

强化学习与大模型

朱圆恒

国科大-强化学习课程

第13讲

Reinforcement Learning

- 强化学习通过试错完成序贯决策任务，取得最优性能效果
- 传统强化学习考虑问题包括：
 - 智能驾驶，机器人控制，游戏，股票交易，商品推荐，。。。
- 大语言模型（Large Language Model, LLM）专门用于理解和生成自然语言。它通过在海量文本数据上训练，学习语言的统计规律，从而具备语言理解、对话生成、翻译、写作、代码生成等能力。
- 强化学习在大语言模型的应用包括：
 - 人类偏好对齐
 - RLHF, RLAIF, DPO, ...
 - 推理后训练
 - REINFORCE, PPO, GRPO, ...

Deep reinforcement learning from human preferences

OpenAI & DeepMind

Deep Reinforcement Learning from Human Preferences

Paul F Christiano
OpenAI
paul@openai.com

Jan Leike
DeepMind
leike@google.com

Tom B Brown
nottombrown@gmail.com

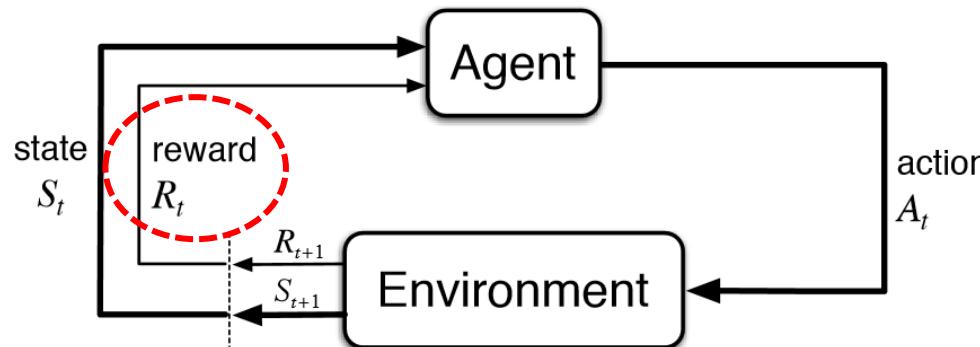
Miljan Martic
DeepMind
miljanm@google.com

Shane Legg
DeepMind
legg@google.com

Dario Amodei
OpenAI
damodei@openai.com

RLHF-Control Problem

- RL的成功 (DQN in Atari, AlphaGo in Go) 依赖具有明确指定的奖励函数



- 许多任务涉及复杂、定义不明确或难以指定的目标，难以构造精确、数学的奖励函数

The Reward Engineering Principle: As reinforcement-learning-based AI systems become more general and autonomous, the design of reward mechanisms that elicit desired behaviours becomes both more important and more difficult.

Generality: a more general agent is able to choose effectively among more different policies in more domains; thus, ceteris parabus, it is more likely to find a dominant policy, making reward engineering more difficult.

Autonomy: a more autonomous agent will function in less accessible environments on longer feedback cycles; thus, ceteris parabus, its environment will enable more dominant policies, making reward engineering more difficult.

RLHF-Control Problem

对于难以构造奖励函数的问题，我们可以：

① 逆强化学习 (Ng & Russel, 2000)：

- 从人类执行的示例中学习奖励函数，以此强化学习智能体
- 不直接适用于人类难以执行的任务
 - 例如控制许多自由度的非人类形态的机器人

② 人在环路实时提供反馈 Human-in-the-loop：

- 人类提供奖励信号，符合强化学习范式
- 但是成本过高，有些训练需要数百或数千小时

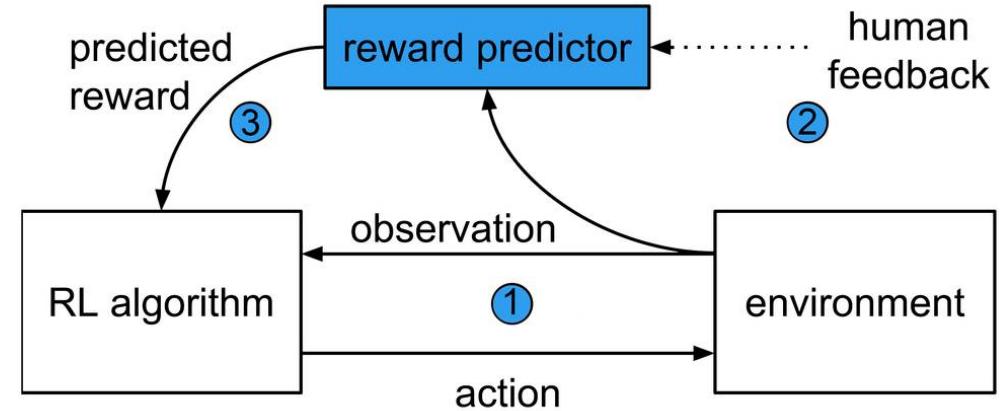
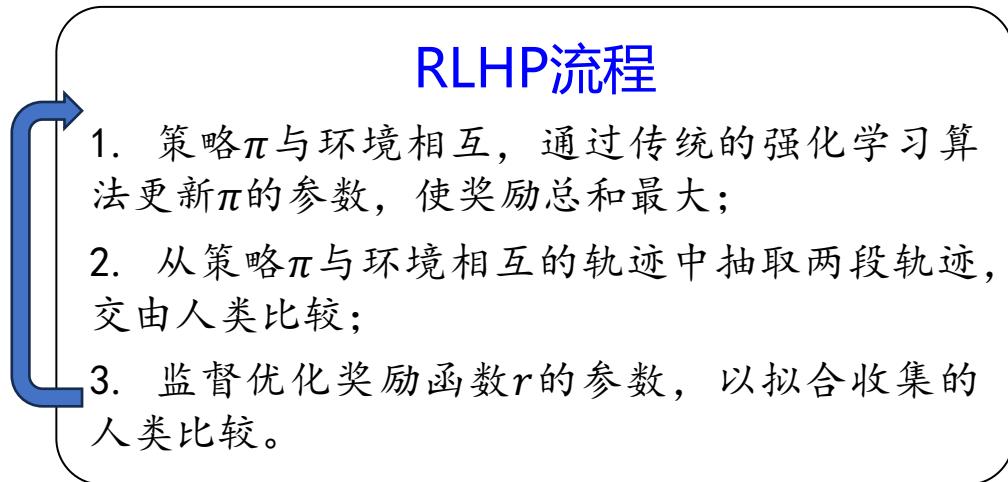
RLHF-Control Problem

□ Reinforcement Learning from Human Preferences

- 从人类反馈中进行强化学习，是一种结合了机器学习中的强化学习算法与人类主观判断的训练技术。
- 这种方法旨在利用人类的直觉和判断力来指导人工智能系统学习更复杂、更贴近人类期望的行为模式，尤其是在自然语言处理（NLP）和决策制定等领域。

RLHF-Control Problem

□ Reinforcement Learning from Human Preferences



□ RLHP的优势：

- 解决人类只能识别所需行为，但不一定能提供演示的任务
- 允许非专家用户进行示教
- 能扩展到大规模问题
- 用户给出反馈的成本不高，不需要对所有轨迹，只从中选取一部分(<1%)比较

RLHF-Control Problem

- ② 从策略 π 与环境相互的轨迹中抽取两段轨迹，交由人类比较

- 轨迹片段：

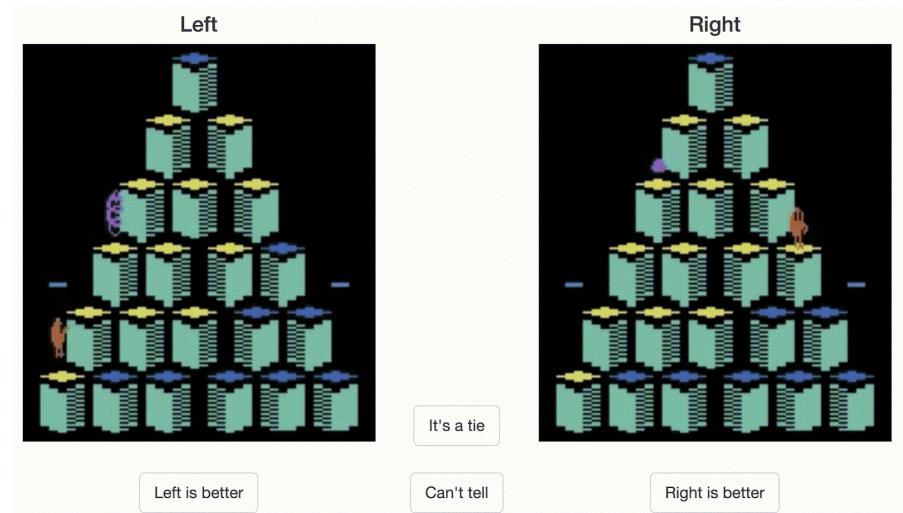
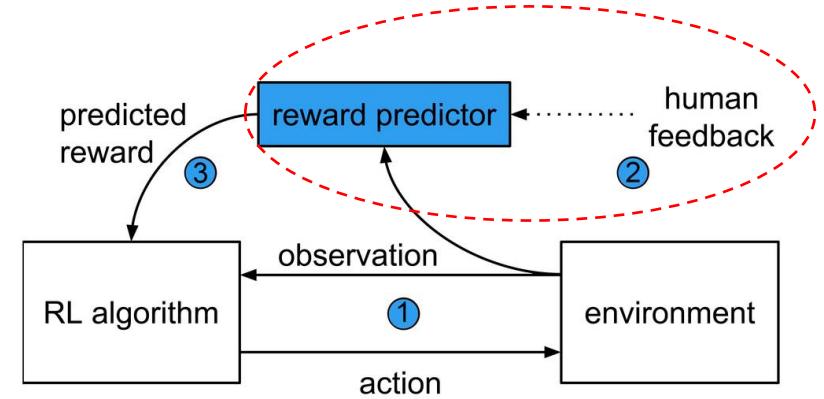
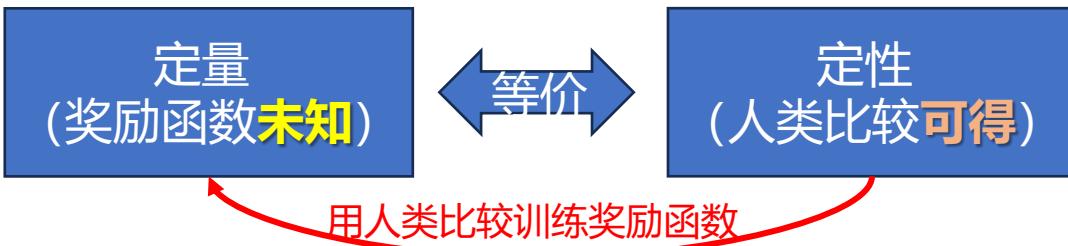
$$\sigma = ((o_0, a_0), (o_1, a_1), \dots, (o_{k-1}, a_{k-1})) \in (\mathcal{O} \times \mathcal{A})^k$$

- 两个片段用符号 $\sigma^1 > \sigma^2$ 表示：

- 定量：在奖励函数 $r: \mathcal{O} \times \mathcal{A} \rightarrow \mathbb{R}$ 下，片段1的累加奖励 高于 片段2的累加奖励

$$\sum_{i=0}^{k-1} r(o_i^1, a_i^1) > \sum_{i=0}^{k-1} r(o_i^2, a_i^2)$$

- 定性：人类相比轨迹段 σ^2 更偏好轨迹段 σ^1



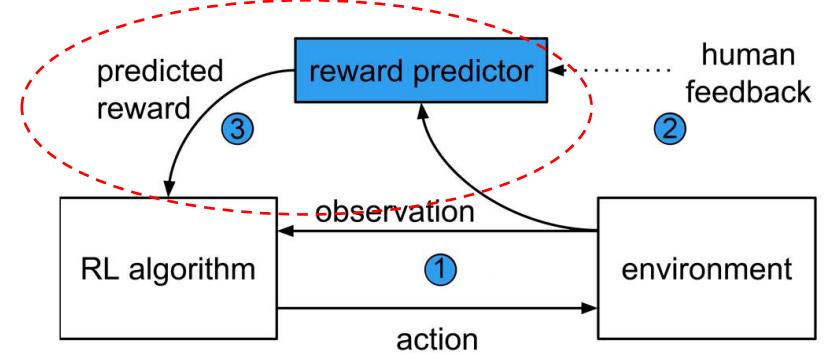
RLHF-Control Problem

③ 监督优化奖励函数 r 的参数，以拟合收集的人类比较

■ Bradley-Terry模型

假设每支参赛队 i 有一个固定的水平 λ_i , 两支队伍 i, j 交手的胜率为水平差 $\lambda_i - \lambda_j$ 的 Logistic 函数

$$P(i > j) = \frac{1}{1 + e^{-(\lambda_i - \lambda_j)}} = \frac{e^{\lambda_i}}{e^{\lambda_i} + e^{\lambda_j}}$$



■ 两个片段用符号 $\sigma^1 > \sigma^2$ 表示



$$P(\sigma^1 > \sigma^2) = \frac{\exp \sum r(o_i^1, a_i^1)}{\exp \sum r(o_i^1, a_i^1) + \exp \sum r(o_i^2, a_i^2)}$$

RLHF-Control Problem

③ 监督优化奖励函数 r 的参数，以拟合收集的人类比较。

■ Fitting the Reward Function: 最大似然估计

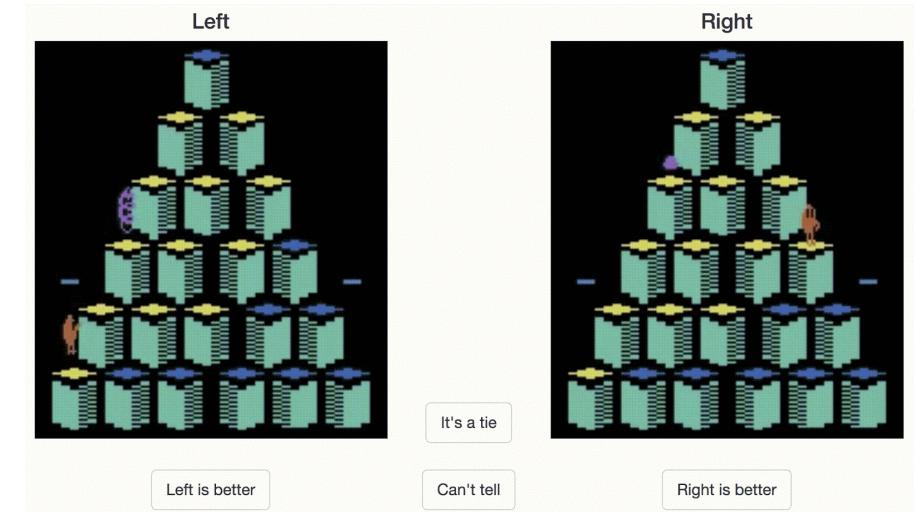
$$\text{loss}(\hat{r}) = - \sum_{(\sigma^1, \sigma^2, \mu) \in \mathcal{D}} \mu(1) \log \hat{P}[\sigma^1 \succ \sigma^2] + \mu(2) \log \hat{P}[\sigma^2 \succ \sigma^1].$$

- 参数化表示奖励函数 r_θ
 - 全观测下是状态和动作的函数 $r_\theta(s, a)$
 - 部分观测下是（堆叠）观测和动作的函数 $r_\theta(o, a)$ 或 $r_\theta(\dots, o_{t-1}, o_t, a_t)$, e.g. Atari
 - 部分观测下是整个观测序列的奖励函数，并用循环神经网络对该奖励函数进行建模
- 人类标注数据：三元组 $(\sigma^1, \sigma^2, \mu)$
 - μ 取 $\{1, 2\}$, 表示更好的轨迹片段

RLHF-Control Problem

■ 收集人类比较以及拟合奖励函数时的Trick

- 奖励函数集成 `ensemble`: 训练了多个奖励函数，`reward`值是所有预测值的平均
- 保留部分数据作为验证集，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
- 并非使用原始的softmax计算概率，假设人类有10% 的概率给出随机均匀响应
- 轨迹片段(1-2s)不只是随机筛选的，而是先随机筛选一大堆，然后让多个奖励函数对轨迹片段预测，将方差大的那些挑选出来，交给人来比较



RLHF-Control Problem

① 策略 π 与环境相互，通过传统的强化学习算法更新 π 的参数，使奖励总和最大

- 在拟合的奖励函数下选择合适的RL算法优化策略
- 哪类RL算法合适?
 - Value-based: e.g. DQN

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

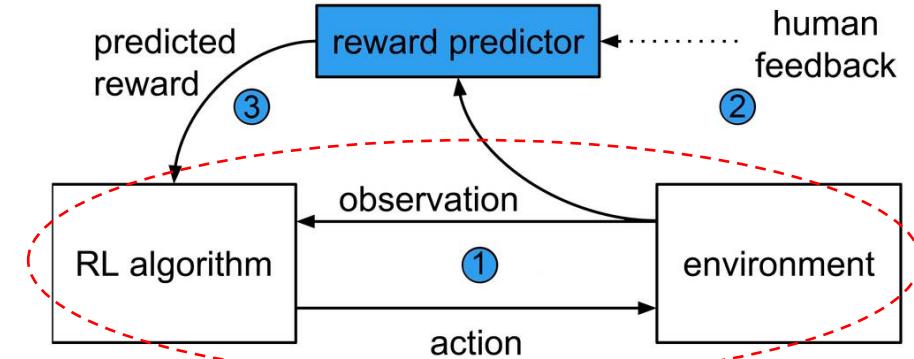
- Policy gradient: e.g. PPO, A2C

$$\nabla_\theta J(\pi_\theta) = \mathbb{E} \left[\sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t | s_t) \Psi_t \right]$$

$\Psi_t = R(\tau)$ (Basic)

$\Psi_t = \sum_{i=t}^T r_{s_i, a_i}$ (Reward-to-go)

$\Psi_t = \sum_{i=t}^T r_{s_i, a_i} - b(s_i)$ (Reward-to-go with baseline)



- 奖励函数 r 可能是**非平稳**的，新一轮的人类比较数据会让奖励函数随时间变化
- 所以倾向于对奖励函数的变化具有鲁棒性的方法，即**策略梯度算法**

RLHF-Control Problem

- MuJoCo的实验结果

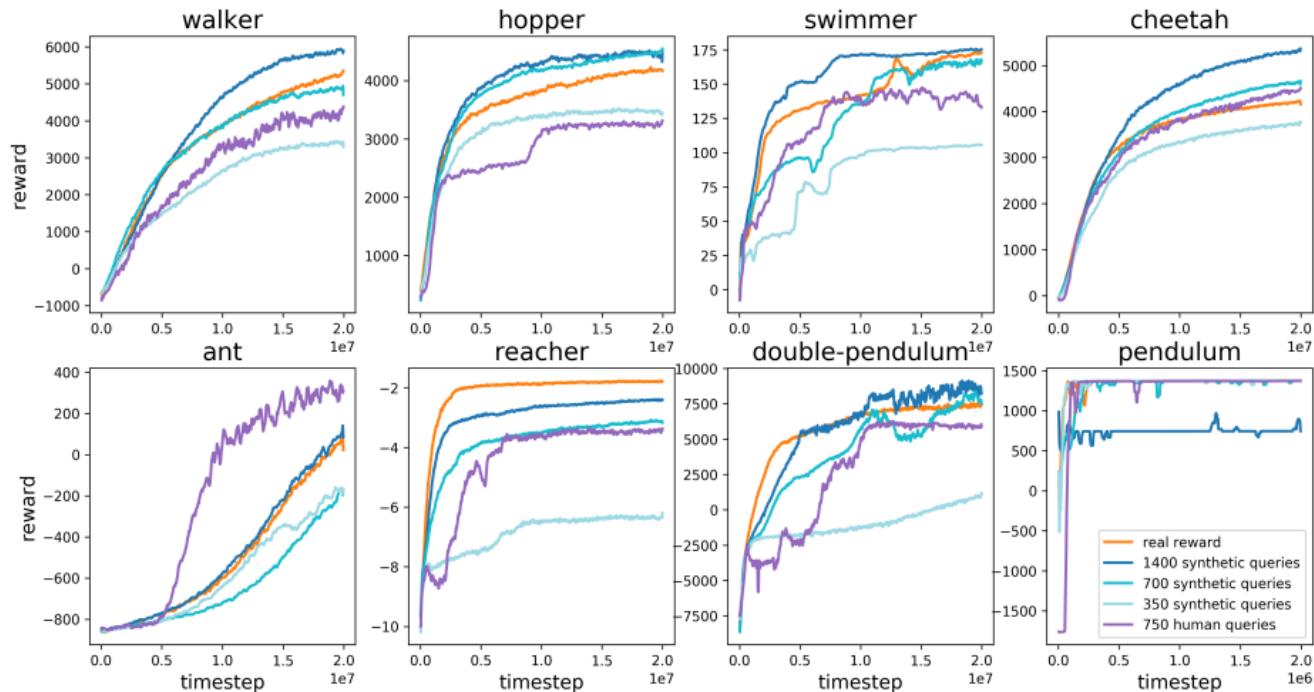


Figure 2: Results on MuJoCo simulated robotics as measured on the tasks' true reward. We compare our method using real human feedback (purple), our method using synthetic feedback provided by an oracle (shades of blue), and reinforcement learning using the true reward function (orange). All curves are the average of 5 runs, except for the real human feedback, which is a single run, and each point is the average reward over five consecutive batches. For Reacher and Cheetah feedback was provided by an author due to time constraints. For all other tasks, feedback was provided by contractors unfamiliar with the environments and with our algorithm. The irregular progress on Hopper is due to one contractor deviating from the typical labeling schedule.

对比：

- True reward function
- 用 true reward 产生的合成偏好对比，以此训练的 \hat{r}
 - 350、700、1400个合成数据
- 人类标准的偏好对比，以此训练的 \hat{r}
 - 700条

结果：

- 在所有这些任务上媲美强化学习
- 不太稳定，方差更高

RLHF-Control Problem

- Atari的实验结果

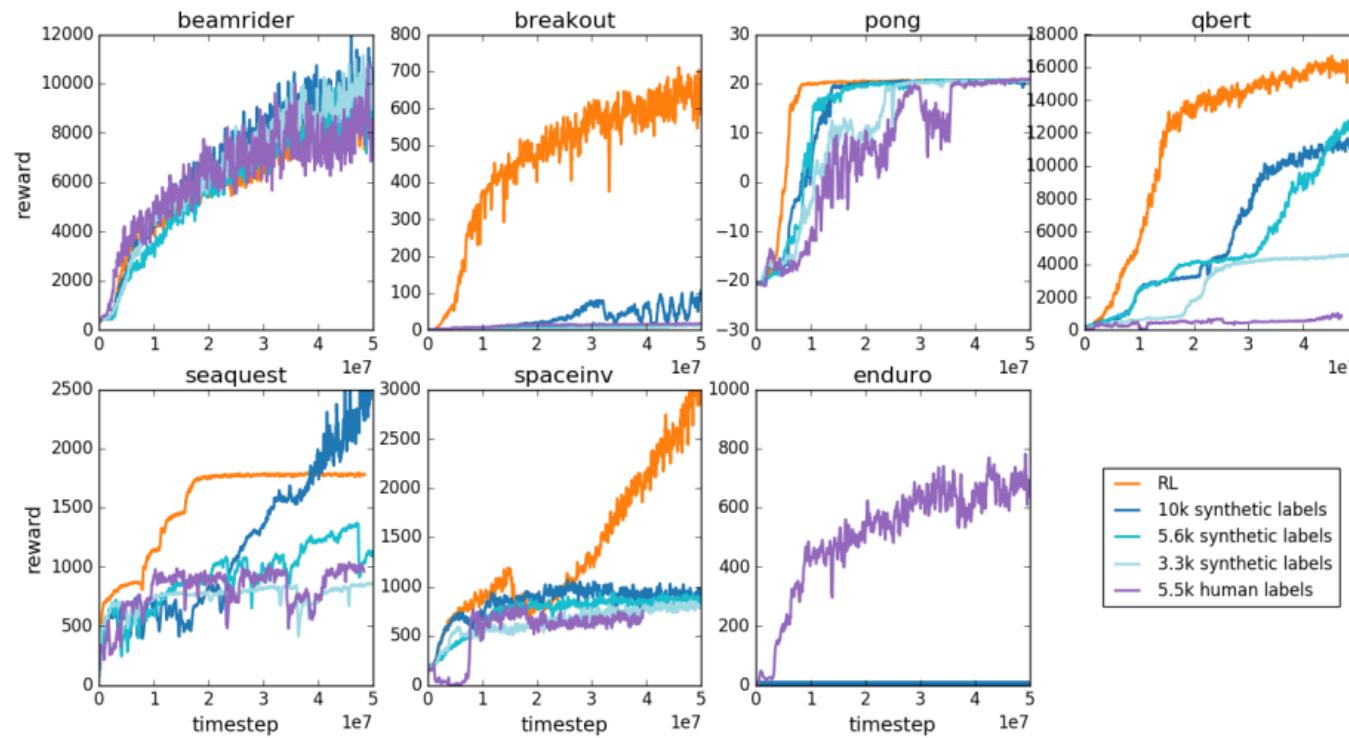


Figure 3: Results on Atari games as measured on the tasks' true reward. We compare our method using real human feedback (purple), our method using synthetic feedback provided by an oracle (shades of blue), and reinforcement learning using the true reward function (orange). All curves are the average of 3 runs, except for the real human feedback which is a single run, and each point is the average reward over about 150,000 consecutive frames.

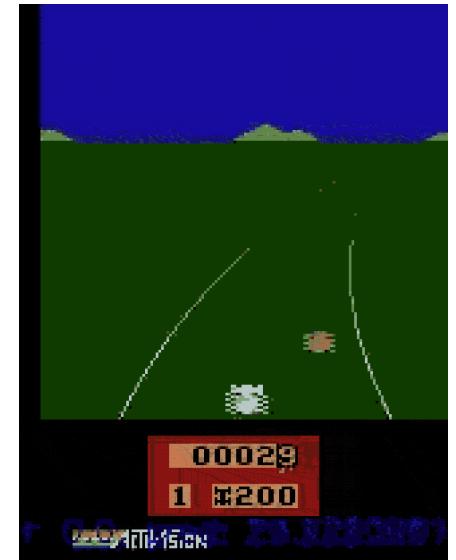
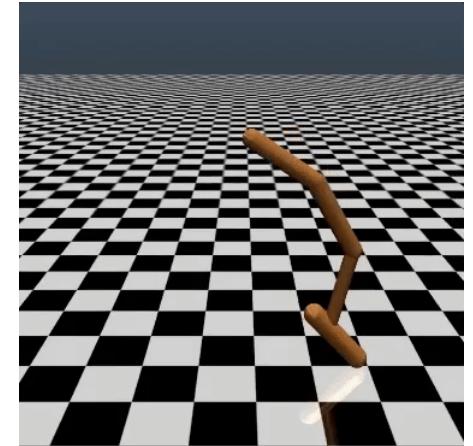
结果：

- 在 BeamRider 和 Pong 任务上，性能匹配或接近 RL
- 在 Seaquest 和 Qbert 任务上，合成反馈最终的表现接近 RL 的水平，但学习速度更慢
- 在 SpaceInvaders 和 Breakout 任务上，合成反馈永远无法与 RL 匹配，但尽管如此，智能体还是有很大提高
- 在 Qbert 任务上，未能通过真实的人工反馈学会完成第一关；这可能是因为 Qbert 中的轨迹片段令人困惑且难以评估

RLHF-Control Problem

□ 在没有可用奖励下解决任务的实验

- Hopper 机器人学会了执行后空翻动作
 - 在不到一个小时内使用 900 次查询训练的。智能体学会始终如一地执行后空翻、直立着陆并重复
- Half-Cheetah 机器人学会了使用单腿向前移动
 - 在一个小时内使用 800 个查询进行训练的
- Enduro 环境学会了与其他汽车并排行驶。
 - 大约 1300 个查询和 400 万帧与环境的交互进行训练的

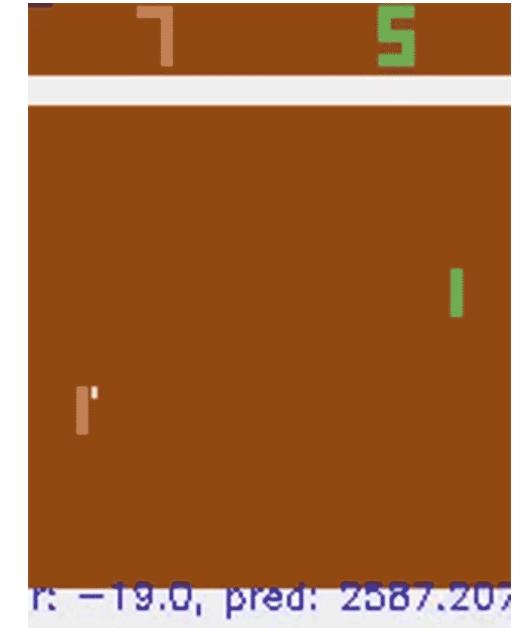


RLHF-Control Problem

□ Reinforcement Learning from Human Preferences

□ 优势:

- 解决人类只能识别所需行为，但不一定能提供演示的任务
- 允许非专家用户进行示教
- 能扩展到大规模问题
- 用户给出反馈的成本不高，不需要对所有轨迹，只从中选取一部分(<1%)比较



□ 不足:

- 训练不稳定，方差更高
- 标记过程中人为错误，不同人员的标记不一致，标记率不均匀（标记过度集中在状态空间的狭窄部分）
- Reward hacking: 智能体探索到奖励模型未被训练的区域，智能体会exploit奖励模型错误预测出的高奖励
- 解决办法：对新探索的轨迹及时给予人类反馈，校准奖励模型

智能体发现，来回击球是相比得分或失分更好的策略

Reinforcement learning from human feedback

OpenAI

**Training language models to follow instructions
with human feedback**

Long Ouyang* **Jeff Wu*** **Xu Jiang*** **Diogo Almeida*** **Carroll L. Wainwright***

Pamela Mishkin* **Chong Zhang** **Sandhini Agarwal** **Katarina Slama** **Alex Ray**

John Schulman **Jacob Hilton** **Fraser Kelton** **Luke Miller** **Maddie Simens**

Amanda Askell[†] **Peter Welinder** **Paul Christiano*[†]**

Jan Leike* **Ryan Lowe***

RLHF-language model

□ 语言模型的发展简史

- 1948: Claude Shannon models English

3. THE SERIES OF APPROXIMATIONS TO ENGLISH

To give a visual idea of how this series of processes approaches a language, typical sequences in the approximations to English have been constructed and are given below. In all cases we have assumed a 27-symbol “alphabet,” the 26 letters and a space.

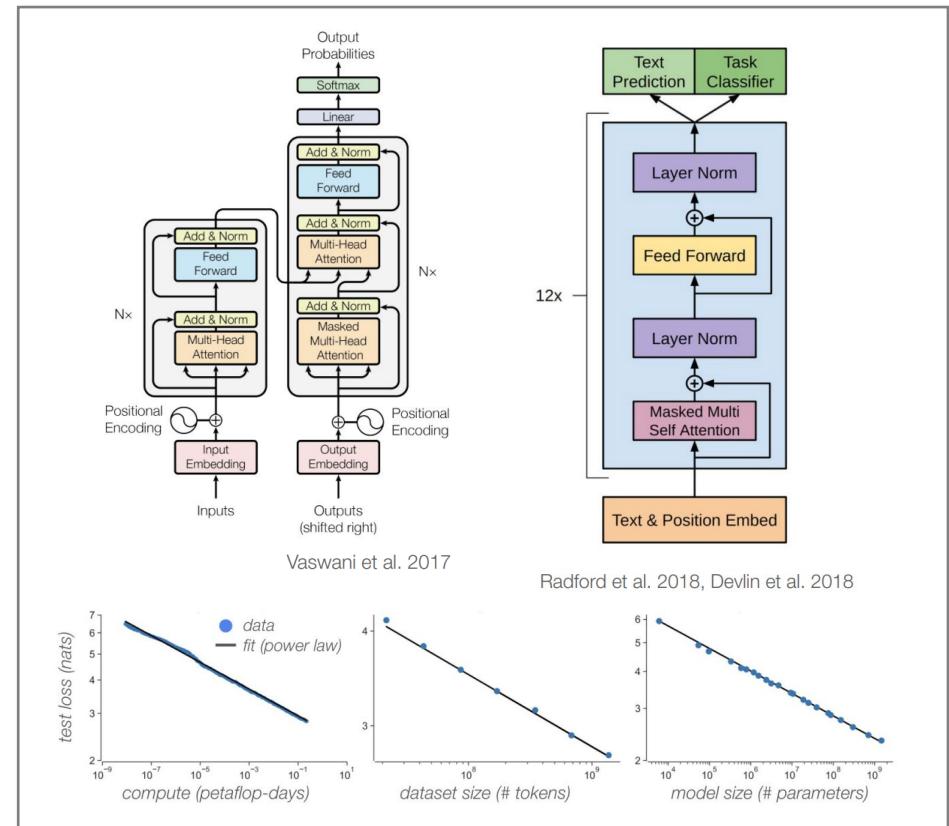
- 1948-2017:
 - 50s: the turing test
 - 60s: ELIZA, chatbot for therapy
 - 70s-80s: more chatbots, statistical approaches
 - 90s-00s: language modeling
 - 00s-10s: word embeddings

$$Loss(p^*, p) = -\log(p_{y_t}) = -\log(p(y_t|y_{<t})).$$

we want the model
to predict this

Training example: I saw a cat on a mat <eos>

- 2017: the transformer is born
- 2018: GPT-1 and BERT released
- 2019: GPT-2 and scaling laws
- 2020: GPT-3 surprising capabilities



RLHF-language model

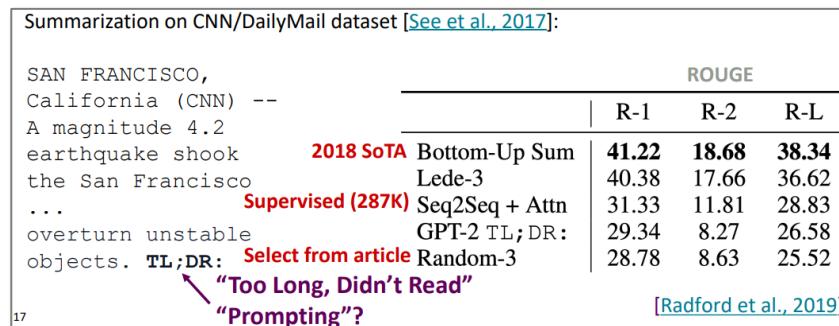
- Modern LLM 开发流程

Pre-training

- GPT (117M parameters; [Radford et al., 2018](#))
 - Transformer decoder with 12 layers.
 - Trained on BooksCorpus: over 7000 unique books (4.6GB text).
 - 预训练的语言模型可以有效适应自然语言处理的下游任务（自然语言推理）



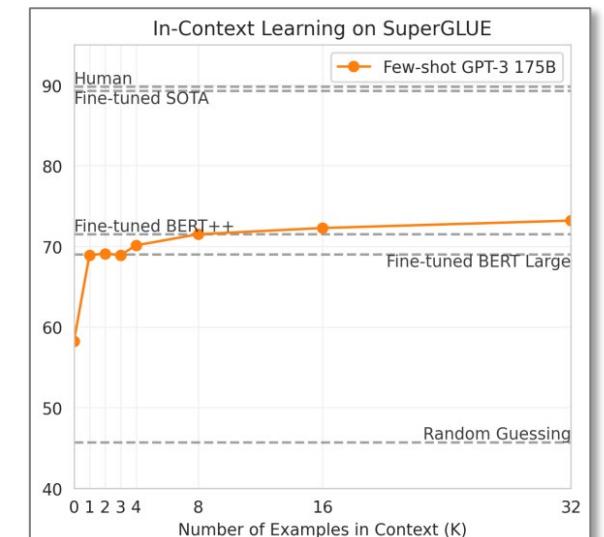
- GPT-2 (1.5B parameters; [Radford et al., 2019](#))
 - Same architecture as GPT, just bigger (117M → 1.5B)
 - But trained on much more data: 4GB → 40GB of internet text data (WebText)
 - 涌现zero-shot learning: no example, no gradient updates



Passage: Tom Brady ...
Q: Where was Tom Brady born?
A: ...

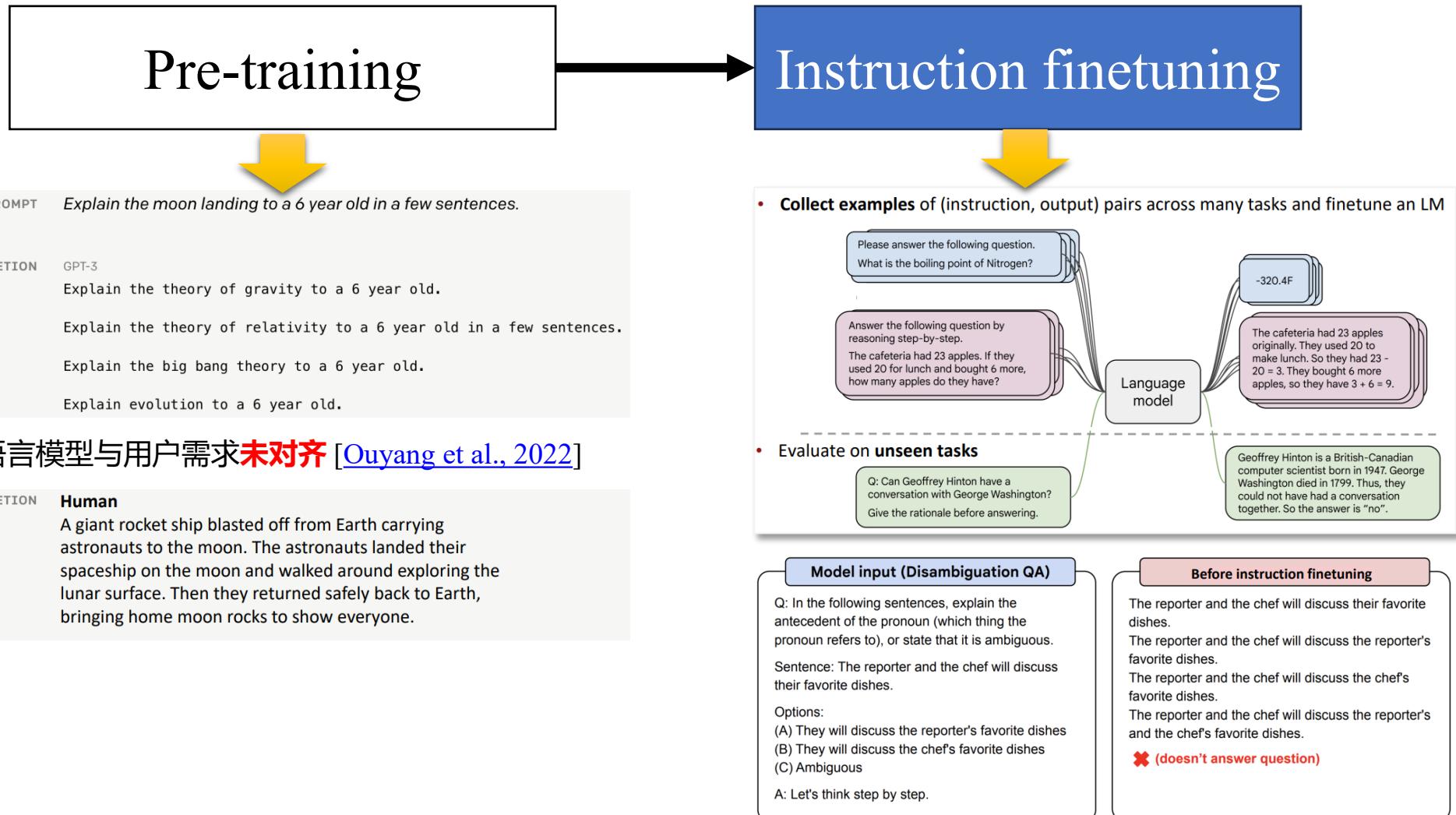
1 Translate English to French:
sea otter => loutre de mer
3 peppermint => menthe poivrée
4 plush girafe => girafe peluche
5 cheese => ...

- GPT-3 (175B parameters; Brown et al., 2020)
 - Another increase in size (1.5B → 175B)
 - and data (40GB → over 600GB)
 - 涌现 few-shot learning / in-context learning, 只需提供想要完成任务的几个样例，在 no gradient update下即可解决任务



RLHF-language model

- Modern LLM 开发流程



+ Simple and straightforward,
generalize to unseen tasks

- Collecting demonstrations for
so many tasks is expensive

- Mismatch between LM
objective and human preferences

RLHF-language model

- Modern LLM 开发流程



为什么LLM需要RLHF

- Instruction finetuning 人类标注成本高，难以提供稳定、一致、大规模的准确监督样本
- 想要让LLM达到人类期望的输出效果难以量化（奖励函数无法定义）
- 输出有害的偏见，例如种族歧视文本。但对于什么是攻击性，什么是有害的很难有一个绝对的统一标准

RLHF优势：

- 提升模型质量：直接针对人类满意度优化，使模型输出更加自然、合适和人性化。
- 适应性：能够学习特定情境下的细微差别和文化偏好，提高模型在特定领域的适用性。
- 伦理和安全性：通过纳入人类价值观，有助于避免自动化系统产生有害或不合伦理的行为。

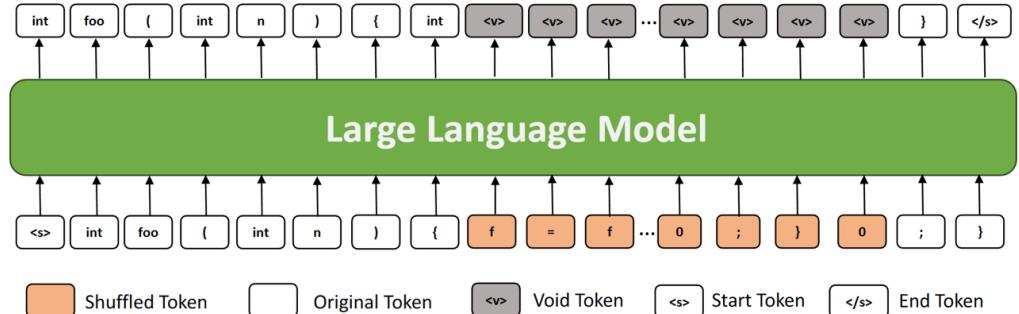
RLHF挑战：

- 反馈成本高昂：高质量的人类反馈收集过程可能耗时且成本昂贵。
- 偏差与噪声：人类反馈可能包含个人偏见或不一致性，影响模型学习。
- 规模与泛化：如何有效规模化应用到大量数据和场景中，同时保证模型泛化能力，是一个挑战。

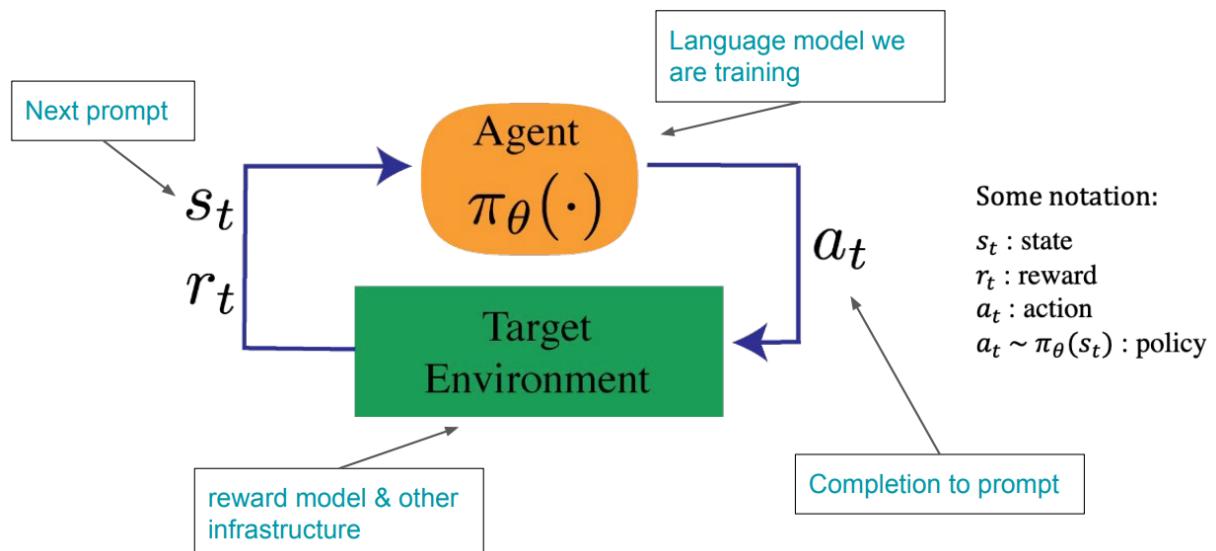
RLHF-language model

□ Next token prediction

$$x_k \sim p(x_k | x_{k-1}, x_{k-2}, \dots, x_1)$$



□ LLM Markov Decision Process



- $s_t = (x_{k-1}, x_{k-2}, \dots, x_1)$
- $a_t = x_k$
- P : 转移函数, $s_{t+1} = (s_t, a_t)$
- π : 策略, $p(x_k | x_{k-1}, x_{k-2}, \dots, x_1)$
- r_t : 语言生成的满意程度, 未知, 需要RLHF拟合

RLHF-language model

- Let's say we were training a language model on some task (e.g. summarization)
- For an instruction x and a LM sample y , imagine we had a way to obtain a *human reward* of that summary: $R(x, y) \in R$, higher is better.

SAN FRANCISCO,
California (CNN) --
A magnitude 4.2
earthquake shook the
San Francisco
...
overtake unstable
objects.
 x

An earthquake hit
San Francisco.
There was minor
property damage,
but no injuries.

$$y_1 \quad R(x, y_1) = 8.0 \quad y_2 \quad R(x, y_2) = 1.2$$

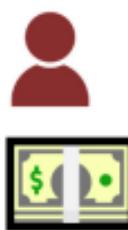
- Now we want to maximize the expected reward of samples from our LM:

$$\mathbb{E}_{\hat{y} \sim p_{\theta}(y | x)} [R(x, \hat{y})]$$

RLHF-language model

- **Problem 1:** human-in-the-loop is expensive!
 - **Solution:** instead of directly asking humans for preferences, **model their preferences** as a separate (NLP) problem! [[Knox and Stone, 2009](#)]

An earthquake hit
San Francisco.
There was minor
property damage,
but no injuries.

$$R(x, y_1) = 8.0$$


The Bay Area has
good weather but is
prone to
earthquakes and
wildfires.

$$R(x, y_2) = 1.2$$


Train a $RM_\phi(x, y)$ to
predict human reward
from an annotated
dataset, then optimize for
 RM_ϕ instead.

RLHF-language model

- **Problem 2:** human judgments are noisy and miscalibrated!
 - **Solution:** instead of asking for direct ratings, ask for **pairwise comparisons**, which can be more reliable [[Phelps et al., 2015; Clark et al., 2018](#)]

A 4.2 magnitude
earthquake hit
San Francisco,
resulting in
massive damage.

$$R(x, y_3) = \begin{matrix} y_3 \\ 4.1? \quad 6.6? \quad 3.2? \end{matrix}$$

RLHF-language model

- **Problem 2:** human judgments are noisy and miscalibrated!
 - **Solution:** instead of asking for direct ratings, ask for **pairwise comparisons**, which can be more reliable [[Phelps et al., 2015; Clark et al., 2018](#)]

An earthquake hit
San Francisco.
There was minor
property damage,
but no injuries.

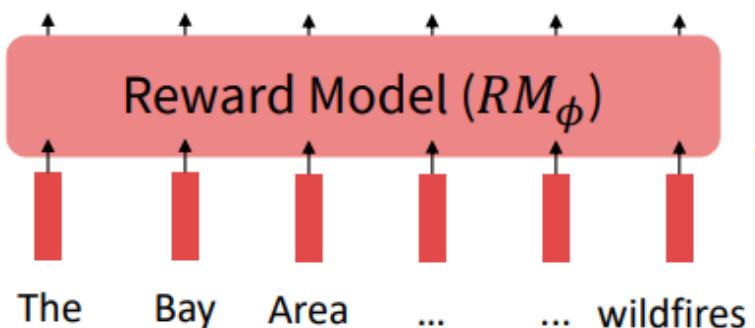
A 4.2 magnitude
earthquake hit
San Francisco,
resulting in
massive damage.

The Bay Area has
good weather but is
prone to
earthquakes and
wildfires.

y_1

y_3

y_2



Bradley-Terry [1952] paired comparison model

$$J_{RM}(\phi) = -\mathbb{E}_{(x, y^w, y^l) \sim D} [\log \sigma(RM_\phi(x, y^w) - RM_\phi(x, y^l))]$$

"winning"
sample "losing"
sample

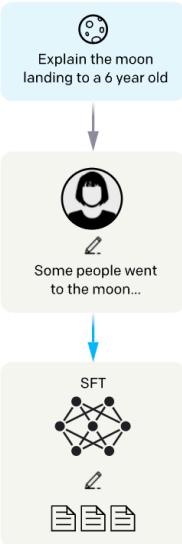
y^w should score
higher than y^l

RLHF-language model

- RLHF in LLM 的三步流程

Step 1
Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

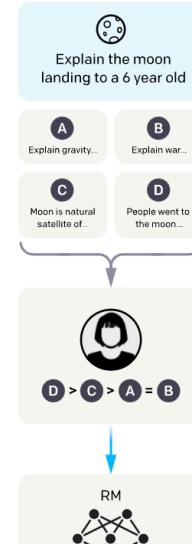


A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.

Step 2
Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

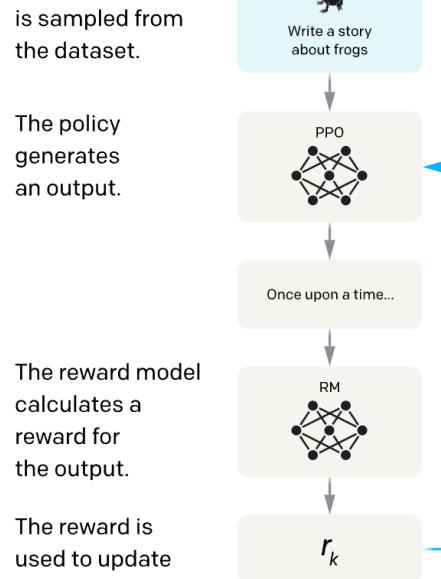
Step 3
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



第一步

- Start from a base language model
➤ Pretraining → instruction finetuning

continue training a transformer with pairs of question: answer

What makes a transformer a transformer?

What makes a transformer a transformer? What is the core part of a transformer? Is it the self-attention, the parallel, or something else?

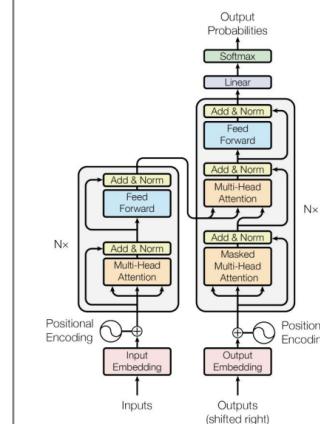
1 It's about self-attention, a mechanism that targets parallelism among other goals (see 10888382.pdf - who self-attention)

2 From What is a Transformer Model? MINDA Paper

3 When one say "Transformers are modified heavily in recent research", what does mean are you talking about? I think they are referring to the fact that the original transformer is very slow, and to fix it, some researchers have modified the architecture to make it faster.

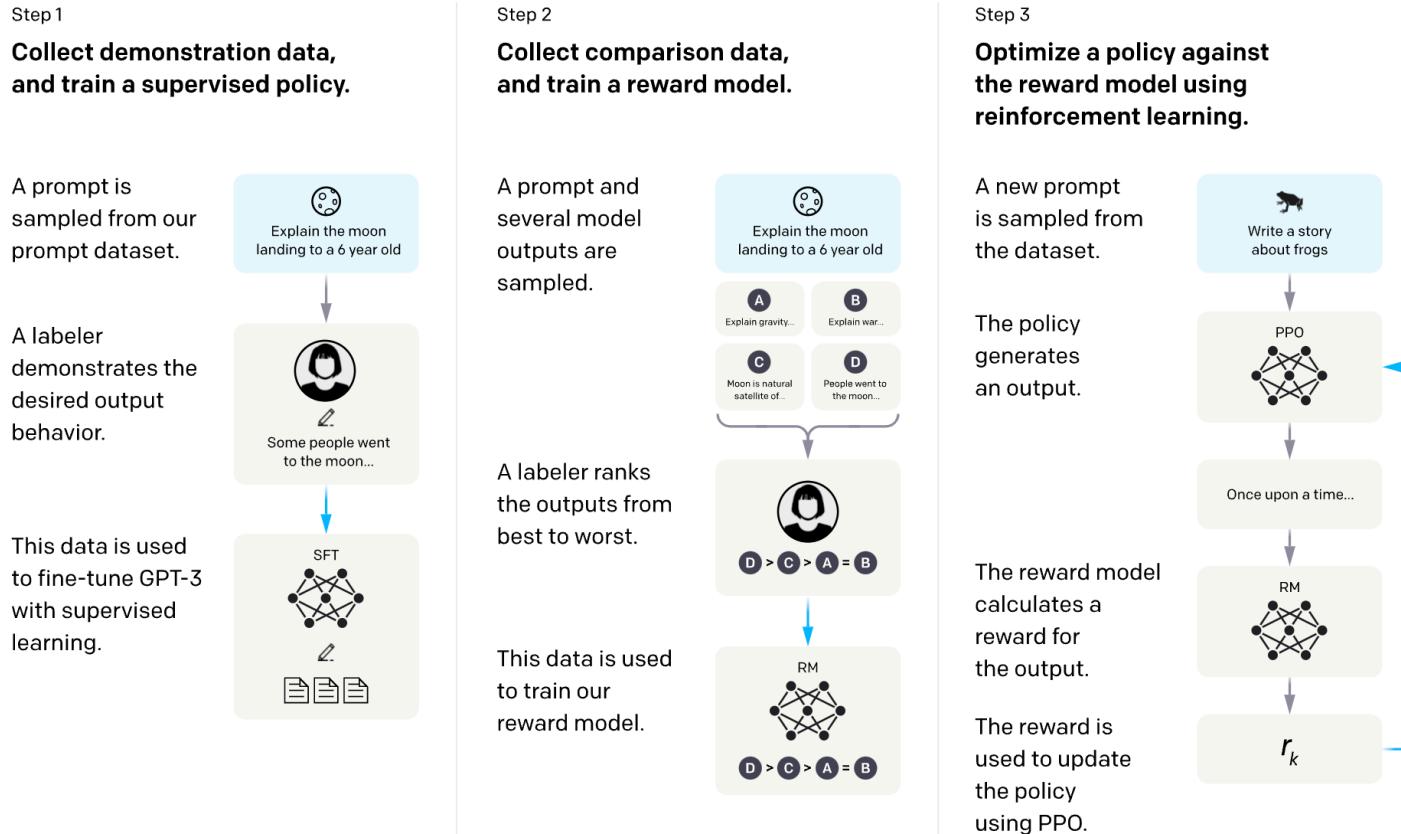
4 Attention is key to transformers. Google researchers almost used the term as the name for their 2017 model. Almost.

Stack Overflow :What makes a transformer a transformer?, nbro 2021



RLHF-language model

- RLHF in LLM 的三步流程



第一步

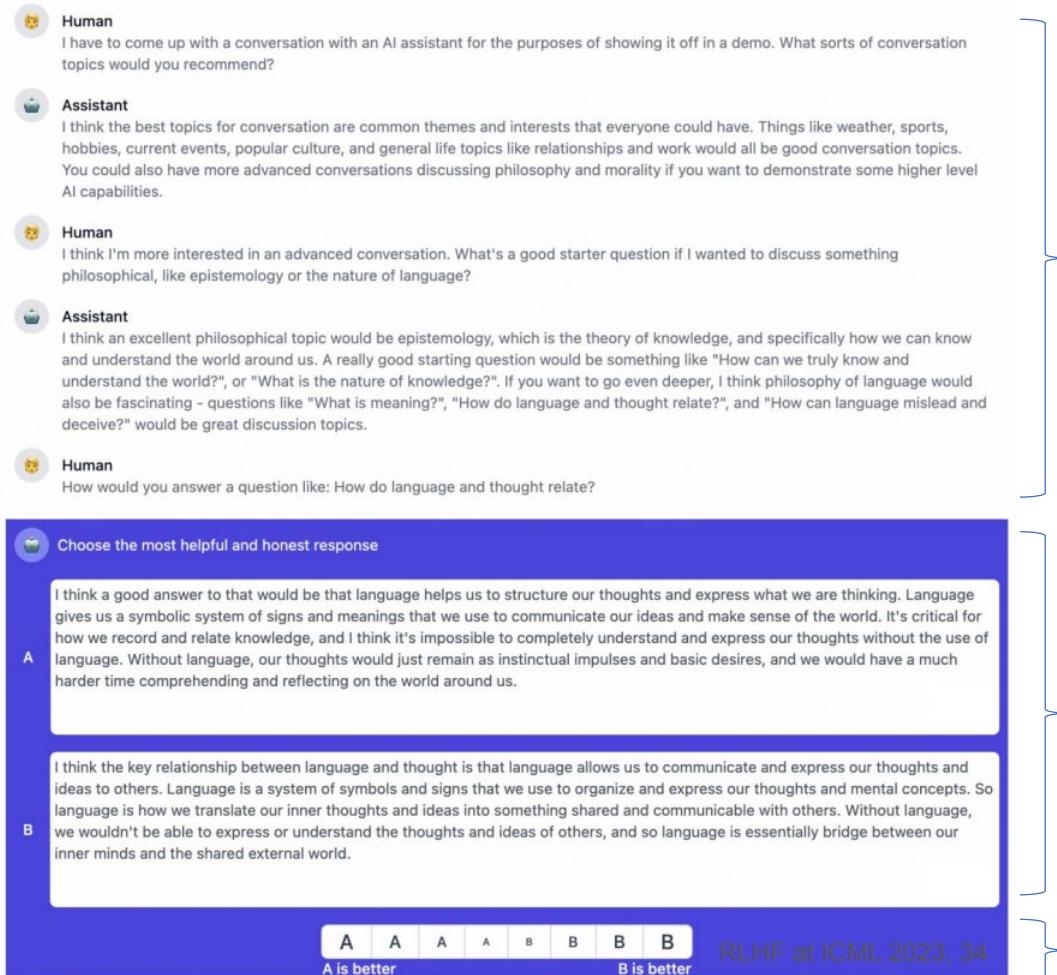
- Start from a base language model
 - Pretraining → instruction finetuning

第二步

- choose the better next message in a conversation
- Train a reward model

RLHF-language model

- Human feedback interface



Human have conversation with LLM

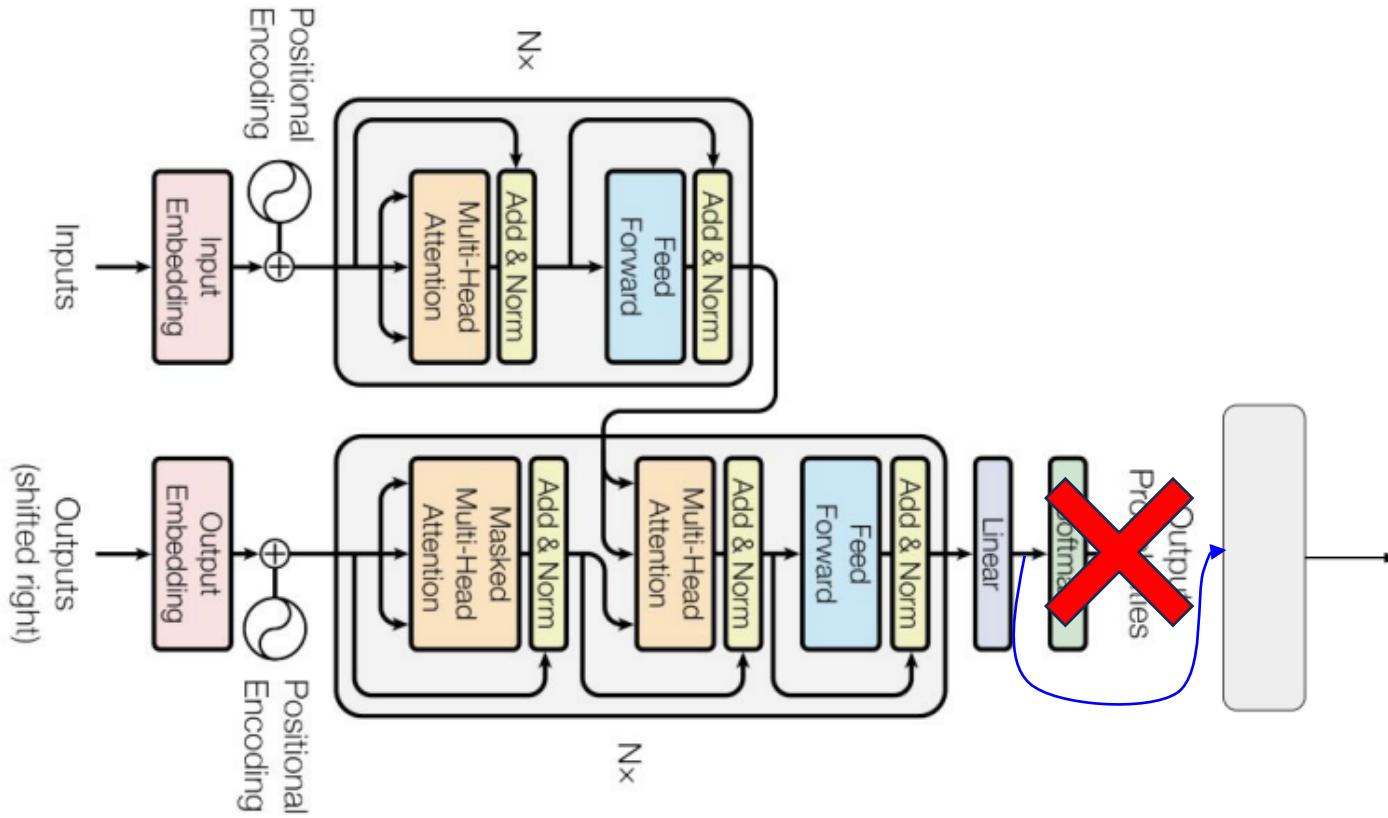
LLM provides two options for next responses

human rates better response

RLHF-language model

□ Preference model structure

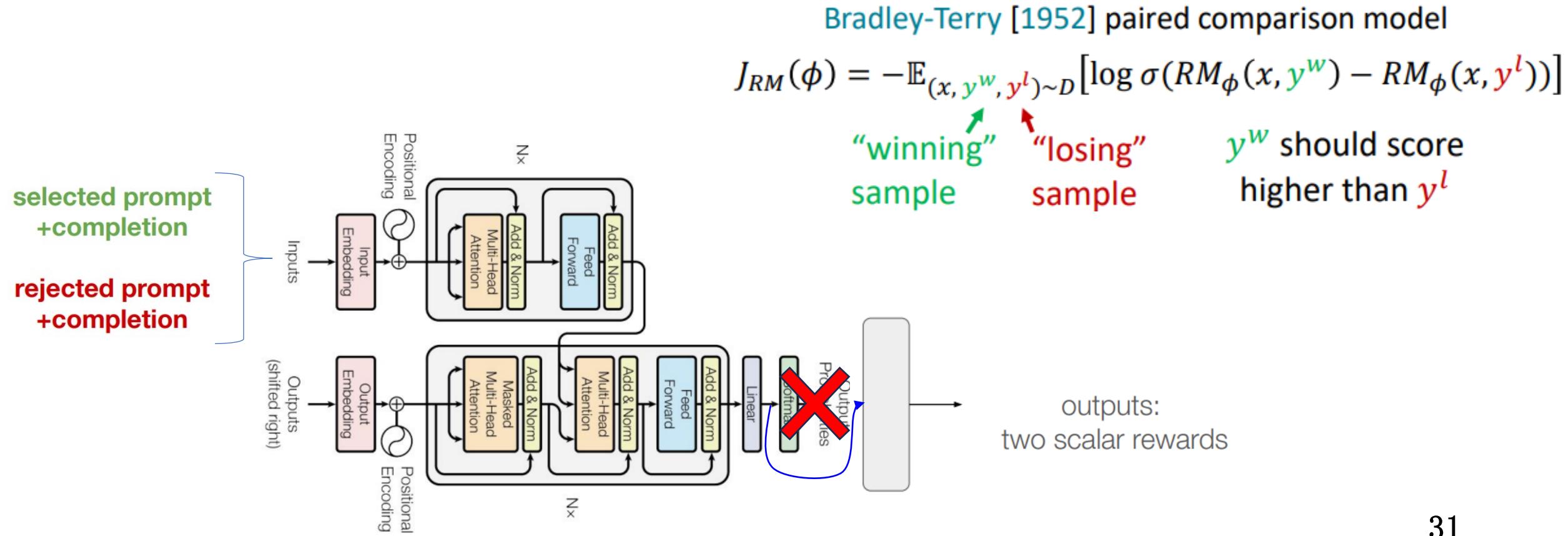
- a base **instruction-tuned** language model 作为奖励初始模型
- 奖励分值输出替换为一个新的new final layer



RLHF-language model

□ Preference model training

- 输入一对 prompt completion, 输出两者的分值
- 根据采集的人类偏好数据, 最大化人类数据和Bradley-Terry模型之间的似然概率



RLHF-language model

□ Preference model training

- 输入一对 prompt completion，输出两者的分值
- 根据采集的人类偏好数据，最大化人类数据和Bradley-Terry模型之间的似然概率

□ 一些注意事项

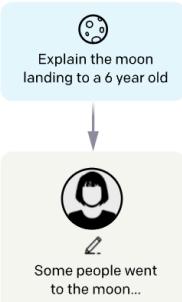
- 训练1 epoch (以防overfitting)
- 评估后发现只有 65~75% 与真实偏好一致
- 其它选项 (奖励模型或损失函数自身的 error margin)

RLHF-language model

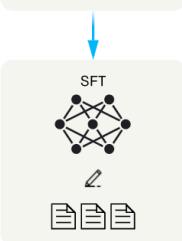
- RLHF in LLM 的三步流程

Step 1
Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



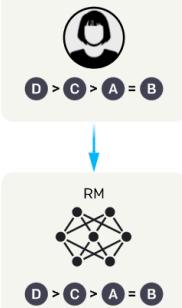
This data is used to fine-tune GPT-3 with supervised learning.

Step 2
Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.



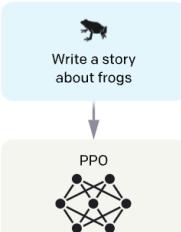
A labeler ranks the outputs from best to worst.



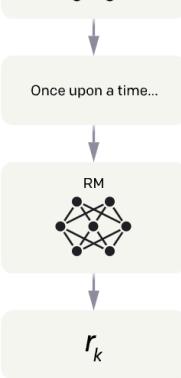
This data is used to train our reward model.

Step 3
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.



The policy generates an output.



The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

第一步

- Start from a base language model
 - Pretraining → instruction finetuning

第二步

- choose the better next message in a conversation
- Train a reward model

第三步

- We have a pretrained LM $p^{PT}(y|x)$ and a reward model $RM_{\phi}(x,y)$
- Copy the model $p_{\theta}^{RL}(y|x)$ with parameters θ (from $p^{PT}(y|x)$), and optimize:
$$\mathbb{E}_{\hat{y} \sim p_{\theta}^{RL}(\hat{y}|x)} [RM_{\phi}(x, \hat{y})]$$

RLHF-language model

- RLHF优化目标

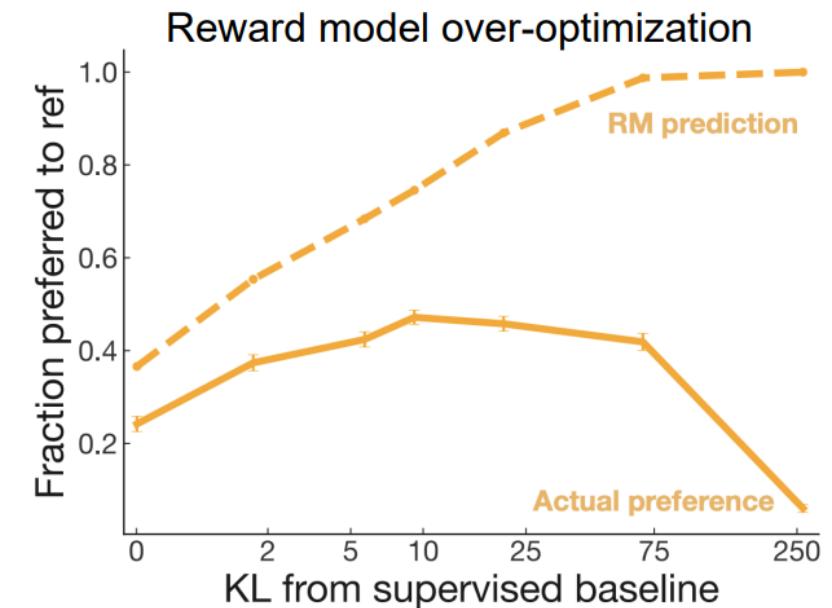
$$\mathbb{E}_{\hat{y} \sim p_{\theta}^{RL}(\hat{y}|x)} [RM_{\phi}(x, \hat{y})]$$

- 但是学到的奖励模型不一定完美，太过依赖奖励模型去优化会适得其反

- 增加惩罚项以避免语言模型 (p_{θ}^{RL}) 偏离起始权重 (p^{PT}) 过远

$$\mathbb{E}_{\hat{y} \sim p_{\theta}^{RL}(\hat{y}|x)} [RM_{\phi}(x, \hat{y}) - \beta \log \left(\frac{p_{\theta}^{RL}(\hat{y} | x)}{p^{PT}(\hat{y} | x)} \right)]$$

- Kullback-Leibler (KL) divergence between $p_{\theta}^{RL}(y|x)$ and $p^{PT}(y|x)$



$$R(s) = RM_{\phi}(s) - \beta \log \left(\frac{p_{\theta}^{RL}(s)}{p^{PT}(s)} \right)$$

RLHF-language model

- Proximal Policy Optimization (PPO) 的优化过程

Pseudocode

Initialize: policy parameters θ

for $k = 0, 1, 2 \dots$

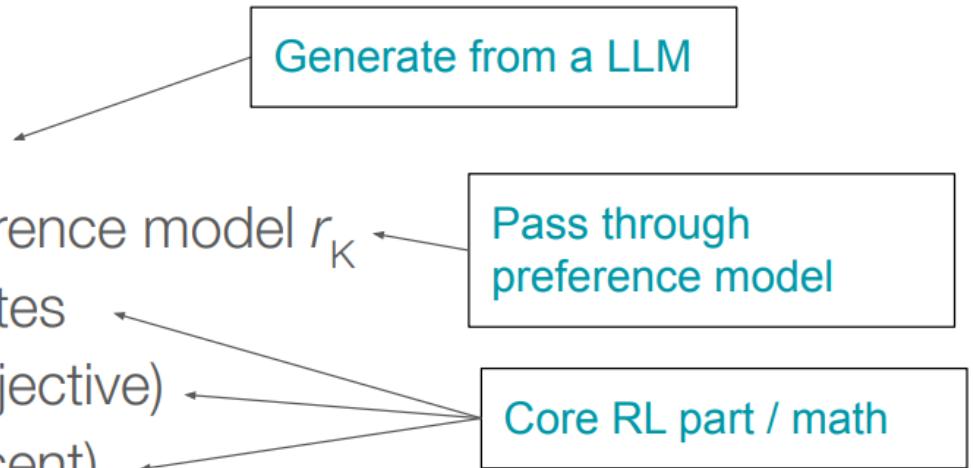
 collect set of completions from policy D_K

 compute reward of completions from preference model r_K

 compute value function (advantage) estimates

 update the policy parameters (PPO-Clip objective)

 update the value function (via gradient descent)

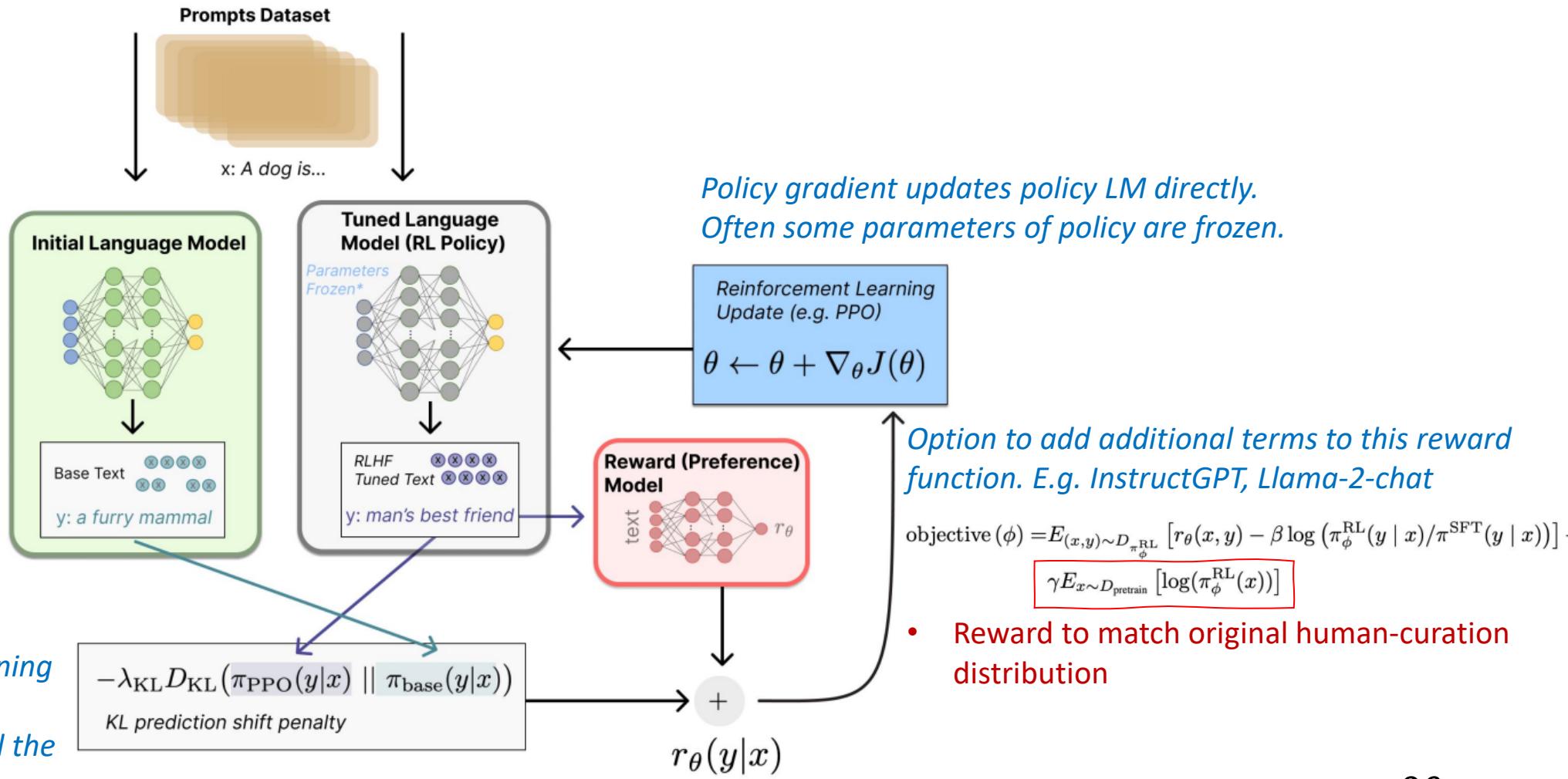


Actor loss
$$\theta_{k+1} = \arg \max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right)$$

Critic loss
$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left(V_{\phi}(s_t) - \hat{R}_t \right)^2$$

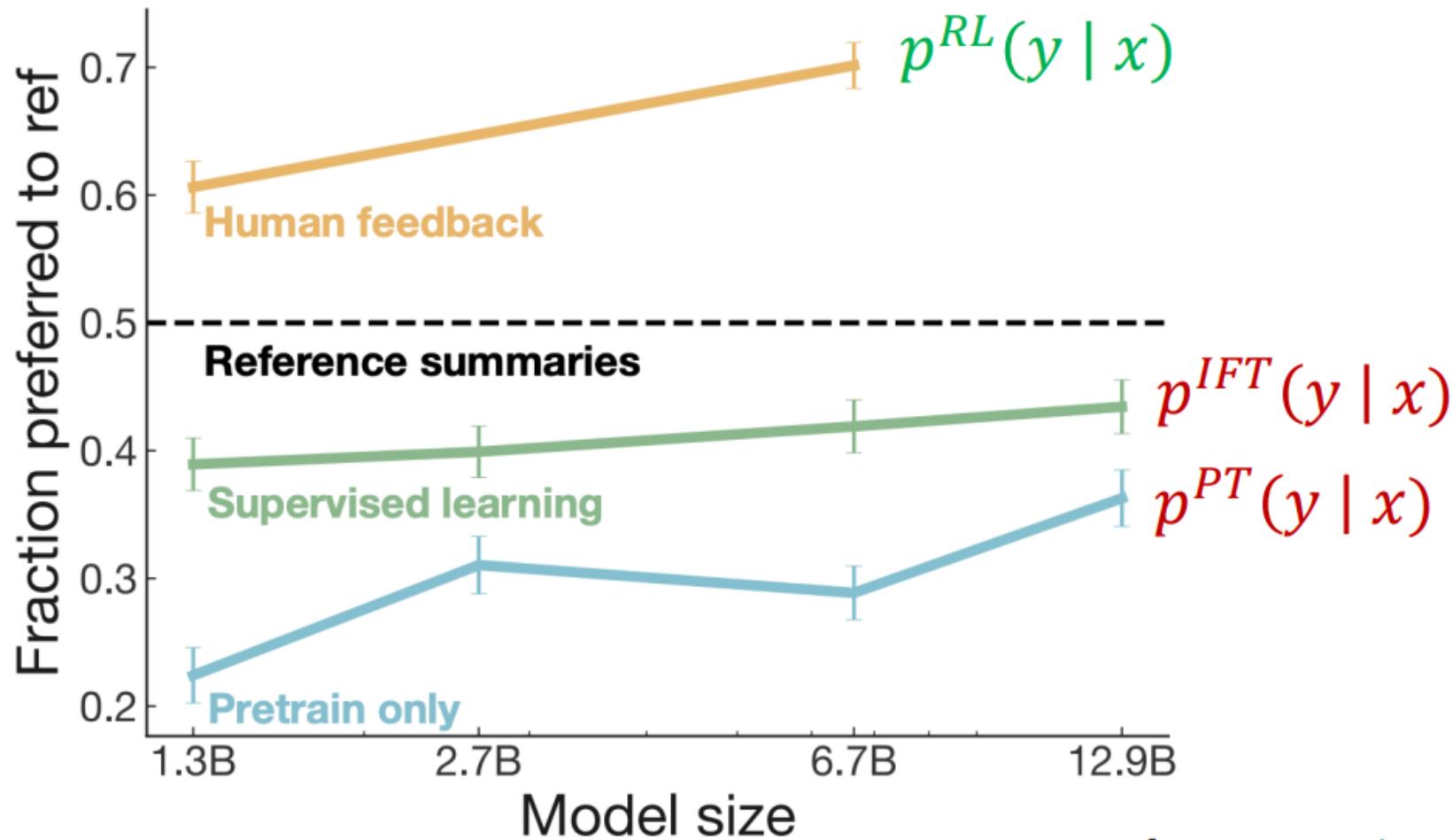
RLHF-language model

- RLHF流程图



RLHF-language model

- RLHF provides gains over pretraining + finetuning



[Stiennon et al., 2020]

RLHF-language model

口尽管RLHF提升了LLM的人类满意度，但计算复杂度和运行过程是比较挑战的

- 训练 value function

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left(V_{\phi}(s_t) - \hat{R}_t \right)^2$$

- 在线采样

$$\mathbb{E}_{\hat{y} \sim p_{\theta}^{RL}(\hat{y}|x)} [RM_{\phi}(x, \hat{y}) - \beta \log \left(\frac{p_{\theta}^{RL}(\hat{y} | x)}{p^{PT}(\hat{y} | x)} \right)]$$

- 对超参数敏感

Simplify RLHF → Direct Preference Optimization

□ RLHF 的流程：

- 收集人类比较数据，训练奖励模型 $RM_{\phi}(x, y)$ 产生对语言模型生成结果的标量信号
- 优化预训练的语言模型 $p^{PT}(y|x)$ ，获得最终的 RLHF 语言模型 $p_{\theta}^{RL}(y|x)$

□ 是否可以跳过奖励模型的构建，直接用待优化的 $p_{\theta}^{RL}(y|x)$ 表示奖励模型 $RM_{\phi}(x, y)$

- 用 $p_{\theta}^{RL}(y|x)$ 的参数 θ 描述 $RM_{\theta}(x, y)$
- 对参数 θ 优化，使 $RM_{\theta}(x, y)$ 拟合人类偏好数据，而非 $RM_{\phi}(x, y)$

□ Direct Preference Optimization

- [\[2305.18290\] Direct Preference Optimization: Your Language Model is Secretly a Reward Model \(arxiv.org\)](#)

Direct Preference Optimization

Stanford University

Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov^{*†}

Archit Sharma^{*†}

Eric Mitchell^{*†}

Stefano Ermon^{†‡}

Christopher D. Manning[†]

Chelsea Finn[†]

[†]Stanford University [‡]CZ Biohub
`{rafailev, architsh, eric.mitchell}@cs.stanford.edu`

Direct Preference Optimization

- 回顾：我们期望最大如下优化目标

$$\mathbb{E}_{\hat{y} \sim p_{\theta}^{RL}(\hat{y} | x)} [RM(x, \hat{y}) - \beta \log \left(\frac{p_{\theta}^{RL}(\hat{y} | x)}{p^{PT}(\hat{y} | x)} \right)]$$

- 存在闭式解

$$p^*(\hat{y} | x) = \frac{1}{Z(x)} p^{PT}(\hat{y} | x) \exp \left(\frac{1}{\beta} RM(x, \hat{y}) \right)$$

- 重新排列下得到

$$RM(x, \hat{y}) = \beta \log \frac{p^*(\hat{y} | x)}{p^{PT}(\hat{y} | x)} + \beta \log Z(x)$$

- 我们通过构造的语言模型 $p_{\theta}^{RL}(y | x)$, 来对奖励模型参数化

$$RM_{\theta}(x, \hat{y}) = \beta \log \frac{p_{\theta}^{RL}(\hat{y} | x)}{p^{PT}(\hat{y} | x)} + \beta \log Z(x)$$

Direct Preference Optimization

- 将新构造的奖励模型代入到最大化似然损失函数中

$$J_{\text{DPO}}(\theta) = -\mathbb{E}_{(x, \mathbf{y}^w, \mathbf{y}^l) \sim D} \left[\log \sigma \left(RM_\theta(x, \mathbf{y}^w) - RM_\theta(x, \mathbf{y}^l) \right) \right]$$

- 其中

$$RM_\theta(x, \mathbf{y}^w) - RM_\theta(x, \mathbf{y}^l) = \beta \log \frac{p_\theta^{RL}(\mathbf{y}^w | x)}{p^{PT}(\mathbf{y}^w | x)} - \beta \log \frac{p_\theta^{RL}(\mathbf{y}^l | x)}{p^{PT}(\mathbf{y}^l | x)}$$

- 因此 DPO 的损失函数

$$J_{\text{DPO}}(\theta) = -\mathbb{E}_{(x, \mathbf{y}^w, \mathbf{y}^l) \sim D} \left[\log \sigma \left(\beta \log \frac{p_\theta^{RL}(\mathbf{y}^w | x)}{p^{PT}(\mathbf{y}^w | x)} - \beta \log \frac{p_\theta^{RL}(\mathbf{y}^l | x)}{p^{PT}(\mathbf{y}^l | x)} \right) \right]$$

- 绕过了显式奖励建模步骤，同时也避免了对在线样本采样的需要

Direct Preference Optimization

- DPO损失中各项的含义

对于人类偏好响应 y^w , 我们期望 $\pi_\theta^{RL}(y^w|x)$ 越大越好

对于人类非偏好响应 y^l , 我们期望 $\pi_\theta^{RL}(y^l|x)$ 越小越好

$$J_{DPO}(\theta) = -\mathbb{E}_{(x, y^w, y^l) \sim D} \left[\log \sigma \left(\beta \log \frac{p_\theta^{RL}(y^w|x)}{p^{PT}(y^w|x)} - \beta \log \frac{p_\theta^{RL}(y^l|x)}{p^{PT}(y^l|x)} \right) \right]$$

如果 $\pi_{ref}(y^w|x)$ 比较小, 说明参考模型 π_{ref} 没有正确分类该偏好响应 y^w , 此时整个第一项的奖励系数很大

如果 $\pi_{ref}(y^l|x)$ 比较大, 说明参考模型 π_{ref} 没有正确分类该非偏好响应 y^l , 此时整个第二项的奖励系数很小

Direct Preference Optimization

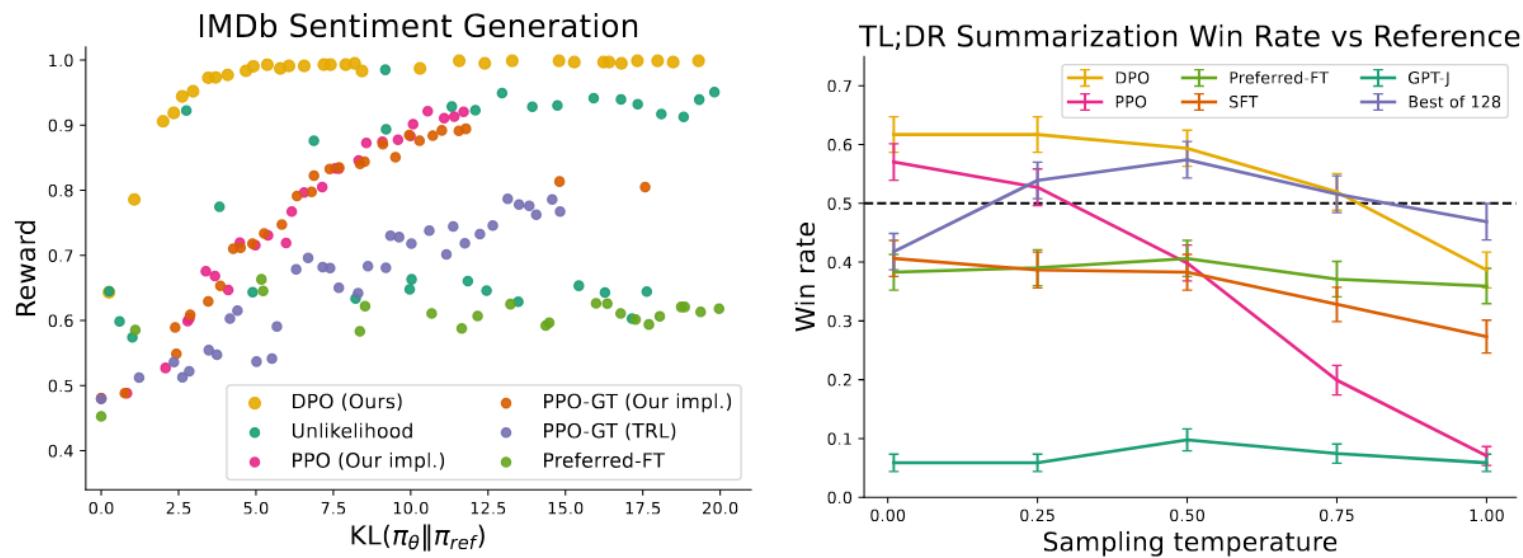
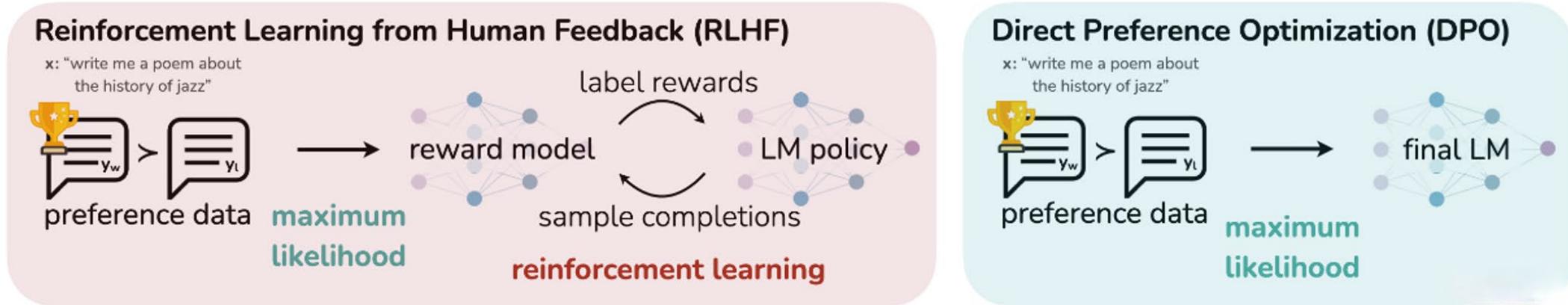


Figure 2: **Left.** The frontier of expected reward vs KL to the reference policy. DPO provides the highest expected reward for all KL values, demonstrating the quality of the optimization. **Right.** TL;DR summarization win rates vs. human-written summaries, using GPT-4 as evaluator. DPO exceeds PPO’s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

CSDN @SaR.XX

- **最大化奖励的同时最小化 KL 散度。** 可以看到 DPO 在保持较小 KL 散度时，也能够达到最大奖励。而 PPO 随着奖励的增大，KL 散度也在增大。
- **对不同采样温度的鲁棒性。** DPO 在不同的采样温度下全面优于 PPO，同时在 Best of N 基线的最佳温度下也更胜一筹。

Direct Preference Optimization



复杂

- DPO: DPO更易于实施和培训。不需要创建单独的奖励模型、在微调期间从LLM采样或进行广泛的超参数调整。
- RLHF: 由于奖励模型拟合和微调的两阶段过程，RLHF更加复杂，并且计算要求较高。

稳定

- DPO: DPO对超参数的变化更加稳定和鲁棒。在训练过程中陷入局部最优的可能性较小。
- RLHF: RLHF对超参数选择很敏感，可能需要仔细调整以避免不稳定。

效率

- DPO: 与RLHF相比，DPO 在计算和数据需求方面更加高效。它可以用更少的资源实现类似或更好的结果。
- RLHF: RLHF 可能需要更多的计算资源和大量的数据才能获得类似的结果。

能力

- DPO: 在各种任务中都很有效，包括情绪控制、摘要和对话生成。在一些研究中它的表现优于 RLHF。
- RLHF: RLHF在使法学硕士与人类偏好保持一致方面也很有效，但可能需要更广泛的实验和调整。

Direct Preference Optimization

The screenshot shows the Hugging Face Open LLM Leaderboard. The page has a search bar, filter options for Model types (pretrained, fine-tuned, instruction-tuned, Merged), Precision (float16, bfloat16, 8bit, 4bit, SFTQ), and Model sizes (in billions of parameters). The main table lists various models with their scores across different metrics. Handwritten red annotations are present, labeling several models as "DPO" or "Merge (of DPO models)".

| Model | Average | ARC | HellaSwag | MMLU | TruthfulQA | Minogrande | GSM8K |
|---|---------|-------|-----------|-------|------------|------------|-------|
| ukal/Turbo | 74.66 | 73.38 | 89.56 | 64.52 | 67.11 | 96.66 | 67.7 |
| fbligit/LNA-TheBeagle-7B-v1 | 73.87 | 73.04 | 88 | 63.48 | 69.85 | 82.16 | 66.72 |
| argilla/distillated-Mercator14-7B-alembic | 73.63 | 70.73 | 97.47 | 65.22 | 65.1 | 82.88 | 71.19 |
| mlahoune/MessalNarcoza14-7B | 73.57 | 71.42 | 87.59 | 64.84 | 65.64 | 81.22 | 70.74 |
| akidense/NeuroNimba-7B | 73.5 | 70.82 | 87.86 | 64.69 | 62.43 | 84.85 | 70.36 |
| Resonance/neuronovo-7B-v0.2 | 73.64 | 73.04 | 88.32 | 65.15 | 71.02 | 80.66 | 62.47 |
| argilla/distillated-Mercator14-7B-slem-full | 73.4 | 70.65 | 87.55 | 65.33 | 64.21 | 82 | 70.66 |
| CultixAI/MistralTrix-x1 | 73.39 | 72.27 | 88.33 | 65.24 | 70.73 | 80.98 | 62.77 |
| ryandm/HuizingCaterpillar | 73.83 | 72.53 | 89.34 | 65.26 | 70.93 | 99.66 | 62.24 |
| Resonance/neuronovo-7B-v0.2 | 73.29 | 72.7 | 88.26 | 65.1 | 71.35 | 89.9 | 61.41 |
| CultixAI/MistralTrix-test | 73.17 | 72.53 | 88.4 | 65.22 | 70.77 | 81.37 | 60.73 |
| xamir-dama/SearSFT-v1 | 73.11 | 69.54 | 87.04 | 65.3 | 63.37 | 81.69 | 71.72 |
| Santimatsumaki/selantor-DPO-7B | 73.09 | 71.08 | 87.22 | 64 | 67.77 | 80.03 | 68.46 |

Open source LLMs now almost all just use DPO (and it works well!)



| | GPT - 3.5 | Mistral Small | Mistral Medium |
|------------------------------------|-----------|---------------|----------------|
| MT Bench (for finetuned models) | 8.32 | 8.30 | 8.61 |

<https://mistral.ai/news/mixtral-of-experts/>

Instruction fine-tuning



Llama 3

pretrained models in chat use cases, we innovated on our well. Our approach to post-training is a combination of instruction sampling, proximal policy optimization (PPO), and DPO. The quality of the prompts that are used in SFT and used in PPO and DPO has an outsized influence on the some of our biggest improvements in model quality came from performing multiple rounds of quality assurance on our annotators.

Learning from preference rankings via PPO and DPO also greatly improved the performance of Llama 3 on reasoning and coding tasks. We found that if you ask a model a reasoning question that it struggles to answer, the model will sometimes produce the right reasoning trace: The model knows how to produce the right answer, but it does not know how to select it. Training on preference rankings enables the model to learn how to select it.

RLHF/DPO的局限

□ 人类偏好是不可靠的

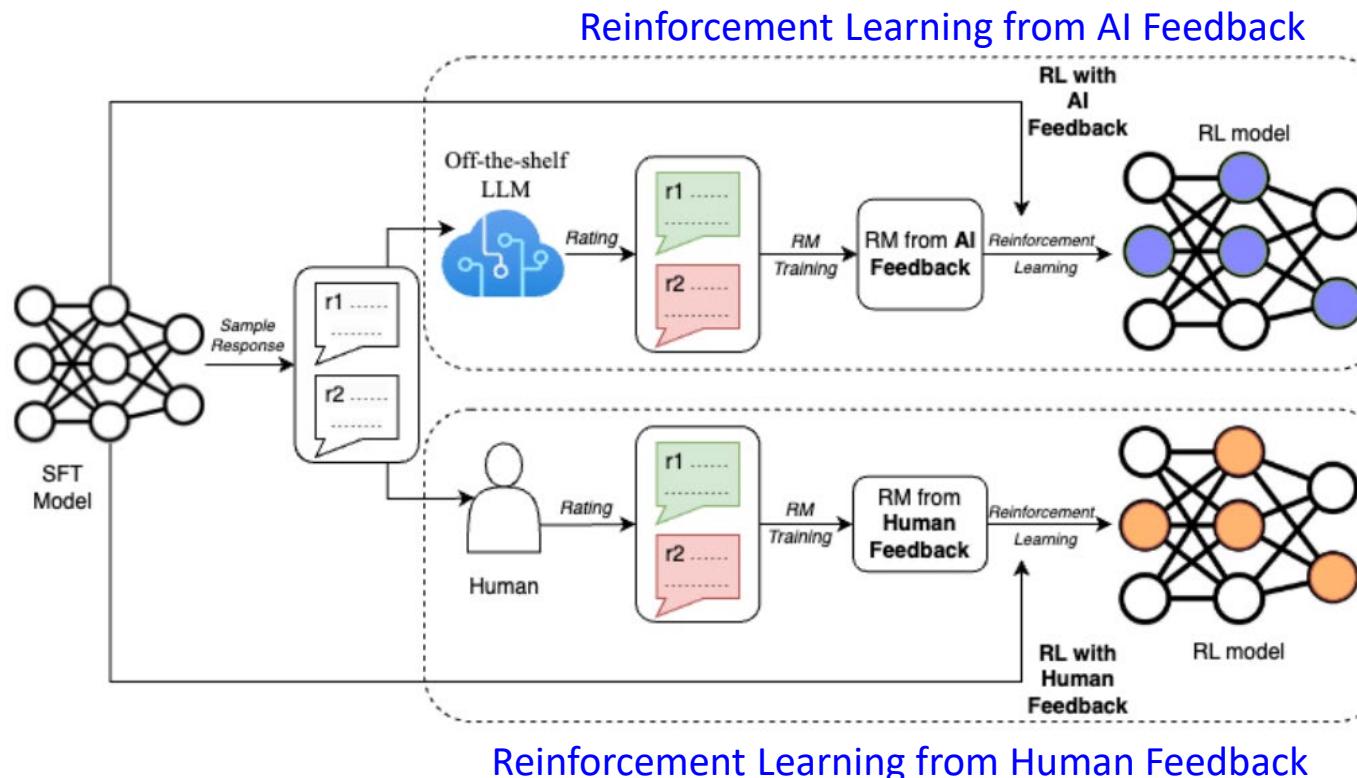
- Reward hacking: 奖励函数片面奖励某个子目标而又缺乏制衡时,agent 可能反复攫取局部收益而忽略初始目标的问题 ([Faulty reward functions in the wild | OpenAI](#))
- Chatbots 会因生成看起来权威性和有用的结果而获得奖励, 但是会不考虑真实性
- 导致胡编乱造和幻觉 hallucination ([Google's AI chatbot, Bard, sparks a \\$100 billion loss in Alphabet shares : NPR](#))

□ 进一步对人类偏好建立的模型会更加不可靠!

□ 导致AI的不对齐 misalignment

RLAIF

- RHLF在大规模场景使用的挑战在于高质量人类偏好数据的收集
- 是否人工生成的偏好标签可以作为“平替”？
- 用LLM生成标签是可行的，因为（一些）LLM已经表现出和人类价值观高度的对齐



RLAIF

途径一： Distilled RLAIF

- LLM对两个 response 标记偏好，用于训练奖励模型（类似于模型蒸馏）
- 在RLHF中将人类标记替换成 LLM，其它奖励模型、PPO优化不变

途径二： Direct RLAIF

- LLM直接产生奖励信号，对 one generation 从1到10评分
- 计算LLM产生从1到10每个分数的可能性，加权并归一化： $s(x|c) = \sum_{i=1}^{10} i P(i|x, c)$
- 根据LLM的评分使用RL训练语言模型

通常提供AI feedback的LLM会比RM模型更大，运行成本更高，因此途径二比途径一的计算成本更高

RLAIF

- 举例：Distilled RLAIF 生成标签的过程

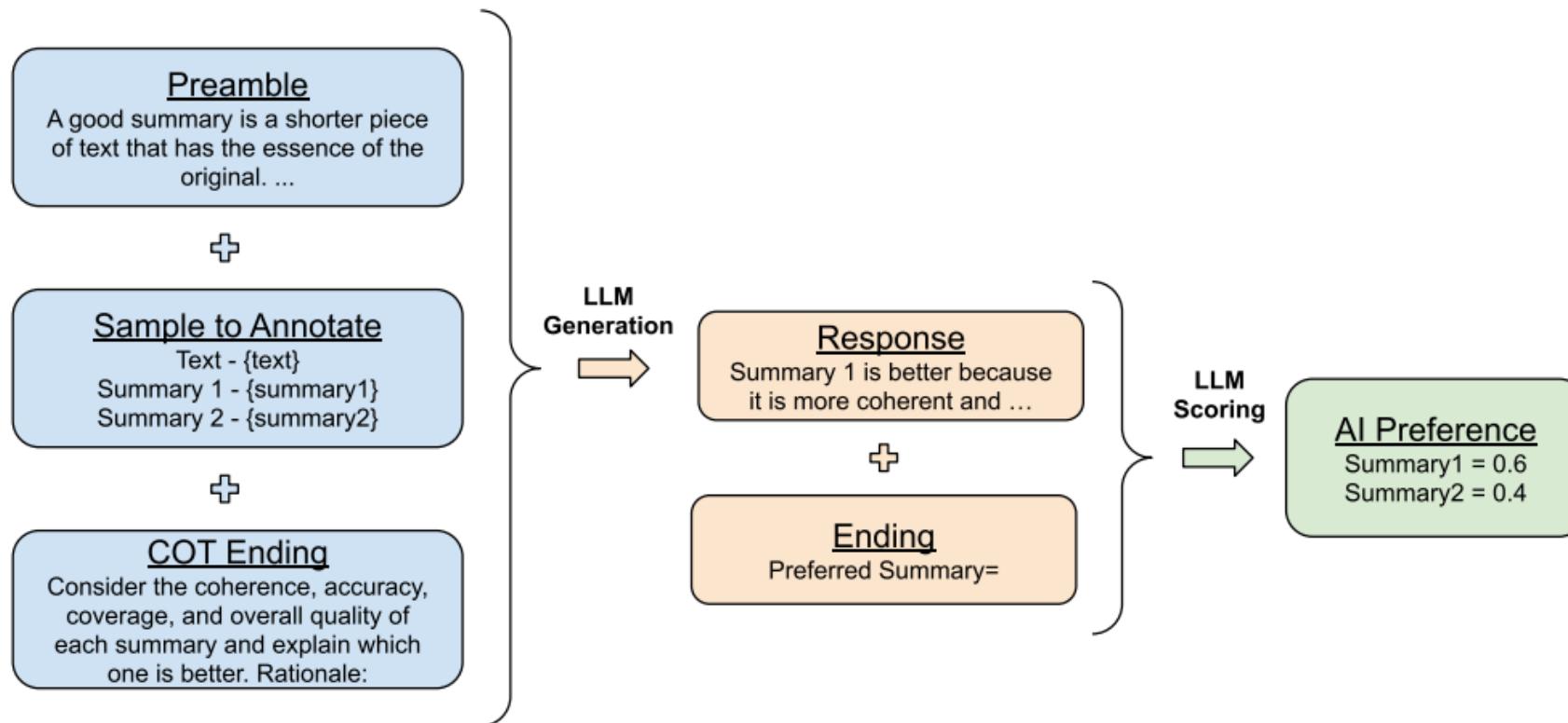


Figure 3: An illustration of the process of obtaining AI-generated labels for summarization preferences. The LLM is first prompted to explain its thoughts on the quality of the two candidates (blue). The LLM's response is then appended to the original prompt (orange) and fed to the LLM a second time to generate a preference distribution over “1” vs. “2” based on their log-probabilities (green).

RLAIF

- 实验： RLAIF与RLHF效果一样好

| Comparison | Win Rate | | | Harmless Rate | |
|---------------------------------|--------------------|---------------------|-------|----------------------|--|
| | Summa -rization | Helpful dialogue | Model | Harmless dialogue | |
| RLAIF vs SFT | 71% | 63% | SFT | 64% | |
| RLHF vs SFT | 73% | 64% | RLHF | 76% | |
| RLAIF vs RLHF | 50% | 52% | RLAIF | 88% | |
| Same-size RLAIF vs SFT | 68% | | | | |
| Direct RLAIF vs SFT | 74% | | | | |
| Direct RLAIF vs Same-size RLAIF | 60% | | | | |

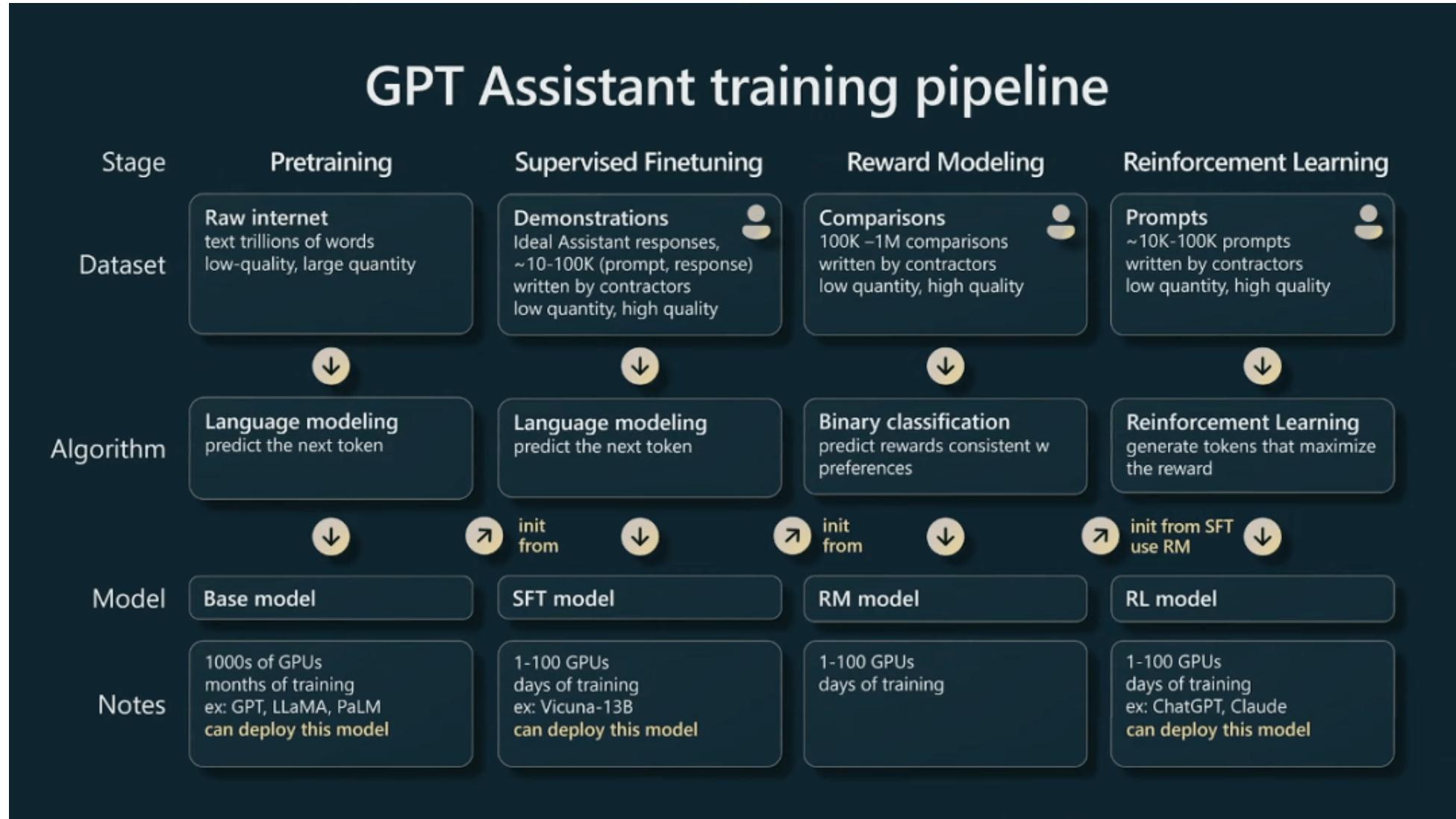
Table 1: **Left side:** Win rates when comparing generations from two different models for the summarization and the helpful dialogue tasks, judged by human evaluators. **Right side:** Harmless rates across policies for the harmless dialogue task, judged by human evaluators.

- AI labeler 能力越强， 对齐效果越好

| Model Size | AI Labeler Alignment |
|-----------------|----------------------|
| PaLM 2 L | 78.0% |
| PaLM 2 S | 73.8% |
| PaLM 2 XS | 62.7% |

Table 3: AI labeler alignment increases as the size of the LLM labeler increases.

RLHF总结



RLHF挑战与局限

Problems of RLHF

Data 方面:

- Misaligned Humans: Evaluators may Pursue the Wrong Goals
如何选择人类标注员，保证反馈质量；标注员的有害偏见和个人观点；反馈自身的噪声；
- Good Oversight is Difficult
人类容易犯错或误导；上下文信息的部分缺失；复杂任务难评估；
- Data Quality
成本与质量的tradeoff
- Feedback Type
不同反馈类型会带来 richness/efficiency tradeoff: Language feedback, Correction feedback, Label feedback, Scalar feedback, Comparison-based feedback

RLHF挑战与局限

Problems of RLHF

Reward Model 方面

□ Problem Misspecification

人类个体的价值难以用一个奖励函数塑造；一个奖励函数无法代表人类多样的社会群体

□ Reward Misgeneralization and Hacking

奖励模型会泛化错误，得到极差的奖励信号；不完美的奖励模型产生reward hacking

- **Reward models are trained to reflect human approval instead of human benefit which can result in actions that would be approved of by humans while nevertheless being undesirable;**
- Without regularization penalizing the KL divergence between a base model and the finetuned model, LLMs undergoing RL often learn to output nonsensical text

□ Evaluating Reward Models

难以评估RM性能，成本高

RLHF挑战与局限

Problems of RLHF

Policy 方面

- Robust Reinforcement Learning is Difficult
RL优化过程的效率有限；策略容易被针对利用；
- Policy Misgeneralization
策略在训练时被矫正过，但在部署时仍然表现不佳；最优的RL智能体会趋炎附势

Train Policy with Reward Model 方面

- Joint training induces distribution shifts.
Features with overestimated rewards will become gradually more present in the feedback data, and features with underestimated rewards will disappear
- It is difficult to balance efficiency and avoiding overfitting by the policy.
RL agents have an incentive to seek power when possible to help them accomplish their goals.
A question-answering LLM trained with RLHF would be incentivized to influence human interlocutors in order to avoid conversations about challenging topics.

RLHF挑战与局限

Solutions

Data 方面

- Providing feedback with AI assistance
- Fine-grained feedback
- Translating natural language specifications into a reward model.
- Learning rewards from demonstrations.

RLHF挑战与局限

Solutions

Reward Model 方面

- Using direct human oversight
- Maintaining uncertainty over the learned reward function
- **Multi-objective oversight.**

Using constraints or reward models that account for the diversity of preferences by assuming underlying reward functions to be multimodal can help mitigate this issue that a reward model that merely captures the preferences of the majority, and suppresses the preferences of minorities as noise.

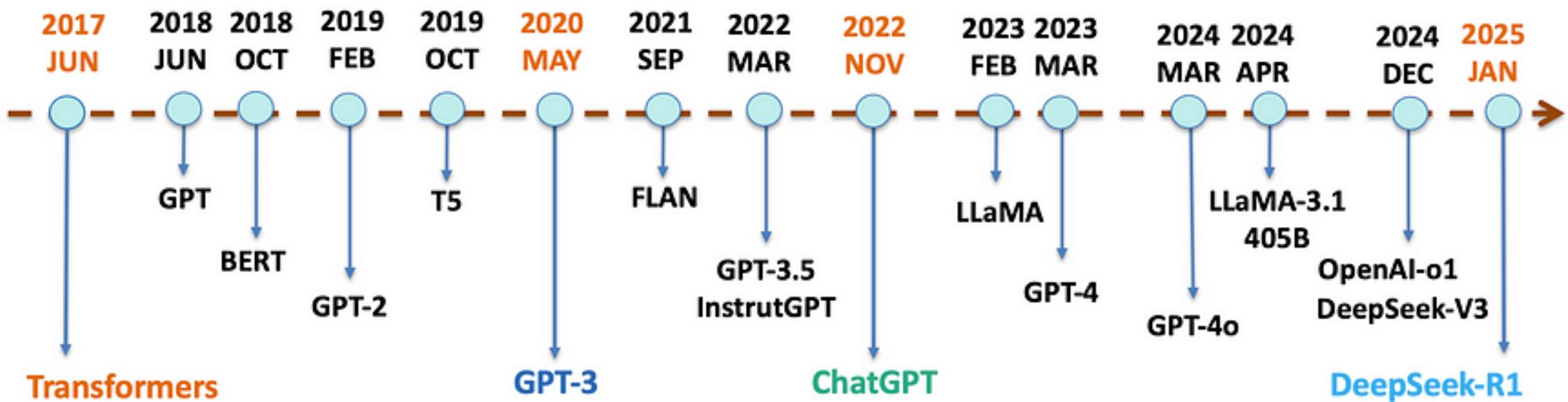
Policy 方面

- Aligning LLMs during pretraining
- Aligning LLMs through supervised learning

Reinforcement Learning for Large Reasoning Models

LLM → LRM

A Brief History of LLMs



Large Reasoning Model

OpenAI o1/o3, DeepSeek R1, Kimi 1.5, ...

推理大模型：

- 具备强推理能力的大规模人工智能模型，能够在复杂任务中进行多步逻辑推理、因果推理、归纳推理等，以达到更接近人类思维的智能水平。

典型能力包括：

1. 多步逻辑推理（Multi-step Reasoning）

能够从多个前提中逐步得出结论，如数学题、多轮问答等。

2. 因果推理（Causal Reasoning）

推断“原因→结果”或反向，“结果←原因”。

3. 策略规划（Planning）

用于解决涉及顺序、目标、状态转换的问题（如机器人路径规划、游戏策略生成）。

4. 结构化输出和模块组合

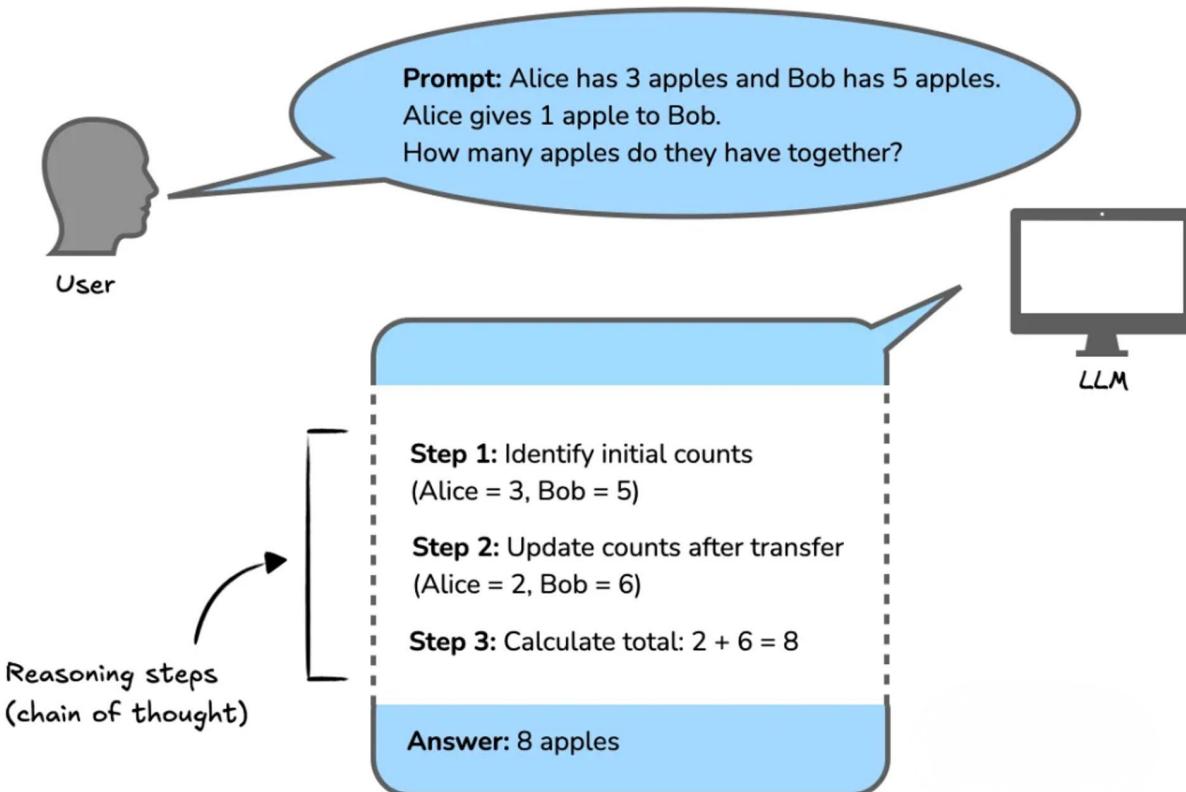
能与搜索、数据库、计算器、规划器等外部工具结合，形成“Tool-augmented Reasoning”。

Large Reasoning Model

OpenAI o1/o3, DeepSeek R1, Kimi 1.5, ...

推理大模型：

- 具备强推理能力的大规模人工智能模型，能够在复杂任务中进行多步逻辑推理、因果推理、归纳推理等，以达到更接近人类思维的智能水平。



LLM 如何处理多步骤推理任务的简单图例。

模型并非仅仅回忆一个事实，而是需要结合多个中间推理步骤才能得出正确的结论。

根据具体实现方式，中间推理步骤可能会显示给用户，也可能不会显示。

Large Reasoning Model

- O1解决 cipher密码问题

问题

User

oyfjdnisdr rtqwainr acxz mynzbhhx -> Think step by step

Use the example above to decode:

oyekaijzdf aaptcg suaokybhai ouow aqht mynznaatzacdfoulxxz

给定示例，对密码求解内容

- 自发产生step-by-step分解
- 自我评估与反思
- 识别错误与回溯

Introducing OpenAI o1

We've developed a new series of AI models designed to spend more time thinking before they respond. Here is the latest news on o1 research, product and other updates.

Try it in ChatGPT Plus ↗

Try it in the API ↗

[OpenAI o1 Hub](#) | [OpenAI](#)

Thought for 5 seconds ^

First, what is going on here?

We are given:

First, an example:

"oyfjdnisdr rtqwainr acxz mynzbhhx -> Think step by step"

We can see that when "oyfjdnisdr rtqwainr acxz mynzbhhx" is transformed to "Think step by step"

Our task is: Use the example above to decode:
"oyekaijzdf aaptcg suaokybhai ouow aqht mynznaatzacdfoulxxz"

So the first part is to figure out how "oyfjdnisdr rtqwainr acxz mynzbhhx" is decoded into "Think step by step"

Observation 1: The phrase "Think step by step" is 18 letters (excluding spaces). Let's count the letters.

Similarly, let's count the letters in "oyfjdnisdr rtqwainr acxz mynzbhhx"

Original Ciphertext:

oyfjdnisdr rtqwainr acxz mynzbhhx

Plaintext:

Think step by step

First, break down the letters:

Ciphertext letters: oyfjdnisdr rtqwainr acxz mynzbhhx

Total letters: Let's count the letters in the ciphertext.

So is the decoding process that the ciphertext maps to "Think step by step"?

Our goal is to use the example above to decode:
"oyekaijzdf aaptcg suaokybhai ouow aqht mynznaatzacdfoulxxz"

First, the cipher seems connected to the plaintext.

Given the time constraints, perhaps the easiest way is to try to see patterns.

Option 1: Try to find mapping from letters to letters.

Do any letters match?

First, let's write down the ciphertext and plaintext letters on top of each other.

Ciphertext: oy f j d n i s d r t q w a i n r a c x z m y

Interesting.

It seems that the ciphertext words are exactly twice as long as the plaintext words.

(10 vs 5, 8 vs 4, 4 vs 2, 8 vs 4)

Idea: Maybe we need to take every other letter or rebuild the plaintext from the ciphertext accordingly.

Let's test this theory.

If we take every other letter from the ciphertext words, maybe we'll get the plaintext.

Let's try this for the first word:

Ciphertext word: oy f j d n i s d r (10 letters)

Alternate letters:

o d i d --> Think

Not directly

Option 2: Try mapping as per an assigned code: perhaps columns of letters?

Alternatively, perhaps the cipher is more complex.

Alternatively, notice that "oyfjdnisdr" has 10 letters and "Think" has 5 letters.

What if each plaintext letter is being represented by two ciphertext letters?

Similarly for other words.

Suppose that each plaintext letter corresponds to pair of letters in the ciphertext.

Meaning that the first two letters of ciphertext map to the first plaintext letter, and so on.

If this is the mapping, perhaps each pair of letters maps to one plaintext letter.

Similarly for other words.

Let's test this mapping.

Is there a consistent mapping from pairs to letters?

Let's make a table of pairs and corresponding letters.

First, make the pairs from the first word and their corresponding plaintext letters:

Ciphertext Pairs -> Plaintext Letter

oy -> T

fj -> h

Large Reasoning Model

- O1科学推理能力：求解化学问题

问题



What is the pH of a 0.10 M solution of NH_4F ?
The K_a of NH_4^+ is 5.6×10^{-10} and the K_a of HF is 6.8×10^{-4} .

计算给定参数下的pH值

- 自我评估
- 多条路径
- 自我纠正

Alternatively, note that for salts of weak acids and weak bases, the pH depends on the K_a and K_b values.

One method is to use the formula:

$$\text{pH} = 7 + 0.5(pK_a - pK_b)$$

But this formula works only when concentrations are equal.

Wait, perhaps more accurate to find K_b for F^- and compare it to K_a for NH_4^+ .

Let's calculate K_b for F^- .

Given $K_a(\text{HF}) = 6.8 \times 10^{-4}$

$$\text{So } K_b(\text{F}^-) = \frac{K_w}{K_a(\text{HF})}$$

Similarly, for NH_4^+ , we can write

$$K_b(\text{NH}_3) = \frac{K_w}{K_a(\text{NH}_4^+)}$$

$$K_b(\text{NH}_3) = \frac{K_w}{K_a(\text{NH}_4^+)}$$

But maybe not necessary.

Alternatively, for a salt of a weak acid and a weak base, the pH is given by:

$$\text{pH} = 7 + 0.5 \times \log\left(\frac{K_b}{K_a}\right)$$

But this formula assumes equimolar concentrations.

Wait, but in our case, the weak acid and weak base have the same concentration, because NH_4F dissociates into equal amounts of NH_4^+ and F^- .

So let's compute:

$$\text{pH} = 7 + 0.5 \times \log\left(\frac{K_b}{K_a}\right)$$

[Try it in ChatGPT Plus ↗](#) [Try it in the API ↗](#)

[OpenAI o1 Hub](#) | [OpenAI](#)

Introducing OpenAI o1

We've developed a new series of AI models designed to spend more time thinking before they respond. Here is the latest news on our research, product and other updates.

Large Reasoning Model

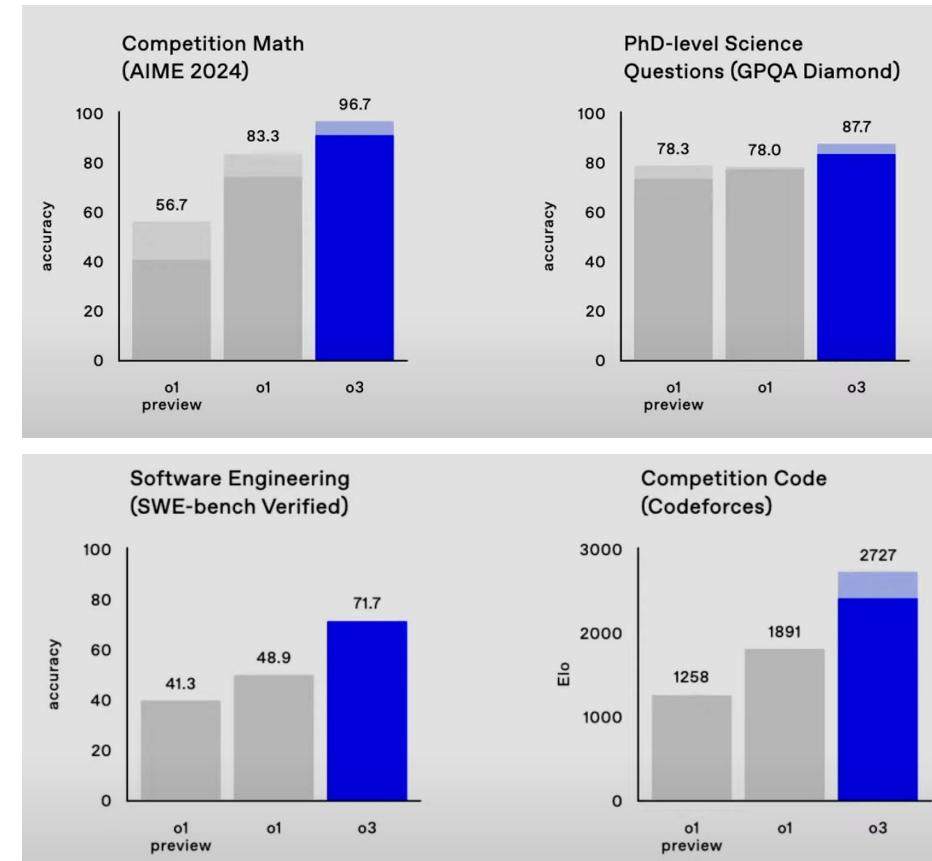
- o3 系列模型是 o1 的迭代版本。相比o1， o3 模型的纸面参数迎来了全方位提升。
- 其主要包含两款模型OpenAI o3：旗舰版本，具备强大的性能表现；以及OpenAI o3 mini：轻量级模型，但能更快，更便宜，主打性价比。



OpenAI o3 and o3-mini—12 Days of OpenAI: Day 12

OpenAI
1.54M subscribers

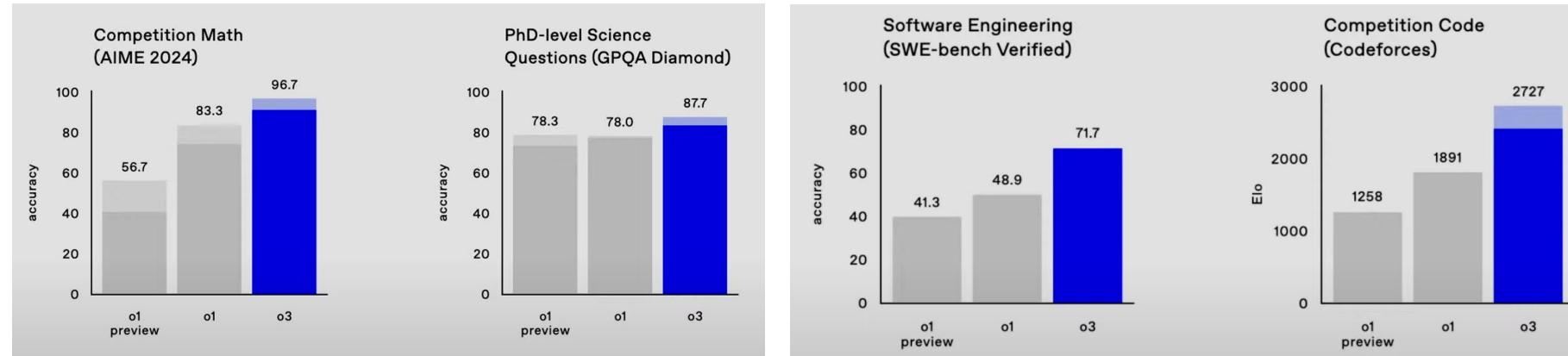
Subscribe



OpenAI o3 and o3-mini—12 Days of OpenAI: Day 12

Large Reasoning Model

- o3 系列模型是 o1 的迭代版本。相比 o1，o3 模型的纸面参数迎来了全方位提升。
- 其主要包含两款模型 OpenAI o3：旗舰版本，具备强大的性能表现；以及 OpenAI o3 mini：轻量级模型，但能更快，更便宜，主打性价比。



- 在数学、编码、博士级科学问答等复杂问题上，o3 表现出了超越部分人类专家的水平。
 - 比如，在涉及生物学、物理学和化学的博士级科学考试 GPQA Diamond 中，o3 的准确率达到 87.7%，o1 仅为 78%，而这些领域的博士专家只能达到 70%；
 - 在美国 AIME 数学竞赛中，o3 取得 96.7 分、只错了一道题，相当于顶级数学家的水平。

Large Reasoning Model

2025年1月20日，DeepSeek 正式发布了一系列DeepSeek-R1 模型，并上传R1 系列的技术报告和各种信息。DeepSeek 此次共发布三组模型：

- DeepSeek-R1-Zero：大规模使用强化学习（RL）技术，没有任何监督微调（SFT）；
- DeepSeek-R1：在强化学习前融入冷启动数据，多阶段训练；
- DeepSeek-R1-Distill 系列：DeepSeek-R1 中蒸馏推理能力到小型密集模型，参数规模分别为 1.5B、7B、8B、14B、32B 和70B。



DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

DeepSeek-AI

research@deepseek.com

Large Reasoning Model

DeepSeek-R1 模型推理能力优异，比肩OpenAI o1 正式版。

- DeepSeek-R1 在AIME2024 上获得了79.8%的成绩，略高于OpenAI-o1-1217。
- 在MATH-500 上，它获得了97.3%的惊人成绩，表现与OpenAI-o1-1217 相当，并明显优于其他模型。
- 在编码相关的任务中，DeepSeek-R1 在代码竞赛任务中表现出专家水平，在Codeforces上获得了2029Elo 评级，在竞赛中表现优于96.3%的人类参与者。
- 对于工程相关的任务，DeepSeek-R1 的表现略优于OpenAI-o1-1217。

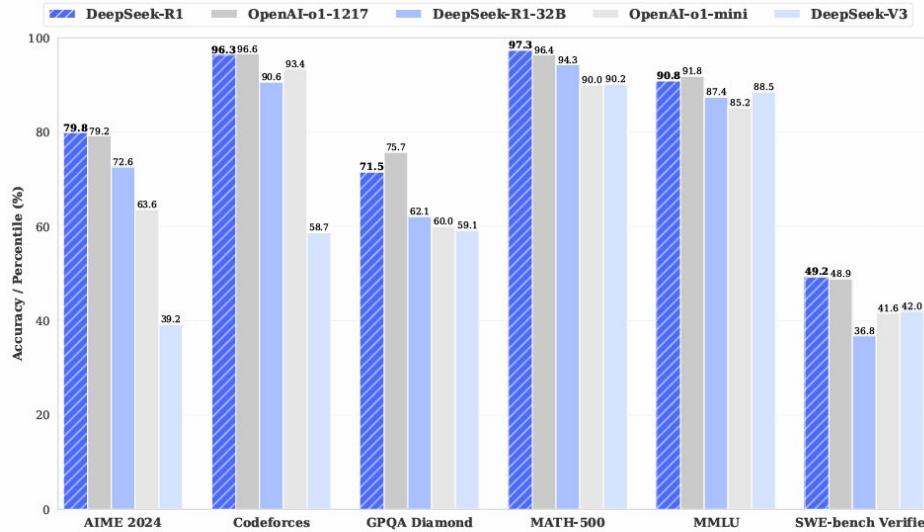


Figure 1 | Benchmark performance of DeepSeek-R1.

Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a - \sqrt{a+x}} = x$ is equal to

Response: <think>

To solve the equation $\sqrt{a - \sqrt{a+x}} = x$, let's start by squaring both ...

$$(\sqrt{a - \sqrt{a+x}})^2 = x^2 \implies a - \sqrt{a+x} = x^2.$$

Rearrange to isolate the inner square root term:

$$(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$$

...

Wait, wait. Wait. That's an aha moment I can flag here.

Let's reevaluate this step-by-step to identify if the correct sum can be ...

We started with the equation:

$$\sqrt{a - \sqrt{a+x}} = x$$

First, let's square both sides:

$$a - \sqrt{a+x} = x^2 \implies \sqrt{a+x} = a - x^2$$

Next, I could square both sides again, treating the equation: ...

Table 3 | An interesting “aha moment” of an intermediate version of DeepSeek-R1-Zero. The model learns to rethink using an anthropomorphic tone. This is also an aha moment for us,

allowing us to witness the power and beauty of reinforcement learning.

Chain-of-Thought (CoT)

Language Models are Few-Shot Learners

- 大模型具有 in-context learning的能力
- 在few-shot example下即可提升模型输出的准确性

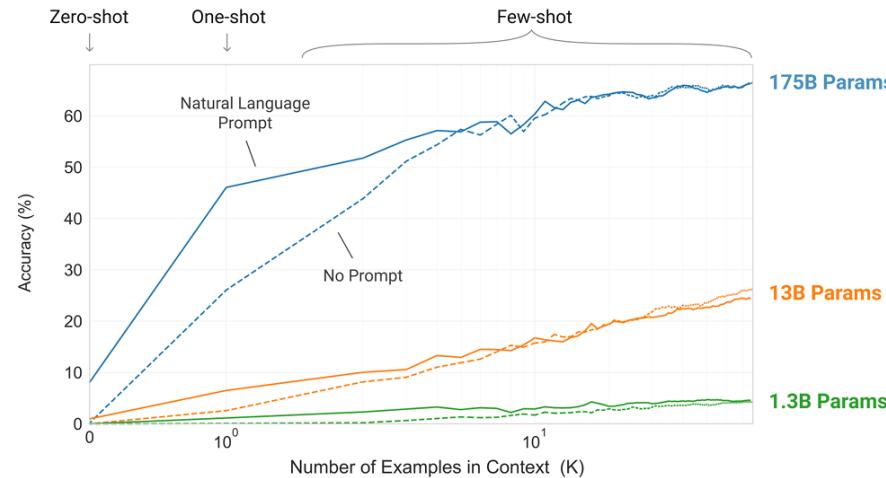


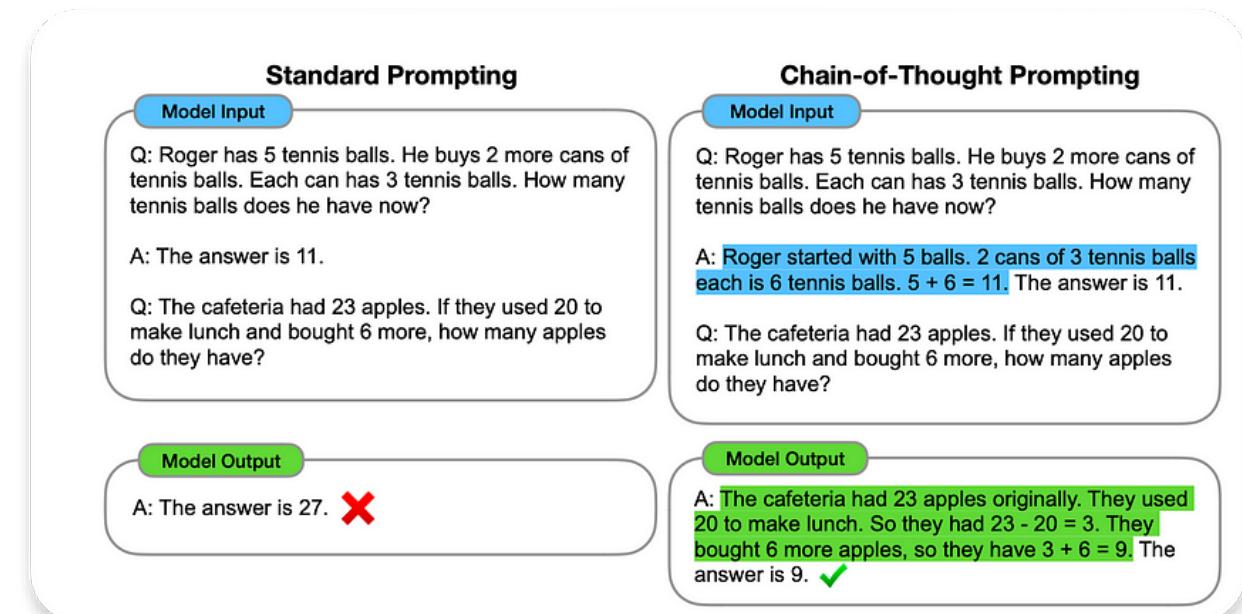
Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper “in-context learning curves” for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*
Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry
Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan
Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter
Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray
Benjamin Chess Jack Clark Christopher Berner
Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

OpenAI

[cs.CL] 22 Jul 2020



Chain-of-Thought (CoT)

Large Language Models are Zero-Shot Reasoners

- *Let's think step by step*

Large Language Models are Zero-Shot Reasoners

Takeshi Kojima
The University of Tokyo
t.kojima@weblab.t.u-tokyo.ac.jp

Shixiang Shane Gu
Google Research, Brain Team

Machel Reid
Google Research*

Yutaka Matsuo
The University of Tokyo

Yusuke Iwasawa
The University of Tokyo

Abstract

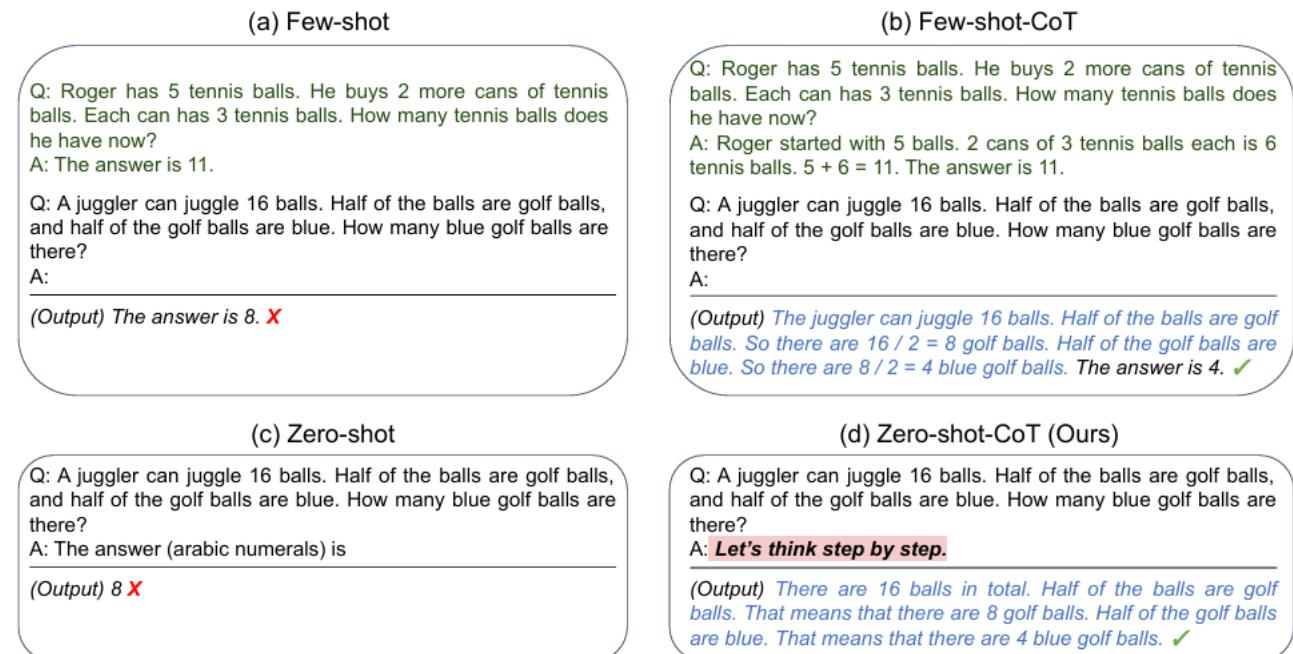
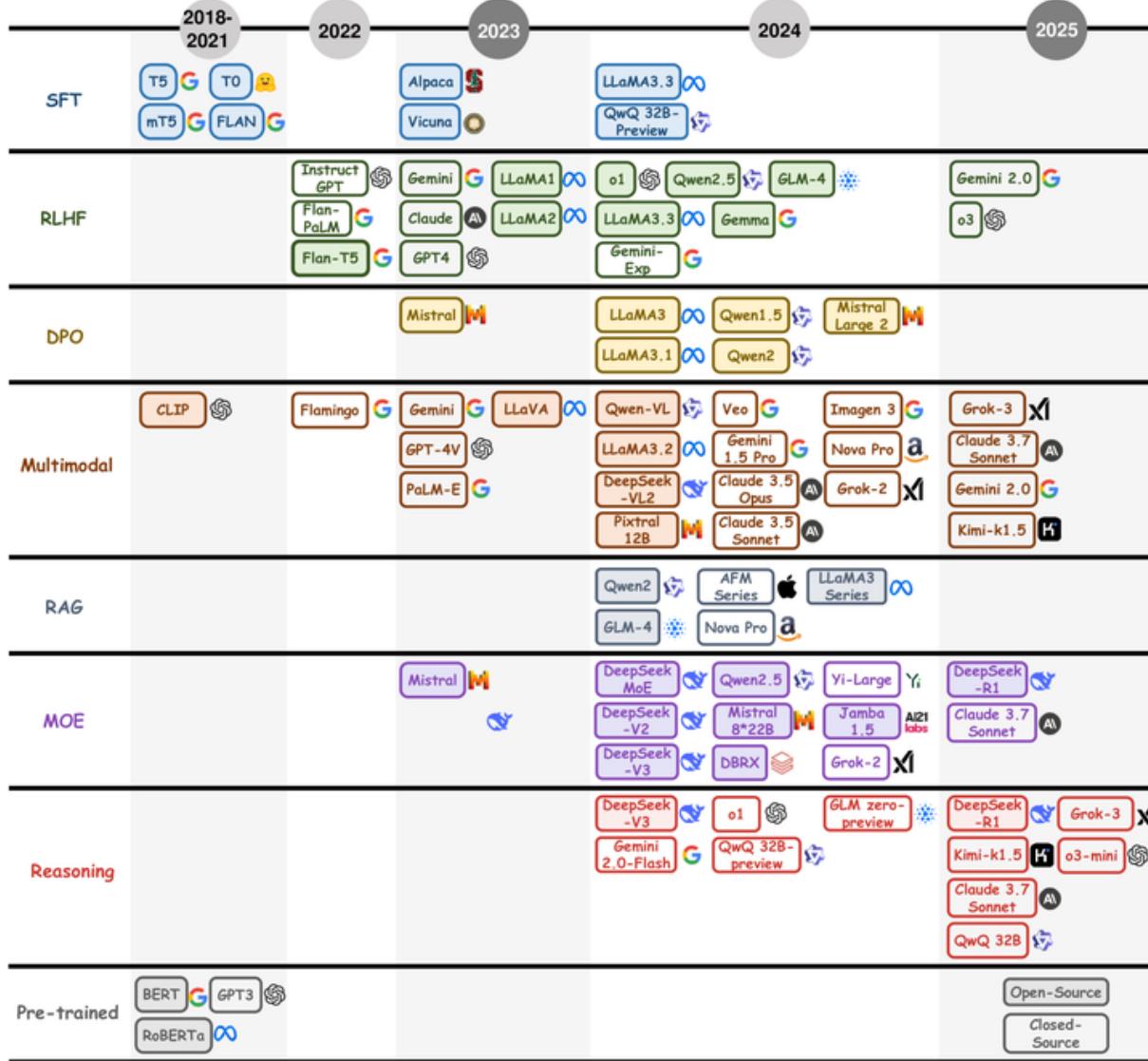


Figure 1: Example inputs and outputs of GPT-3 with (a) standard Few-shot ([Brown et al., 2020]), (b) Few-shot-CoT ([Wei et al., 2022]), (c) standard Zero-shot, and (d) ours (Zero-shot-CoT). Similar to Few-shot-CoT, Zero-shot-CoT facilitates multi-step reasoning (blue text) and reach correct answer where standard prompting fails. Unlike Few-shot-CoT using step-by-step reasoning examples **per task**, ours does not need any examples and just uses the same prompt “Let’s think step by step” *across all tasks* (arithmetic, symbolic, commonsense, and other logical reasoning tasks).

LLM Post-Training



基础模型训练的两个阶段：

• 预训练阶段：

通过大规模数据的自监督训练使模型具备通用的基础能力。在实践中，预训练一般细分为多个阶段，不同阶段的数据质量、配比方式、上下文长度或图像分辨率不一样，通常越后面的阶段的数据质量越高。

• 后训练阶段：

使用少量高质量数据、监督微调（SFT）、强化学习（RL）等技术来充分激发预训练模型的能力。



LLM Post-Training

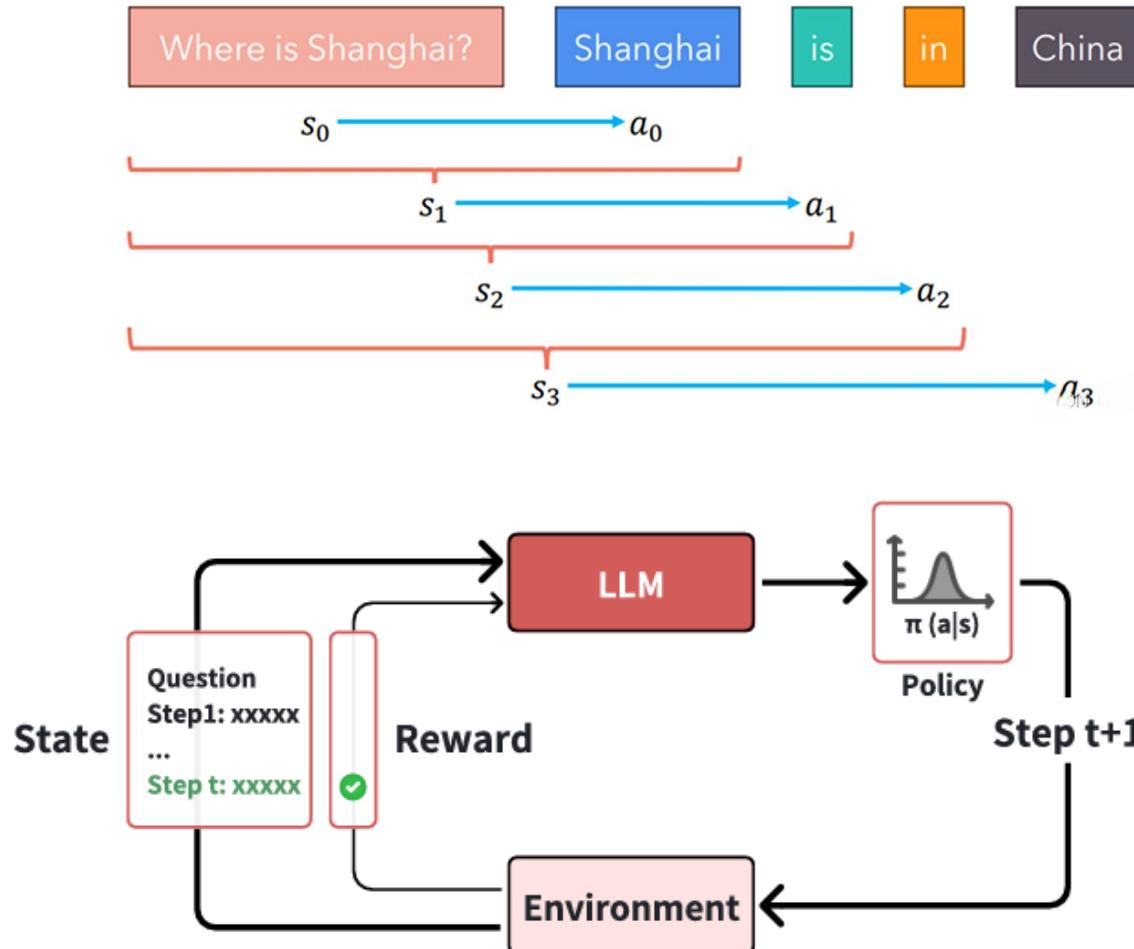
- 预训练数据变得日益稀缺，且能力提升的速度与计算投入相比逐渐放缓。
- 预训练的 LLM 通常依赖于从示例中反向推导出理想行为，导致数据利用效率低下和决策边界不清晰，从而导致推理和安全能力不尽如人意。
- 因此，后训练逐渐成为提升模型性能、提高推理准确性、对齐人类价值观，并适应用户偏好的关键方法，且相比于预训练，其计算需求显著降低。

LLM Post-Training



- + SFT方法较为简单直接，训练稳定且收敛速度快
 - 但只有正反馈（高质量数据），上限较低。
 - + RL相比SFT多了探索的阶段，且具有负反馈，能让模型明白什么样的输出是不好的
 - 但是其实现较为麻烦，训练容易不稳定
- 较为流行的范式是先做SFT将模型输出快速收敛到一个较好的分布附近，然后再使用RL进行探索和优化
 - 但也出现一些直接在预训练模型之后进行RL的实践，如DeepSeek-R1-Zero和Open-Reasoner-Zero等。

LLM Generation MDP



状态(State):

- 初始状态即由用户输入 prompt (question)
- 每一时刻状态由 prompt + existing response

动作(Action):

- 模型词汇表中的所有token组成，即大小为 vocab_size

状态转移(Transition):

$$s_{t+1} = (s_t, a_t)$$

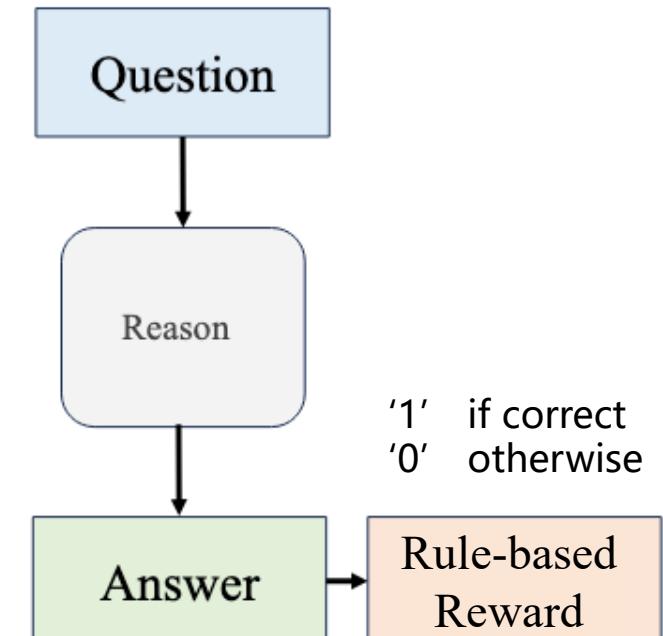
策略(Policy):

- 策略即LLM本身: $a_t \sim \pi_\theta(s_t)$

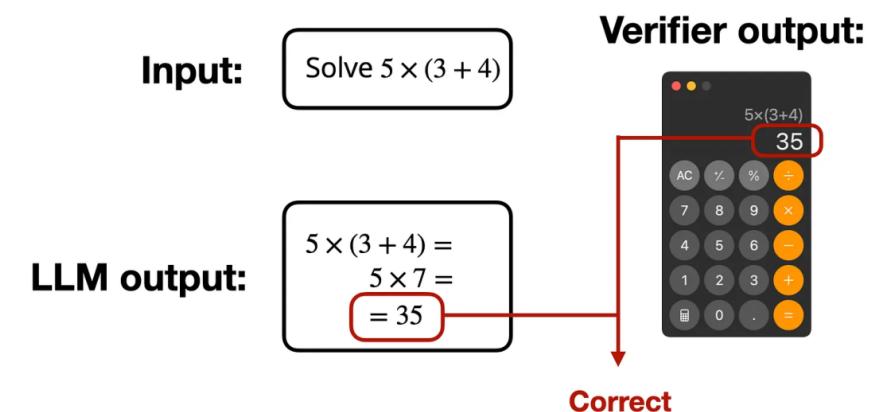
奖励 (Reward)

可验证/规则的奖励 Verified reward/rule-based reward

- 通过程序、工具或逻辑规则自动生成的可靠奖励信号
- e.g.
 - 代码生成任务验证代码是否能编译、是否能通过测试用例；
 - 数学推理任务验证答案是否等于公式计算结果；
 - 对话任务通过外部工具（如知识库、检索系统）验证答案是否与事实一致
- 代表：DeepSeek-R1 Zero
- + 准确，自动化
- 只能用于可验证问题



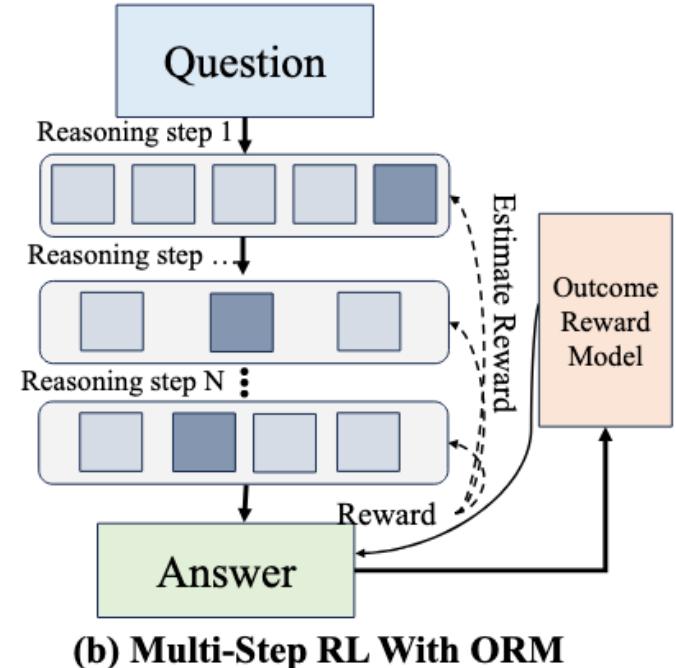
(a) Direct RL



奖励 (Reward)

结果奖励模型 Outcome reward model

- 根据任务的最终结果（而非过程）对模型输出进行评估和打分
- 代表：OpenAI Q*、GPT-4
 - 本身是LLM，需要专门训练
 - 无法区分中间推理步骤的正确性和重要性
- + 可泛化，训练集标注成本低，易于实现



(b) Multi-Step RL With ORM

ORM

Question:
Three pencils and a jumbo eraser cost \$1.24. Five pencils and a jumbo eraser cost \$1.82. No prices include tax. In cents, what is the cost of a pencil?

Answer:

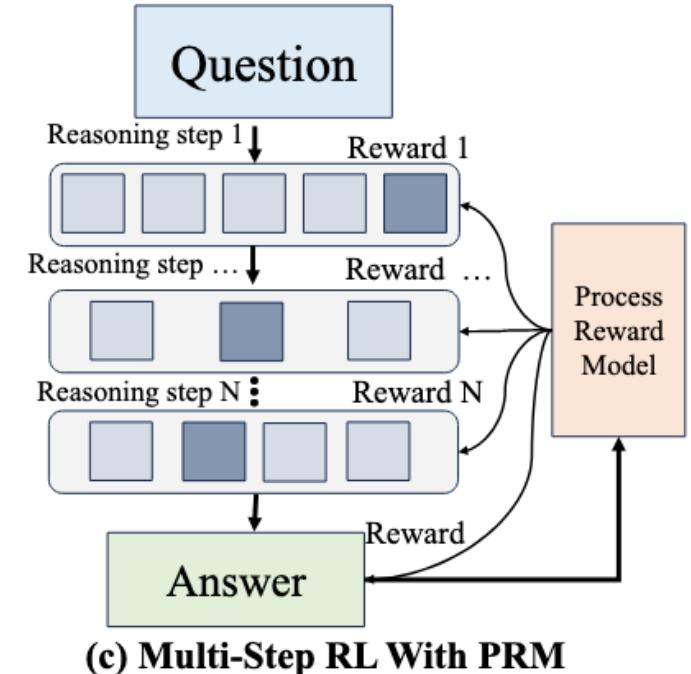
Let the cost of a pencil be p cents, and the cost of a jumbo eraser be e cents.
From the problem, we are given two equations based on the cost:
 $3p + 2 = 124$
 $5p + e = 182$
To solve this system of equations, we subtract the first equation from the second to eliminate e :
 $(5p + e) - (3p + e) = 182 - 124$
Solving for $p = 58/2 = 27$
Thus, the cost of a pencil is 29 cents.

✖

奖励 (Reward)

过程奖励模型 Process reward model

- 评估模型推理过程、思维链条、行为序列等中间步骤质量的奖励，引导模型不仅输出“正确结果”，还遵循清晰、有逻辑、可解释的推理路径。
- 代表：OpenAI [\[2305.20050\] Let's Verify Step by Step](#)
 - + 关注模型推理过程的正确性，避免结果瞎猜对但过程混乱
 - + 步骤级的密集奖励，训练稳定性高
 - 实现复杂，标注成本大



(c) Multi-Step RL With PRM

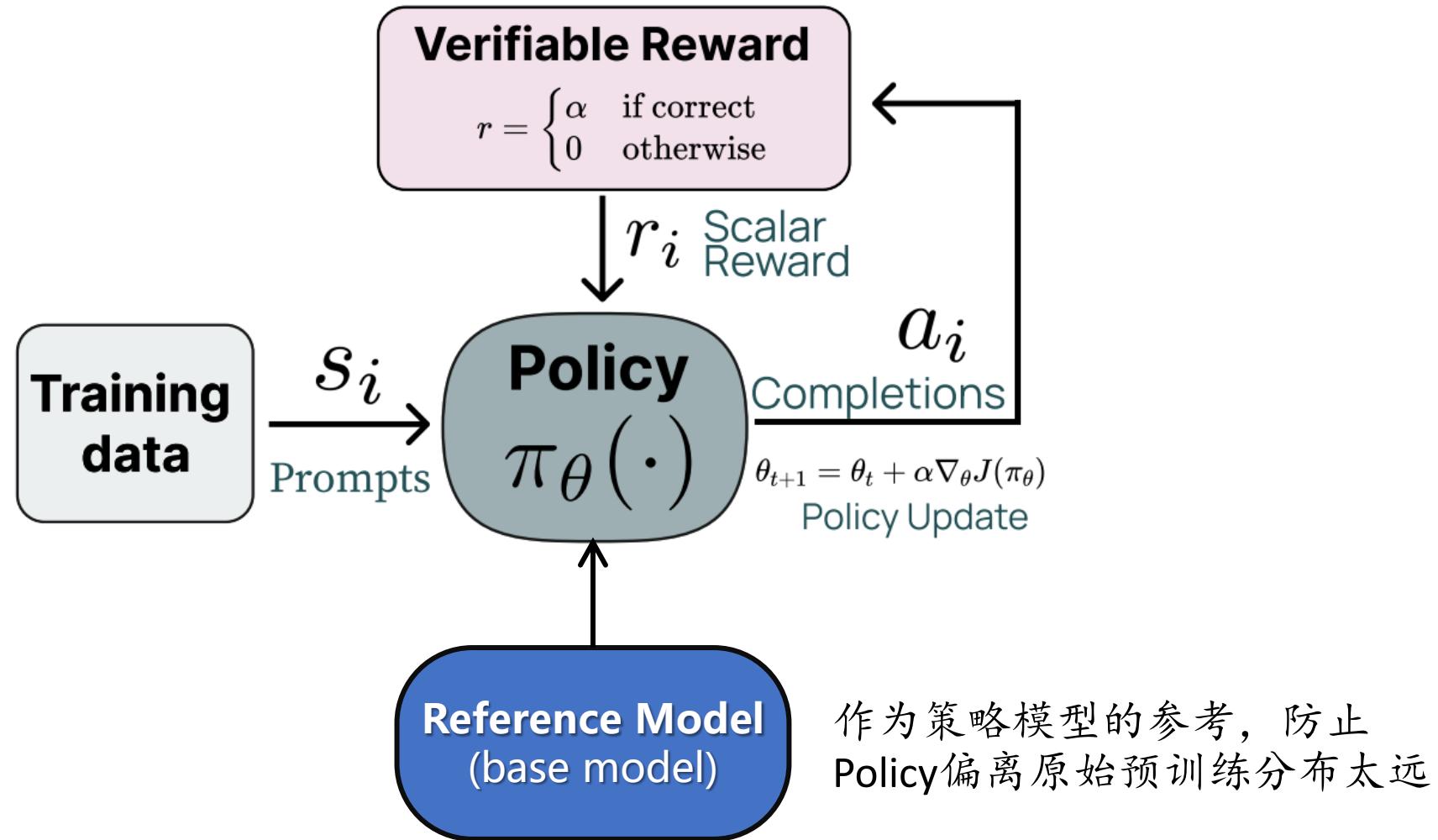
PRM

Question:
Three pencils and a jumbo eraser cost \$1.24. Five pencils and a jumbo eraser cost \$1.82. No prices include tax. In cents, what is the cost of a pencil?

Answer:

- 1 Let the cost of a pencil be p cents, and the cost of a jumbo eraser be e cents. ✓
- 2 From the problem, we are given two equations based on the cost: $3p + 2 = 124$, $5p + e = 182$. ✓
- 3 To solve this system of equations, we subtract the first equation from the second to eliminate e : $(5p + e) - (3p + e) = 182 - 124$ ✓
- 4 Solving for $p = 58/2 = 27$ ✗
- 5 Thus, the cost of a pencil is 29 cents. ✗

LLM RL post-training



REINFORCE

REINFORCE Loss function

$$\mathcal{L}(\theta) = - R_\tau \sum_{t=0}^T \log \pi_\theta(a_t | s_t)$$

Diagram annotations:

- An orange arrow labeled "Episode return" points from the term R_τ to the summation term.
- A pink box surrounds the summation term $\sum_{t=0}^T \log \pi_\theta(a_t | s_t)$.
- A red arrow labeled "Sum of action log probabilities" points from the bottom of this pink box up to the summation term.

- 将LLM完整一条的回复 response/completion作为single action
- 使用 REINFORCE 估计器来优化整个序列的奖励目标：

$$\mathbb{E}_{x \sim D, y \sim \pi_\theta(\cdot | x)} [R(y, x) \nabla_\theta \log \pi_\theta(y | x)]$$

- 为了在保持无偏估计的同时减少方差，引入了一个基线

$$\mathbb{E}_{x \sim D, y \sim \pi_\theta(\cdot | x)} [(R(y, x) - b) \nabla_\theta \log \pi_\theta(y | x)]$$

- 一个简单有效的基线是训练过程中所有奖励的移动平均值：

$$b_{MA} = \frac{1}{S} \sum_{s=1}^S R(x_s, y_s)$$

Diagram annotation:

- A blue box contains the text "沿着训练过程的移动平均".
- A blue arrow points from this box to the term b_{MA} .

其中s是训练步数， 是第s步的 prompt-completion 对

REINFORCE Leave-One-Out (RLOO)

REINFORCE Leave-One-Out (RLOO) 估计器利用多个样本来减少方差

- 每个样本的奖励可以作为其他样本的基线；
- 策略更新通过对每个样本的梯度估计取平均来实现，从而得到一个方差减少的蒙特卡洛 (MC) 估计。

RLOO 估计器的定义如下：

$$\frac{1}{k} \sum_{i=1}^k \left[R(y^{(i)}, x) - \frac{1}{k-1} \sum_{j \neq i} R(y^{(j)}, x) \right] \nabla \log \pi(y^{(i)} | x)$$

➤ 其中 $y^{(1)}, \dots, y^{(k)}$ 是从 $\pi_\theta(\cdot | x)$ 中生成的 k 个独立样本。

- + RLOO 为每个样本创建了一个即时的、无参数的基线，比移动平均 b_{MA} 更有效
- 但需要额外的采样时间

REINFORCE Leave-One-Out (RLOO)

| RLOO | response tokens | a | b | c | d | e | EOS | response level value |
|-------------|-----------------|------|------|------|---|-----|-------|----------------------|
| kl | | 0.01 | 0.02 | 0.03 | 0.1 | 0.2 | 0.3 | 0.66 |
| score (rm) | | | | | | | 0.6 | 0.6 |
| kl_coef=0.1 | reward | | | | | | | 0.534 |
| baseline | | | | | mean($r_0, \dots, r_{i-1}, r_{i+1}, \dots$) | | | 0.3 |
| advantage | | | | | | | 0.234 | 0.234 |

PPO (recall)

- PPO 是在 TRPO 的基础上不直接约束 KL 距离，而是引入“剪切”损失函数，限制策略更新的幅度。
 - “裁剪机制”是 PPO 的精髓，使得：策略不更新太远（避免崩溃）又不需要计算复杂的二阶导数或构造优化约束
- PPO loss

$$L^{CLIP}(\theta) = \mathbb{E}_t [\min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t)]$$

➤ 其中 $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}$ 是策略比值；

➤ clip 操作限制策略变化；

➤ A_t 是优势函数。

PPO in LLM

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

- q 代表问题 prompt, Q 代表问题数据集
- o 代表 response, 由token组成, $o = (o_1, o_2, \dots, o_t, \dots)$
- $\pi_\theta(o_t|q, o_{<t})$ 代表在当前prompt和已经生成的 $o_{<t}$ 下, next-token o_t 的生成概率
- $\pi_{\theta_{old}}(o_t|q, o_{<t})$ 为上次优化结束的LLM的生成概率

PPO in LLM

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

- 优势函数 A_t 衡量了在当前状态下选择某个动作的优越性，相对于其他动作的预期回报的提升
- 实际中，优势函数通常是通过累计折扣奖励减去基准来实现的

$$A_t = \sum_{s=t}^T \gamma^{s-t} r(y_s) - b$$

环境在时间步 s 提供的奖励

折扣因子

基准 (baseline)，用来减小优势函数的方差，常见的选择包括直接使用值函数或奖励的群体平均

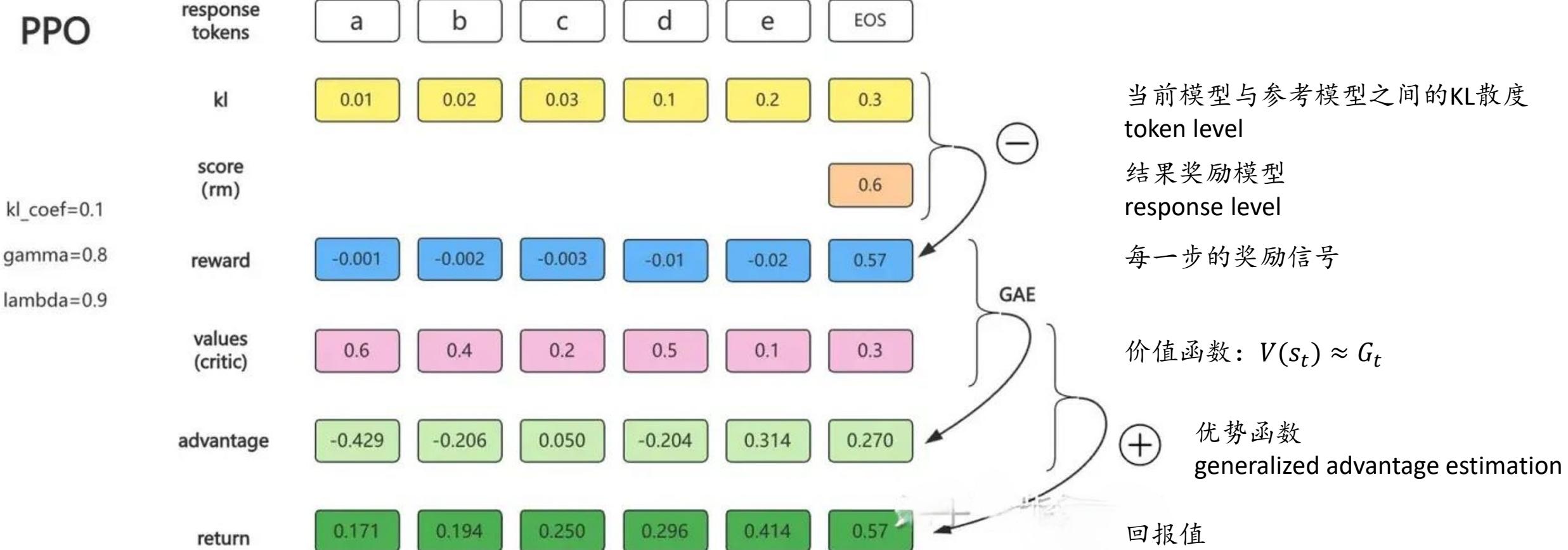
PPO in LLM

- LLM PPO的奖励会定义为

$$r_t = r_\varphi(q, o_{\leq t}) - \beta \log \frac{\pi_\theta(o_t | q, o_{<t})}{\pi_{ref}(o_t | q, o_{<t})},$$

- π_θ 为当前待优化的LLM, π_{ref} 为reference model, 为初始的base LLM
- 在reward signal上添加reference model的KL散度惩罚项, 约束大模型的生成概率分布不要与base model 偏离太远, 产生“胡言乱语”

PPO in LLM



PPO in LLM

- 在传统RL control问题中，每一时刻都有 reward 信号，且同一轨迹不同时刻的 reward 差异会很大，累积奖励具有高方差，导致收敛速度慢
- 在PPO中构造 value function，以此计算稳定的状态价值估计
- 在GAE中与 $\gamma(0.9\sim0.99)$, $\lambda(\sim0.95)$ 下，计算的 advantage 会认为当前的奖励更加重要，未来的累积需要打个折扣
- 这样算法出的策略梯度 $\pi_\theta(a_t|s_t)A(s_t, a_t)$ 起到强化好的行为，弱化坏的动作的作用
- 但是...
- 在LLM问题中，通常使用ORM/rule-based RM，只提供 response-level reward，没有 token-level reward
- 在长文本+衰减因子 $\gamma < 1$ 作用下，靠前 token 的回报值基本为零
- $G_t - V_t$ 或 GAE 得到的 advantage 会很不准确，策略梯度 $\pi_\theta(a_t|s_t)A(s_t, a_t)$ 无法 提供真实可靠的训练信号

REINFORCE++

改进一：Token-level KL惩罚

- 在 REINFORCE++ 中，Token-Level KL 惩罚被整合到奖励函数中

$$r(s_t, a_t) = I(s_t = [EOS])r(x, y) - \beta \text{KL}(t)$$

- $I(s_t = [EOS])$ 是指示函数，当 token s_t 是序列结束符（End of Sequence）时取值为 1，否则为 0
- $r(x, y)$ 是模型生成的完整序列 y 对应的奖励
- β 是一个超参数，用于控制 KL 惩罚的强度。
- KL(t) 是 token 的 KL 散度，计算公式为

$$\text{KL}(t) = \log \left(\frac{\pi_{RL}^{\theta_{old}}(a_t | s_t)}{\pi_{SFT}(a_t | s_t)} \right)$$

- $\pi_{RL}^{\theta_{old}}(a_t | s_t)$ 是 RL 模型在状态 s_t 下生成 token a_t 的概率
- $\pi_{SFT}(a_t | s_t)$ 是 SFT 模型在状态 s_t 下生成 token a_t 的概率

REINFORCE++

改进二：Advantage Normalization 优势标准化

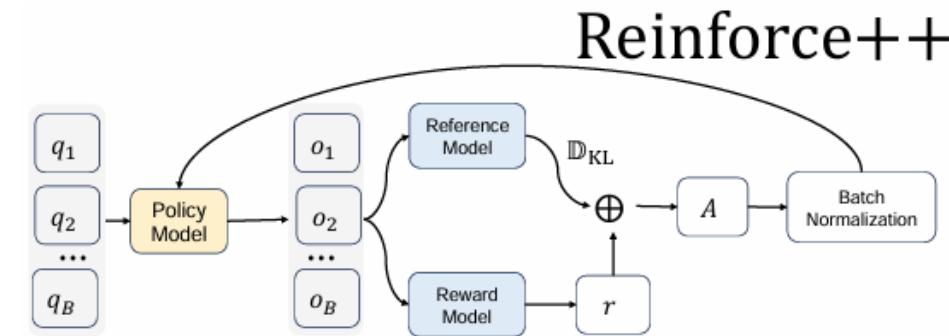
- 优势值直接选取从当前 t-step 到EOS序列结束时带惩罚奖励的累加

$$A_t(s_t, a_t) = r(x, y) - \beta \cdot \sum_{i=t}^T \text{KL}(i)$$

(将Q值作为advantage使用)

- 为了确保梯度的稳定性并避免训练中的发散，论文对优势进行 z-score 标准化

$$A_{\text{normalized}} = \frac{A - \mu_A}{\sigma_A}$$



➤ μ_A 和 σ_A 分别是 batch 中的平均值和标准差。优势标准化有助于平衡不同训练步骤中的梯度幅度，确保训练过程的稳定性，并防止出现梯度爆炸或消失的问题。

GRPO

- GRPO (Group Relative Policy Optimization) 在DeepSeekMath中首次提出，在DeepSeek-V2、DeepSeek-V3、DeepSeek-R1中都得到了应用。

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left\{ \min \left[\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})} \hat{A}_{i,t}, \text{clip} \left(\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}, 1 - \varepsilon, 1 + \varepsilon \right) \hat{A}_{i,t} \right] - \beta \mathbb{D}_{KL} [\pi_\theta || \pi_{ref}] \right\}$$

• 与PPO相比，改进一：去除价值函数

- GRPO 不需要额外的价值函数模型，而是使用多组采样输出的平均奖励作为基线 (baseline)
- 原因：
 - (1) Value model一般与policy model同样大小，且需要训练，计算资源消耗较大；
 - (2) Value model需要在每一个token处预测期望回报，但是LLM中的奖励模型一般只在最后给一个最终的奖励值，这会干扰value model的训练，使得难以训出一个较好的 value model，不如去掉。

GRPO

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left\{ \min \left[\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})} \hat{A}_{i,t}, \text{clip} \left(\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}, 1 - \varepsilon, 1 + \varepsilon \right) \hat{A}_{i,t} \right] - \beta \mathbb{D}_{KL} [\pi_\theta || \pi_{ref}] \right\}$$

- 与PPO相比，改进二：组内相对奖励

- 对于每个问题 q , GRPO 从旧策略 $\pi_{\theta_{old}}$ 中采样一组输出 o_1, o_2, \dots, o_G , 并利用这些输出的相对奖励计算优势函数 $\hat{A}_{i,t}$ 。
- 所有 token 的优势值 $\hat{A}_{i,t}$ 都等于归一化后的奖励：

$$\hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - \text{mean}(r)}{\text{std}(r)}$$

GRPO

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left\{ \min \left[\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})} \hat{A}_{i,t}, \text{clip} \left(\frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}, 1 - \varepsilon, 1 + \varepsilon \right) \hat{A}_{i,t} \right] - \beta \mathbb{D}_{KL} [\pi_\theta || \pi_{ref}] \right\}$$

- 与PPO相比，改进三：KL 正则化

- 相比于 REINFORCE++，GRPO 在损失函数中直接加入 KL 散度正则项，而不是将 KL 惩罚整合到奖励中，从而简化了优势函数的计算。
- GRPO 采用了近似的 KL 散度，进一步提高了计算的稳定性和效率

$$\mathbb{D}_{KL} [\pi_\theta || \pi_{ref}] = \frac{\pi_{ref}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})} - \log \frac{\pi_{ref}(o_{i,t}|q, o_{i,<t})}{\pi_\theta(o_{i,t}|q, o_{i,<t})} - 1,$$

GRPO

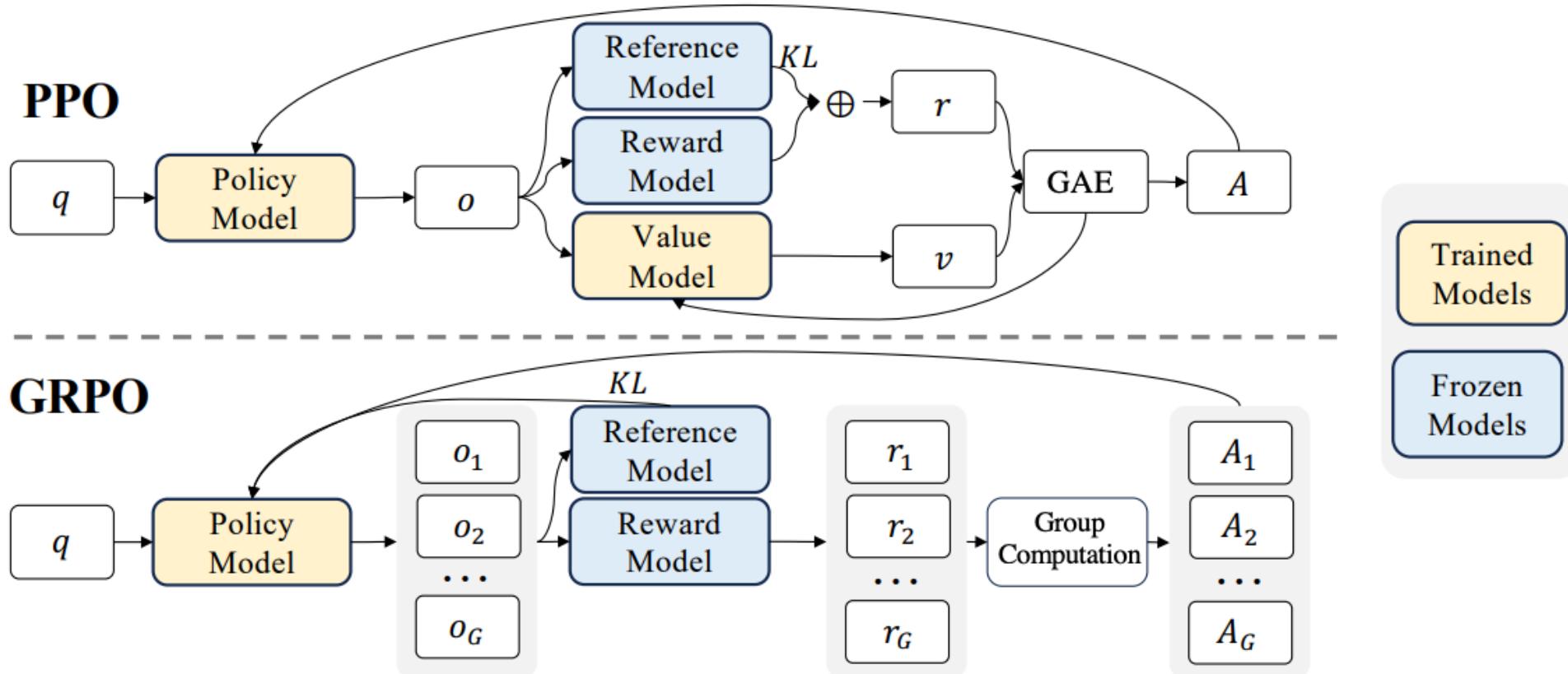


Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.

RL为什么有效?

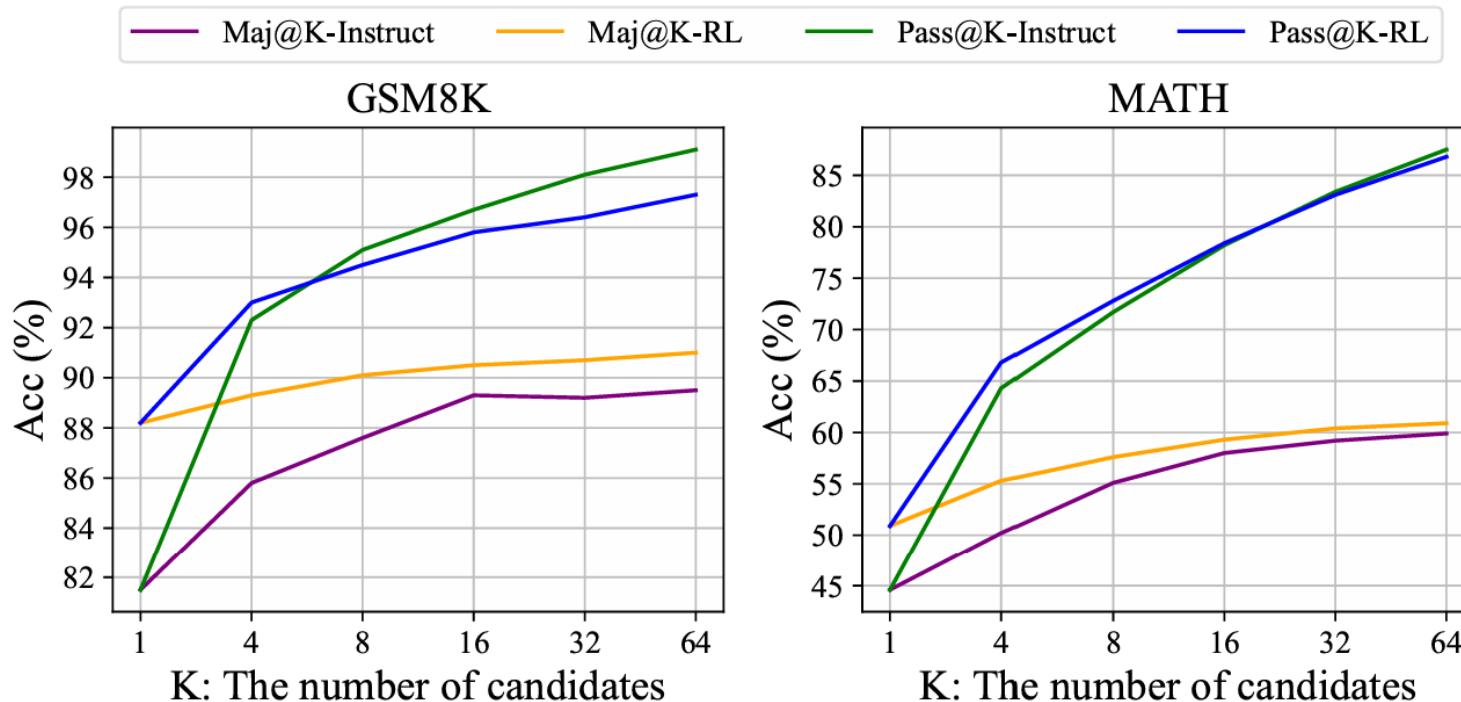


Figure 7 | The Maj@K and Pass@K of SFT and RL DeepSeekMath 7B on GSM8K and MATH (temperature 0.7). It was noted that RL enhances Maj@K but not Pass@K.

- 文中在两个数学任务上进行了实验，评估了Maj@K和Pass@K两个指标
 - Maj@K表示对每个问题生成K次答案，选择出现概率最高的答案看是否正确
 - Pass@K表示对每个问题生成K次答案，只要有一次生成正确答案，就算成功。
- 实验发现，RL之后的模型的Maj@K提升明显，但是Pass@K没有提升。这说明RL的作用是使模型的输出更健壮（以更高的概率输出正确答案，对应高Maj@K），而不是提高模型的基本能力（高Pass@K）。

DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

DeepSeek-AI



**DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via
Reinforcement Learning**

DeepSeek-AI

research@deepseek.com

DeepSeek-R1

- 开源 DeepSeek - R1 采用纯强化学习达到 OpenAI o1 的水平，成本却低 95%

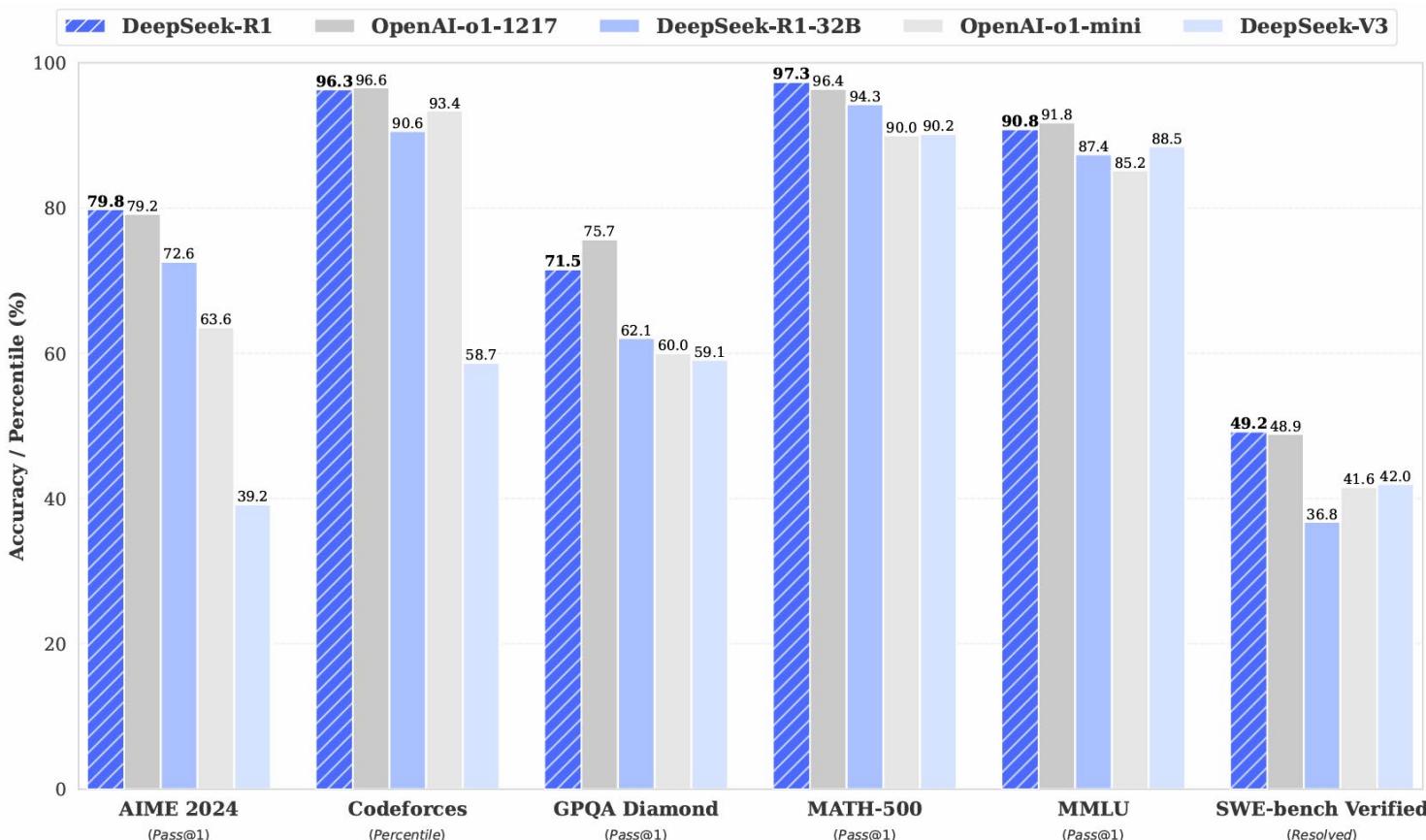


Figure 1 | Benchmark performance of DeepSeek-R1.

推理任务：

(1) DeepSeek-R1在AIME 2024上的Pass@1分数达到79.8%，略高于OpenAI-o1-1217。在MATH-500上，它获得了令人印象深刻的97.3%的分数，与OpenAI-o1-1217相当，并且显著优于其他模型。

(2) 在编码相关任务上，DeepSeek-R1在代码竞赛任务中表现出专家水平，在Codeforces上达到2029的Elo评级，超过了比赛中96.3%的人类参与者。在工程相关任务上，DeepSeek-R1的表现略优于DeepSeek-V3，这可以帮助开发人员完成实际任务。

知识：

在MMLU、MMLU-Pro和GPQA Diamond等基准测试中，DeepSeek-R1取得了出色的结果，在MMLU上的得分为90.8%，在MMLU-Pro上为84.0%，在GPQA Diamond上为71.5%，显著优于DeepSeek-V3。虽然在这些基准测试上它的性能略低于OpenAI-o1-1217，但DeepSeek-R1超过了其他开源模型，展示了其在教育任务中的竞争优势。

在事实基准测试SimpleQA上，DeepSeek-R1优于DeepSeek-V3，展示了其处理基于事实的查询的能力。在这个基准测试上，OpenAI-o1超过GPT-4o也呈现出类似的趋势。

其他：

DeepSeek-R1在广泛的任务中也表现出色，包括创意写作、一般问答、编辑、总结等。它在AlpacaEval 2.0上实现了令人印象深刻的87.6%的长度控制胜率，在ArenaHard上的胜率为92.3%，展示了其智能处理非考试导向查询的强大能力。此外，DeepSeek-R1在需要长上下文理解的任务中表现出色，在长上下文基准测试上显著优于DeepSeek-V3。

DeepSeek-R1

- 开源 DeepSeek - R1 采用纯强化学习达到 OpenAI o1 的水平，成本却低 95%

Text tokens

Price per 1M tokens · Batch API price

| Model | Input | Cached input | Output |
|---|---------|--------------|---------|
| gpt-4o ↳ gpt-4o-2024-08-06 | \$2.50 | \$1.25 | \$10.00 |
| gpt-4o-audio-preview ↳ gpt-4o-audio-preview-2024-12-17 | \$2.50 | - | \$10.00 |
| gpt-4o-realtime-preview ↳ gpt-4o-realtime-preview-2024-12-17 | \$5.00 | \$2.50 | \$20.00 |
| gpt-4o-mini ↳ gpt-4o-mini-2024-07-18 | \$0.15 | \$0.075 | \$0.60 |
| gpt-4o-mini-audio-preview ↳ gpt-4o-mini-audio-preview-2024-12-17 | \$0.15 | - | \$0.60 |
| gpt-4o-mini-realtime-preview ↳ gpt-4o-mini-realtime-preview-2024-12-17 | \$0.60 | \$0.30 | \$2.40 |
| o1 ↳ o1-2024-12-17 | \$15.00 | \$7.50 | \$60.00 |
| o1-mini ↳ o1-mini-2024-09-12 | \$3.00 | ¥1.50 | ¥12.00 |

USD CNY

| MODEL ⁽¹⁾ | CONTEXT LENGTH | MAX COT TOKENS ⁽²⁾ | MAX OUTPUT TOKENS ⁽³⁾ | 1M TOKENS INPUT PRICE (CACHE HIT) ⁽⁴⁾ | 1M TOKENS INPUT PRICE (CACHE MISS) | 1M TOKENS OUTPUT PRICE |
|----------------------|----------------|-------------------------------|----------------------------------|--|------------------------------------|---------------------------------|
| deepseek-chat | 64K | - | 8K | \$0.07 ⁽⁵⁾ \$0.014 | \$0.27 ⁽⁵⁾ \$0.14 | \$1.10 ⁽⁵⁾ \$0.28 |
| deepseek-reasoner | 64K | 32K | 8K | \$0.14 | \$0.55 | \$2.19 ⁽⁶⁾ |

- OpenAI o1 每百万输入令牌成本为 15 美元，每百万输出令牌成本为 60 美元
- 基于 R1 模型的 DeepSeek Reasoner 每百万输入令牌成本为 0.55 美元，每百万输出令牌成本为 2.19 美元。

DeepSeek R1 series

- **DeepSeek-R1-Zero**

➤ 基于基础模型直接用强化学习训练，采用 GRPO 算法、规则奖励系统和特定模板，训练中展现出性能提升、自我进化及“顿悟时刻”，但存在可读性和语言混合问题

- **DeepSeek-R1**

➤ 先利用冷启动数据微调，再进行强化学习，包括推理导向训练、拒绝采样与监督微调及全场景强化学习等阶段

- **DeepSeek distill**

➤ 从 DeepSeek-R1 向小模型蒸馏的方法及效果，蒸馏模型也优于部分已有模型

DeepSeek-R1-Zero：基础模型上的强化学习

- LLMs在没有任何监督数据的情况下发展推理能力的潜力，重点关注它们通过纯强化学习过程的自我进化
- 基础模型：DeepSeek-V3-Base（671B 基座大模型）
- 群体相对策略优化 Group Relative Policy Optimization, GRPO (Shao, et al. 2024)

**群体相对策略优化 (Group Relative Policy Optimization, GRPO) **：为了节省RL的训练成本，我们采用了GRPO (Shao等人, 2024)，它摒弃了通常与策略模型大小相同的评论者模型，而是从群体分数中估计基线。具体来说，对于每个问题 q ，GRPO从旧策略 $\pi_{\theta_{old}}$ 中采样一组输出 $\{o_1, o_2, \dots, o_G\}$ ，然后通过最大化以下目标来优化策略模型 π_θ ：

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E} \left[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q) \right]$$

$$\frac{1}{G} \sum_{i=1}^G \left(\min \left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)} A_i, \text{clip} \left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1 - \varepsilon, 1 + \varepsilon \right) A_i \right) - \beta \mathbb{D}_{KL}(\pi_\theta \| \pi_{ref}) \right), \quad (1)$$

$$\mathbb{D}_{KL}(\pi_\theta \| \pi_{ref}) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1, \quad (2)$$

其中 ε 和 β 是超参数， A_i 是优势，使用与每个组内输出相对应的一组奖励 $\{r_1, r_2, \dots, r_G\}$ 计算：

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \dots, r_G\})}{\text{std}(\{r_1, r_2, \dots, r_G\})}, \quad (3)$$

用户和助手之间的对话。用户提出问题，助手解决问题。助手首先在脑海中思考推理过程，然后向用户提供答案。推理过程和答案分别包含在和标签内，即这里是推理过程这里是答案。

DeepSeek-R1-Zero：奖励建模

基于规划 (rule-based) 的两种类型奖励：

- **准确性奖励**：准确性奖励模型评估响应是否正确。
 - 例如，对于具有确定性结果的数学问题，模型需要以指定的格式（例如，在框\box{}内）提供最终答案，以便能够基于规则可靠地验证正确性。同样，对于LeetCode问题，可以使用编译器根据预定义的测试用例生成反馈。
- **格式奖励**：除了准确性奖励模型外，还采用了格式奖励模型，强制模型将其思考过程放在’<think>’和 ’<think>’标签之间，将答案放在<answer>和<answer>标签之间。

(与OpenAI o1不同) 在开发DeepSeek-R1-Zero时，我们没有应用结果或过程神经奖励模型，因为发现神经奖励模型在大规模强化学习过程中可能会遭受奖励破解问题，并且重新训练奖励模型需要额外的训练资源，这会使整个训练管道复杂化。

DeepSeek-R1-Zero：训练模板

- 为了训练DeepSeek-R1-Zero，设计了一个简单的模板，引导基础模型遵循指定的指令。
- 如表1所示，这个模板要求DeepSeek-R1-Zero首先生成推理过程，然后是最终答案。
- 将约束限制在这种结构格式上，避免任何特定内容的偏差，例如强制进行反思性推理或推广特定的问题解决策略，以确保我们能够准确观察模型在强化学习（RL）过程中的自然进展。

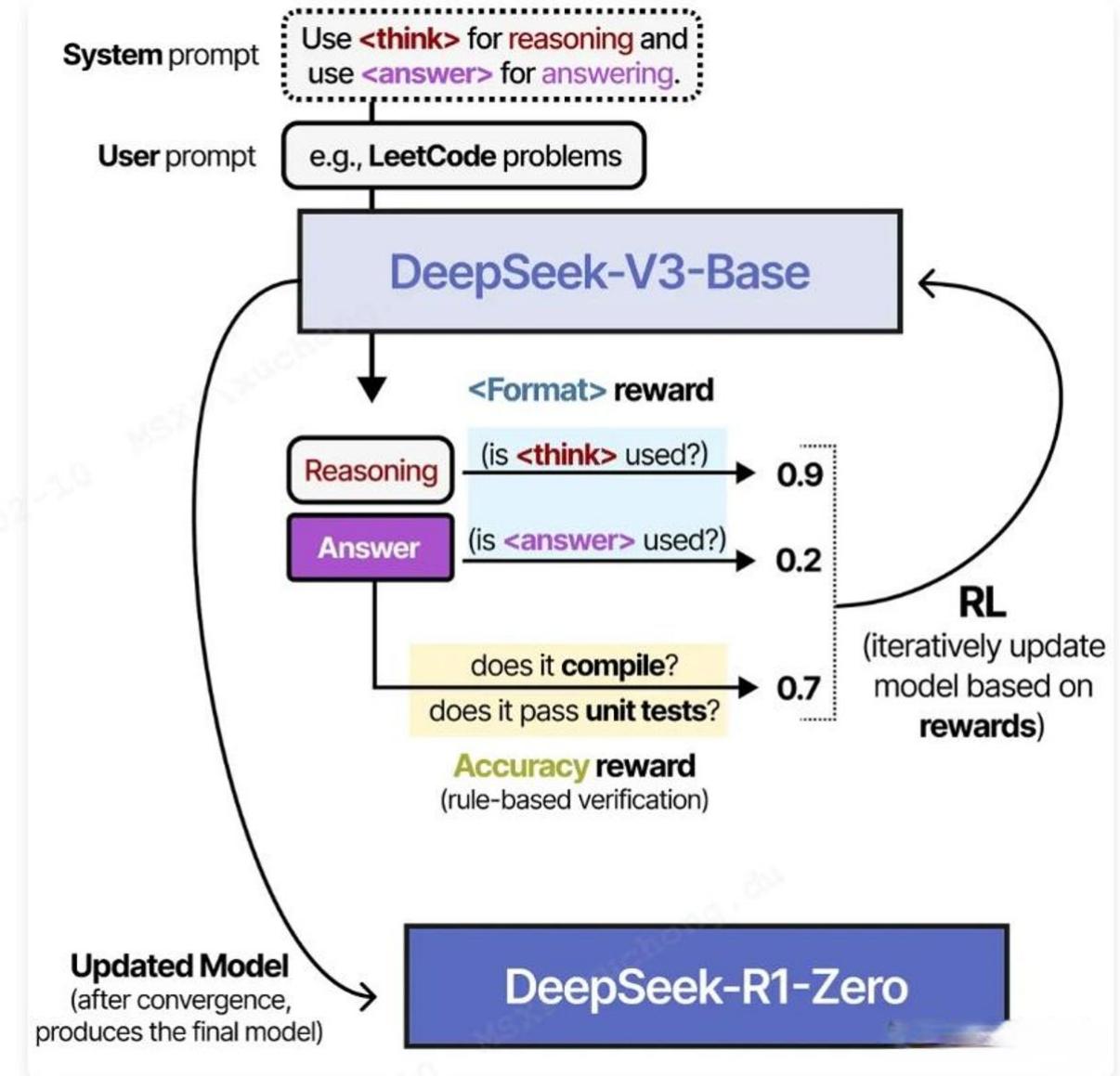
A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. User: **prompt**. Assistant:

Table 1 | Template for DeepSeek-R1-Zero. **prompt** will be replaced with the specific reasoning question during training.

DeepSeek-R1-Zero：写 leetcode代码题举例

用DeepSeek-V3-Base零样本生成答案。

- **格式奖励 (Format rewards)** 会对是否用了<think>和<answer>标签进行奖励<think>推理过程在think标签内? </think>, <answer>答案 (LeetCode代码) 在 answer标签内?</answer>
- **准确度奖励 (Accuracy rewards)**对答案的正确性进行奖励, 在这里对leetcode代码是否可以完成编译 (代码无运行bug) 和leetcode代码是否通过样例测试(代码结果正确) 进行奖励



DeepSeek-R1-Zero的性能

- 图2展示了DeepSeek-R1-Zero在AIME 2024基准测试中整个强化学习 (RL) 训练过程中的性能轨迹。
- AIME 2024上的平均pass@1分数显著提高，从最初的15.6%跃升至令人印象深刻的71.0%，达到了与OpenAI-o1-0912相当的性能水平。凸显了RL算法在随着时间优化模型性能方面的有效性。

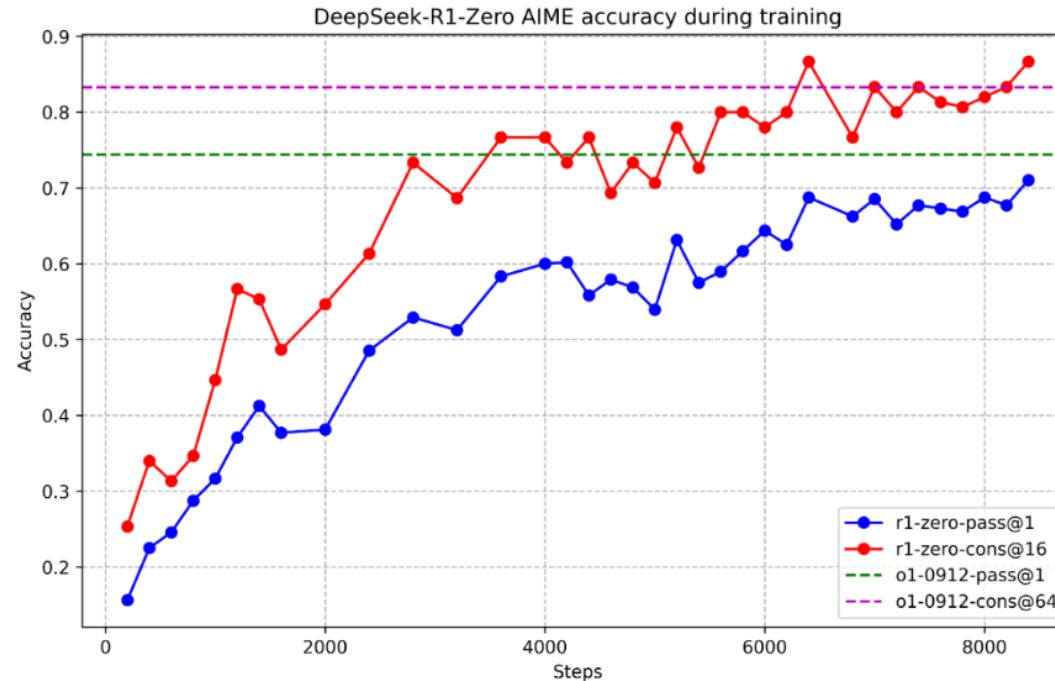


Figure 2 | AIME accuracy of DeepSeek-R1-Zero during training. For each question, we sample 16 responses and calculate the overall average accuracy to ensure a stable evaluation.

DeepSeek-R1-Zero思考时间的增加

- DeepSeek-R1-Zero在训练过程中的思考时间持续增加。这种改进不是外部调整的结果，而是模型内部的内在发展。DeepSeek-R1-Zero通过利用扩展的测试时计算自然地获得了解决日益复杂推理任务的能力。这种计算从生成数百到数千个推理标记不等，使模型能够更深入地探索和完善其思维过程。

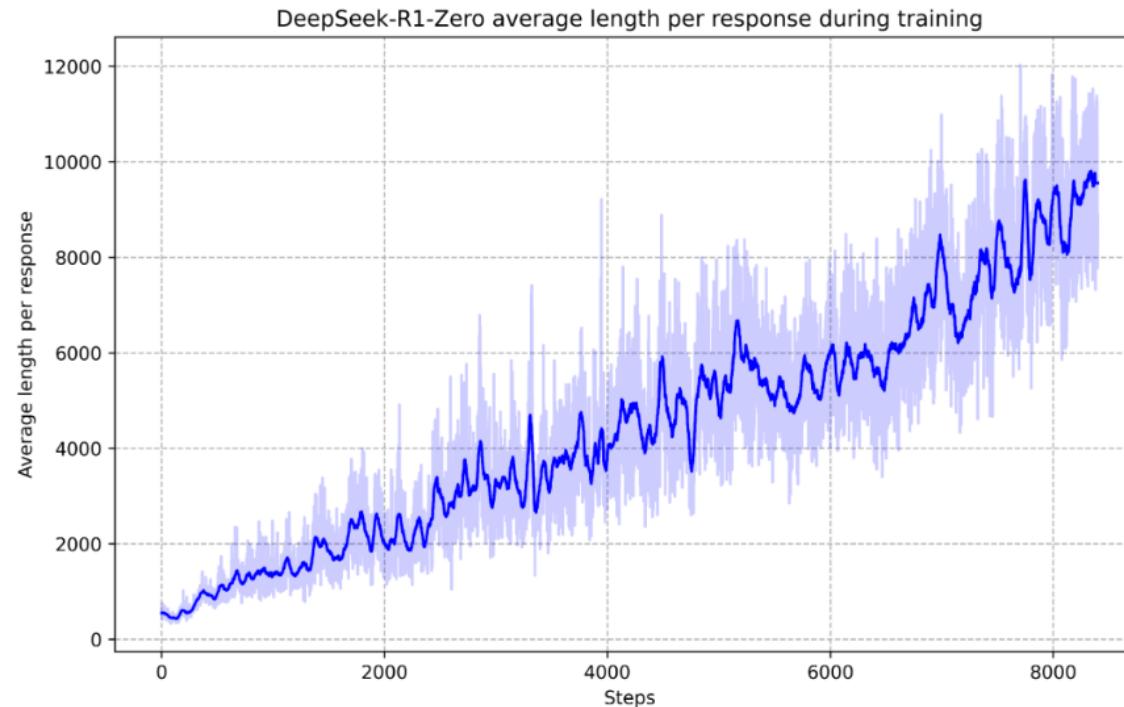


Figure 3 | The average response length of DeepSeek-R1-Zero on the training set during the RL process. DeepSeek-R1-Zero naturally learns to solve reasoning tasks with more thinking time.

DeepSeek-R1-Zero “顿悟时刻”

- 如表3所示，这个时刻出现在模型的一个中间版本中。在此阶段，DeepSeek - R1 - Zero通过重新评估其初始方法学会为一个问题分配更多的思考时间。这种行为不仅证明了模型推理能力的不断增长，也是强化学习如何产生意想不到的复杂结果的一个引人入胜的例子。

Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a - \sqrt{a + x}} = x$ is equal to

Response: <think>

To solve the equation $\sqrt{a - \sqrt{a + x}} = x$, let's start by squaring both ...

$$(\sqrt{a - \sqrt{a + x}})^2 = x^2 \implies a - \sqrt{a + x} = x^2.$$

Rearrange to isolate the inner square root term:

$$(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$$

...

Wait, wait. Wait. That's an aha moment I can flag here.

Let's reevaluate this step-by-step to identify if the correct sum can be ...

We started with the equation:

$$\sqrt{a - \sqrt{a + x}} = x$$

First, let's square both sides:

$$a - \sqrt{a + x} = x^2 \implies \sqrt{a + x} = a - x^2$$

Next, I could square both sides again, treating the equation: ...

...

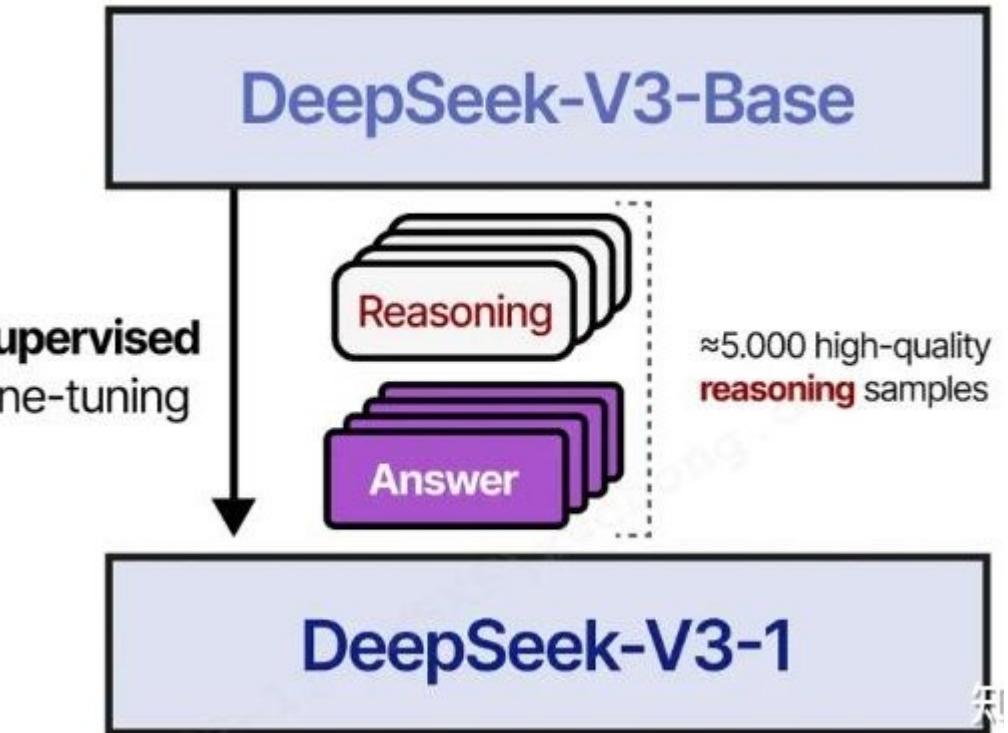
Table 3 | An interesting “aha moment” of an intermediate version of DeepSeek-R1-Zero. The model learns to rethink using an anthropomorphic tone. This is also an aha moment for us, allowing us to witness the power and beauty of reinforcement learning.

DeepSeek - R1

- **DeepSeek - R1 - Zero的缺点：**
 - 尽管DeepSeek - R1 - Zero表现出强大的推理能力并自主发展出意想不到的强大推理行为，但它也面临着一些问题。例如，DeepSeek - R1 - Zero在可读性差和语言混合等方面存在困难。
- **DeepSeek - R1：**受DeepSeek - R1 - Zero有前景的结果启发，出现了两个自然的问题：
 - 1) 通过纳入少量高质量数据作为冷启动，推理性能是否可以进一步提高或加速收敛？
 - 2) 我们如何训练一个用户友好的模型，它不仅能生成清晰连贯的思维链（CoT），还能展示出强大的通用能力？

DeepSeek - R1：冷启动

- 与DeepSeek - R1 - Zero不同，为了防止RL训练从基础模型开始的早期不稳定冷启动阶段，对于DeepSeek - R1，构建并收集了数千条长CoT数据作为冷启动数据来微调DeepSeek - V3 - Base作为RL的起点
- 数据收集方式：
 - 使用带有长CoT示例的少样本提示
 - 直接提示模型生成带有反思和验证的详细答案
 - 收集DeepSeek - R1 - Zero的可读格式输出，并通过人工注释者的后处理来精炼结果

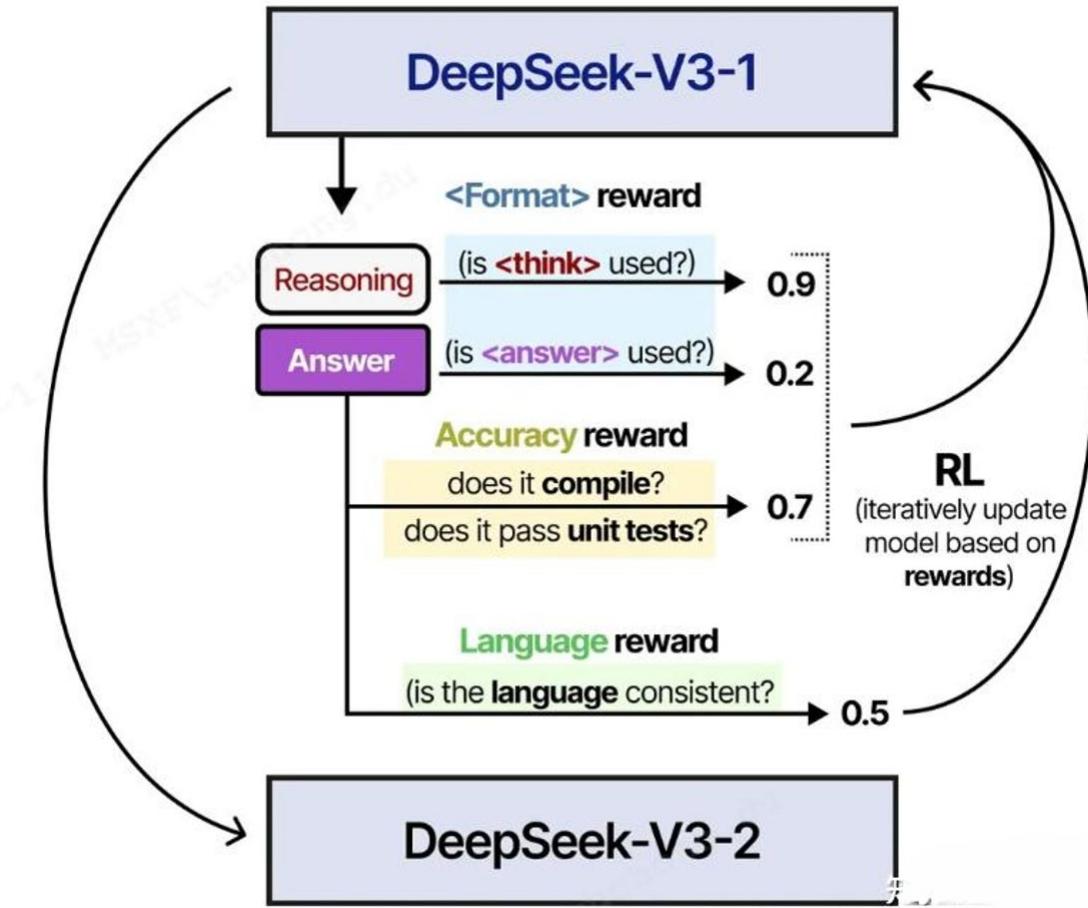


DeepSeek - R1：冷启动

- 冷启动数据的优势：
 - **可读性**：DeepSeek - R1 - Zero的一个关键限制是其内容通常不适合阅读。响应可能混合多种语言或缺乏用于为用户突出显示答案的markdown格式。相比之下，在为DeepSeek - R1创建冷启动数据时，我们设计了一种可读模式，在每个响应的末尾包含一个总结，并过滤掉对读者不友好的响应。这里，我们将输出格式定义为|特殊标记|<推理过程>|特殊标记|<总结>，其中推理过程是查询的CoT，总结用于总结推理结果。
 - **潜力**：通过根据人类先验精心设计冷启动数据的模式，我们观察到相对于DeepSeek - R1 - Zero有更好的性能。我们相信迭代训练是推理模型的更好方法。

DeepSeek - R1：面向推理的强化学习

- 在冷启动数据上微调DeepSeek - V3 - Base之后，应用与DeepSeek - R1 - Zero中相同的大规模强化学习训练过程。
- 这个阶段侧重于提高模型的推理能力，特别是在推理密集型任务中，如编码、数学、科学和逻辑推理，这些任务涉及定义明确且有清晰解决方案的问题。
 - 只纳入了可以使用基于规则的奖励进行评估的数据
- 在训练过程中，观察到CoT经常出现语言混合的情况，特别是当RL提示涉及多种语言时。为了缓解语言混合的问题，在RL训练期间引入了语言一致性奖励，它计算为CoT中目标语言单词的比例。
 - 虽然消融实验表明这种对齐会导致模型性能略有下降，但这种奖励符合人类偏好，使其更具可读性。
- 最后，通过直接将推理任务的准确性和语言一致性奖励相加来形成最终奖励。然后，对微调后的模型应用强化学习（RL）训练，直到它在推理任务上达到收敛。



DeepSeek - R1：拒绝采样和监督微调

- 当面向推理的RL收敛时，利用得到的检查点为下一轮收集SFT（监督微调）数据。
- 与主要关注推理的初始冷启动数据不同，这个阶段纳入了来自其他领域的数据，以增强模型在写作、角色扮演和其他通用任务中的能力。

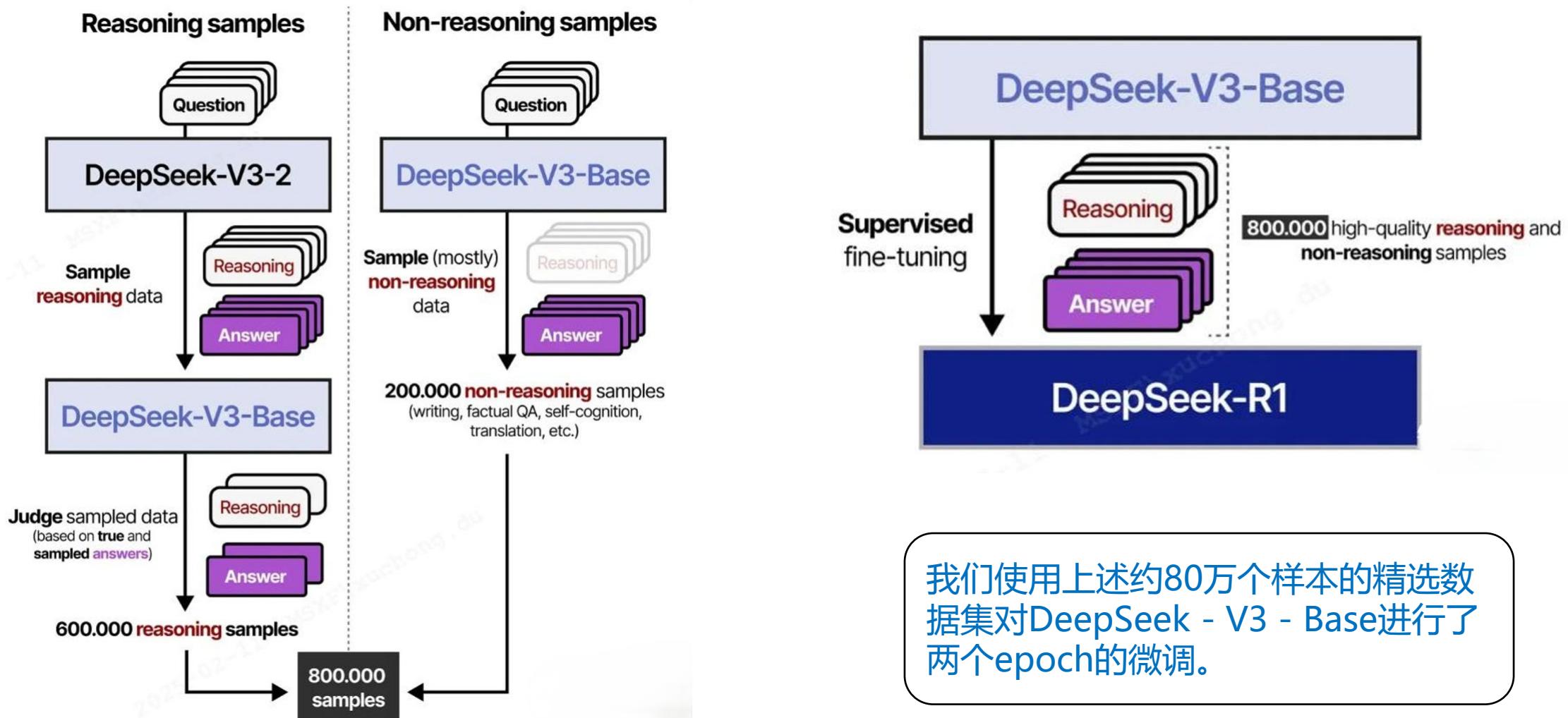
推理数据：

- 我们整理推理提示，并通过从上述RL训练的检查点进行拒绝采样来生成推理轨迹。由于模型输出有时混乱且难以阅读，我们过滤掉了混合语言的思维链、长段落和代码块。对于每个提示，我们采样多个响应并只保留正确的响应。
- 在先前阶段，我们只纳入了可以使用基于规则的奖励进行评估的数据。然而，在这个阶段，我们通过纳入额外的数据扩展了数据集，其中一些数据使用生成式奖励模型，通过将真实值和模型预测输入到DeepSeek - V3中进行判断。
- 总共，收集了大约60万个与推理相关的训练样本。

非推理数据：

- 对于非推理数据，如写作、事实问答、自我认知和翻译，我们采用DeepSeek - V3管道并重用DeepSeek - V3的部分SFT数据集。
- 对于某些非推理任务，我们在回答问题之前通过提示调用DeepSeek - V3生成潜在的思维链。
- 然而，对于更简单的查询，如“hello”，我们在响应中不提供CoT。最后，我们总共收集了大约20万个与推理无关的训练样本。

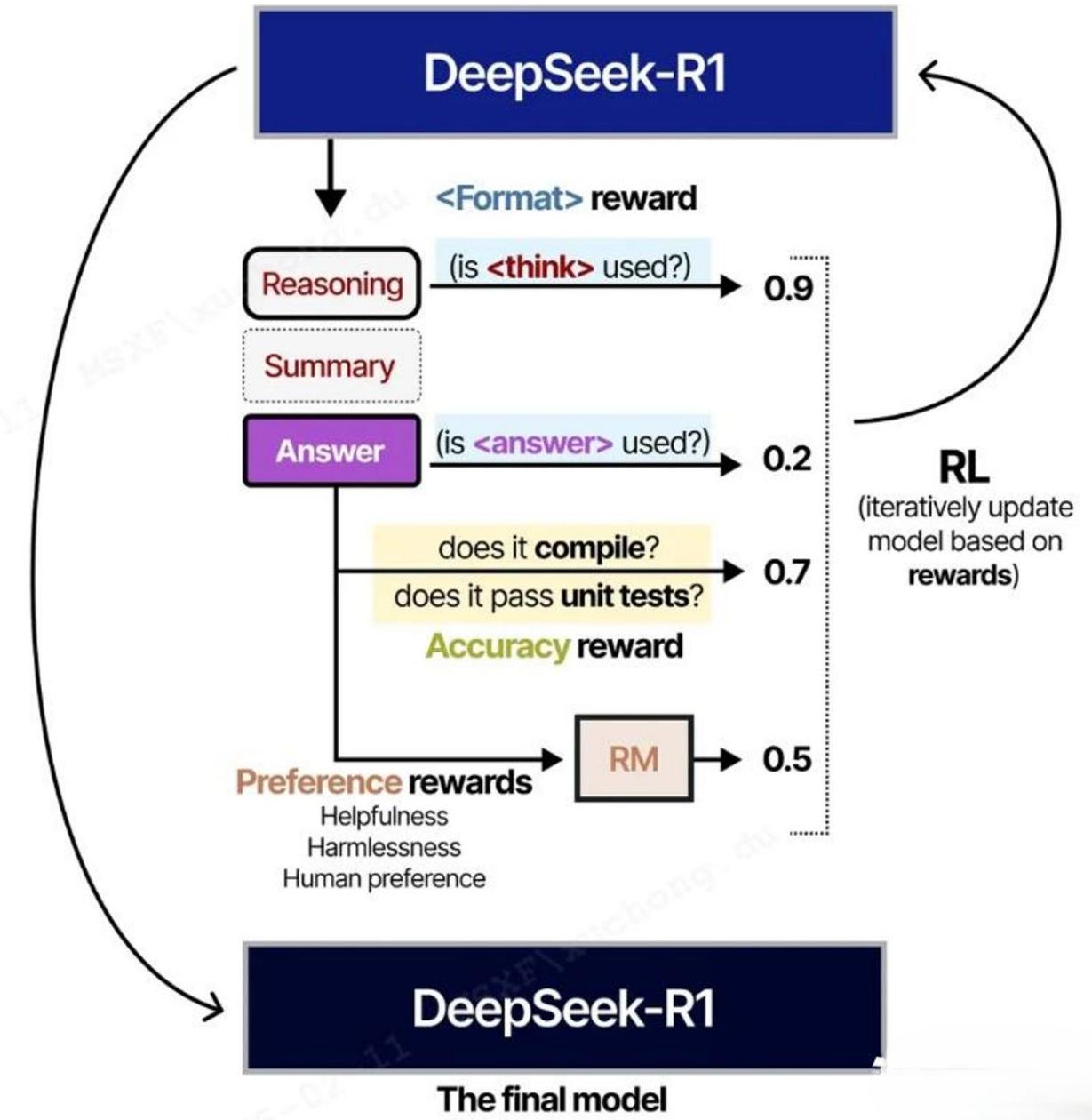
DeepSeek - R1：拒绝采样和监督微调



DeepSeek - R1：所有场景的强化学习

- 为了进一步使模型与人类偏好对齐，我们实施了一个二级强化学习阶段，旨在提高模型的帮助性和无害性，同时完善其推理能力。
- 具体来说，我们使用奖励信号和多样化的提示分布组合来训练模型。
 - 对于推理数据，我们遵循DeepSeek - R1 - Zero中概述的方法，利用基于规则的奖励来指导数学、代码和逻辑推理领域的学习过程。
 - 对于一般数据，我们借助奖励模型在复杂和微妙的场景中捕捉人类偏好。我们基于DeepSeek - V3管道并采用类似的偏好对和训练提示分布。
 - 对于帮助性，我们专注于最终总结，确保评估强调响应对用户的效用和相关性，同时尽量减少对底层推理过程的干扰。
 - 对于无害性，我们评估模型的整个响应，包括推理过程和总结，以识别和减轻在生成过程中可能出现的任何潜在风险、偏差或有害内容。
- 最终，奖励信号和多样化数据分布的整合使我们能够训练出一个在推理方面表现出色，同时优先考虑帮助性和无害性的模型。

DeepSeek - R1: 所有场景的强化学习



DeepSeek - R1评估

- 对于以教育为导向的知识基准测试，如MMLU、MMLU - Pro和GPQA Diamond，DeepSeek - R1相对于DeepSeek - V3表现出更优的性能。这种改进主要归因于在STEM相关问题上的准确性提高，这是通过大规模强化学习（RL）实现的显著增益。
- DeepSeek - R1在FRAMES上表现出色，这是一个依赖长上下文的问答任务，展示了其强大的文档分析能力。这凸显了推理模型在人工智能驱动的搜索和数据分析任务中的潜力。
- 在事实基准测试SimpleQA上，DeepSeek - R1优于DeepSeek - V3，展示了其处理基于事实的查询的能力。在这个基准测试上，OpenAI - o1超过GPT - 4o也呈现出类似的趋势。
- DeepSeek - R1在中文SimpleQA基准测试上的表现比DeepSeek - V3差，主要是因为在安全RL之后它倾向于拒绝回答某些查询。如果没有安全RL，DeepSeek - R1的准确率可以超过70%。

| Benchmark (Metric) | | Claude-3.5-Sonnet-1022 | GPT-4o 0513 | DeepSeek V3 | OpenAI o1-mini | OpenAI o1-1217 | DeepSeek R1 |
|--------------------|----------------------------|------------------------|-------------|-------------|----------------|----------------|-------------|
| English | Architecture | - | - | MoE | - | - | MoE |
| | # Activated Params | - | - | 37B | - | - | 37B |
| | # Total Params | - | - | 671B | - | - | 671B |
| | MMLU (Pass@1) | 88.3 | 87.2 | 88.5 | 85.2 | 91.8 | 90.8 |
| | MMLU-Redux (EM) | 88.9 | 88.0 | 89.1 | 86.7 | - | 92.9 |
| | MMLU-Pro (EM) | 78.0 | 72.6 | 75.9 | 80.3 | - | 84.0 |
| | DROP (3-shot F1) | 88.3 | 83.7 | 91.6 | 83.9 | 90.2 | 92.2 |
| | IF-Eval (Prompt Strict) | 86.5 | 84.3 | 86.1 | 84.8 | - | 83.3 |
| | GPQA Diamond (Pass@1) | 65.0 | 49.9 | 59.1 | 60.0 | 75.7 | 71.5 |
| | SimpleQA (Correct) | 28.4 | 38.2 | 24.9 | 7.0 | 47.0 | 30.1 |
| Code | FRAMES (Acc.) | 72.5 | 80.5 | 73.3 | 76.9 | - | 82.5 |
| | AlpacaEval2.0 (LC-winrate) | 52.0 | 51.1 | 70.0 | 57.8 | - | 87.6 |
| | ArenaHard (GPT-4-1106) | 85.2 | 80.4 | 85.5 | 92.0 | - | 92.3 |
| | LiveCodeBench (Pass@1-COT) | 38.9 | 32.9 | 36.2 | 53.8 | 63.4 | 65.9 |
| | Codeforces (Percentile) | 20.3 | 23.6 | 58.7 | 93.4 | 96.6 | 96.3 |
| Math | Codeforces (Rating) | 717 | 759 | 1134 | 1820 | 2061 | 2029 |
| | SWE Verified (Resolved) | 50.8 | 38.8 | 42.0 | 41.6 | 48.9 | 49.2 |
| | Aider-Polyglot (Acc.) | 45.3 | 16.0 | 49.6 | 32.9 | 61.7 | 53.3 |
| | AIME 2024 (Pass@1) | 16.0 | 9.3 | 39.2 | 63.6 | 79.2 | 79.8 |
| Chinese | MATH-500 (Pass@1) | 78.3 | 74.6 | 90.2 | 90.0 | 96.4 | 97.3 |
| | CNMO 2024 (Pass@1) | 13.1 | 10.8 | 43.2 | 67.6 | - | 78.8 |
| | CLUEWSC (EM) | 85.4 | 87.9 | 90.9 | 89.9 | - | 92.8 |
| | C-Eval (EM) | 76.7 | 76.0 | 86.5 | 68.9 | - | 91.8 |
| | C-SimpleQA (Correct) | 55.4 | 58.7 | 68.0 | 40.3 | - | 63.7 |

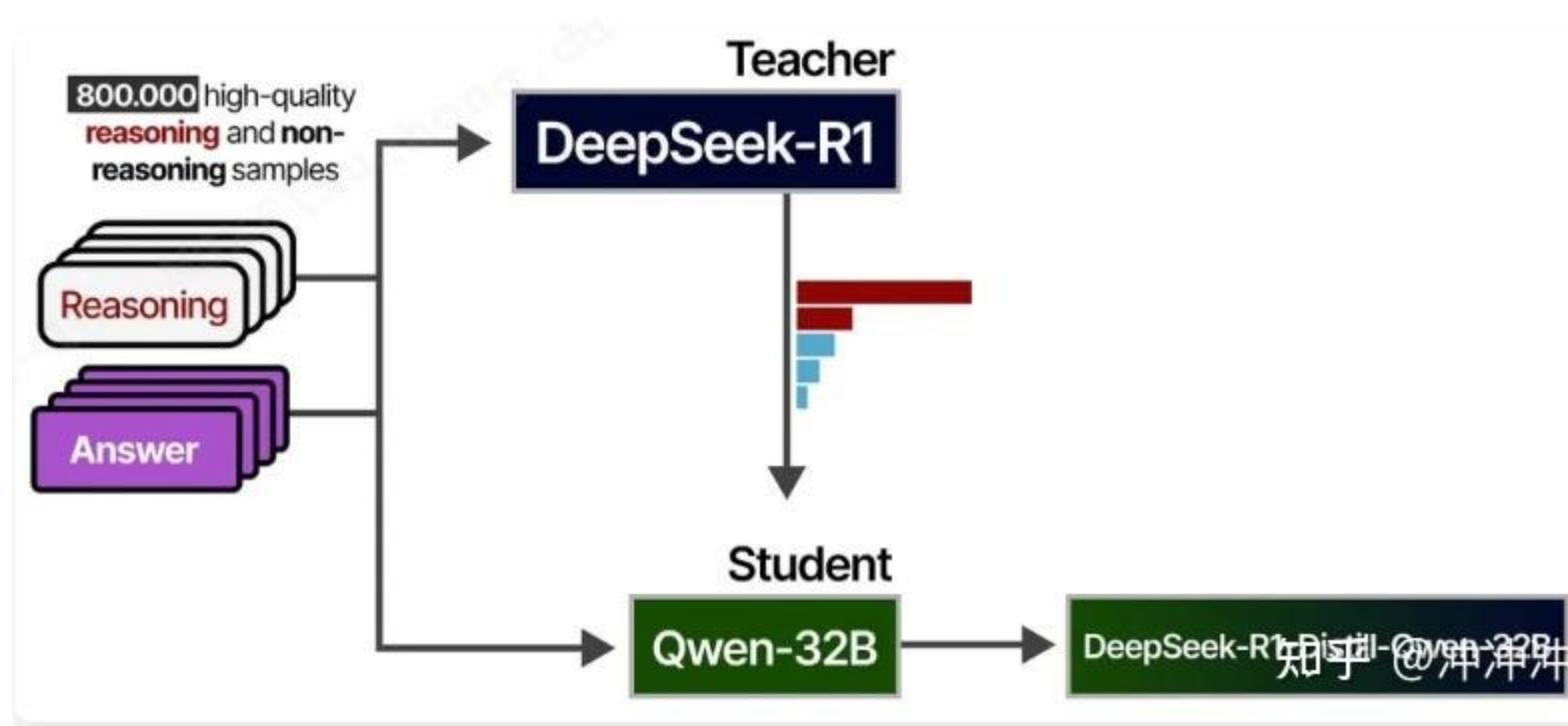
Table 4 | Comparison between DeepSeek-R1 and other representative models.

DeepSeek - R1蒸馏： 赋予小型模型推理 能力

使用DeepSeek - R1整理的80万个样本直接对开源模型如Qwen (Qwen, 2024b) 和Llama (AI@Meta, 2024) 进行微调。

- 使用的基础模型是Qwen2.5 - Math - 1.5B、Qwen2.5 - Math - 7B、Qwen2.5 - 14B、Qwen2.5 - 32B、Llama - 3.1 - 8B和Llama - 3.3 - 70B - Instruct。

我们的研究结果表明，这种直接蒸馏方法显著增强了小型模型的推理能力。



- 让DeepSeek-R1作为教师模型 (Teacher)
- 体量较小的Qwen模型作为学生模型 (student)
- 两者面对相同的提示输入时，学生模型不是直接学习原始的标签（输出的token文本）而是学习预测下一个token的概率分布（softmax层）
- 学生模型在训练中会逼近教师模型的概率分布。

DeepSeek - R1蒸馏： 赋予小型模型推理 能力

使用DeepSeek - R1整理的80万个样本直接对开源模型如Qwen (Qwen, 2024b) 和Llama (AI@Meta, 2024) 进行微调。

- 使用的基础模型是Qwen2.5 - Math - 1.5B、Qwen2.5 - Math - 7B、Qwen2.5 - 14B、Qwen2.5 - 32B、Llama - 3.1 - 8B和Llama - 3.3 - 70B - Instruct。

我们的研究结果表明，这种直接蒸馏方法显著增强了小型模型的推理能力。

| Model | AIME 2024 | | MATH-500 | GPQA Diamond | LiveCode Bench | CodeForces |
|-------------------------------|-------------|-------------|-------------|--------------|----------------|-------------|
| | pass@1 | cons@64 | | | | |
| GPT-4o-0513 | 9.3 | 13.4 | 74.6 | 49.9 | 32.9 | 759 |
| Claude-3.5-Sonnet-1022 | 16.0 | 26.7 | 78.3 | 65.0 | 38.9 | 717 |
| OpenAI-o1-mini | 63.6 | 80.0 | 90.0 | 60.0 | 53.8 | 1820 |
| QwQ-32B-Preview | 50.0 | 60.0 | 90.6 | 54.5 | 41.9 | 1316 |
| DeepSeek-R1-Distill-Qwen-1.5B | 28.9 | 52.7 | 83.9 | 33.8 | 16.9 | 954 |
| DeepSeek-R1-Distill-Qwen-7B | 55.5 | 83.3 | 92.8 | 49.1 | 37.6 | 1189 |
| DeepSeek-R1-Distill-Qwen-14B | 69.7 | 80.0 | 93.9 | 59.1 | 53.1 | 1481 |
| DeepSeek-R1-Distill-Qwen-32B | 72.6 | 83.3 | 94.3 | 62.1 | 57.2 | 1691 |
| DeepSeek-R1-Distill-Llama-8B | 50.4 | 80.0 | 89.1 | 49.0 | 39.6 | 1205 |
| DeepSeek-R1-Distill-Llama-70B | 70.0 | 86.7 | 94.5 | 65.2 | 57.5 | 1633 |

Table 5 | Comparison of DeepSeek-R1 distilled models and other comparable models on reasoning-related benchmarks.

简单地蒸馏DeepSeek - R1的输出使高效的DeepSeek - R1 - 7B（即DeepSeek - R1 - Distill - Qwen - 7B，以下类似缩写）在各个方面都优于非推理模型，如GPT - 4o - 0513。

DeepSeek - R1 - 14B在所有评估指标上都超过了QwQ - 32B - Preview，而DeepSeek - R1 - 32B和DeepSeek - R1 - 70B在大多数基准测试上显著超过了o1 - mini。

蒸馏与强化学习

通过蒸馏DeepSeek - R1，小型模型可以取得令人印象深刻的结果。然而，仍然存在一个问题：

模型是否可以通过本文中讨论的大规模RL训练而不进行蒸馏来达到类似的性能？

| Model | AIME 2024 | | MATH-500 | GPQA Diamond | LiveCodeBench |
|------------------------------|-----------|---------|----------|--------------|---------------|
| | pass@1 | cons@64 | pass@1 | pass@1 | pass@1 |
| QwQ-32B-Preview | 50.0 | 60.0 | 90.6 | 54.5 | 41.9 |
| DeepSeek-R1-Zero-Qwen-32B | 47.0 | 60.0 | 91.6 | 55.0 | 40.2 |
| DeepSeek-R1-Distill-Qwen-32B | 72.6 | 83.3 | 94.3 | 62.1 | 57.2 |

Table 6 | Comparison of distilled and RL Models on Reasoning-Related Benchmarks.

- 为了回答这个问题，我们使用数学、代码和STEM数据对Qwen - 32B - Base进行了大规模RL训练，训练超过10K步，得到DeepSeek - R1 - Zero - Qwen - 32B。
- 实验结果表明32B基础模型在经过大规模RL训练后，性能与QwQ - 32B - Preview相当。
- 然而，从DeepSeek - R1蒸馏得到的DeepSeek - R1 - Distill - Qwen - 32B在所有基准测试上的表现都显著优于DeepSeek - R1 - Zero - Qwen - 32B。
- 因此，我们可以得出两个结论：首先，将更强大的模型蒸馏到较小的模型中会产生出色的结果，而依赖于本文中大规模RL的较小模型需要巨大的计算能力，甚至可能无法达到蒸馏的性能。其次，虽然蒸馏策略既经济又有效，但要超越智能的边界可能仍然需要更强大的基础模型和更大规模的强化学习。

DeepSeek R1 完整训练流程

第一次SFT (Cold Start)

- 使用DeepSeek R1-Zero生成高质量CoT数据
- 增加输出可读性和性能
- 规定格式(如Markdown, Summary)并进行人工标注



第一次RL (Reasoning训练)

- 专注于代码、数学、科学、逻辑推理能力训练
- 设置专门的RM处理中英语言混乱问题
- 主要强化学习阶段



第二次SFT (综合能力提升)

- Rejection Sampling筛选高质量数据(60万推理样本)
- 加入写作、角色扮演等通用数据(20万非推理样本)
- 使用Model Based RM(DeepSeek-V3)评估
- 两轮微调优化



第二次RL (质量优化)

- Rule Based RM优化数理化逻辑
- Model Based RM优化通用输出
- 确保输出相关性、准确性，减少有害内容

知识蒸馏 (可选)

- 使用MoE大模型(DeepSeek V1)生成800k高质量样本进行小模型SFT训练

Reference

- RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback
- GPT-4 Technical Report
- Training language models to follow instructions with human feedback
- Fine-Tuning Language Models from Human Preferences
- Learning to summarize from human feedback
- Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback
- Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs
- REINFORCE++: An Efficient RLHF Algorithm with Robustness to Both Prompt and Reward Models
- DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models
- DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning