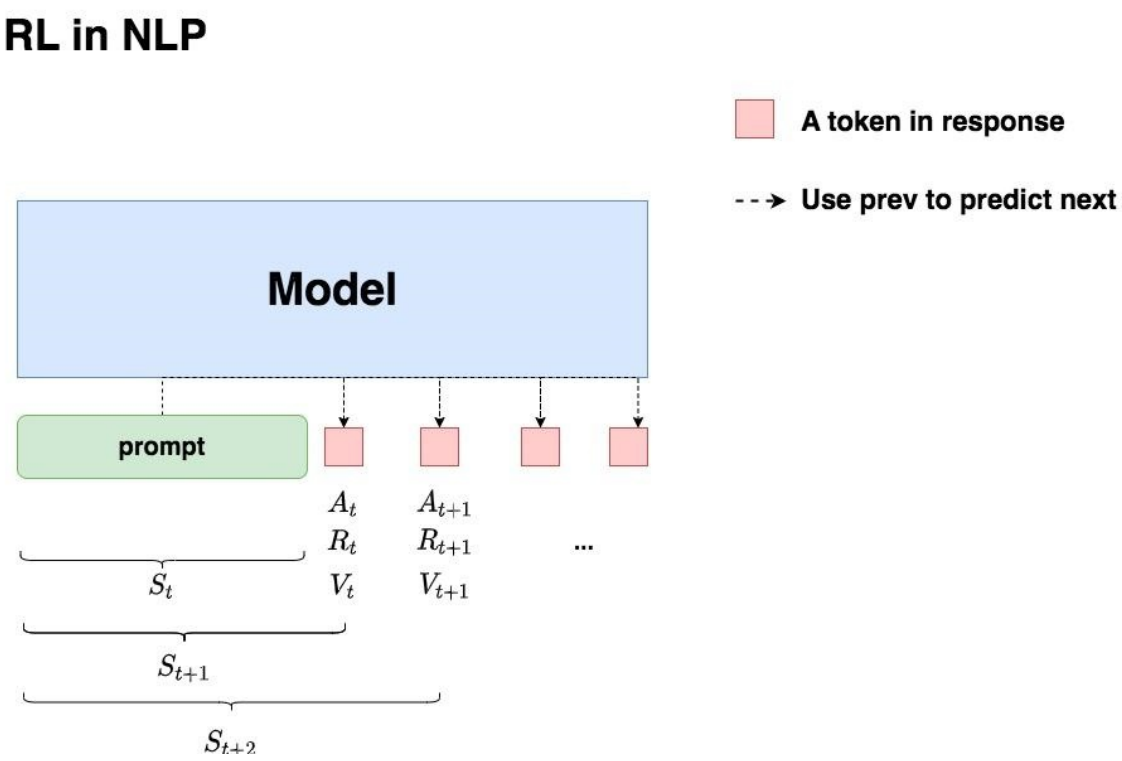
$$loss(\hat{r}) = -E_{(\sigma^+, \sigma^-, y, \in \mathcal{D})}[log \hat{P}[\sigma^+ \succ \sigma^-]] =$$
$$-E_{(\sigma^+, \sigma^-, y, \in \mathcal{D})}[log \frac{exp \sum \hat{r}(o_t^+, a_t^+)}{exp \sum \hat{r}(o_t^+, a_t^+) + exp \sum \hat{r}(o_t^-, a_t^-)}]$$

还做了以下的改进：

1. 我们拟合了一组预测因子，每个预测因子都使用从 $\mathcal{D}$ 中有替换地抽取的 $|\mathcal{D}|$ 个三元组进行训练（这说的就是是 `Bootstrapping自助抽样法`）。独立地对这些预测值进行归一化，然后对结果取平均来定义 $\hat{r}$
2. 数据的一小部分（$\frac{1}{e}$）被用作每个预测值的验证集。我们使用 l_2 **正则化**，调整正则化系数，使验证损失保持在训练损失的1.1到1.5倍之间。在某些领域中，我们还应用 dropout 进行正则化

3. 我们假设人类有 10% 的概率给出随机均匀响应，而不是像第一个公式中直接应用softmax。从概念上讲，这种调整是必要的，因为人类评分员有一个恒定的出错概率，即使是奖励差异变得极端时，错误概率也不会衰减为0.

NLP 场景下的强化学习



> 本章有些图中用A表示动作，本章的文字部分动作表示为a，A表示优势。

目的：给模型一个prompt，模型能生成符合人类喜好的回复。在时刻t产生一个token，t+1时刻的输入是t时刻的输入+t产生的token，如此自回归的生成。接下给出在NLP场景下强化学习的一些概念：

- **策略** $\pi(a_t|s_t)$: 在状态 s_t 下生成token a_t 的概率
- **动作** a_t **：** 生成的token，动作空间是词表，
- **状态** s_t ： 用户输入的prompt+t时刻前生成的所有token
- **奖励** R_t ： 在状态 s_t 下生成token a_t 的即时奖励。
- **价值** V_t ： 状态价值函数，包含着即时奖励和未来期望奖励， $V_t = E[G_t|s_t] = E(R_t + \gamma G_{t+1}|s_t)$

注意NLP场景下状态转移是确定的，也就是在状态 s_t 下生成token a_t 转移到的下一个状态 s_{t+1} 是固定的，即状态 s_t 拼接生成的token a_t 。

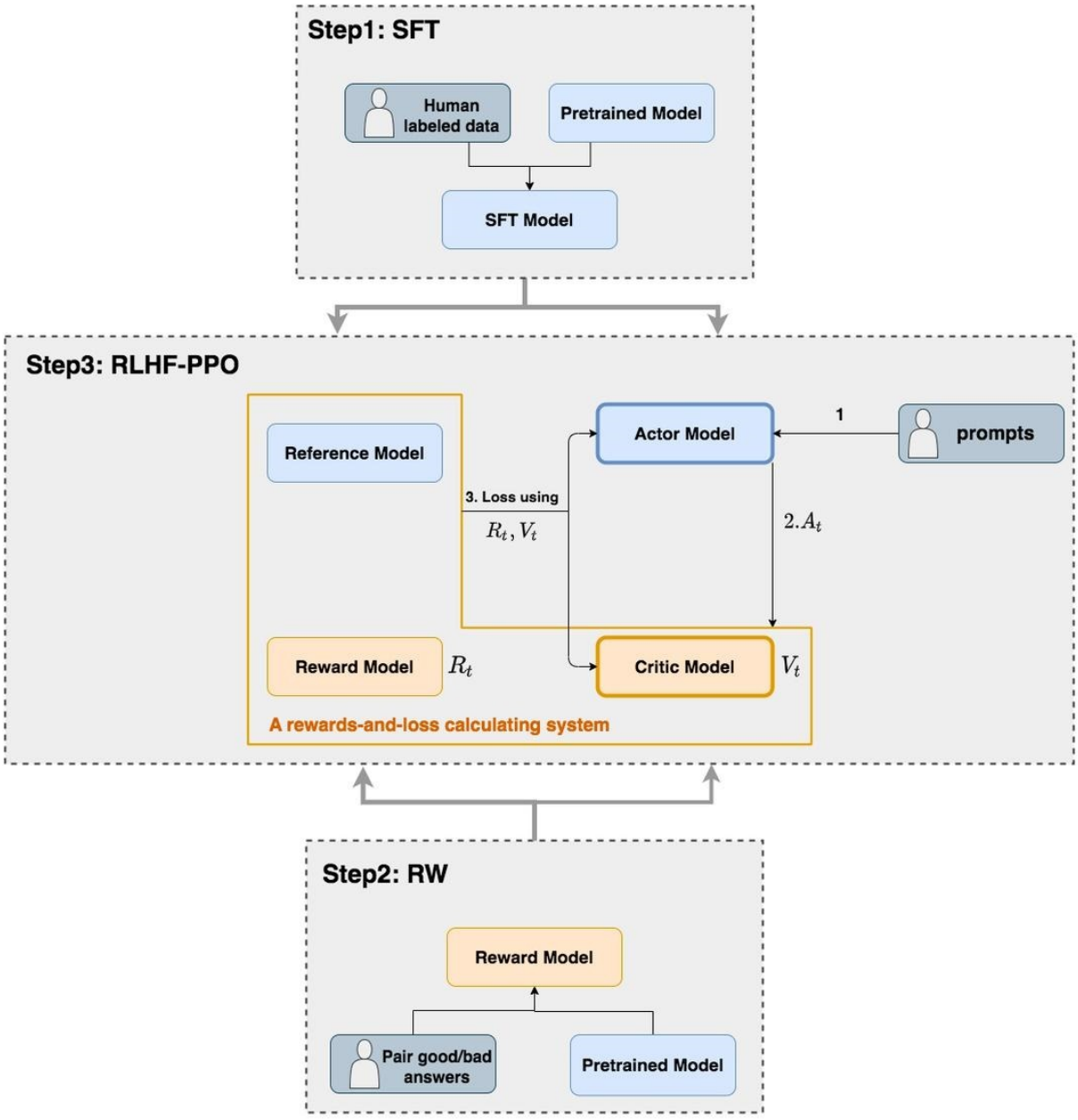
- **训练目标**：最大化期望回报，其中 x, y 分别表示用户查询和模型输出：

$$\max_{\pi} E_{x\sim\mathcal{D}, y\sim\pi}[R(x, y)]$$

更常见的训练目标则是带有行为约束的强化学习，对策略更新幅度进行了限制，π_{ref} 一般采用上一轮的 π 策略。

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

RLHF-PPO



RLHF-PPO包含四个模型，其中**Actor/Critic Model**需要训练，**Reward/Reference Model**不用训练

- **Actor Model**：策略模型，要训练的目标语言模型，Actor模型直接用SFT训练出的模型进行初始化
- **Critic Model**：评论家模型，预计期望总收益，Critic模型在SFT模型的基础上加数值头后训练得到。

- **Reward Model**：奖励模型，计算即时收益，Reward模型在SFT模型的基础上加数值头后训练得到（这里是指奖励模型训练，不是指RLHF-PPO训练）
- **Reference Model**：参考模型，它的作用是在RLHF阶段给语言模型增加一些“约束”，防止语言模型训歪。Reference 模型直接用SFT训练出的模型进行初始化

Actor Model

Actor模型直接用SFT训练出的模型进行初始化，训练目的是让Actor模型能产生符合人类喜好的回复，输入给Actor模型用户查询，将产生的回复和用户查询一起计算损失，用于更新Actor模型。

这里参照猛猿大佬的思路，讨论一下Actor模型的损失函数是怎么推导出来的。

1. 直观上来看，损失可以设计为 $Actor\_loss = -\sum_{t}^{T} V_t log\pi(a_t|s_t)$ ，当 $V_t > 0$ 时，表示Critic 模型对Actor模型采取的工作进行正反馈，就需要增大 $\pi(a_t|s_t)$ ，以降低loss，反之亦然。之后都省略求和号。
2. V_t 是Critic模型预测出来的，采取行动 a_t 后的实际收益为 $Q(s_a, a_t) = R_t + \gamma V_{t+1}$ ，采用优势 $R_t + \gamma V_{t+1} - V_t$ 以使得方差更小更平滑，损失就变成了 $Actor\_loss = -A_t log\pi(a_t|s_t)$
3. 对 R_t 改造。在NLP场景下中，奖励或者人类偏好往往是句子级别的，也就是可以用最后一个时刻的奖励 R_t 表示整个句子的奖励：

- 当 t≠T 时，我们更加关心Actor是否有在Ref的约束下生产token a_t (T表示最后一个时刻)
- 当 t=T 时，我们不仅关心Actor是否遵从了Ref的约束，也关心真正的即时收益 R_t

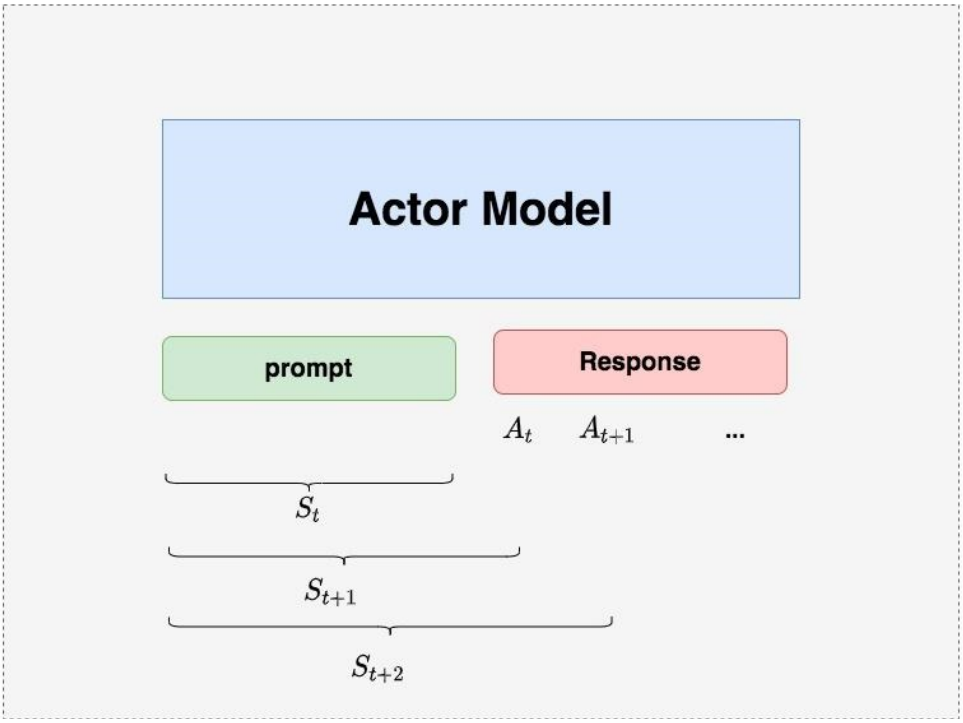
$$R_t = \begin{cases} -kt\_ctl * (log\frac{\pi(a_t|s_t)}{\pi_{ref}(a_t|s_t)}), t \neq T \\ -kt\_ctl * (log\frac{\pi(a_t|s_t)}{\pi_{ref}(a_t|s_t)}) + R_t, t = T \end{cases}$$

其中 kt_ctl 是缩放因子，deepspeed-chat中设置为0.1，

在Reward模型的训练阶段（这个训练不是指PPO训练，而是在SFT模型的基础上要训练才能得到一个奖励模型），就用最后时刻的 R_t 表示对整个用户查询+回复的奖励值。其余时刻的即时奖励就用Actor是否遵循了Ref的约束。

4. 对 A_t 改造。采用**GAE**，采用广义优势估计来平衡方差和偏差，$A_t = (R_t + \gamma V_{t+1} - V_t) + \gamma\lambda A_{t+1}$ ， λ 是控制因子。
5. 引入**重要性采样**，让一个batch的数据能复用多次。 $Actor\_loss = -A_t \frac{\pi(a_t|s_t)}{\pi_{old}(a_t|s_t)}$ ，这里去掉了log，原因参考《详解近端策略优化(ppo，干货满满)》
6. 引入**clip**，避免 π_{ref} 与 π 距离过大。

$$Actor\_loss = -min(A_t \frac{\pi(a_t|s_t)}{\pi_{old}(a_t|s_t)}, A_t clip(\frac{\pi(a_t|s_t)}{\pi_{old}(a_t|s_t)}, 1-\epsilon, 1+\epsilon))$$



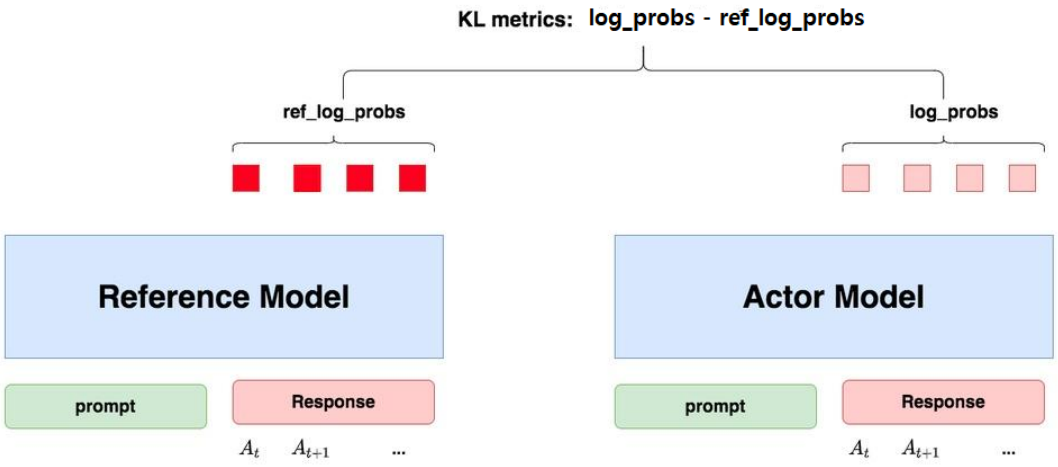
Reference Model

Reference模型在训练过程中保持冻结，目的是防止Actor模型训练歪，也就是不要与SFT模型差异过大。

如下图所示，Actor模型对用户输入生成回复，过程中会为每个token生成对应的log_probs，Reference模型类似生成ref_log_probs，优化目标为：

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

这里不是严格的等于，只是KL散度的近似，这个值越小意味着两个分布的相似性越高。



Critic Model

Critic Model用于预测期望收益 V ，需要参与训练，提升模型对人类喜好量化判断的能力，用奖励模型初始化。

设置待更新的评估模型为 V_t^{old} ，由GAE的定义，递归展开有：

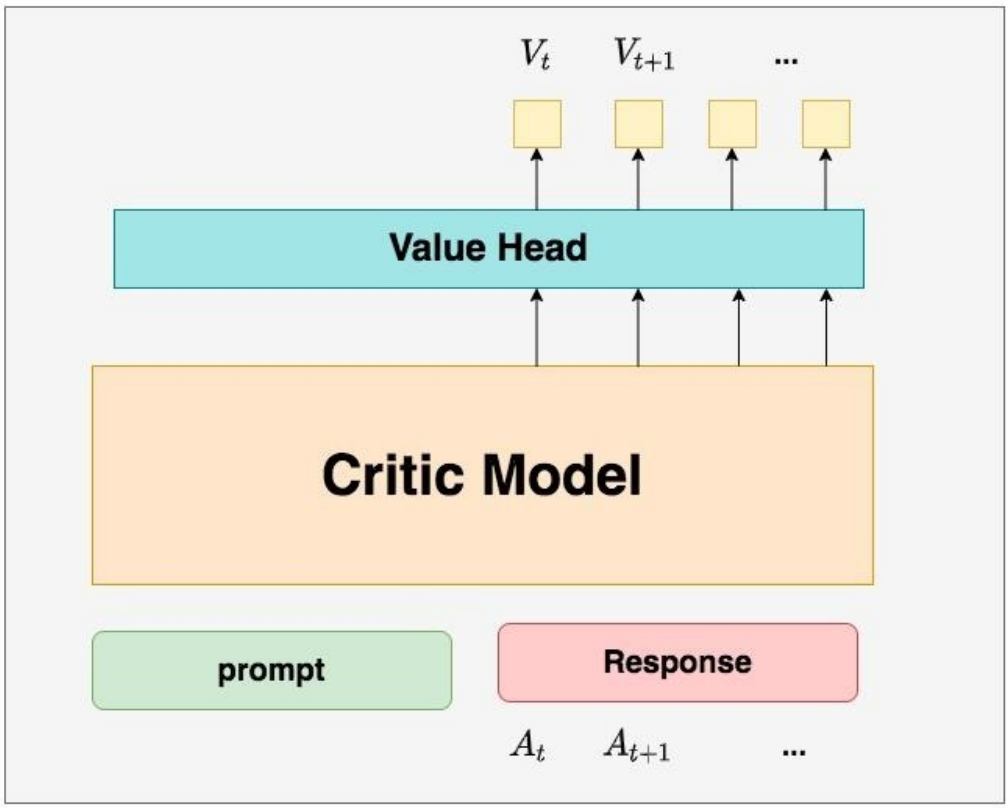
$$A_t^{GAE} = \delta_t + \gamma A_{t+1}^{GAE} = (r_t + \gamma V_{t+1}^{old} - V_t^{old}) + \gamma\lambda A_{t+1}^{GAE}$$

$$R_t = A_t^{GAE} + V_t^{old} = (r_t + \gamma V_{t+1}^{old}) + \gamma\lambda A_{t+1}^{GAE}$$

优势可以从后往前推导出每个动作的优势。评估模型的损失函数为：

$$L_{loss} = ((V_t^{new} - R_t)^2 = [(V_t^{new} - (r_t + \gamma V_{t+1}^{old})) - \gamma\lambda A_{t+1}^{GAE}]^2$$

与Actor-Critic模型的评估模型损失一致，只不过这里引入了GAE对优势做了指数平滑而已。在此基础上还加了**clip**操作 $V_t^{clip} = clip(V_t^{new}, V_t^{old} + \epsilon, V_t^{old} - \epsilon)$ ，类似于策略模型中的clip操作。



Reward Model

计算生成token a_t 的即时收益，参数冻结。只取最后token处的奖励值，作为整个句子的奖励 $R(x, y)$ 。

采用**对比学习**的方式进行训练，针对同一个用户问题，生成了prefer的回复 y_w 和dispref的回复 y_l ，将查询和回复拼接起来交给奖励模型进行评分，训练目标是最大化模型对prefer回复的奖励与对dispref回复之间奖励的差值，奖励模型训练的损失为：

$$Reward\_loss = -E_{(x,y_w,y_l)\in\mathcal{D}}[log\sigma(R(x,y_w) - R(x,y_l))]$$

其中 x, y_w, y_l 表示用户输入，偏好输出(正样本)和不偏好输出(负样本)， $\sigma(x) = \frac{1}{1+e^{-x}}$ ，带入 $\sigma(R(x,y_w) - R(x,y_l))$ ，奖励为：

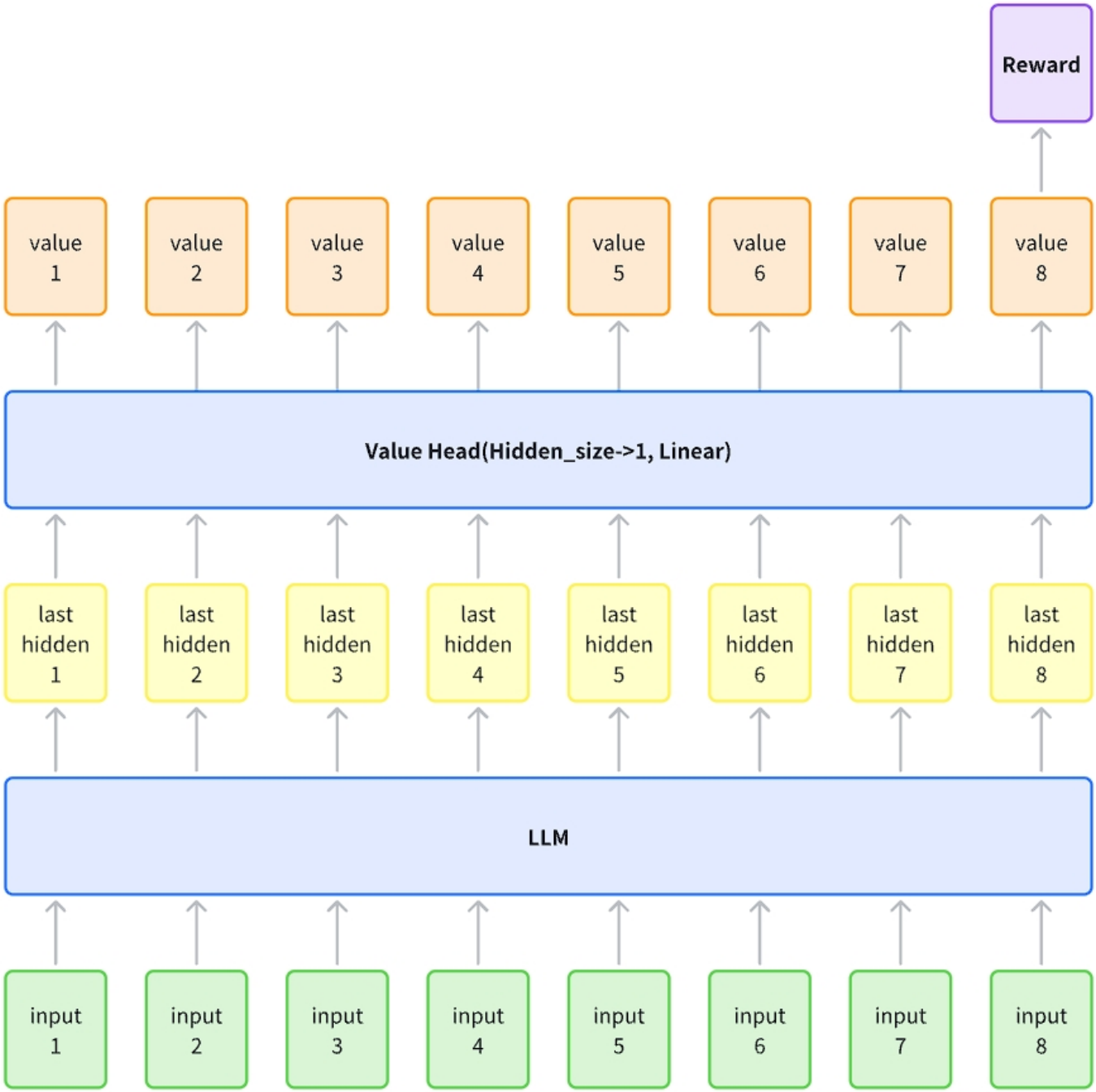
$$Reward\_loss = -E_{(x,y_w,y_l)\in\mathcal{D}}[log\frac{exp(R(x,y_w))}{exp(R(x,y_w)) + exp(R(x,y_l))}]$$

这里的奖励与传统RLHF中的损失不同,传统RLHF考虑了一个trajectory中所有(s, a)对，并且初始状态可以不同：

$$loss(\hat{r}) = -E_{(\sigma^+,\sigma^-,y,\in\mathcal{D})}[log\frac{exp\sum\hat{r}(s_t^+,a_t^+)}{exp\sum\hat{r}(s_t^+,a_t^+) + exp\sum\hat{r}(s_t^-,a_t^-)}]$$

而在nlp领域只针对整个恢复计算奖励，初始的用户输入为 x 相同，因为奖励或者人类偏好往往是句子级别的。

可以**将用户查询到模型回复视为单步MDP**：即 s_0 = 用户输入，a_0 = 模型回复 ，每个状态和动作都是一个token序列，整个trajectory只有一个动作，第一个动作的奖励 $R(x,y) = R_0(x,y)$



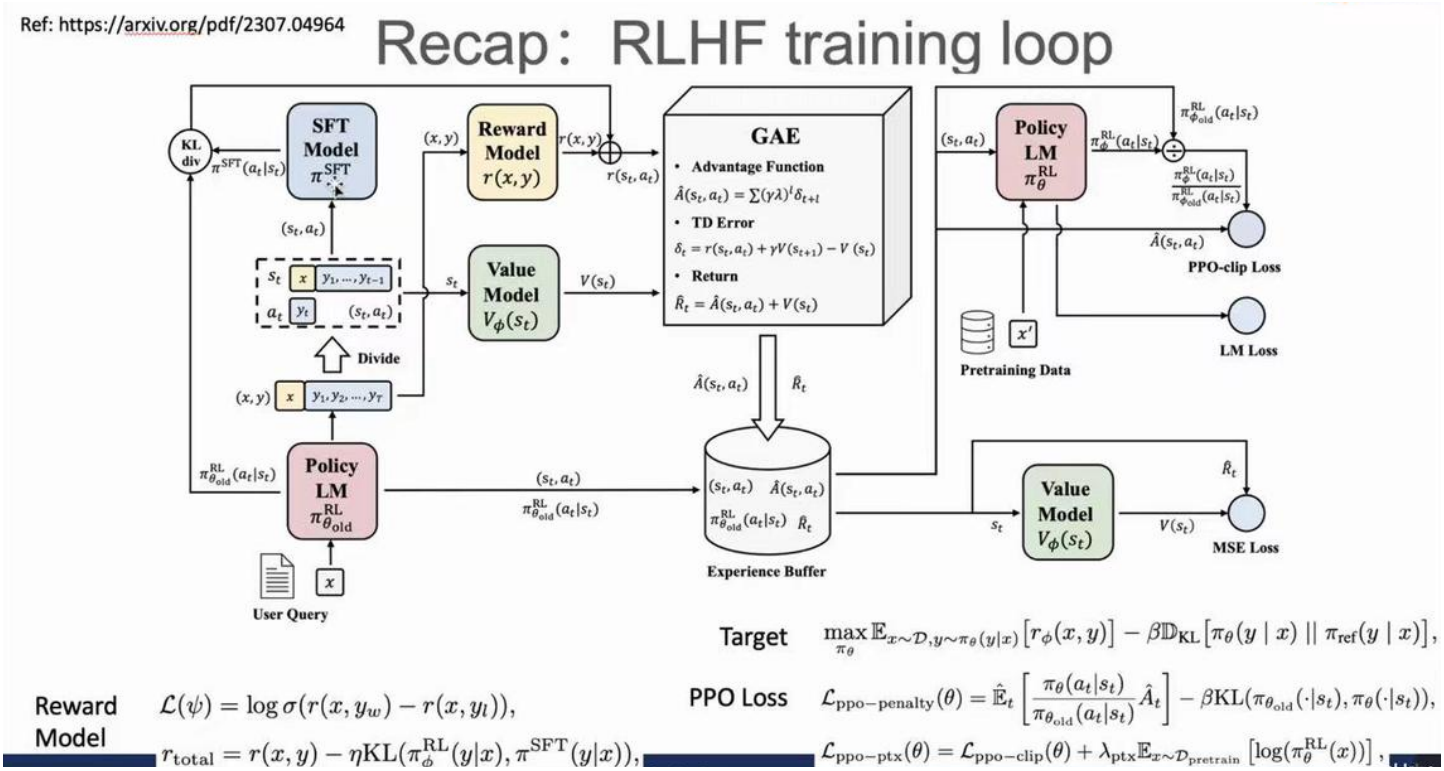
RLHF-PPO总结

RLHF-PPO的核心思想是通过限制策略更新的幅度，确保新策略与旧策略的差异不会过大，从而稳定训练，包含以下4个模型：

- **Actor Model**：策略模型，这就是我们想要训练的目标语言模型。
- **Reference Model**：参考模型，它的作用是在RLHF阶段给语言模型增加一些“约束”，防止语言模型训歪。我们希望训练出来的Actor模型既能达到符合人类喜好的目的，又尽量让它和SFT模型不要差异太大。即希望两个模型的输出分布尽量相似，通过与Actor Model之间的KL散度控制。
- **Critic Model**：评估模型/价值模型，它的作用是期望回报，在RLHF中，我们不仅要训练模型生成符合人类喜好的内容的能力（Actor），也要提升模型对人类喜好量化判断的能力（Critic）。

- **Reward Model**：奖励模型，它的作用是计算即时收益。奖励模型可以是人为规定的，也可以用神经网络实现。

其中Actor和Critic Model是需要训练的，Reward和Reference Model是参数冻结的。Actor和Reference model用同一个sft模型初始化，Reward 和Critic model用同一个奖励模型初始化。



RLHF-PPO训练流程：

1. 以**sft模型**初始化Reference和Actor模型，sft模型使用有监督数据集训练得到。格式为**instruction+input+output**，input是用户输入，output是标签，instruction是任务描述。
2. 以**奖励模型**初始化Critic模型，奖励模型使用偏好数据集训练得到，格式为**instruction+input+chosen+rejected**，instruction是任务描述，input是输入，chosen是正确答案，rejected是sft模型的输出。
3. 使用初始的模型，采样一些mini-batch的input生成轨迹，初始化缓冲区（包含**输入、输出、状态、动作、奖励、下一状态**）。
4. 从缓冲区采样一个**mini-batch**的input，由Actor模型生成输出，Critic模型计算value，Reward模型生成奖励，然后更新actor和critic模型。
5. 更新缓冲区的数据，回到步骤3。

RLHF-PPO的优缺点

- **优点**：clip操作能有效防止策略突变，稳定性高；支持数据复用，off-policy；训练相较于TRPO更简单，训练效率更高。
- **缺点**：对超参数敏感（如 ϵ、λ）;需要维护4个模型，对显存需要较高；clip机制是经验性设计，缺乏理论保证。

⚽ **RLHF-PPO的一些额外的知识点：**

1. 在InstructGPT中，在Actor模型的损失函数上又加了一项在预训练数据集上的损失，避免遗忘预训练学到的知识，也被称为**减轻对齐税（这个问题在小模型上更明显）**。
2. **对奖励进行归一化**能稳定训练
3. **对优势标准化**能稳定网络训练，特别是在分布式训练中。
4. **Reward hacking**：模型可能会通过添加特定的词语或短语来使度量对齐的指标获得高分，但降低了语言的整体质量。