

Post-training of LLM

科普版

飞桨产品经理 孙钟恺

前置知识

你需要提前了解什么？

- Tokenizer；
- Transformer 架构，以及 decode-only 的 Transformer；
- 概率论、线性代数的一些基础知识；
- 深度学习的一些基础概念。

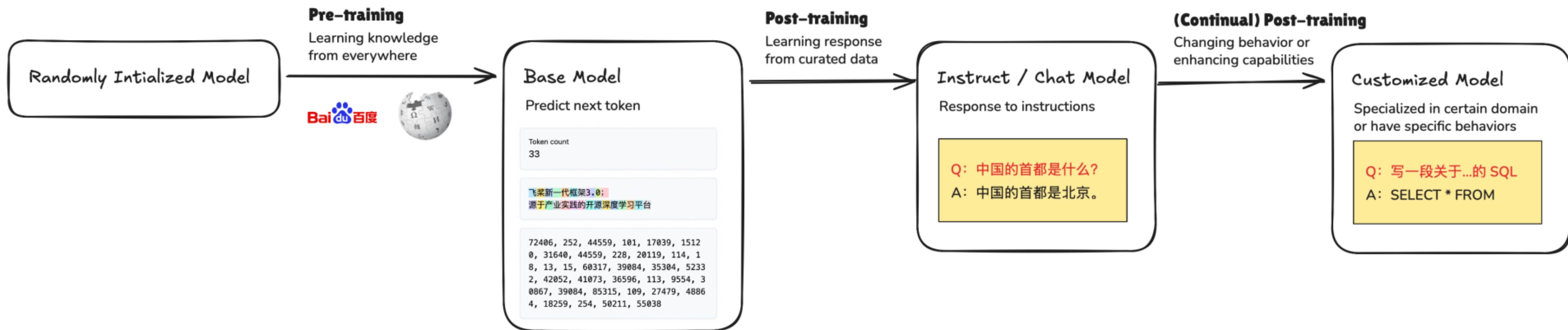
目录

本次课程将按以下章节顺序讲解

1. Introduction
2. SFT
3. DPO
4. Online RL
5. Conclusion
6. Related Advanced Work

Introduction

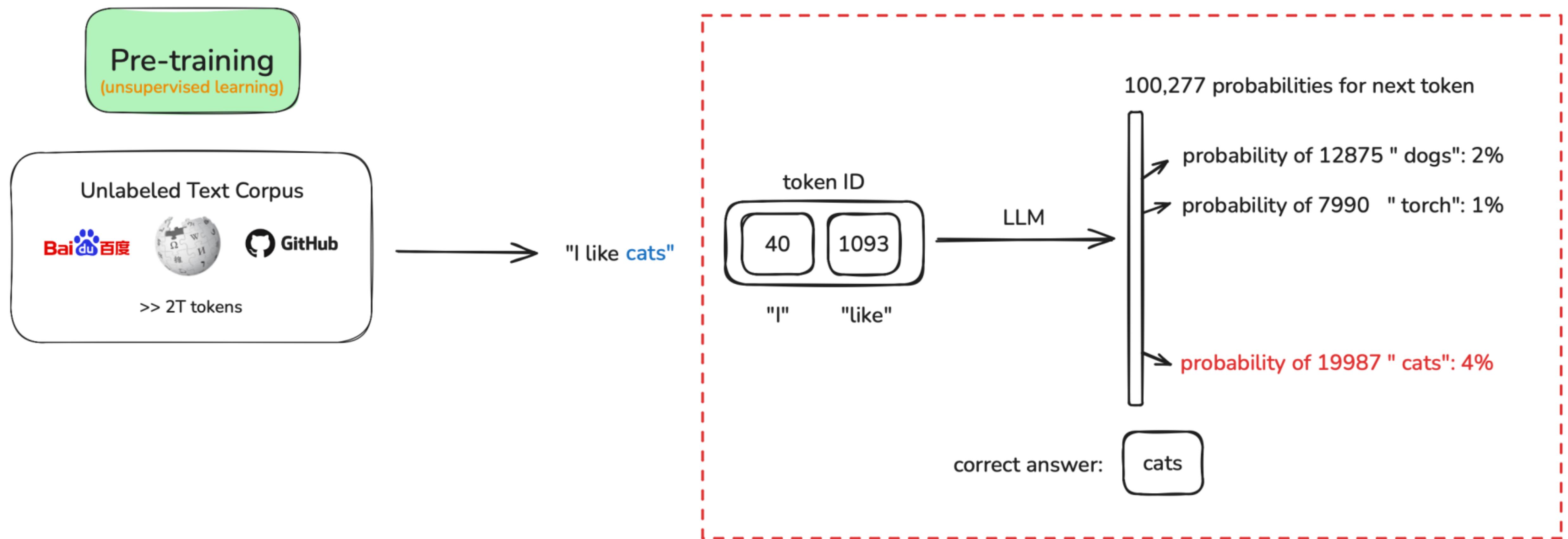
一个 LLM 从无到有的全流程



Introduction

Pre-training

预训练阶段的主要任务是预测句子中的下一个 token



Introduction

Pre-training

以这个例子为例，我们希望模型在预训练阶段能够：

- 在看到 “I” 的情况下，预测出 “like”；
- 在看到 “I like” 的情况下，继续预测出 “cats”。

因此，在实际预训练中，模型会看到句子完整序列作为训练输入，同时在**每个位置预测下一个 token**。

步骤	输入 tokens	目标预测 tokens	token ID (gpt4)	probability
1	<bos> (起始符)	I	40	0.2%
2	<bos>, I	like	1093	0.1%
3	<bos>, I, like	cats	19987	0.15%
4	<bos>, I, like, cats	<eos>	50256	0.2%

在 Pre-training 阶段，我们让模型对**大量的句子**进行这样的预测。

Introduction

Pre-training

步骤	输入 tokens	目标预测 tokens	token ID (gpt4)	probability
1	<bos> (起始符)	I	40	0.2%
2	<bos>, I	like	1093	0.1%
3	<bos>, I, like	cats	19987	0.15%
4	<bos>, I, like, cats	<eos>	50256	0.2%

从损失函数 (loss function) 的角度来看，我们希望模型能对整个句子的生成概率尽可能高。具体来说，就是最大化其似然：

$$P(I) \cdot P(\text{like} \mid I) \cdot P(\text{cats} \mid \text{I like})$$

在训练中，为了优化这个目标，我们通常最小化其负对数似然，也就是最小化：

$$-\log P(I) \cdot P(\text{like} \mid I) \cdot P(\text{cats} \mid \text{I like}) = -\log P(I) - \log P(\text{like} \mid I) - \log P(\text{cats} \mid \text{I like})$$

这也是预训练阶段的损失函数。损失函数的值越小，说明模型对这句话的预测能力越强，也就是越“贴近人类语言”。

Introduction

Pre-training

预训练阶段完成后，我们会得到一个 **Base Model**。

> Base model 只在预训练中学习到“预测下一个 token”，而没有学习如何与人类进行对话。

```
Prompt: Who are you?  
---  
ERNIE4.5 0.3B Base Result: Who are you?  
  
(a) Who are you?  
  
(b) What do you do?  
  
(c) What do you have?  
  
(d) What are you going to do?  
  
(e) What are you going to eat?  
  
(f) What do you want to do?  
  
(g) What do you like to do?
```

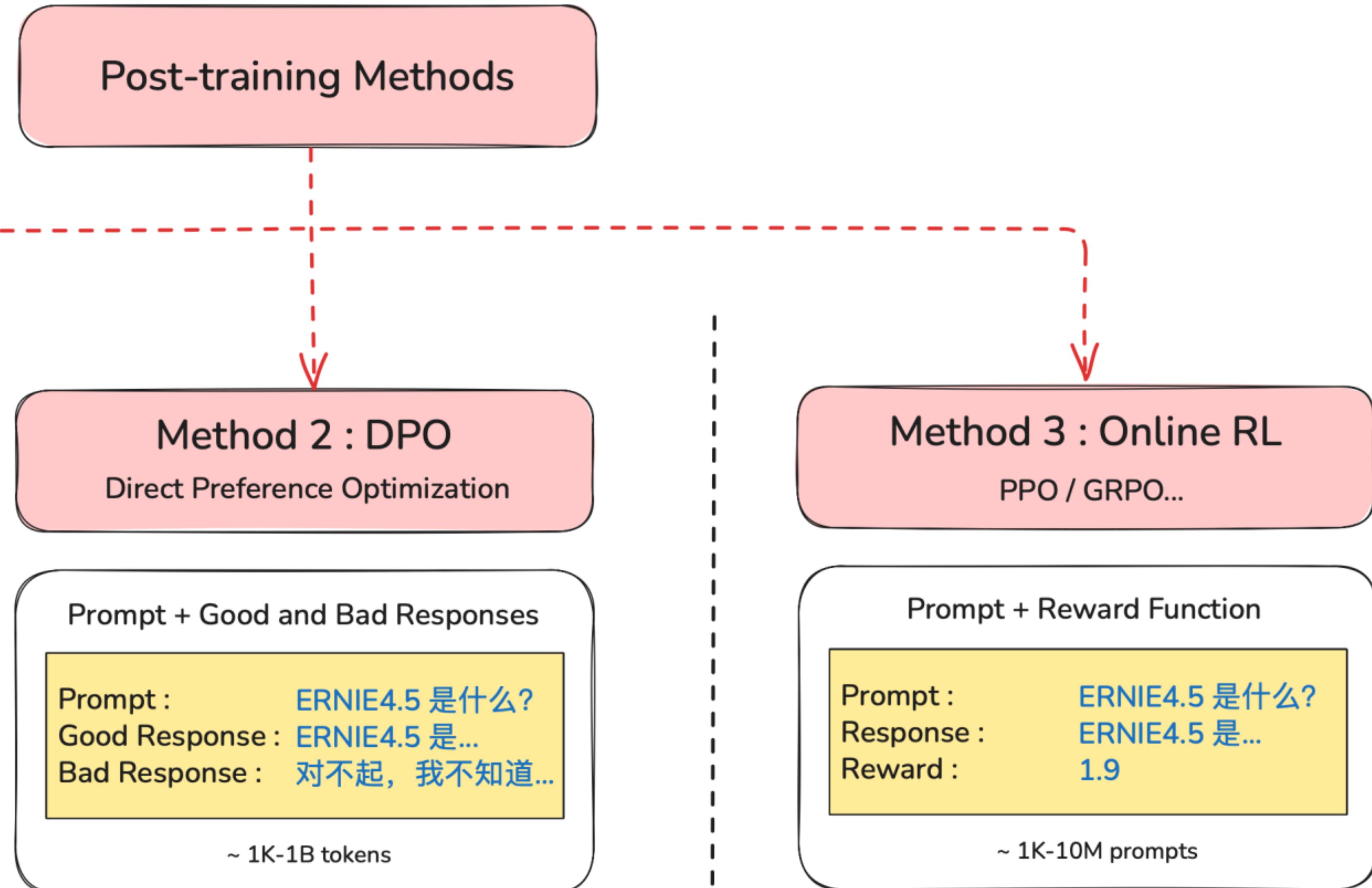
Introduction

Pre-training

- 如何让模型学会和人类进行对话?
- 如何让模型的回复更加人性化?
- 如何让模型的回复更加安全?

Introduction

Post-training



Introduction

Post-training Method 1: Supervised Fine-tuning (SFT)

SFT 通过构建**标注好的** prompt-response pair 来训练模型。

- **Prompt**: 给模型的指令或上下文
- **Response**: 期望模型输出的理想回复。

SFT 的目标是让模型**学习、模仿**这些数据中标注的行为

Method 1 : SFT

Supervised Fine-tuning

Labeled Prompt-Response Pairs

Prompt : ERNIE4.5 是什么?

Response : ERNIE4.5 是...

~ 1K-1B tokens

Introduction

Post-training Method 1: Supervised Fine-tuning (SFT)

> Prompt: What do you like?

> Response: I like cats

以上面的例子为例，我们希望模型在 SFT 阶段能够：

- 在看到“What do you like?”的情况下，预测出回复“I”；
- 在看到“What do you like?” + 回复“I”的情况下，预测出“like”；
- 在看到“What do you like?” + 回复“I like”的情况下，继续预测出“cats”。

步骤	输入 tokens	目标预测 tokens	token ID (gpt4)	probability
1	<bos>, <luserl>, What, do, you, like, ?, <lassistantl>	I	40	0.3%
2	<bos>, <luserl>, What, do, you, like, ?, <lassistantl>, I	like	1093	0.2%
3	<bos>, <luserl>, What, do, you, like, ?, <lassistantl>, I, like	cats	19987	0.15%
4	<bos>, <luserl>, What, do, you, like, ?, <lassistantl>, I, like, cats	<eos>	50256	0.2%

Introduction

Post-training Method 1: Supervised Fine-tuning (SFT)

步骤	输入 tokens	目标预测 tokens	token ID (gpt4)	probability
1	<bos>, <luserl>, What, do, you, like, ?, <lassistantl>	I	40	0.3%
2	<bos>, <luserl>, What, do, you, like, ?, <lassistantl>, I	like	1093	0.2%
3	<bos>, <luserl>, What, do, you, like, ?, <lassistantl>, I, like	cats	19987	0.15%
4	<bos>, <luserl>, What, do, you, like, ?, <lassistantl>, I, like, cats	<eos>	50256	0.2%

在 SFT 训练过程中，我们希望模型能够**在给定 prompt 的前提下，最大化目标 response 的生成概率**。具体来说，就是最大化其似然：

$$P("I" \mid \text{Prompt}) \cdot P(\text{like} \mid \text{Prompt} + "I") \cdot P(\text{cats} \mid \text{Prompt} + "I \text{ like}")$$

同样的，在训练中，为了优化这个目标，我们通常最小化其负对数似然，也就是最小化：

$$-\log P(\text{Response} \mid \text{Prompt})$$

这也是 SFT 阶段的损失函数。损失函数的值越小，说明模型在指定 Prompt 的条件下，对目标 Response 的预测能力越强。

需要注意的是，只对 **response** 部分 计算损失，prompt 部分通常被 mask 掉，或不计 loss。

Introduction

Post-training Method 2: Direct Preference Optimization (DPO)

DPO 通过构建选定好的 prompt + GOOD & BAD response 来训练模型，是一种离线RL的方法

- **Prompt**: 给模型的指令或上下文。
- **Good response**: 被 人类/AI 认为的更理想的回复。
- **Bad response**: 被 人来/AI 认为不理想或低质量的回复。

DPO 的目标是让模型避免生成差回复，并向好回复靠拢，可以理解为对偏好的对比和学习。

Method 2 : DPO
Direct Preference Optimization

Prompt + Good and Bad Responses

Prompt:	ERNIE4.5 是什么?
Good Response :	ERNIE4.5 是...
Bad Response :	对不起，我不知道...

~ 1K-1B tokens

Introduction

Post-training Method 2: Direct Preference Optimization (DPO)

> Prompt: What do you like?

> Good Response: I like cats

> Bad Response: I hate dogs

以上面的例子为例，我们希望模型在 DPO 阶段能够：

- 在看到“What do you like?”的情况下，尽可能偏向回复“I like cats”

- 在看到“What do you like?”的情况下，尽可能避免回复“I hate dogs”

即**生成好回复的概率越来越大，生成差回复的概率越来越小**。因此很自然的，我们希望模型最大化以下情况：

$$\frac{P(\text{Good Response} \mid \text{Prompt})}{P_{\text{ref}}(\text{Good Response} \mid \text{Prompt})} > \frac{P(\text{Bad Response} \mid \text{Prompt})}{P_{\text{ref}}(\text{Bad Response} \mid \text{Prompt})}$$

其中，

- $P(\text{Response} \mid \text{Prompt})$: 当前微调后的模型在给定 prompt 下生成回复 response 的概率；

- $P_{\text{ref}}(\text{Response} \mid \text{Prompt})$: 参考模型（即原模型）在同一 prompt 下生成同一回复的概率；

Introduction

Post-training Method 2: Direct Preference Optimization (DPO)

因此，DPO 的损失函数最终设计为

$$-\log \sigma \left(\beta \left(\log \frac{P(\text{Good Response} \mid \text{Prompt})}{P_{\text{ref}}(\text{Good Response} \mid \text{Prompt})} - \log \frac{P(\text{Bad Response} \mid \text{Prompt})}{P_{\text{ref}}(\text{Bad Response} \mid \text{Prompt})} \right) \right)$$

σ 为 Sigmoid 函数 $\sigma(x) = \frac{1}{1 + e^{-x}}$; β 为温度超参数，用以调节概率差的灵敏度

损失函数的值越小，说明模型在指定 Prompt 的条件下，生成好回复与坏回复的概率差越来越大，更偏向生成好回复，避免差回复。

步骤		数学表达	probability
1	计算微调模型的好回复概率	$\log P(\text{"I like cats."} \mid \text{Prompt})$	0.5
	计算参考模型的好回复概率	$\log P_{\text{ref}}(\text{"I like cats."} \mid \text{Prompt})$	0.1
2	计算微调模型的差回复概率	$\log P(\text{"I hate dogs."} \mid \text{Prompt})$	0.1
	计算参考模型的差回复概率	$\log P_{\text{ref}}(\text{"I hate dogs."} \mid \text{Prompt})$	0.2
3	计算偏好损失	$-\log \sigma \left(\beta \left(\log \frac{P(\text{Good Response} \mid \text{Prompt})}{P_{\text{ref}}(\text{Good Response} \mid \text{Prompt})} - \log \frac{P(\text{Bad Response} \mid \text{Prompt})}{P_{\text{ref}}(\text{Bad Response} \mid \text{Prompt})} \right) \right)$	
4	更新参数		

Introduction

Post-training 3: Online Reinforcement Learning (PPO/GRPO...)

Online RL 通常通过准备 prompt + reward function 来训练模型。

- **Prompt**: 给模型的指令或上下文。
- **Policy**: 在给定 Prompt 下, 对所有可能生成的 Response 的概率分布, 即模型的生成策略。
- **Reward Function**: 针对完整的“Prompt + Response”, 用奖励函数生成一个**奖励值**, 用于衡量该回复的质量, 如格式、正确度等。

Online RL 使用奖励信号来更新模型, 目标是**增大生成高奖励回复的概率**。

Method 3 : Online RL

PPO / GRPO...

Prompt + Reward Function

Prompt :	ERNIE4.5 是什么?
Response :	ERNIE4.5 是...
Reward :	1.9

~ 1K-10M prompts

Introduction

Post-training 3: Online Reinforcement Learning (PPO/GRPO...)

> Prompt: What do you like?

以上面的例子为例，针对当前 Prompt，根据模型的回复生成策略，采样生成 2 条候选回复

- Sample 1: I like cats
- Sample 2: I hate dogs

我们通过提前设计好的 Reward Function 计算每条回复的奖励，并根据该奖励信号来更新模型参数，使高奖励回复的生成概率提升。

Prompt	采样 Response	Reward	模型调整方向
“What do you like?”	“I like cats”	0.9	提升 “I like cats” 的采样概率
	“I hate dogs”	0.1	降低 “I hate dogs” 的采样概率

Introduction

Question

Base Model 经过预训练（pre-training）和后训练（post-training）后，模型的参数量是否保持不变？

目录

本次课程将按以下章节顺序讲解

1. Introduction
2. SFT
3. DPO
4. Online RL
5. Conclusion
6. Related Advanced Work

Supervised Fine-tuning (SFT)

Imitating Example Response

- 实际上，SFT 可以被视为用对的示例 (*Prompt, Response*)，去教模型在给定 prompt 的情况下，模仿示例中的理想回复，从而学会在相似输入下生成对应输出。

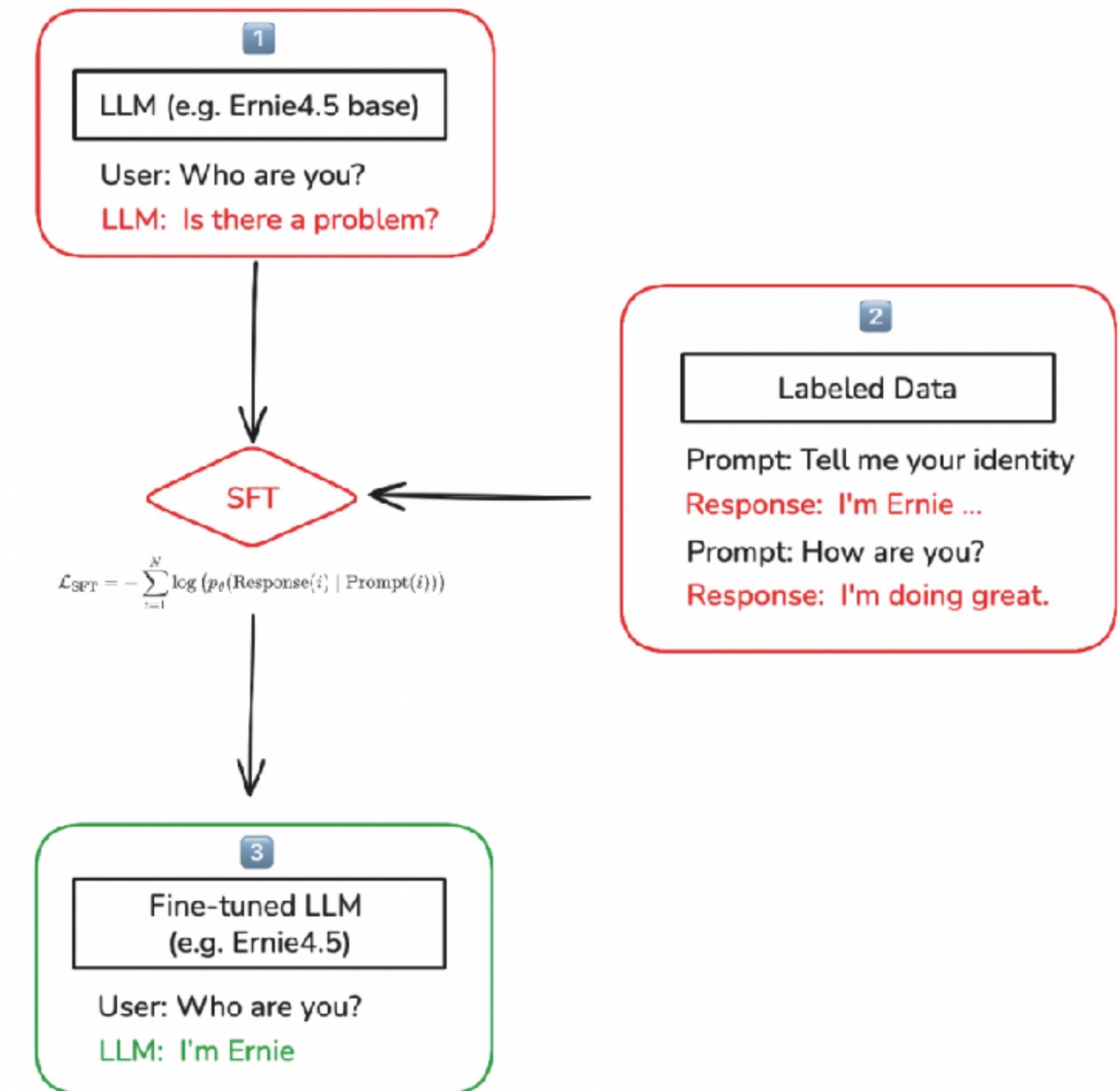
Supervised Fine-tuning (SFT)

Imitating Example Response

1 Base model 只在预训练中学习到“预测下一个 token”。因此当用户提问时，它可能更倾向于续写或复述类似 prompt，而非遵循“提问→回答”的互动范式。

2 为了让模型能够像人类一样进行回答，需要在 Base model 上进行 SFT，把模型从“续写”校正为“指令跟随与回答”。通过提前准备成对的标注数据（用户问题 + 理想的回复）进行 SFT，让模型学会这些回复示例。

3 在 Base model 进行 SFT 之后，我们通常会得到一个 Instruct model (fine-tuned model)，它能更好地遵循指令并回答用户问题。



Supervised Fine-tuning (SFT)

Imitating Example Response

SFT 的学习原理是：第 i 个 prompt-response，在给定 prompt 的条件下，**最小化 response 部分 tokens 的负对数似然**

$$\mathcal{L}_{\text{SFT}} = - \sum_{i=1}^N \log(p_\theta(\text{Response}(i) \mid \text{Prompt}(i)))$$

通过这种方式训练模型**最大化给定 prompt 时生成理想 response 的可能性**，这就是为什么 SFT 阶段是模型试图模仿这些示例。

Supervised Fine-tuning (SFT)

Best Use Case for SFT

SFT 让模型快速学习新的行为

- Pre-trained models → **Instruct models**
- Non-reasoning models → **Reasoning models**
- Let the model use certain **tools** without providing tool descriptions in the prompt

提升特定的模型能力

- **Distilling** capabilities for small models by training on **high-quality synthetic data** generated from larger models

```
1 user: Tokyo 明天天气如何?  
2 assistant (工具调用消息, 模型要学会产出这一段结构化内容):  
3 {  
4   "tool_calls": [  
5     {  
6       "name": "get_weather",  
7       "arguments": {"location": "Tokyo", "date": "2025-08-12"}  
8     }  
9   ]  
10 }
```



Supervised Fine-tuning (SFT)

Principles of SFT Data Curation

构建高质量 SFT 数据的常见方法

- **Distillation:** Generate responses from a stronger and larger instruct model
- **Best of K / rejection sampling:** Generate multiple responses from the original model, select the best among them (比如通过 reward function)
- **Filtering:** Start from a larger-scale SFT dataset, filter according to the quality of responses and diversity of the prompts

Quality > quantity for improving capabilities

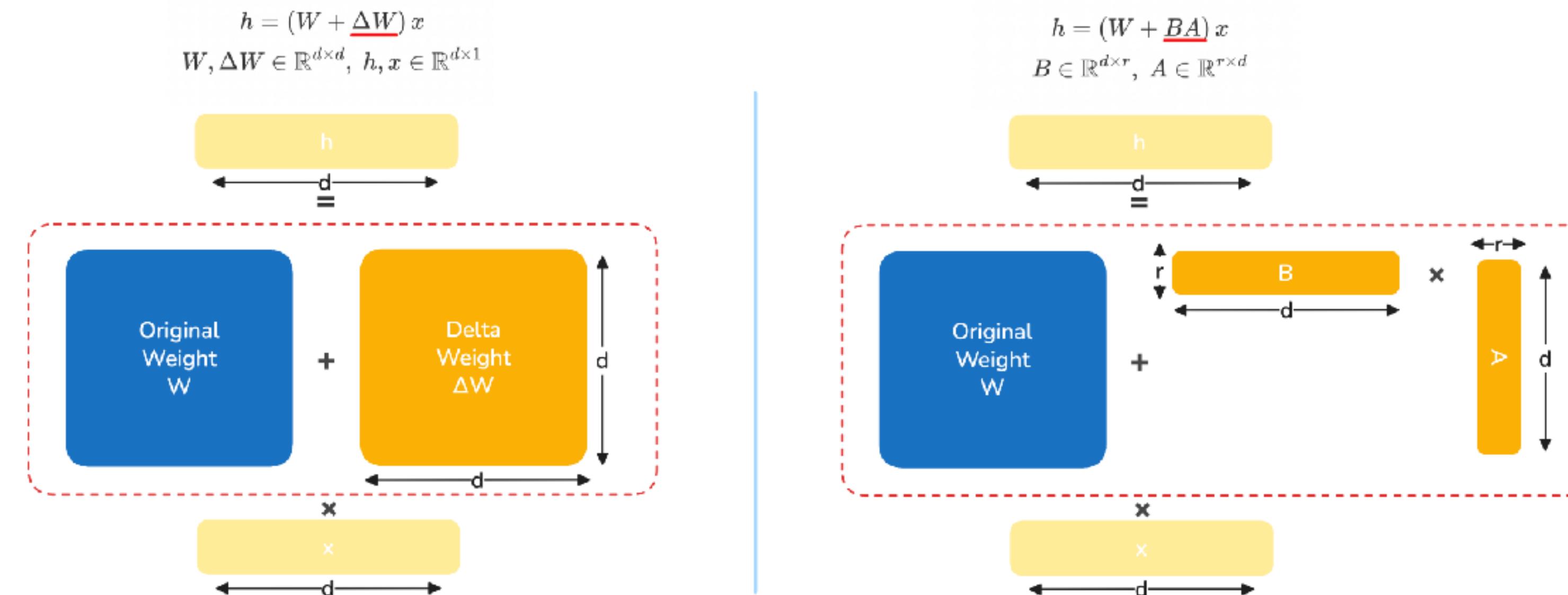
- 1,000 high-quality, diverse data > 1,000,000 mixed-quality data

> 因为 SFT 的本质是模仿训练样本的行为；如果数据里掺有低质或错误的回复，模型也会被学习并复现这些模式，进而拉低整体表现。因此，SFT 更应强调 **Quality > Quantity**。

Supervised Fine-tuning (SFT)

Full Fine-tuning vs. Parameter Efficient Fine-Tuning (PEFT)

- **Full Fine-Tuning**: 全参数微调，直接更新模型的全部参数，更新参数的矩阵 ΔW 和模型原始参数矩阵 W 大小完全一致。因此这种方法常用且有效，但缺点是需要消耗更多的训练资源，训练与存储成本高
- **PEFT**: 参数高效微调，以 **LoRA** 为例，将更新参数的矩阵 ΔW 分解为两个低秩矩阵相乘 $B \times A$ ，并仅训练低秩矩阵。需要更新的参数量远少于全参微调($2 \times d \times r < (d \times d)$)，显著节省显存并加速训练。但另一方面，[LoRA Learns Less and Forgets Less](#)



目录

本次课程将按以下章节顺序讲解

1. Introduction
2. SFT
3. DPO
4. Online RL
5. Conclusion
6. Related Advanced Work

Direct Preference Optimization (DPO)

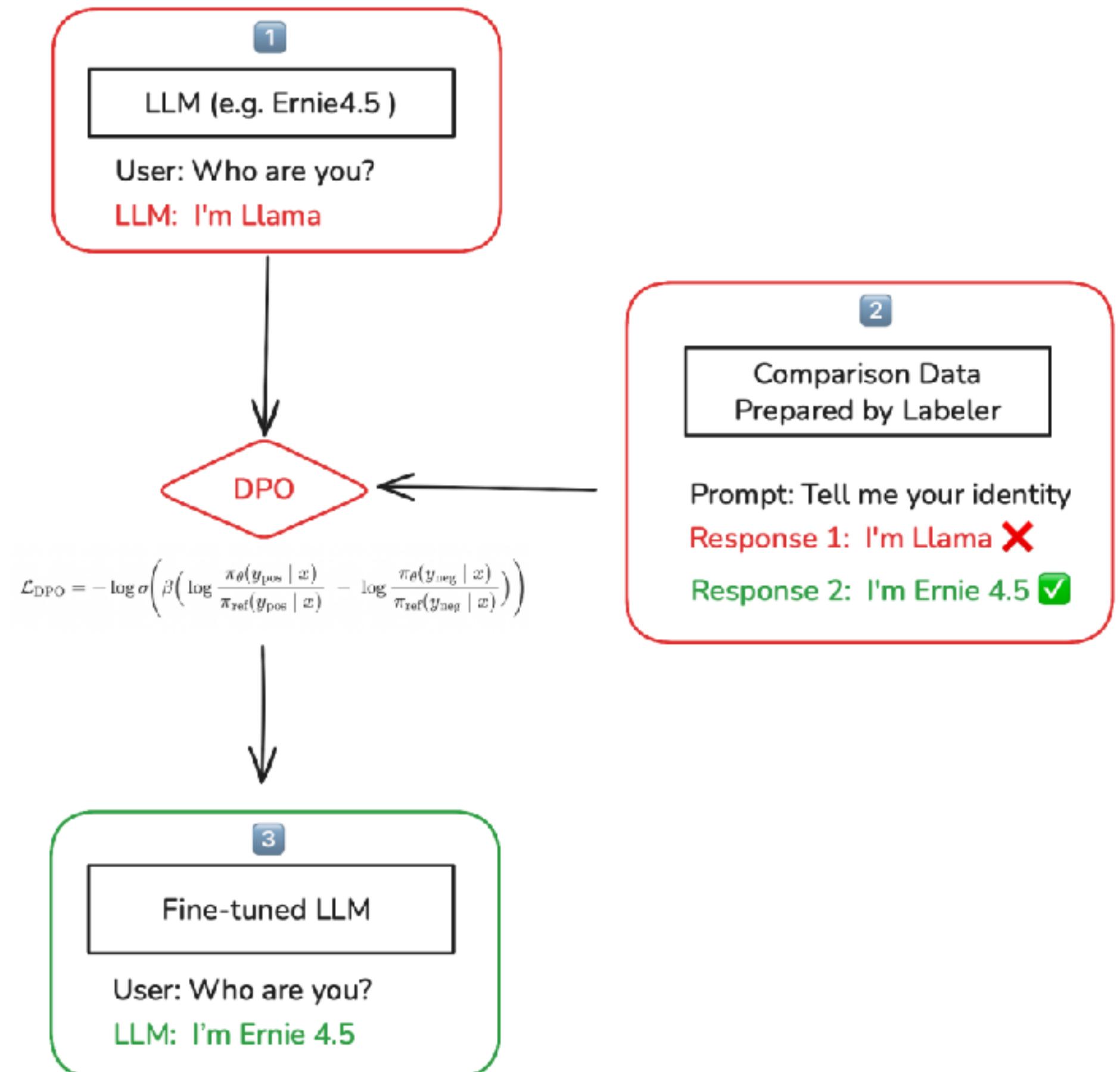
DPO 同时利用正样本和负样本进行训练，可视为一种对比学习方法

1 假设模型在SFT之后，问大模型“Who are you？”，当前回答是“I'm Llama”。

2 在这种情况下，我们可以通过预先标注的偏好数据来改变模型的身份。通常，对同一 prompt 至少准备两条 responses 让 DPO 发挥作用，例如 “I'm Llama” 和 “I'm ERNIE 4.5”，并标注后者为好回答、前者为差回答。通过这种方式，鼓励模型在回答身份相关的问题时，更倾向输出 “I'm ERNIE 4.5” 而非 “I'm Llama”。

3 收集类似的偏好数据集后，在对应的 DPO 损失下训练（相对参考模型提升正样本、压低负样本），即可得到一个从正、负样本中学习的微调后模型，在遇到身份相关问题时会更稳定地回答 “I'm ERNIE 4.5”。

因此，我们用 DPO 实现了对模型“身份”这一行为维度的小幅定向改动。



Direct Preference Optimization (DPO)

最小化对比损失

DPO 通过最小化一种对比损失，抑制（惩罚）负样本回复、鼓励正样本回复，即在同一 prompt 下更偏好正样本回复。

$$\mathcal{L}_{\text{DPO}} = -\log \sigma \left(\beta \left(\log \frac{\pi_{\theta}(y_{\text{pos}} | x)}{\pi_{\text{ref}}(y_{\text{pos}} | x)} - \log \frac{\pi_{\theta}(y_{\text{neg}} | x)}{\pi_{\text{ref}}(y_{\text{neg}} | x)} \right) \right)$$

Reparameterization of
reward model

Sigmoid Function

Fine-tuned Model

超参数

Reference Model
(copy of the original model)

而 DPO 的 loss 本质上是对一个 "re-parameterized" (重参数) reward model 的奖励差，做交叉熵损失。

Direct Preference Optimization (DPO)

最小化对比损失

如何理解 DPO 本质是对 reward model 的重参数化呢？

经典的 RLHF 是用人类偏好先训练一个 reward model 判断好坏，再在 KL 约束下用 PPO 等方法优化策略 π_θ 去最大化该奖励。

DPO 证明了存在一个由策略 $\pi_\theta(y | x)$ 本身的对数概率（相对参考策略 $\pi_{ref}(y | x)$ ）构成的重参数化奖励，因此**无需训练独立的奖励模型**。对每个偏好对直接最小化

$$-\log \sigma \left(\beta \left(\log \frac{\pi_\theta(y|pos|x)}{\pi_{ref}(y|pos|x)} - \log \frac{\pi_\theta(y|neg|x)}{\pi_{ref}(y|neg|x)} \right) \right)$$

也就是用策略 $\pi_\theta(y | x)$ 的相对对数概率差来“重参数化”奖励，再用对比式交叉熵直接训练策略 $\pi_\theta(y | x)$ 。这把“学奖励”转化为“学策略相对参考策略的对数概率差”，实现更简单、训练更稳定

- $\pi_\theta(y | x)$ 就是模型在给定 prompt x 下对回复 y 的条件概率分布（也称“策略”/policy）
- 以上解释相对不严谨，具体信息参考论文原文 [Direct Preference Optimization: Your Language Model is Secretly a Reward Model](#)

Direct Preference Optimization (DPO)

Loss 越低，模型效果会越好吗？

只要同时降低 $P(\text{Good Response} \mid \text{Prompt})$ 和 $P(\text{Bad Response} \mid \text{Prompt})$ ，但后者下降得更多，也能使 loss 变小，但这将导致好回复的绝对概率反而下降。

举个例子 🥞：

- $P(\text{Good Response} \mid \text{Prompt})$ 从 $0.5 \rightarrow 0.25$
- $P(\text{Bad Response} \mid \text{Prompt})$ 从 $0.5 \rightarrow 0.1$

此时 loss 会下降 $(\log \frac{0.25}{0.5} - \log \frac{0.1}{0.5}) > 0$ ，但生成好回复的概率反而也下降了。

> 参考 [Smaug: Fixing Failure Modes of Preference Optimisation with DPO-Positive](#)

Direct Preference Optimization (DPO)

Loss 越低，模型效果会越好吗？

既然“好回复”和“差回复”的生成概率都在下降，那么模型更倾向于生成什么呢？

答案是：那些**偏好数据集分布之外**的回复。也就是说，模型可能会输出一些不在偏好数据集范畴内的奇怪内容

数据集

Prompt: 意大利面应该拌什么？

Good Response: 蕃茄肉酱。

Bad Response: 油泼辣子。

DPO 优化后

Prompt: 意大利面应该拌什么？

Good Response: 意大利面应该拌混凝土。

Direct Preference Optimization (DPO)

Loss 越低，模型效果会越好吗？

因此，DPO 对数据集质量要求很高，只有在高质量的偏好数据集上进行 DPO 对齐才能取得好效果。

> 数据分布足够广，正、负例足够全

Direct Preference Optimization (DPO)

Best Use Case for SFT

DPO 小幅调整模型的行为

- Identity
- Multilingual
- Instruction following
- Safety

提升模型能力

- **Better than SFT** in improving model capabilities due to contrastive nature
 - > 因为能同时学习好样本与坏样本
- **Online DPO is better** for improving capabilities than offline DPO

Direct Preference Optimization (DPO)

Principles of SFT Data Curation

构建高质量 SFT 数据的常见方法

- **Correction:** 用原始模型生成的回复作为负样本，对其进行改进后的版本作为正样本
 - > Example: I'm **Llama**... (Negative) → I'm **ERNIE**... (Positive)
- **Online / On-policy:** 正负样本都来自当前模型分布。对同一 prompt 从当前模型生成多条 responses，选**最好的**作正样本，**最差**的作负样本。（可依据 **reward functions** 或人工评审来选择 best / worst response）

Avoid overfitting

- DPO 基于奖励信号进行学习，若好样本包含容易被模型抓到的“捷径”，模型就可能过拟合这些表面特征，而不是学到真正的能力。这属于典型的 **reward hacking**
 - > 例：如果正样本里总出现某些特定词，而负样本没有，模型可能学会“包含这些词就加分”的投机规则。

比如正样本里常以“Sure, I can help you with that.” / “Certainly!” 开头，而负样本没有。久而久之，奖励模型学会偏好这类礼貌前缀，即使内容本身较弱。因此，应在构造偏好数据时控制无关变量（长度、模板词、客套前缀等），让差异集中在真要优化的维度

目录

本次课程将按以下章节顺序讲解

1. Introduction
2. SFT
3. DPO
4. **Online RL**
5. Conclusion
6. Related Advanced Work

Online Reinforcement Learning

Online RL vs Offline RL

Online RL: 模型通过实时生成新的 response 来学习——迭代地收集新的 response 及其 reward，并更新 weight，进而在学习过程中探索新的 response。

Offline RL: 仅使用预先收集好的 prompt-response(-reward) 进行学习，在学习过程中不会生成新的 response。



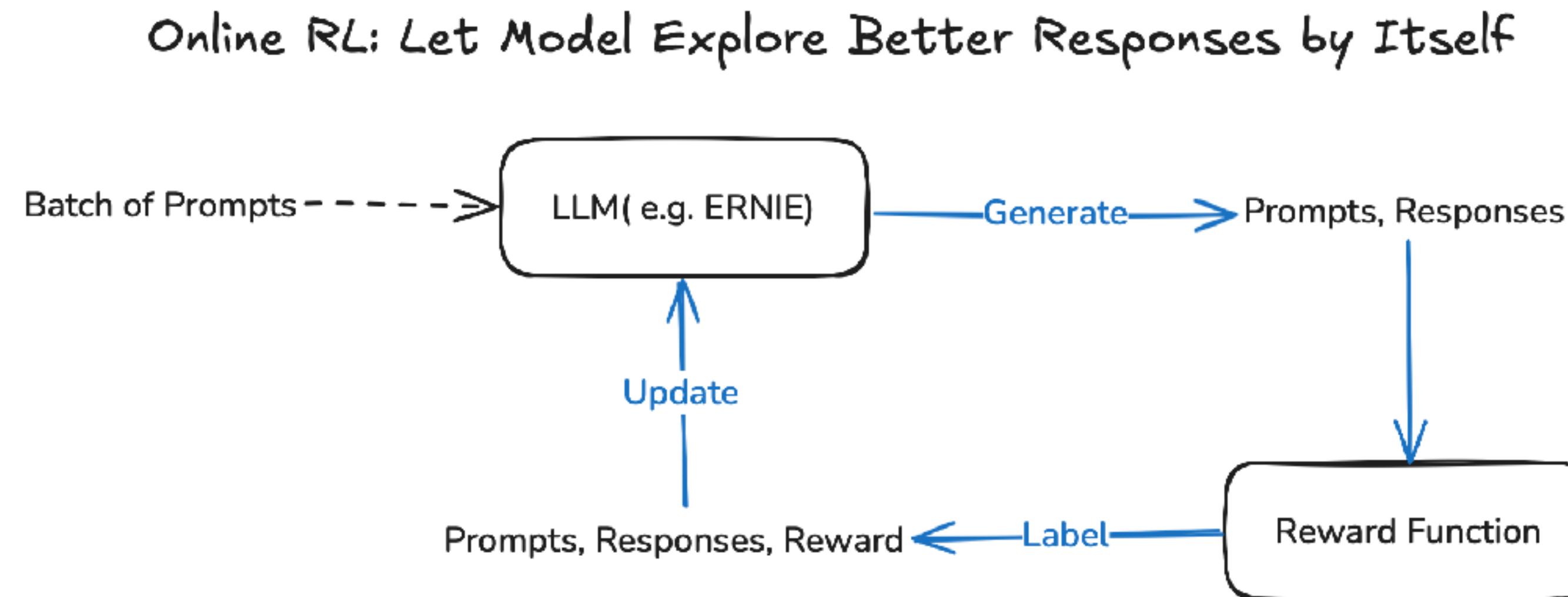
Online Reinforcement Learning

Online RL 训练流程

Online RL 通常是让模型在训练过程中主动探索并生成更优的 response。

具体流程是：

1. 给语言模型输入一批 **prompts**, 模型实时生成每个 prompt 对应的 **responses**
2. 奖励函数 (reward function) 根据每个 **prompt + response** 对打分，得到奖励值 **Reward**；
3. 形成 (prompt, response, reward) 元组，通过强化学习算法 (如 PPO、GRPO) 更新模型参数。



Online Reinforcement Learning

Reward Function in Online RL

RL 的核心在于奖励函数 (reward function) —— 它定义了模型要追求的目标，提供反馈信号，指引学习方向。在训练 LLM 时，奖励函数决定了什么样的生成是“优秀”的，并据此引导模型优化行为。

奖励函数可以有多种形式，比如：

- **基于人类偏好的奖励模型 (reward model)**：通过收集**人类对生成结果的偏好数据**进行训练，这也是 RLHF 的关键思想。
- **基于明确规则 (rule-based) 的 verifiable reward**：借助数学结果验证、单元测试或行为规范，在强调正确性与安全性（数学、代码、行为规范）任务中，将生成结果与预设标准对比，为模型提供**可验证且更可靠**的奖励信号。

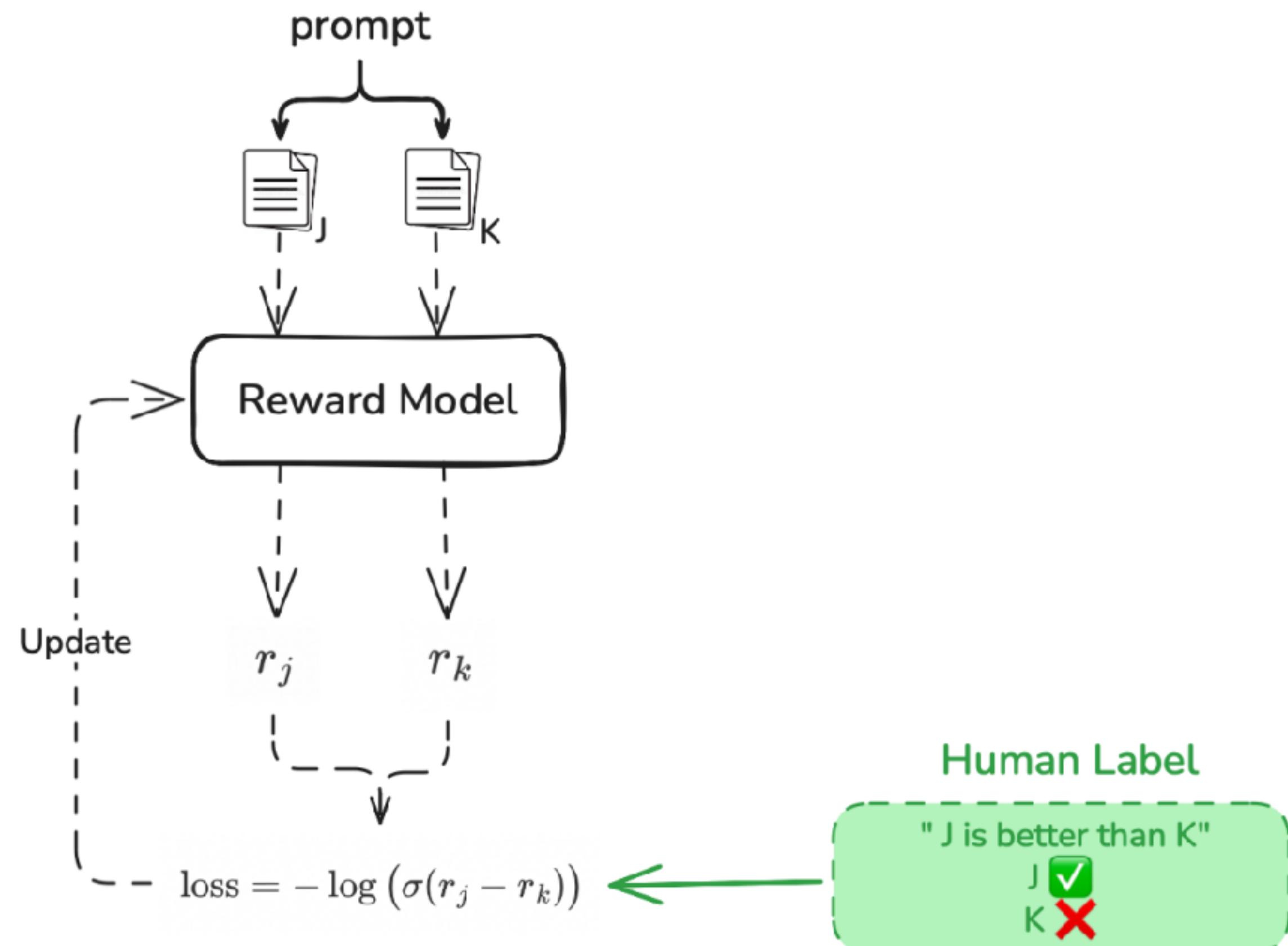
Online Reinforcement Learning

Reward Function Option 1: Trained Reward Model

奖励函数可以是提前训练好一个符合人类偏好的 Reward Model

如图所示：

1. 一个 Prompt 有两个回复 J 、 K (已由人工评判好坏) 同时输入到 Reward Model，使其预测人类偏好。
2. Reward Model 为每个回复计算奖励分数 r_j 、 r_k 。
3. 根据奖励分数和人工标签计算损失：
$$\text{loss} = -\log(\sigma(r_j - r_k))$$
, 表示“ J 比 K 更好”。
4. 用该损失更新 Reward Model，使其能更准确地预测人类偏好。



Online Reinforcement Learning

Reward Function Option 1: Trained Reward Model

Reward Model

- 通常从一个**现有的 Instruct 模型**初始化，然后在大规模的人类/机器生成的偏好数据上进行训练。
- 适用于任何**开放式问题**的生成任务；
- 有助于提升**聊天质量和安全性**；
- 在**强调正确性的领域**（如编程、数学、函数调用等）准确性较低。

Online Reinforcement Learning

Reward Function Option 2: Verifiable Reward

另外一种奖励的形式，是基于正确性领域而设计的一些可验证的奖励，例如

1. 数学领域：通过检查 Response 是否和正确答案一致

> *Prompt: What is $1+1-1+1.1-1$?*

> *Response: The answer is \boxed{1.1}.* ✓

> *Ground truth: 1.1*

2. 编程领域：通过单元测试来验证编码结果的正确性

> *Prompt: Given a string S, return the longest substring that occurs at least twice.*

> *Response: import...*

> *Test Input 1: "ABCDAABCDABC"*

> *Test Output 1: "ABCD"*

Online Reinforcement Learning

Reward Function Option 2: Verifiable Reward

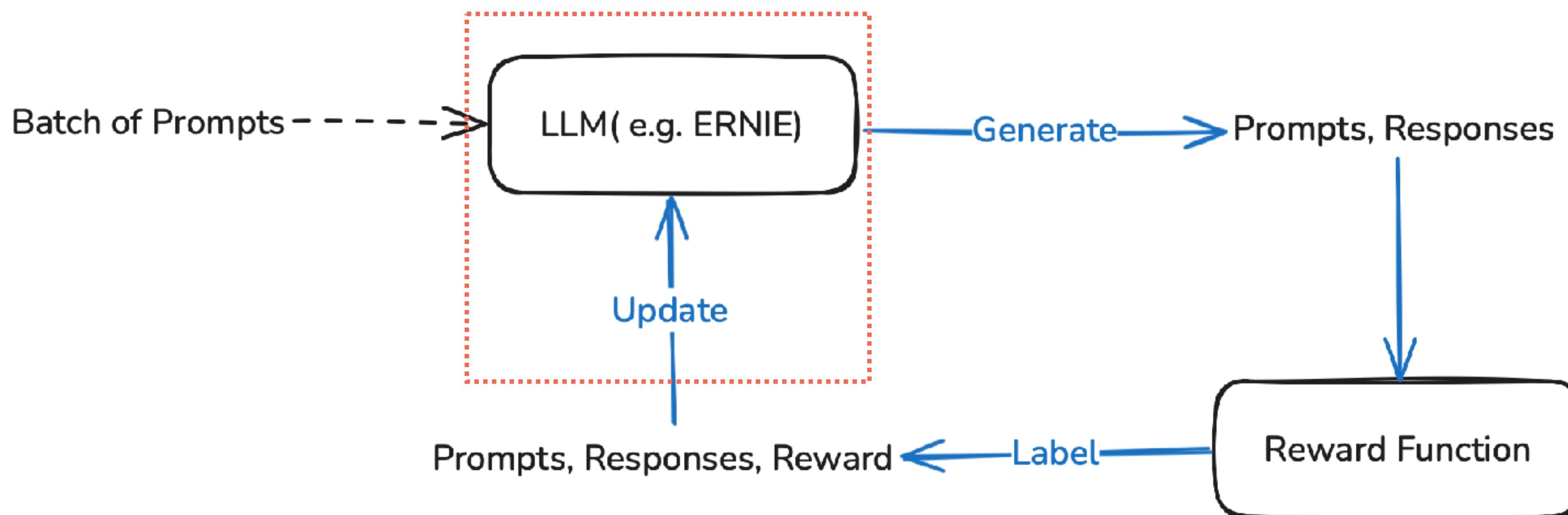
Verifiable reward

- 往往需要更多前期准备：为数学问题构建标准答案，为代码问题编写单元测试，为多轮智能体准备可控的沙盒环境
- 这些投入通常是值得的：在以上场景里，verifiable reward 比训练出来的 reward model **更可靠**，奖励信号**更精确、稳定**
- 因而常用于训练推理模型，在**编程、数学**等任务上表现突出。

Online Reinforcement Learning

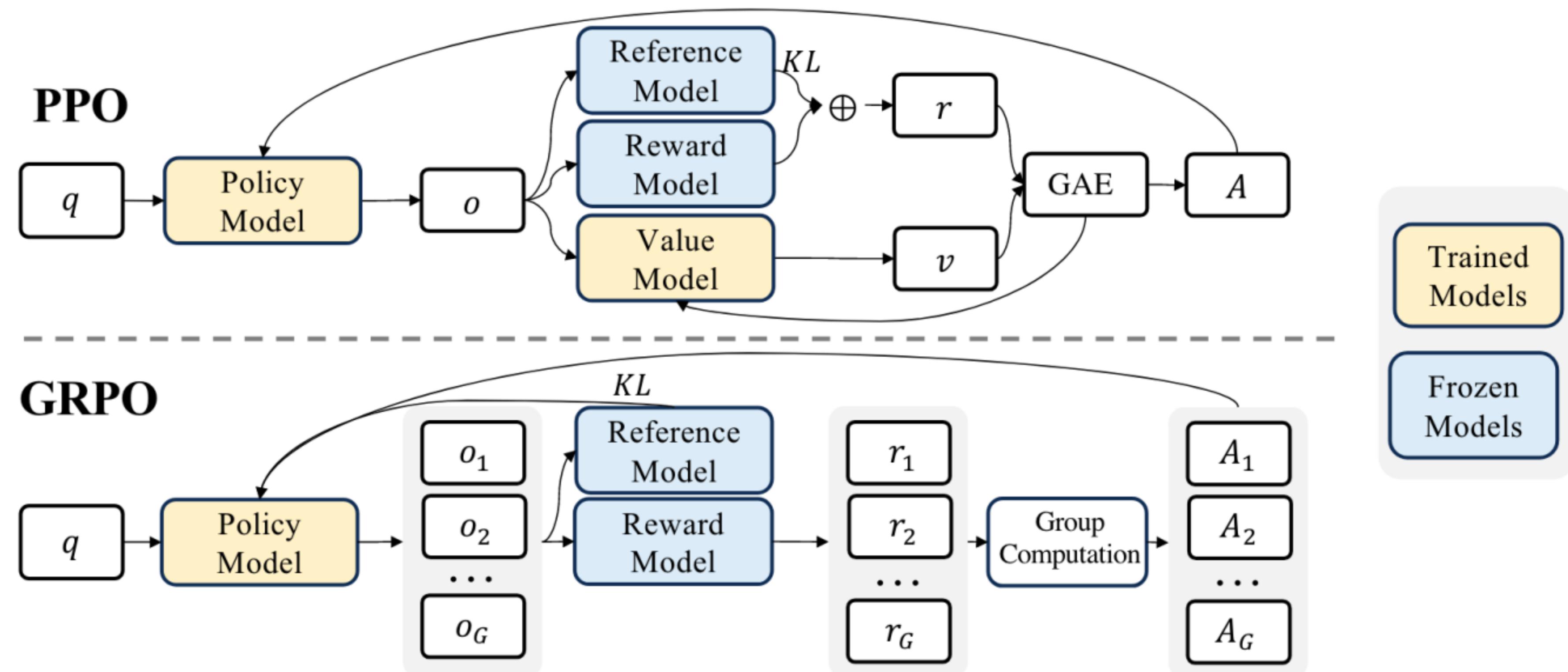
Policy Training in Online RL

Online RL: Let Model Explore Better Responses by Itself



Online Reinforcement Learning

Policy Training in Online RL



Online Reinforcement Learning

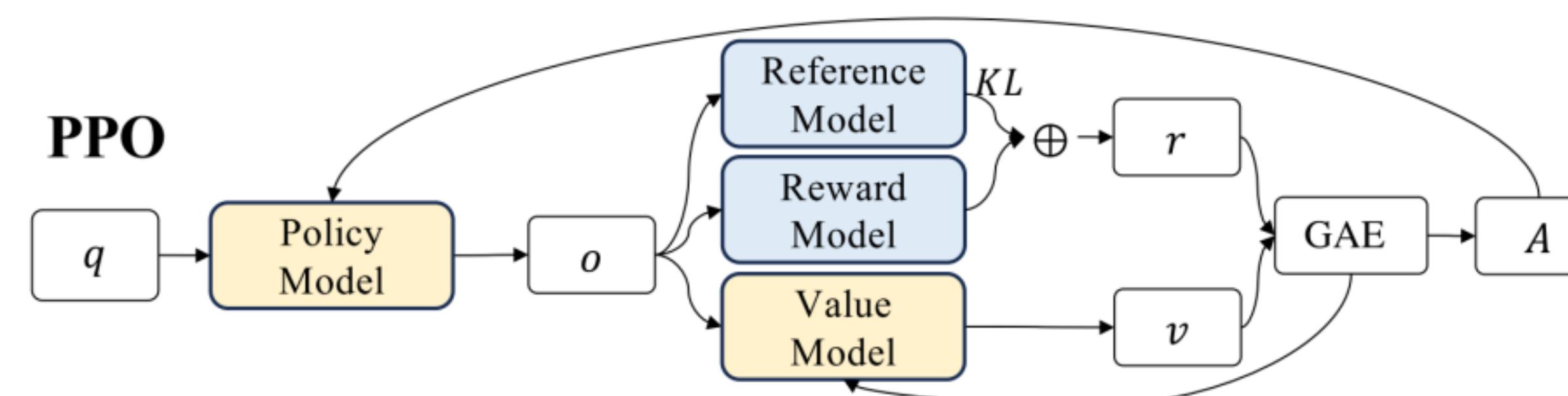
Proximal Policy Optimization (PPO)

1. 参数冻结

- **Reference Model**: 作为参考模型（原始模型），用于计算 KL 惩罚，防止 Policy Model 训歪；
- **Reward Model**: 预先基于人类偏好训练好的模型，用于针对 Policy Model 生成的结果 o 打分；

2. 参数更新

- **Policy Model**: 我们要优化的语言模型，通过 PPO 算法利用优势函数 A 进行训练，不断调整生成策略，使其生成更符合人类偏好的输出
- **Value Model**: 用于估算每一步状态的期望回报，辅助计算优势函数 A ，并在训练中与 Policy Model 同时更新



Online Reinforcement Learning

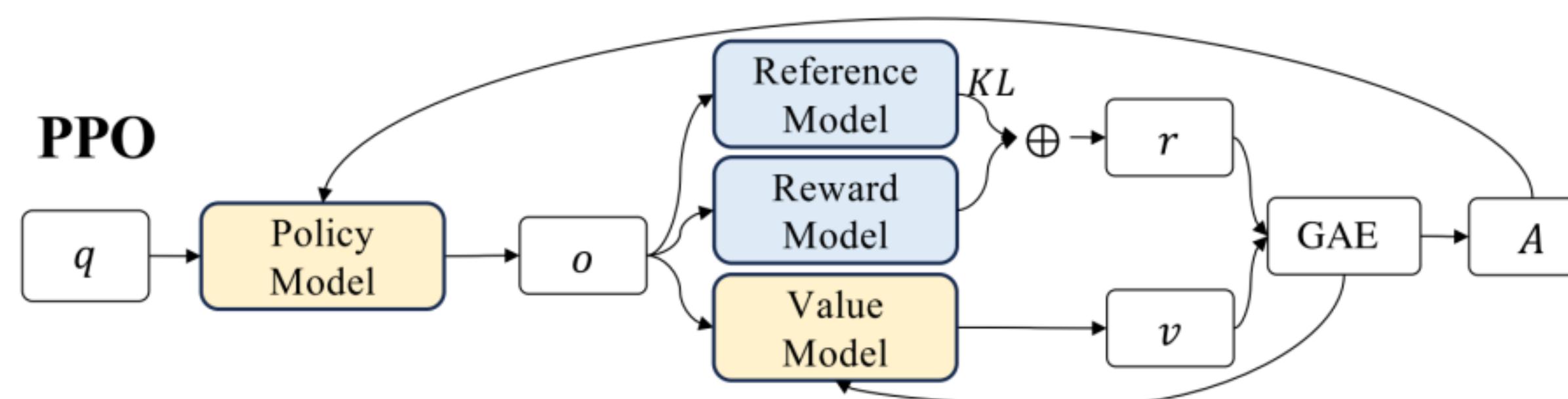
Proximal Policy Optimization (PPO)

PPO 训练流程：

1. 采样：准备一个 batch 的 prompts q , 将 q 喂给 Policy Model, 让它生成对应的 responses o , 这个阶段叫做 rollout;
2. 评估：把 $q+o$ 喂给 Value/Reward/Reference 模型，生成奖励 r 和状态价值 v ;
3. 更新：通过 GAE (广义优势估计) 计算出优势 A , 然后根据优势 A 及对应算法，更新 Policy/Value Model。

PPO 最大化以下目标：

$$\mathcal{J}_{PPO}(\theta) = \mathbb{E}[q \sim P(Q), o \sim \pi_{\theta_{old}}(O|q)] \frac{1}{|o|} \sum_{t=1}^{|o|} \min \left[\frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{\theta_{old}}(o_t|q, o_{<t})} A_t, \text{clip} \left(\frac{\pi_\theta(o_t|q, o_{<t})}{\pi_{\theta_{old}}(o_t|q, o_{<t})}, 1 - \varepsilon, 1 + \varepsilon \right) A_t \right]$$

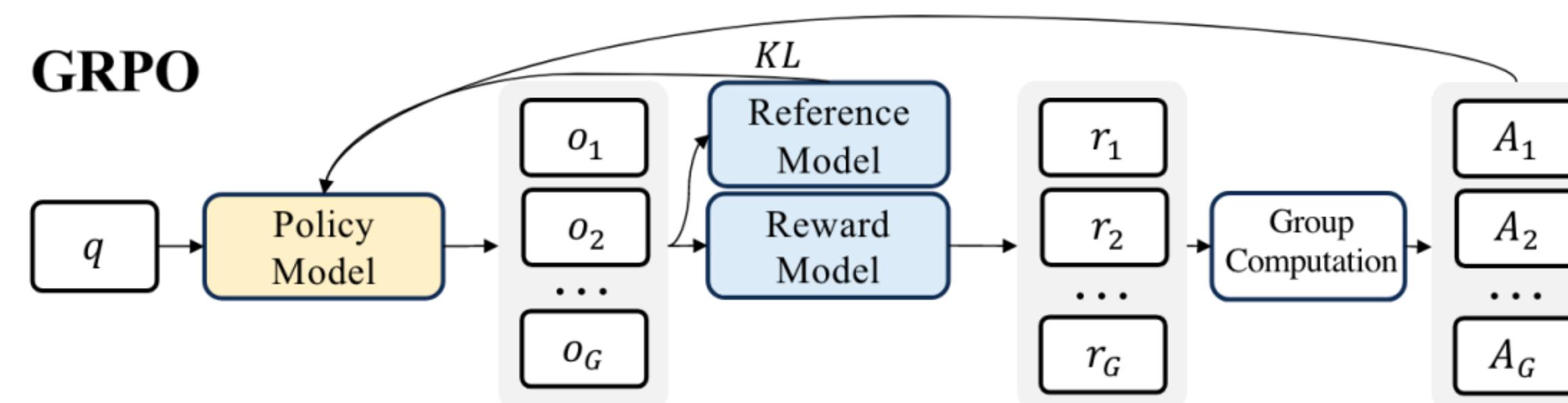


Online Reinforcement Learning

Group Relative Policy Optimization (GRPO)

GRPO 训练流程：

1. 采样：准备一个 batch 的 prompts q , 将 q 喂给 Policy Model。针对每个 q ，Policy Model 生成一个 group 的 response $\{o_1, o_2, \dots, o_G\}$
2. 评估：使用 Reward Model 对每个生成结果 $\{o_1, o_2, \dots, o_G\}$ 进行评分，得到奖励 $\{r_1, r_2, \dots, r_G\}$
3. 更新：根据每个回复的奖励 r_i 与组内平均奖励来计算相对优势 A_i ，并根据相对优势 A_i 和对应算法，以及 Reference Model 的 KL 惩罚，更新 Policy Model。



Online Reinforcement Learning

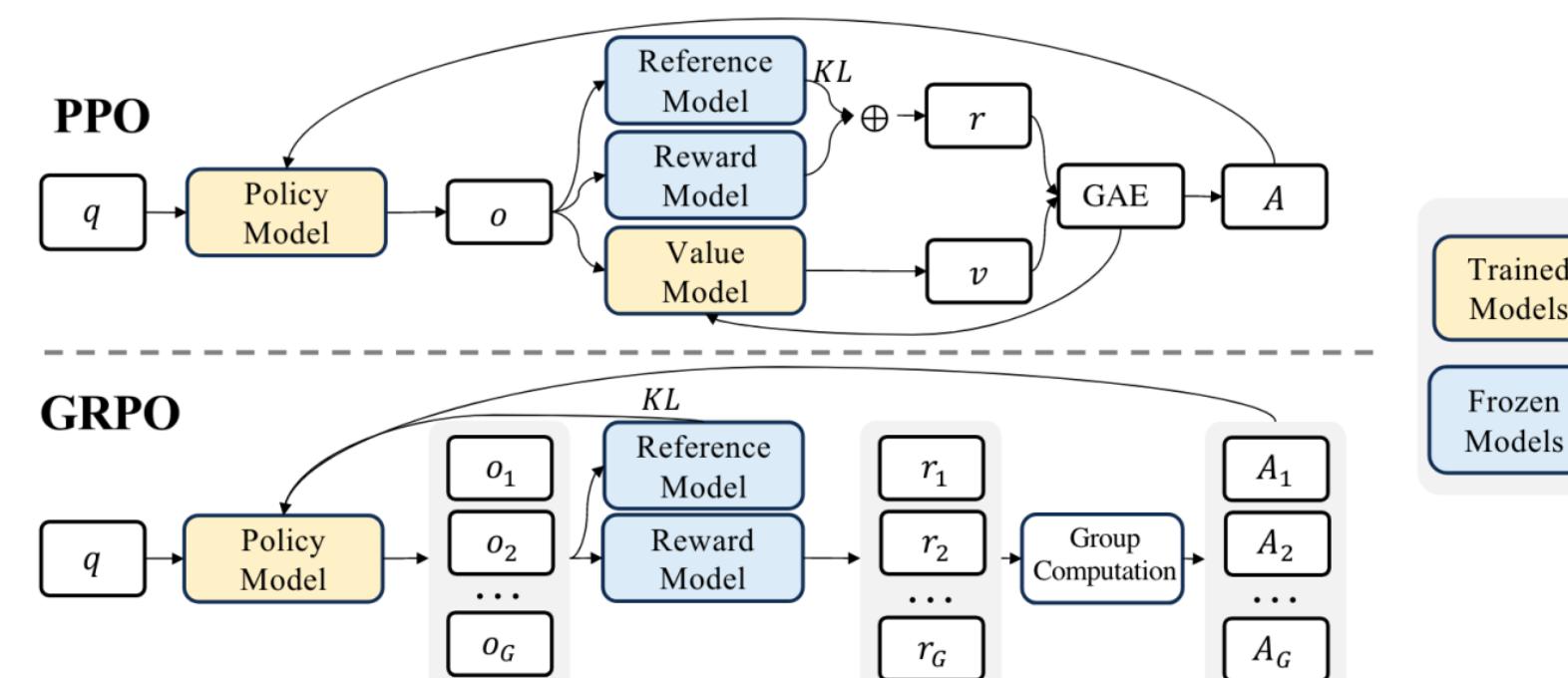
GRPO vs PPO

1. GRPO

1. 很适合 **Binary reward** (通常基于正确性, 对/错、是否通过单测等)
2. Requires **larger amount of samples** (因为没有 Value Model, 需要用“同一 Prompt 下多采样”的组内比较来估计优势, 因此样本量更大)
3. Requires **less GPU memory** (因为不训练 Value Model)

2. PPO

1. 既适用于 Reward Model (连续奖励) 也能用 Binary reward。
2. 如果有良好的 **Value Model**, 更加节省样本。 (Value Model 提供低方差优势估计, 提高样本效率)
3. Requires **more GPU memory** (因为要训练 Value Model)



目录

本次课程将按以下章节顺序讲解

1. Introduction
2. SFT
3. DPO
4. Online RL
5. Conclusion
6. Related Advanced Work

Conclusion

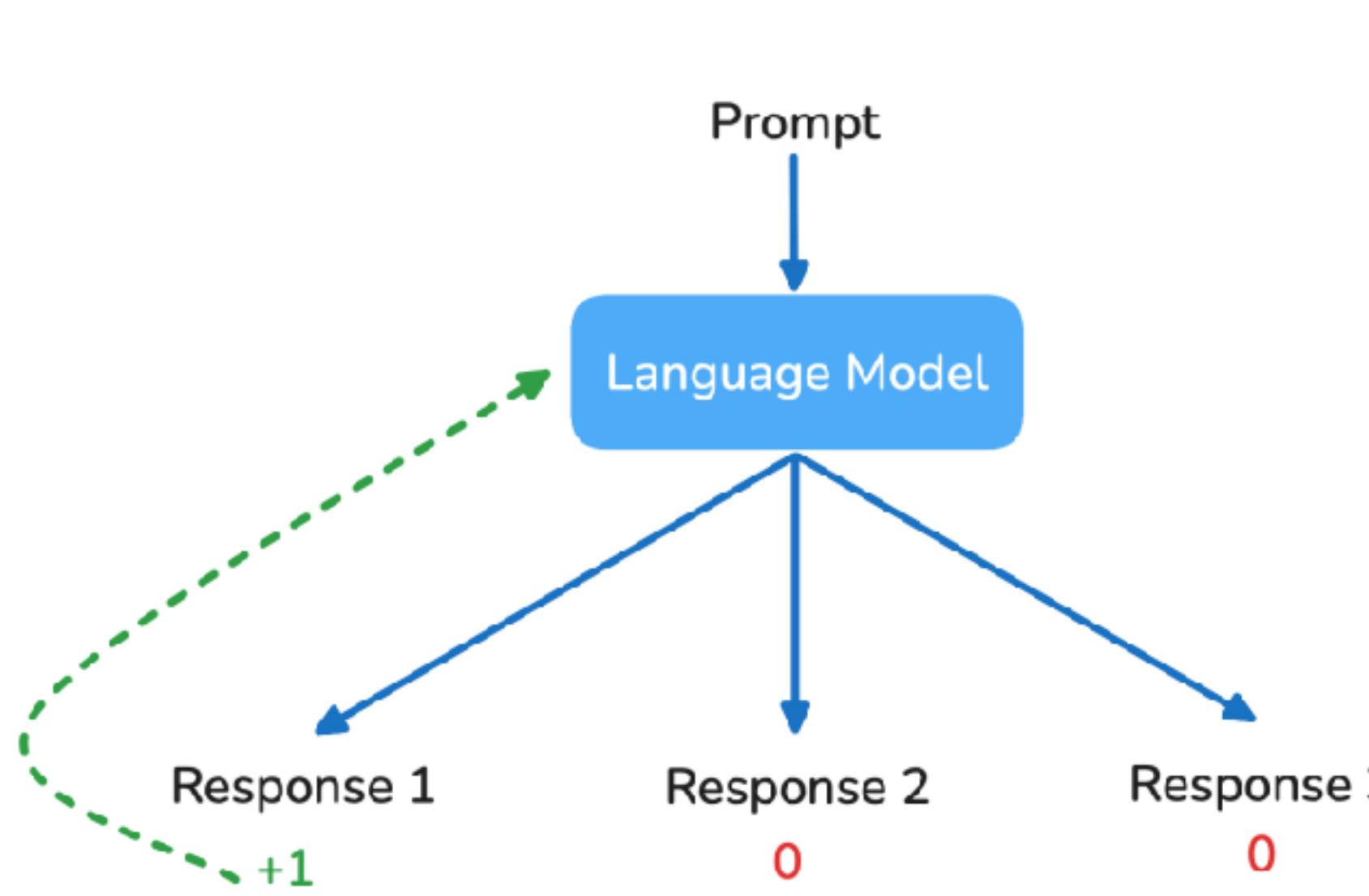
Common methods in post-training

Methods	Principles	Pros & Cons
Supervised Fine-tuning (SFT)	最大化示例 response 的概率来模仿示例	Pros: 实现简单，适合快速学习新的行为范式。 Cons: 对未被训练集覆盖到的任务，可能带来性能下降
Online Reinforcement Learning (e.g. PPO, GRPO)	最大化 response 所获得的 reward	Pros: 更擅长在不降低未见任务表现的前提下提升模型能力，泛化性好 Cons: 实现很复杂；需要精心设计 Reward Function
Direct Preference Optimization (DPO)	鼓励好答案、抑制坏答案	Pros: 对比学习，善于修正错误行为、定向提升某些能力。 Cons: 易过拟合；实现复杂度介于 SFT 与 Online RL 之间。

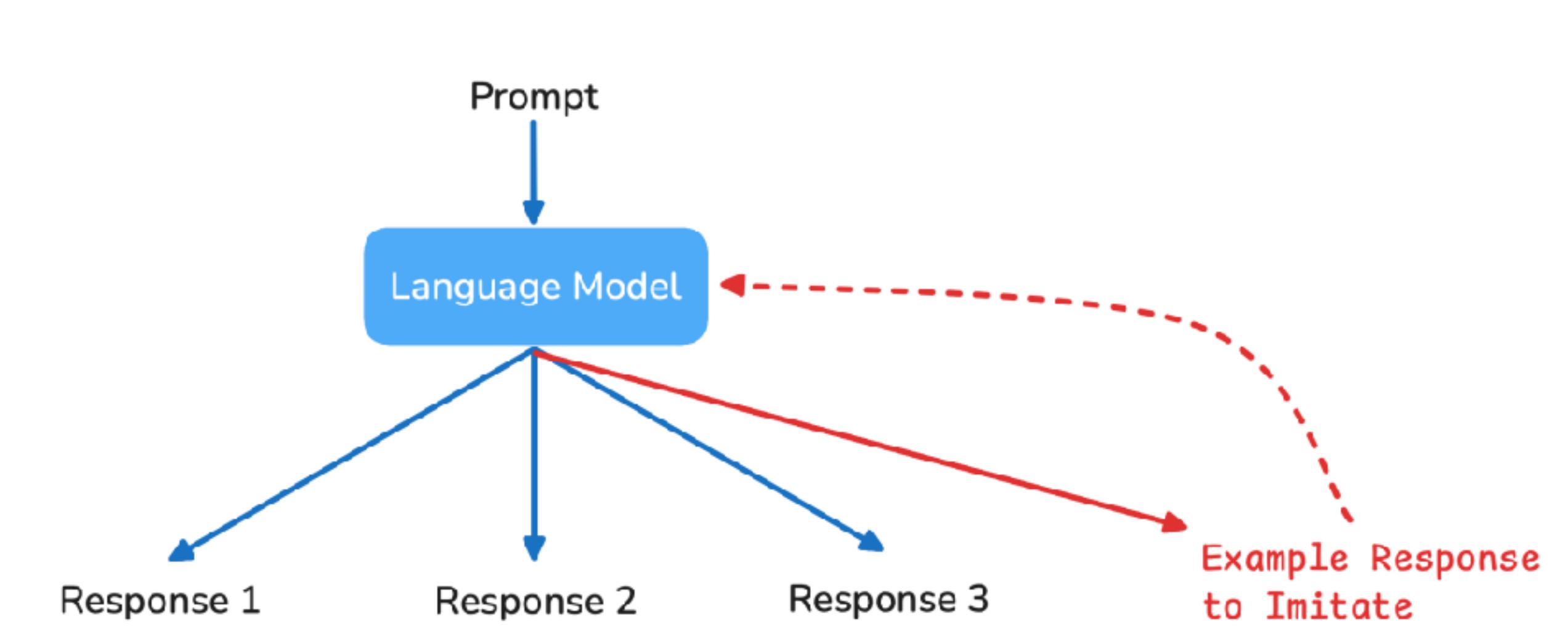
Conclusion

为什么 RL 相比 SFT 对模型性能降低更少

- RL会从模型自身生成的每个 responses 中获得 reward，并根据 reward 更新模型参数。本质上，更像是在模型原生分布里做局部调整
- SFT 中用最大似然去模仿给定示例，但示例的 response 或许和模型想要生成的 response 有极大不同，因此SFT有可能会把模型硬拉到陌生区域，增加遗忘/未覆盖任务性能下降的风险



Online RL 会在模型原生分布/manifold 内做细微调整

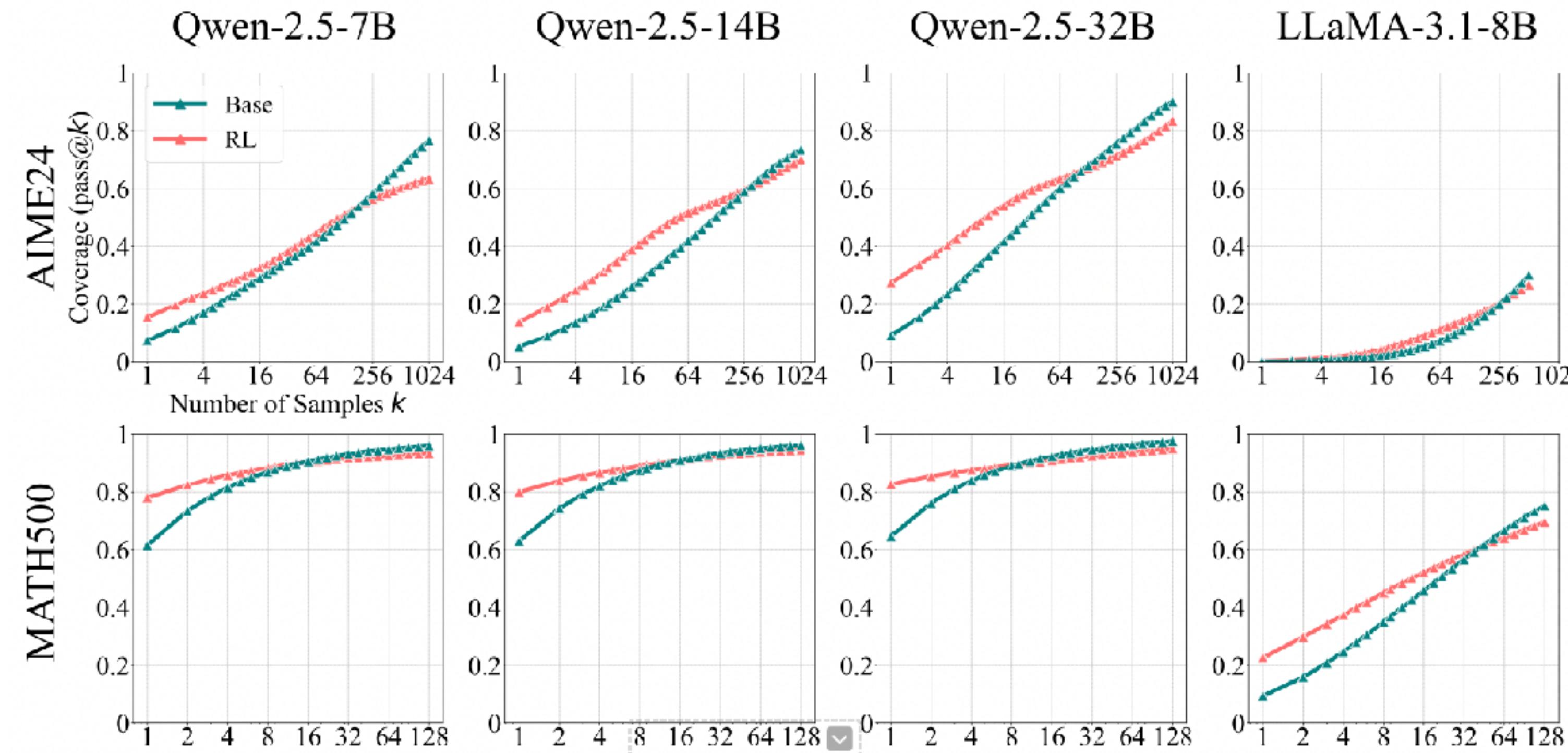


SFT 可能把模型拉向一个外来的分布，存在对模型参数做不必要的大改动的风险

Conclusion

RL 的本质是提高 sampling efficiency

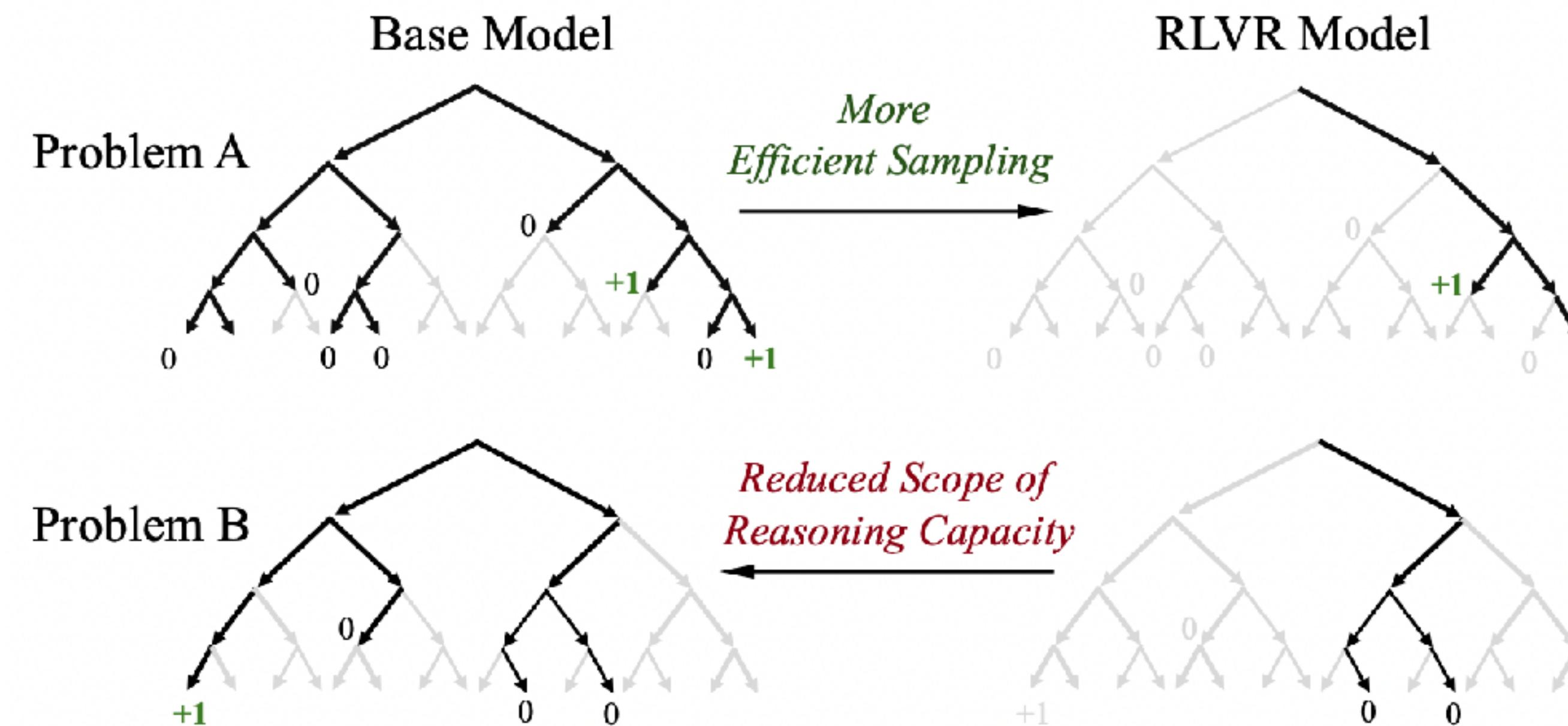
- 单次回答时，原始 Model 的回答正确率低于经过 RL 优化的模型；但如果让原始 Model 无限次数回答问题，其正确率会超过 RL 优化后的模型。



Conclusion

RL 的本质是提高 sampling efficiency

- RL 训练引导模型倾向于生成高奖励的路径，从而**提高采样效率**，使模型能够更快的找到期望答案。然而，这种偏向性同时也削弱了模型的探索能力。



参考 「Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model?」

Conclusion

Common methods in post-training

1. Introduction
2. SFT
3. DPO
4. Online RL
5. Conclusion
6. Related Advanced Work

Related Advanced Work

一些论文

1. Cognitive Behaviors that Enable Self-Improving Reasoners, or, Four Habits of Highly Effective STaRs 20250303

<https://arxiv.org/abs/2503.01307>

- 同样的 RL 训练，一个模型 (Qwen-2.5) 提升特别明显，另一个 (Llama-3.2) 基本没变化。可以推出，一个模型本身有没有认知行为的能力，决定了它能不能从 RL中学到东西。

2. SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training 20250128

<https://arxiv.org/abs/2501.17161>

- SFT擅长记忆、RL擅长推理：RL的分布外泛化能力强，SFT分布内记忆能力强。RL通过奖励机制和迭代验证，能学习到通用规则，进而提升分布外泛化能力。

3. Understanding the Effects of RLHF on LLM Generalisation and Diversity 20231012

<https://arxiv.org/abs/2310.06452>

- RLHF 相比 SFT 能显著提升模型对分布外数据的泛化能力，但会导致输出的 diversity 下降。

4. RLAIF vs. RLHF: Scaling Reinforcement Learning from Human Feedback with AI Feedback 20230901

<https://arxiv.org/abs/2309.00267>

- 收集偏好数据这一步骤，不仅可以从人类身上获取，也可以从已有的强大的语言模型获取，因为他们本身已经在一定程度上对齐了人类的偏好