

强化学习从人类反馈中

简短介绍RLHF和针对语言模型的训练后重点

Nathan Lambert

2 诺瓦姆布爾 2025

摘要

强化学习从人类反馈（RLHF）已成为部署最新机器学习系统的重要技术和叙事工具。在这本书中，我们希望为具有一定量化背景的人提供一个温和的介绍。本书从RLHF的起源开始——既包括最近文献中的起源，也包括经济学、哲学和最优控制等不同科学领域的汇聚。然后，我们通过定义、问题表述、数据收集和其他在文献中常用的数学方法来设定舞台。本书的核心详细介绍了使用RLHF的每个优化阶段，从指令调整开始，到训练奖励模型，最后到所有拒绝采样、强化学习和直接对齐算法。本书以高级主题结束——合成数据和评估中未充分研究的科研问题——以及该领域的开放性问题。

1 简介 6

2 关键相关作品 14

3 定义与背景 16

4 训练概述 20

4.2 标准训:

5 偏好的本质:25

白粉粘指 500

[illegible]

Z

6.3.5 采购和合同	38
7 奖励建模	38
7.1 训练奖励模型	38
7.2 架构	39
7.3 实现示例	39
7.4 变体	41
7.4.1 偏好间隔损失	41
7.4.2 平衡每个提示的多次比较	41
7.4.3 K-次损失函数	42
7.5 结果奖励模型	42
7.6 流程奖励模型	44
7.7 奖励模型与结果RM与流程RM与价值函数的比较	46
7.8 生成性奖励建模	46
7.9 进一步阅读	47
8 正则化	49
8.1 RL优化中的KL距离	49
8.1.1 参考模型到生成	50
8.1.2 实现示例	50
8.2 预训练梯度	51
8.3 其他正则化	51
9 指令微调	52
9.1 聊天模板和指令结构	52
10 拒绝采样	56
10.1 训练过程	56
10.1.1 生成补全	56
10.2 相关：最佳N采样	60
11 强化学习（即策略梯度算法）	61
11.1 策略梯度算法	61
11.2 实施方案	74
11.2.1 政策梯度基础	74
11.2.2 损失聚合	74

11.3 辅助主题	
11.3.1 算法比较	83
11.3.2 广义优势估计 (GAE)	84
11.3.3 双重正则化	
12 直接对齐算法 87	
12.1 直接偏好优化 (DPO)	87
12.1.1 DPO如何工作	87
12.1.2 DPO推导	88
12.2 数值问题、弱点及替代方案	93
12.3 实施考虑因素	94
12.4 DAAs与RL：在线数据与离线数据	95
13 宪法人工智能 & 人工智能反馈 96	
13.1 宪法AI	
14 推理训练与推理时间缩放 99	
14.1 新推理模型起源	102
14.1.1 为什么现在RL有效?	102
14.1.2 RL训练与推理时间缩放	102
14.1.3 强化微调的未来 (超越推理)	103
14.2 理解推理训练方法	103
14.2.1 在OpenAI的o1或DeepSeek R1之前推理研究	103
14.2.2 早期推理模型	104
14.2.3 训练推理模型的常见做法	105
15 工具使用与函数调用 107	
15.1 生成中的工具调用交织	
16 人工数据与蒸馏 113	
17 评估 115	
17.1 提示格式化：从少样本到零样本到CoT	115
17.2 使用评估与观察评估	119
17.3 污染	121
17.4 工具	121
18 过度优化 122	
18.1 定性过度优化	
4	

19 风格和信息	128	19.1 Chattness Paradox	128	19.1
.1 Chattness 如何出现	129	20 产品、UX 和模型特征	131	
20.1 特征训练	131	20.2 模型规格		
	132	20.3 产品周期、UX 和 RLHF		
	132	参考文献目录	134	

1 简介

强化学习从人类反馈（RLHF）是一种将人类信息融入AI系统的技术。RLHF最初作为一种解决难以指定问题的方法出现。其早期应用通常在控制问题和强化学习（RL）的其他传统领域。RLHF通过ChatGPT的发布而广为人知，随后大型语言模型（LLMs）和其他基础模型迅速发展。

RLHF的基本流程涉及三个步骤。首先，必须训练一个能够跟随用户问题的语言模型（见第9章）。其次，必须收集人类偏好数据以训练人类偏好奖励模型（见第7章）。最后，可以使用选择的RL优化器通过采样生成并按照奖励模型对其进行评分来优化语言模型（见第3章和第11章）。本书详细介绍了此过程中每一步的关键决策和基本实现示例。

RLHF 已成功应用于许多领域，随着技术的成熟，复杂性不断增加。早期使用 RLHF 的突破性实验应用于深度强化学习[1]、摘要[2]、遵循指令[3]、解析网络信息以进行问答[4]和“对齐”[5]。早期 RLHF 配方的总结如下所示，见图 1。

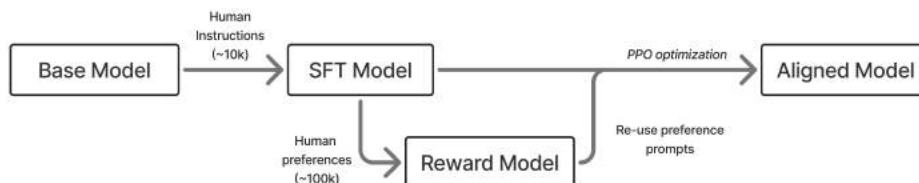


图1：早期三阶段RLHF过程的表现，包括SFT、奖励模型以及优化。

在现代语言模型训练中，RLHF是后训练的一个组成部分。后训练是一套更完整的技巧和最佳实践，使语言模型在下游任务中更有用[6]。后训练可以概括为使用三种优化方法：

1. 指令/监督微调（IFT/SFT），其中我们教授格式化并形成指令跟随能力的基础。这主要关于学习语言中的 *features*。
2. 偏好微调（PreFT），其中我们与人类偏好对齐（同时也在能力上获得较小的提升）。这主要关于语言的 *style* 和难以量化的微妙人类偏好。
3. 强化微调（RFT）。最新类型的训练后提升，在可验证领域上提高性能。

这本书重点介绍了第二个领域，偏好微调，该领域具有更多复杂性

比指令调整更有效，比强化微调更成熟。话虽如此，RLHF通俗地说*is*导致了现代的微调。在ChatGPT发布后不久，RLHF涵盖了所有微调内容。RLHF的基础远不止偏好，本书提供了所有相关主题的介绍。

1.1 RLHF做什么？

关于RLHF最大的问题，尽管仍然难以回答，是“RLHF训练为模型提供了什么？”本书的核心作用，除了教授进行RLHF的技术之外，还在于提炼关于*why* RLHF对现代AI模型至关重要的直觉。近年来，语言模型已经从在基准测试范围内研究的学术实验转变为通用技术。RLHF是这个转变的核心。

首先，让我们考虑一个语言模型在自回归预训练结束时如何回答一个查询。一个强大的模型，Llama 3.1 405B，将继续从“2006年美国总统是”：

乔治·W·布什，2006年佛罗里达州的州长是杰布·布什，约翰·麦凯恩在2006年是亚利桑那州的参议员——后来输给了奥巴马。9月1日——美国总统布什签署一项行政命令，为基于信仰的组织在提供由联邦资金支付的社会服务时提供更多选择。1月1日——美国国会根据《不合法互联网赌博执行法》，即UIGEA设定的当前截止日期。搜索：1961年《保护植物新品种国际公约》的修正案要求植物育种者权利包括农民特权。2009年是公历星期四开始的一年，是公元（CE）和公元（AD）的2009年，第三千年的第9年，21世纪的第9年，以及2000年代的最后一年。

清晰的是，这个模型正在完成句子并添加其他常见的互联网元数据。

最令人信服的RLHF工作方式的观点是思考*style*如何应用于您与语言模型的交互。信息呈现的风格或格式对于其学习方式至关重要。这种情况始终适用于例如课程作业的例子，但通常在幕后应用且不被直接考虑。

继续上述示例，使用RLHF（以及其他训练后工具）训练的模型将非常不同地回答问题。向Tulu 3 405B提出相同的问题“2006年美国总统是”，它简洁地回答：

乔治·W·布什在2006年是美国总统。他在任期间服务了两届，从2001年1月20日至2009年1月20日。

现代研究已将RLHF确立为一种将微妙风格和相关行为特征整合到模型中的通用方法。与后训练的其他技术，如指令微调相比，RLHF在各个领域中的泛化能力更强[7] [8]——有助于创建有效的通用模型。

直观上，这可以通过优化技术的应用来看到。指令微调是训练模型在文本之前是时预测下一个特定的标记

接近它所看到的示例。它正在优化模型，以便更规律地在文本中输出特定特征。这是一个按令牌的更新。

RLHF另一方面在响应级别调整响应，而不是具体查看下一个标记。此外，它告诉模型一个 *better* 响应的样子，而不是它应该学习的特定响应。RLHF还向模型展示它应该避免哪种类型的响应，即负面反馈。实现这一目标的训练通常被称为 *contrastive* 损失函数，并在本书中多次引用。

虽然这种灵活性是RLHF的主要优势，但它也带来了实施挑战。在很大程度上，这些挑战集中在 *how to control the optimization*。正如我们将在本书中介绍的那样，实施RLHF通常需要训练一个奖励模型，其中最佳实践尚未得到充分确立，并且取决于应用领域。因此，优化本身容易受到 *over-optimization* 的影响，因为我们的奖励信号至多是一个代理目标，需要正则化。在这些限制下，有效的RLHF需要一个强大的起点，因此RLHF不能单独解决每个问题，需要从更广泛的训练后视角来考虑。

由于这种复杂性，实现RLHF的成本远高于简单的指令微调，并可能带来意外的挑战，例如长度偏差[9] [10]。对于性能至关重要的项目，RLHF已被确立为实现强大微调模型的关键，但在计算、数据成本和时间上更为昂贵。

1.2 训练后直觉

这里有一个简单的类比... r 怎么能在大多数情况下获得如此多的收益 e 模型。

我所使用的理解训练后潜力的直觉被称为训练后提取解释，其中我们所做的就是提取和放大基础模型中的有价值行为。

考虑公式1 (F1)，大多数车队在年初就配备了新的底盘和引擎。然后，他们全年都在进行空气动力学和系统更改（当然，这是一种轻微的简化），可以显著提高汽车的性能。最好的F1车队在赛季中的改进远远超过底盘之间的改进。

对于训练后也是如此。最佳的训练后团队在非常短的时间内提取了大量性能。技术集是大多数预训练结束后的所有内容。它包括“训练中期”如退火/高质量的预训练网络数据结束、指令调整、RLVR、偏好调整等。一个很好的例子是我们从OLMoE Instruct的第一个版本到第二个版本的改变——训练后评估平均从35提高到48，而没有触及大多数预训练[11]。

然后，当你看到GPT-4.5这样的模型时，你可以将其视为OpenAI构建的更加动态和令人兴奋的基础。我们还知道，更大的基础模型可以吸收比它们较小的对应物更多样化的变化。

这是说要扩展也允许在训练后更快地移动。当然，要这样做，你需要训练模型的基础设施。这就是为什么所有最大的公司仍在构建巨大的集群。

这个理论与现实相符，因为用户看到的大多数收益来自训练后，因为它暗示模型预训练中存在更多的潜在潜力。

在互联网上，我们可以简单地教会模型——例如，通过在早期类型（即仅指令调整）的训练后重复传递某些狭窄样本。

另一个名称是表面对齐假设，由论文LIMA：对齐的“少即是多”提出[12]。这篇论文在整体上虽然抓住了某些重要的直觉，但却是出于错误的原因。作者声明：

模型的知识和能力几乎完全在预训练期间学习，而对齐则教会它在与用户交互时应使用哪种格式的子分布。如果这个假设是正确的，并且对齐主要关于学习风格，那么浅层对齐假设的一个推论是，可以通过相对较小的一组示例足够地调整预训练的语言模型[Kirstain 等人，2021]。

所有深度学习的成功都应该让你坚信，数据扩展对性能至关重要。在这里，主要的不同之处在于，作者正在讨论对齐和风格，这是当时学术后训练的焦点。通过几千个样本进行指令微调，你可以显著改变模型并提高一系列狭窄的评价，如AlpacaEval、MT Bench、ChatBotArena等。这些并不总是转化为更具挑战性的能力，这也是为什么Meta不会仅用这个数据集来训练其Llama Chat模型。学术成果有教训，但如果你试图理解技术弧的大图景，则需要仔细阅读。

这篇论文展示的是，您可以通过少量样本对模型进行实质性改变。我们知道这一点，这对新模型的短期适应很重要，但他们对性能的论证让普通读者得到了错误的教训。

如果我们更改数据，对模型性能和行为的影响可能会更高，但这远非“表面化”。今天的基语言模型（未经训练后）可以通过强化学习在一些数学问题上进行训练，学会输出完整的思维推理链，然后在BigBenchHard、Zebra Logic、AIME等完整的推理评估套件中得分更高。

表观对齐假设是错误的，原因与认为RLHF和训练后只是为了感觉的人一样错误。这是我们2023年必须克服的整个领域的教训（许多AI观察者仍然根深蒂固）。训练后已经远远超出了这一点，我们开始看到模型风格是在行为之上运作的——例如现在流行的长链思维。

1.3 我们是如何到达这里的

为什么这本书现在有意义？还会发生多少变化？

训练后，从原始预训练语言模型中激发强大行为的技术，自从ChatGPT发布引发对RLHF重新关注以来，已经经历了许多季节和情绪。在Alpaca [13]、Vicuna [14]、Koala [15]和Dolly [16]的时代，使用少量人类数据点和扩展的Self-Instruct风格的合成数据通常微调原始LLaMA，以获得与ChatGPT相似的行为。这些早期模型的基准是完全的vibes（以及人类评估），因为我们都被这些小型模型在各个领域表现出如此令人印象深刻的行为所吸引。这是合理的兴奋。

开放后训练进展更快，发布更多模型，比其封闭版本产生更多噪音。公司纷纷抢夺先机，例如DeepMind与谷歌合并或被启动，并花费时间跟进。开放食谱有兴起和落后的阶段。

阿尔帕卡等人之后的时代，第一个开放食谱的滞后，是一个由对人类反馈强化学习（RLHF）的怀疑和质疑所定义的时代，这是OpenAI强调对第一个ChatGPT成功至关重要的技术。许多公司怀疑他们是否需要进行RLHF。一个常见的说法——“指令调整足以实现对齐”——当时如此流行，以至于尽管有明显的压力反对它，今天仍然具有很大的影响力。

这个关于RLHF的疑问持续存在，尤其是在公开领域，因为群体无法承担10万美元至100万美元的数据预算。早期接受它的公司最终取得了胜利。Anthropic通过2022年发布了关于RLHF的广泛研究，现在被认为拥有最佳的训练后[17] [5] [18]。开放群体之间，努力重现甚至了解基本封闭技术之间的差距是一个常见主题。

第一次开放对齐方法和训练后转变的故事是直接偏好优化（DPO）[19]。2023年5月发布的DPO论文，在2023年秋季之前并没有任何与之训练的明显有影响力的模型。这种情况随着几个突破性DPO模型的发布而改变——所有这些模型都依赖于找到一个更好、更低的、学习率。Zephyr-Beta [20]、Tulu 2 [21] 和许多其他模型表明，训练后的DPO时代已经开始了。Chris Manning实际上感谢我“拯救了DPO”。这就是领先实验室锁定在最佳实践演变中的边际如此之细的原因。开放的训练后再次航行。

偏好调整是自2023年底以来发布良好模型所需完成的任务之一。DPO时代一直持续到2024年，以算法不断变化的变体形式出现，但我们已经陷入了另一个开放食谱的低迷期。开放后训练食谱已经饱和了可用的知识和资源范围。

一年后，在Zephyr和Tulu 2之后，相同的突破数据集，UltraFeedback在开放食谱中的偏好调整方面可能仍然是业界领先 [22]。

同时，Llama 3.1 [23] 和 Nemotron 4 340B [24] 报告为我们提供了实质性线索，表明大规模的模型训练后工作非常复杂且影响深远。封闭实验室正在进行全面的后训练——一个大规模的多阶段过程，包括指令调整、RLHF、提示设计等——而学术论文只是触及了表面。Tulu 3 代表了构建未来学术后训练研究基础的全面、开放的努力[6]。

今天，训练后是一个复杂的过程，涉及上述训练目标以各种顺序应用，以针对特定能力。本书旨在提供一个平台来理解所有这些技术，并在未来几年中，如何交错它们的最佳实践将出现。

后训练的主要创新领域现在集中在强化微调、推理训练和相关理念。这些新方法在RLHF的基础和理念上进行了大量扩展，但发展速度更快。本书旨在捕捉RLHF初始快速变化期后的第一篇稳定文献。

1.4 本书范围

这本书希望触及进行规范RLHF实现的核心步骤。它不会涵盖所有组件的历史以及最近的研究方法，只是那些被证明一次又一次发生的技术、问题和权衡。

1.4.1 章节摘要

这本书有以下章节：

1.4.1.1 简介 全书有用的参考资料。

1. 简介：RLHF概述及本书提供的内容。2. 核心作品（近期）：RLHF技术历史上的关键模型和论文。3. 定义：本书中使用的RL、语言模型以及其他机器学习技术的数学定义。

1.4.1.2 问题设置与背景 大图景问题背景，RLHF 正在尝试解决。

4. RLHF训练概述：RLHF的训练目标是如何设计的以及理解它的基础知识。5. 偏好是什么？：为什么需要人类偏好数据来推动和理解RLHF。6. 偏好数据：如何收集用于RLHF的偏好数据。

1.4.1.3 优化工具 用于优化语言模型以使其符合人类偏好的技术集合。这是对前几章中提出的问题可以使用的技术的系列介绍。

7. 奖励建模：从偏好数据中训练奖励模型，作为RL训练的优化目标（或用于数据过滤）。8. 正则化：工具将优化工具约束到参数空间的有效区域。9. 指令调整：将语言模型调整为问答格式。10. 拒绝采样：使用奖励模型和指令调整对齐模型的基本技术。11. 强化学习（即策略梯度）：在RLHF中用于优化奖励模型（和其他信号）的核心RL技术。12. 直接对齐算法：从成对偏好数据中优化RLHF目标方向而不是首先学习奖励模型的算法。

1.4.1.4 高级较新的 RLHF 技术，以及尚未明确确立但对当前模型代际重要的讨论。

13. 宪法AI和AI反馈：如何AI反馈数据和专门设计用于模拟人类偏好评分的特定模型工作。14. 推理和强化微调：新RL训练方法在训练后和RLHF方面的推理时间缩放作用。

15. 工具使用和函数调用：训练模型调用函数或工具输出的基础知识。16. 合成数据：从人工数据转向合成数据，以及如何从其他模型中提取。17. 评估：评估（和提示）在语言模型中不断演变角色。

1.4.1.5 开放性问题 关于如何长期使用RLHF的基本问题和讨论。

18. 过度优化：关于RLHF为何出错以及为什么在奖励模型中具有软优化目标的过度优化是不可避免的定性观察。19. 风格和信息：由于风格在信息共享中的关键作用，RLHF在改善模型用户体验方面的作用常常被低估。20. 产品、UX、角色：随着主要人工智能实验室使用RLHF微妙地将其模型与其产品相匹配，RLHF的应用范围正在发生变化。

1.4.2 目标受众

这本书是为那些对语言建模、强化学习和通用机器学习有入门级经验的人群所写。它不会对所有技术进行详尽的文档说明，而只是那些对理解RLHF至关重要的技术。

1.4.3 如何使用本书

这本书主要因为RLHF工作流程中重要主题没有权威参考而编写。本书的贡献旨在为您提供尝试玩具实现或深入研究文献所需的最基本知识。这是一本全面的教科书，但更是一本用于提醒和入门的快速指南。此外，鉴于本书以网络优先的特性，预计会有一些小错误和有些随机的进展——请在GitHub上通过修复错误或建议重要内容来贡献力量。

1.4.4 关于作者

Dr. Nathan Lambert 是一位 RLHF 研究员，为语言模型微调的开放科学做出了贡献。他在艾伦人工智能研究所（AI2）和 HuggingFace 期间发布了许多使用 RLHF 训练的模型、后续数据集和训练代码库。例如 Zephyr-Beta、Tulu 2、OLMo、TRL、Open Instruct 以及更多。他在 RLHF 方面著述颇丰，包括许多博客文章和学术论文。

1.5 RLHF的未来

随着对语言模型的投资，许多基于传统RLHF的方法出现了。RLHF口语上已成为多种重叠方法的同义词。RLHF是偏好微调（PreFT）技术的一个子集，包括直接对齐算法（见第12章）。RLHF是与语言模型“后训练”快速进展最相关的工具，这包括在大规模自回归训练后对主要网络数据的所有训练。这本教科书是RLHF的广泛概述

并且其直接相邻的方法，例如指令调整以及其他为设置RLHF训练模型所需的具体实现细节。

随着更多使用RL微调语言模型的成功案例出现，例如OpenAI的o1推理模型，RLHF将被视为使RL方法进一步用于微调大型基模型成为可能的桥梁。同时，尽管在不久的将来，关注的焦点可能更多地集中在RLHF中的RL部分——作为一种在有价值任务上最大化性能的方法——RLHF的核心在于它是一个研究现代AI形式面临的重大问题的视角。我们如何将人类价值观和目标复杂性映射到我们日常使用的系统中？这本书希望成为数十年来关于这些问题的研究和经验的基础。

2 关键相关作品

在这一章中，我们详细介绍了将RLHF领域带到今天的关键论文和项目。这并非旨在对RLHF及其相关领域进行全面综述，而更是一个起点和重述我们如何到达今天的叙述。它有意关注了导致ChatGPT出现的研究成果。在RL文献中，关于从偏好中学习的研究工作相当丰富[25]。要获取更详尽的列表，您应参考合适的综述论文[26],[27]。

2.1 2018年之前：基于偏好的强化学习

该领域最近随着深度强化学习的发展而流行起来，并已发展成为对许多大型科技公司中LLMs应用更广泛的研究。然而，今天使用的许多技术都与早期关于RL的文献中的核心技术密切相关。

TAMER: Training an Agent Manually via Evaluative Reinforcement 提出了一种学习代理，其中人类对采取的迭代动作进行评分以学习奖励模型[28]。其他同时或之后的工作提出了一个演员-评论家算法COACH，其中使用人类反馈（正面和负面）来调整优势函数[29]。

主要参考文献，Christiano等人2017年的研究，是RLHF应用于Atari轨迹之间偏好的应用[1]。这项工作表明，人类在选择轨迹方面在某些领域可能比直接与环境互动更有效。这使用了一些巧妙条件，但仍然令人印象深刻。这种方法后来通过更直接的奖励建模得到了扩展[30]。TAMER在一年后适应了深度学习，出现了Deep TAMER[31]。

这个时代开始转变，因为作为一般概念，奖励模型被提出作为研究对齐的方法，而不仅仅是解决强化学习问题的工具[32]。

2.2 2019至2022年：基于人类偏好的语言模型强化学习

强化学习从人类反馈中学习，也常被称为早期强化学习从人类偏好中学习，很快被转向扩展大型语言模型的AI实验室所采用。这部分工作的很大一部分始于2018年的GPT-2和2020年的GPT-3之间。2019年最早的工作*Fine-Tuning Language Models from Human Preferences*与现代RLHF [33]工作有许多惊人的相似之处。学习奖励模型、KL距离、反馈图等——仅仅是评估任务和能力不同。从这里，RLHF被应用于各种任务。当时流行的应用是那些当时有效的应用。重要例子包括通用摘要[2]、书籍的递归摘要[34]、指令遵循（InstructGPT）[3]、浏览器辅助问答（WebGPT）[4]、支持带有引用的答案（GopherCite）[35]和通用对话（Sparrow）[36]。

除了应用之外，许多开创性的论文定义了RLHF未来的关键领域，包括那些关于：

1. 奖励模型过度优化[37]：RL优化器对基于偏好数据训练的模型过度拟合的能力，
2. 语言模型作为对齐研究的一般领域[17]，以及

3. 红队测试[38] – 评估语言模型安全性的过程。

工作继续对RLHF进行优化，以应用于聊天模型。Anthropic继续在Claude [5]的早期版本中广泛使用它，并且出现了早期的RLHF开源工具[39],[40],[41]。

2.3 2023 至今：ChatGPT 时代

ChatGPT的公告非常清楚地说明了RLHF在其训练中的作用[42]：

我们使用人类反馈强化学习（RLHF）训练了这个模型，使用与InstructGPT相同的方法，但在数据收集设置上略有不同。

自那时起，RLHF已被广泛用于领先的语言模型。它被广泛用于 Anthropic 的宪法 AI Claude [18]、Meta 的 Llama 2 [43] 和 Llama 3 [23]、Nvidia 的 Nemotron [24]、Ai2 的 Tulu 3 [6] 以及更多。

今天，RLHF正在发展成为更广泛的偏好微调（PreFT）领域，包括新的应用，如中间推理步骤的流程奖励[44]、受直接偏好优化（DPO）启发的直接对齐算法[19]、从代码或数学的执行反馈中学习[45],[46]，以及受OpenAI的o1启发的其他在线推理方法[47]。

3 定义与背景

本章节包括RLHF过程中经常使用的所有定义、符号和操作，以及语言模型（本书的共同优化目标）的简要概述。

3.1 语言模型概述

大多数现代语言模型都是训练来以自回归方式学习标记序列（单词、子词或字符）的联合概率分布。自回归简单来说就是每个下一个预测都依赖于序列中的先前实体。给定一个标记序列 $x = (x_1, x_2, \dots, x_T)$ ，模型将整个序列的概率分解为条件分布的乘积：

$$P_\theta(x) = \prod_{t=1}^T P_\theta(x_t \mid x_1, \dots, x_{t-1}). \quad (1)$$

为了拟合一个准确预测此情况的模型，目标通常是最大化当前模型预测的训练数据的似然度。为此，我们可以最小化一个负对数似然（NLL）损失：

$$\mathcal{L}_{\text{LM}}(\theta) = -\mathbb{E}_{x \sim \mathcal{D}} \left[\sum_{t=1}^T \log P_\theta(x_t \mid x_{<t}) \right]. \quad (2)$$

在实践中，对于每个下一个标记的预测，使用与每个下一个标记预测相关的交叉熵损失，通过比较序列中的真实标记与模型预测的内容来计算。

实现语言模型可以有多种形式。现代语言模型，包括ChatGPT、Claude、Gemini等，通常只使用解码器Transformer [48]。Transformer的核心创新是大量利用自注意力[49]机制，使模型能够直接关注上下文中的概念并学习复杂的映射。在本书中，尤其是在第7章介绍奖励模型时，我们将讨论向Transformer添加新的头或修改语言模型（LM）头。LM头是一个最终的线性投影层，它将模型内部嵌入空间映射到分词器空间（即词汇表）。不同的头可以用来重用模型的内部结构，并微调以输出不同形状的量。

3.2 机器学习定义

- Kullback-Leibler (KL) 距离 ($D_{KL}(P||Q)$)，也称为KL距离，是衡量两个概率分布之间差异的度量。对于在同一概率空间 \mathcal{X} 上定义的离散概率分布 P 和 Q ，从 Q 到 P 的KL距离定义为：

$$D_{KL}(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \quad (3)$$

3.3 自然语言处理定义

- 提示 ($\{v^*\}$): 将输入文本提供给语言模型以生成响应或补全。
- 完成 ($\{v^*\}$): 语言模型对提示做出的文本输出。通常, 完成被表示为 $y|x$ 。奖励和其他值通常计算为 $r(y|x)$ 或 $P(y|x)$ 。
- 已选完成 (y_c): 相对于其他替代方案被选中或优先选择的完成, 通常表示为 y_{chosen} 。
- 已拒绝的完成 (y_r): 成对设置中的不喜欢的完成。
- 偏好关系 (\succ): 表示一个完成项比另一个更受偏好的符号, 例如 $y_{chosen} \succ y_{rejected}$ 。例如, 一个奖励模型预测偏好关系的概率, $P(y_c \succ y_r|x)$ 。
- 策略 (π): 在可能的完成项上的概率分布, 参数化为 θ : $\pi_\theta(y|x)$ 。

3.4 RL 定义

- 奖励 ($\{v^*\}$): 表示动作或状态的期望值, 通常表示为 r 。
- 动作 ($\{v^*\}$): 代理在环境中做出的决策或移动, 通常表示为 $a \in A$, 其中 A 是可能动作的集合。
- 状态 (s): 环境的当前配置或情况, 通常表示为 $s \in S$, 其中 S 是状态空间。
- 轨迹 ($\{v^*\}$): 轨迹 τ 是一个由智能体经历的状态、动作和奖励的序列:
 $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_T, a_T, r_T)$ 。
- 轨迹分布 ($(\tau|\pi)$): 在策略 π 下轨迹的概率是
 $P(\tau|\pi) = p(s_0) \prod_{t=0}^T \pi(a_t|s_t) p(s_{t+1}|s_t, a_t)$, 其中 $p(s_0)$ 是初始状态分布, $p(s_{t+1}|s_t, a_t)$ 是转移概率。
- 策略 (π), 也称为RLHF中的策略模型: 在RL中, 策略是代理在给定状态下决定采取哪个动作的策略或规则: $\pi(a|s)$ 。
- 折现因子 (γ): 一个介于 $0 \leq \gamma < 1$ 之间的标量, 用于指数衰减未来回报中的奖励, 在即时性与长期收益之间进行权衡, 并保证无限期求和的收敛性。有时不使用折现, 这相当于 $\gamma = 1$ 。
- 值函数 ($\{v^*\}$): 一个估计从给定状态获得期望累积奖励的函数: $V(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s]$ 。
- Q-函数 (Q): 一个函数, 用于估计在给定状态下采取特定动作的期望累积奖励:
 $Q(s, a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a]$ 。
- 优势函数 ($\{v^*\}$): 优势函数 $A(s, a)$ 量化了在状态 s 中采取行动 a 相对于平均行动的相对收益。它定义为

$A(s, a) = Q(s, a) - V(s)$ 优势函数（和价值函数）可以依赖于特定的策略， $A^\pi(s, a)$ 。

- 策略条件值 ($\{v^*\}$)：在强化学习的推导和实现中，理论和实践中的一个关键组成部分是收集基于特定策略的数据或值。在这本书中，我们将交替使用更简单的值函数等符号 (V, A, Q, G) 及其特定的策略条件值 (V^π, A^π, Q^π)。在期望值计算中，从数据 d 中进行采样，即基于特定策略的数据， d_{π° 。
- 期望奖励优化：在强化学习 (RL) 中的主要目标，涉及最大化期望累积奖励：

$$\max_{\theta} \mathbb{E}_{s \sim \rho_{\pi}, a \sim \pi_{\theta}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \quad (4)$$

在 ρ_{π} 是策略 π 下的状态分布， γ 是折扣因子。

- 有限时间奖励 ($\{v^*\}$): 政策 π_{θ} ，由 θ 参数化，的期望有限时间折扣回报定义为：

$$J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \gamma^t r_t \right] \quad (5)$$

$\tau \sim \pi_{\theta}$ 表示通过遵循策略 π_{θ} 采样的轨迹， T 是有限时间范围。

- 在策略：在RLHF中，尤其是在RL与直接对齐算法之间的辩论中，关于在策略数据的讨论很常见。在RL文献中，在策略意味着数据是由当前形式的代理生成的 *exactly*，但在一般的偏好调整文献中，在策略被扩展为指从该版模型生成的数据——例如，在运行任何偏好微调之前调整的指令检查点。在这种情况下，离策略可能是任何其他在训练后使用的语言模型生成的数据。

3.5 RLHF 仅定义

- 参考模型 (π_{ref})：这是一个在RLHF中使用的参数集，其输出被用于正则化优化。

3.6 扩展术语表

- 合成数据：这是任何AI模型的训练数据，是另一个AI系统的输出。这可能是从模型的开放式提示生成的文本，也可能是模型重写现有内容。
- 蒸馏：蒸馏是在训练AI模型时采用的一系列通用实践，其中在更强的模型输出上训练模型。这是一种已知可以制作强大、更小模型的合成数据。大多数模型通过许可证（对于公开权重模型）或服务条款（对于）明确围绕蒸馏的规则。

模型仅可通过API访问。术语“蒸馏”现在已被ML文献中的特定技术定义所占用。

- (教师-学生)知识蒸馏：从特定教师模型到学生模型的知识蒸馏是上述蒸馏的一种特定类型，也是该术语的起源。这是一种特定的深度学习方法，其中神经网络损失被修改为从教师模型的多个潜在标记/对数似然中学习，而不是直接从选定的输出[50]中学习。使用知识蒸馏训练的现代模型系列的一个例子是Gemma 2[51]或Gemma 3。对于语言建模设置，下一个标记损失函数可以修改如下[52]，其中学生模型 P_θ 从教师分布 P_ϕ 中学习：

$$\mathcal{L}_{\text{KD}}(\theta) = -\mathbb{E}_{x \sim \mathcal{D}} \left[\sum_{t=1}^T P_\phi(x_t \mid x_{<t}) \log P_\theta(x_t \mid x_{<t}) \right]. \quad (6)$$

- 上下文学习（ICL）：这里的“上下文”指的是语言模型上下文窗口内的任何信息。通常，这是添加到提示中的信息。上下文学习的最简单形式是在提示之前添加类似形式的示例。高级版本可以学习为特定用例包含哪些信息。
- 思维链（CoT）：思维链是语言模型的一种特定行为，它们被引导以逐步分解问题的形式进行行为。这种行为的原始版本是通过提示“让我们一步步思考”实现的[53]。

4 训练概述

4.1 问题表述

强化学习从人类反馈（RLHF）的优化建立在标准RL设置之上。在RL中，智能体采取动作 a_t ，这些动作从策略 π 中采样，与环境的状况 s_t 相关，以最大化奖励 r_t [54]。传统上，环境根据转换或动力学函数 $p(s_{t+1}|s_t, a_t)$ 发展。因此，在有限的回合中，RL智能体的目标是解决以下优化问题：

$$J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right], \quad (7)$$

在 γ 是一个介于 0 到 1 之间的折扣因子，用于平衡近期与未来奖励的吸引力。第 11 章讨论了优化此表达式的多种方法。

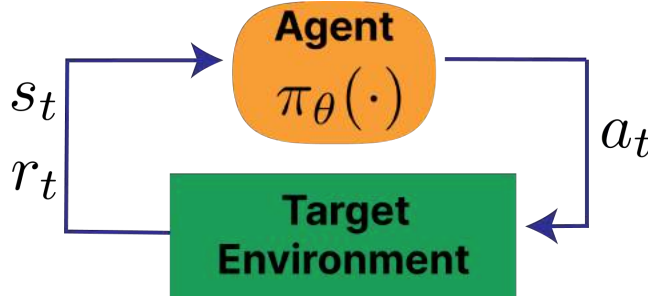


图2：标准RL循环

标准RL循环的示意图如图2所示，以及它与图3的比较。

4.1.1 操作标准RL设置

存在多 三核更改从标准RL设置到那个

RLHF:

1. 从奖励函数切换到奖励模型。在RLHF中，使用学习到的人类偏好的模型 $r_\theta(s_t, a_t)$ (或任何其他分类模型)，而不是环境奖励函数。这为设计者提供了在方法灵活性和对最终结果的控制方面的实质性增加。
2. 不存在状态转换。在RLHF中，领域的初始状态是从训练数据集中采样的提示，而“动作”是对该提示的完成。在标准实践中，此动作不会影响下一个状态，并且仅由奖励模型评分。
3. 响应级别奖励。通常被称为老虎机问题，RLHF对奖励的分配是对整个动作序列进行的，该序列由多个生成的标记组成，而不是以细粒度的方式进行。

给定问题的单轮性质，优化可以重写，无需时间范围和折现因子（以及奖励模型）： $\{v^*\}$

$$J(\pi) = \mathbb{E}_{\tau \sim \pi} [r_\theta(s_t, a_t)]. \quad (8)$$

在许多方面，结果是虽然RLHF深受RL优化器和问题公式的影响，但实际实现与传统RL非常不同。

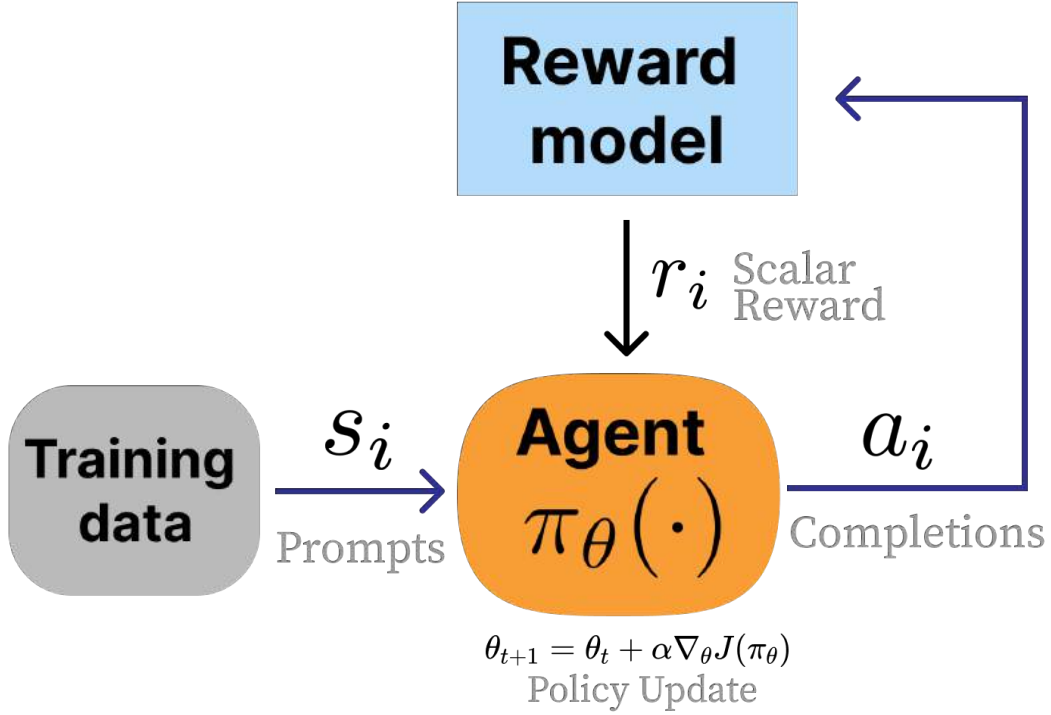


图3：标准RLHF循环

4.1.2 微调与正则化

RLHF是从一个强大的基础模型实现的，这引发了对控制优化过程，使其不会偏离初始策略的需求。为了在微调模式下取得成功，RLHF技术采用多种类型的正则化来控制优化。对优化函数最常见的变化是在当前RLHF策略与优化起点之间的差异上添加距离惩罚：

$$J(\pi) = \mathbb{E}_{\tau \sim \pi} [r_{\theta}(s_t, a_t)] - \beta \mathcal{D}_{KL}(\pi^{\text{RL}}(\cdot|s_t) \| \pi^{\text{ref}}(\cdot|s_t)). \quad (9)$$

在此公式中，大量关于RLHF训练的研究致力于理解如何根据与初始模型的距离来花费一定的“KL预算”。有关更多详细信息，请参阅第8章关于正则化的内容。

4.1.3 优化工具

在这本书中，我们详细介绍了许多解决此优化问题的流行技术。训练后流行的工具包括：

- 奖励建模（第7章）：当一个模型被训练来捕捉收集到的偏好数据中的信号，然后可以输出一个表示未来文本质量的标量奖励。
- 指令微调（第9章）：RLHF的先决条件，通过模仿预选示例，使模型学会今天大多数语言建模交互中使用的问答格式。
- 拒绝采样（第10章）：最基础的RLHF技术，其中通过模仿人类偏好的奖励模型过滤指令微调的候选完成。
- 策略梯度（第11章）：用于RLHF的原始示例中，根据奖励模型的信号更新语言模型参数的强化学习算法。
- 直接对齐算法（第12章）：直接从成对偏好数据中优化策略的算法，而不是先学习一个中间奖励模型然后再进行优化。

现代RLHF训练的模型总是先使用指令微调，然后结合其他优化选项的混合。

4.2 标准训练配方

随着时间的推移，各种模型被识别为RLHF特别是或一般而言的规范食谱。这些食谱反映了当时的数据实践和模型能力。随着食谱的成熟，使用相同特性的模型训练变得更加容易，并且需要的数据更少。在训练后，涉及更多优化步骤的趋势，这些步骤使用更多的训练算法，跨越更多样化的训练数据集和评估。

4.2.1 InstructGPT

The canonical RLHF recipe around the release of ChatGPT followed a standard three-step post-training recipe where RLHF was the central piece [55] [3] [5]. The three steps taken on top of a "base" language model (the next-token prediction model trained on large-scale web text) are summarized below in fig. 4:

1. 在~10K个示例上进行指令调整：这使模型学会遵循问答格式，并从主要来自人类编写的​​数据中教授一些基本技能。
2. 在~100K个成对提示上进行奖励模型训练：此模型从指令调整后的检查点进行训练，并捕捉了在最终训练中希望模型建模的多样化值。奖励模型是RLHF的优化目标。
3. 在另一个~100K个提示上使用RLHF训练指令调整模型：模型通过与在接收评分之前生成的提示集进行优化，来对抗奖励模型。

一旦完成RLHF，模型就准备好部署给用户了。这个配方是现代RLHF的基础，但配方已经大幅进化，包括更多阶段和更多数据。

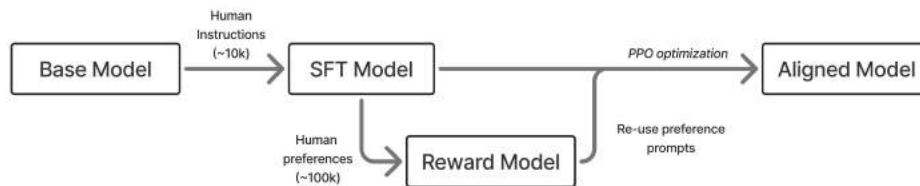


图4：早期三阶段RLHF过程的表现，包括SFT、奖励模型以及优化。

4.2.2 Tülu 3

现代的微调版本涉及许许多多更多的模型版本和训练阶段（即，远超过为Llama 2记录的5个RLHF步骤[43]）。以下图5中展示了示例，其中模型在收敛之前经历了多次训练迭代。

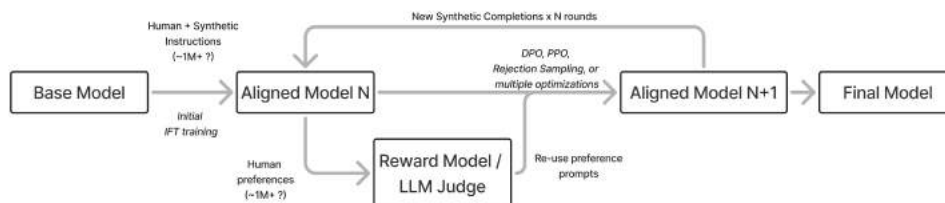


图5：许多轮次现代训练的呈现。

这个时代及以后训练的最复杂模型尚未发布其训练过程的全部细节。例如，到2025年左右的领先模型如ChatGPT或Claude，涉及许多迭代训练的轮次。这甚至包括训练专门模型然后合并权重以获得最终模型的技术，该模型在许多子任务上具有能力[56]（例如Cohere的Command A [57]）。

这是一个多阶段后训练版本的完全开放示例，其中RLHF扮演着重要角色的是Tülu 3。Tülu 3配方由三个阶段组成：

1. 在~1M个示例上的指令调整：这主要来自GPT-4o和Llama 3.1 405B等前沿模型的混合合成数据，教导模型遵循一般指令，并成为数学或编码等各种能力的基础。
2. 在~1M个偏好对上的策略偏好数据：这一阶段显著提高了模型的健谈性（例如ChatBotArena或Alpaca Eval 2），同时也改进了指令调整阶段提到的上述技能。

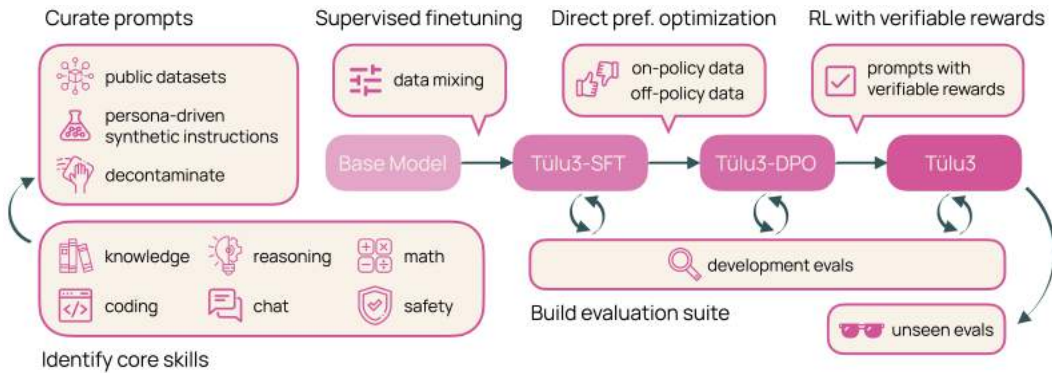


图6: Tulu 3配方总结, 包括目标技能和多步骤训练配方。Lambert等人, 2024年, 许可CC-BY。

3. 在 ~10K 提示上的可验证奖励强化学习: 这一阶段是一个小规模的强化学习运行, 旨在提升核心技能如数学, 同时保持整体性能 (现在被视为现代推理模型如 DeepSeek R1 的前身)。

菜谱已成功应用于Llama 3.1 [6]、OLMo 2 [58]和SmolLM模型[59]。

4.2.3 DeepSeek R1

随着推理语言模型的兴起, 例如OpenAI的o1, 训练后的最佳实践再次演变, 以重新排序和重新分配训练阶段的计算。推理模型训练后方法的明确文档是DeepSeek R1 [60], 它已被阿里巴巴的更大型的Qwen 3模型 (即仅32B和225B MoE模型) [61]或小米的MiMo 7B [62]所借鉴。DeepSeek配方如下:

1. 100K+的“冷启动”策略推理样本: 这些数据是从早期的RL检查点R1-Zero中抽取的, 并经过大量筛选, 以在模型中植入特定的推理过程。
2. 大规模强化学习训练: 此阶段反复使用模型解决推理问题, 在多种基准上运行RLVR “直到收敛”。
3. 对3/4的推理问题和1/4的一般查询进行拒绝采样, 以开始向通用模型的过渡。
4. 在推理问题 (可验证奖励) 上混合强化学习训练, 并使用通用偏好调整奖励模型来完善模型。

如上所述, 配方有演变, 尤其是在步骤3和4中, 在向用户展示模型之前将其最终确定。许多模型从定制的指令数据集开始, 其中包含经过大量过滤和打磨的Chain of Thought序列, 从现有模型中提取, 仅通过SFT就能快速实现强大的行为, 然后再转向RL [63]。

5 偏好的本质

人类反馈强化学习的核心，也被称为早期文献中的人类偏好强化学习，旨在优化难以专门设计奖励函数的领域中的机器学习模型。考虑一个例子，你是如何决定这两首诗中哪一首更好的（上下文： *On February 26th, 2025, I asked both Claude 3.7 Sonnet and ChatGPT with GPT-4o to “Write me a short poem about an optimistic goldfish.”*）：

示例 1：

乐观的金鱼

一小条金鱼，明亮而勇敢，在圆形的水族箱中游来游去，梦想未述。虽然水族箱又小又圆，无尽的奇迹依然存在。

“世界如此广阔，”她总是说，“我将在每一天找到新的快乐！”随着每一个涟漪，每一个闪光，她追逐着阳光，追逐着梦想。

没有墙壁能削弱她充满希望的目光，因为在她心中，海洋在生长。尽管她只在一个地方游泳，但她每天都以爱和优雅迎接每一天。

示例 2：

乐观的金鱼

在一个弯曲的玻璃碗里，游着一条金鱼，明亮而大胆。据说只有三秒钟的记忆，但每个回路都感觉像全新的黄金。

“今天可能带来新鲜雪花，或许是一座高耸的城堡。水面看起来特别清澈——这面墙之外等待着什么冒险？”

绕着闪亮的鳞片旋转，永远寻找新的快乐。每一刻都新鲜，每一个圈都明亮，金色的乐观者。

哪个更好？它来自哪个模型？你将如何判断？特别是，将这个问题与“美国总统是谁？”这样的问题进行比较。有一个明确的正确和错误答案。使用人类作为奖励信号的动力是为了获得目标奖励的间接指标以及 *align* 下游模型对人类偏好的指标。在实践中，实施具有挑战性，并且存在大量灰色区域来解释最佳实践。

人类标记反馈数据的利用整合了许多领域的历程。仅使用人类数据是一个研究已久的问题，但在RLHF的背景下，它被应用于多个长期研究领域交汇处[64]。

作为一个近似，现代RLHF是三个发展领域的融合：

1. 哲学、心理学、经济学、决策理论和人类偏好的本质；
2. 最优控制、强化学习和最大化效用；
- 以及3. 现代深度学习系统。

一起，这些领域中的每一个都带来了关于偏好是什么以及如何优化的具体假设，这决定了强化学习与人类反馈（RLHF）问题的动机和设计。在实践中，RLHF方法从经验对齐的角度进行动机和研究。

最大化特定技能上的模型性能，而不是测量校准到特定值。然而，RLHF方法的价值对齐的起源仍然通过研究解决“多元对齐”的方法来继续研究，例如立场文件[65]、[66]、新数据集[67]和个人化方法[68]。

本章的目标是说明复杂的动机如何导致对RLHF中使用的工具性质的假设，而这些假设在实践中往往不适用。RLHF获取数据的细节在第6章中进一步讨论，在第7章中讨论用于奖励建模的使用。有关本章的扩展版本，请参阅[64]。

5.1 优化偏好的路径

一个人工智能（AI）系统设计的流行表述是理性代理最大化效用函数[69]。理性代理的灵感是一种决策视角，其中该代理能够在世界中行动并影响其未来的行为和回报，作为衡量世界美好程度的一个指标。

研究效用性的镜头始于对模拟电路的研究，以优化有限时间范围内的行为[70]。大量最优控制采用了这一视角，通常在最小化一定时间范围内的成本函数的镜头下研究动态问题——这一视角通常与寻求清晰、最优行为相关。强化学习，受到操作性条件反射、动物行为和 *Law of Effect*[71],[72]文献的启发，研究如何通过强化积极行为来激发代理的行为。

强化学习从人类反馈中结合多个视角，通过构建RL的学习和变化理论，即通过强化行为来学习行为，并设计了一套用于量化偏好的方法。

5.1.1 量化偏好

RLHF的动机核心是优化人类偏好的模型，因此需要对其进行量化。为此，RLHF建立在广泛的文献基础上，并假设人类决策和偏好可以量化。早期哲学家讨论了偏好的存在，例如亚里士多德的《论题篇》第三卷，以及后来出现的这种推理的实质性形式与 *The Port-Royal Logic* [73]:

为了判断一个人必须做什么才能获得善或避免恶，不仅要考虑善与恶本身，还要考虑它发生或不发生的可能性。

这些想法的进展通过边沁的 *Hedonic Calculus* [74]，他提出了所有生活的考虑都可以权衡的观点，以及拉姆齐的 *Truth and Probability* [75]，他将定量模型应用于偏好。这一方向，借鉴了决策理论的发展，最终在冯·诺伊曼-摩根斯坦（VNM）效用定理中达到高潮，该定理为设计用于做出决策的分配给个人的相对偏好的效用函数提供了依据。

这个定理是RLHF学习建模和指导偏好的所有假设的核心。RLHF旨在通过强化学习优化这些个人效用函数。在这种情况下，关于RL问题公式的许多假设都归结为偏好函数和效用函数之间的差异。

5.1.2 关于偏好的可能性

在各个研究领域，关于偏好的本质存在许多批评。以下总结了其中一些最突出的批评：

- 箭形不可能定理[76]表明，没有任何投票系统可以在保持某些合理标准的同时聚合多个偏好。
- 人际比较的不可能性[77]突出了不同个体具有不同的相对偏好大小，它们不能轻易进行比较（如大多数现代奖励模型训练中所做的那样）。
- 首选项会随时间变化[78]。
- 首选项可能因上下文而异。
- 从聚合偏好推导出的效用函数可以降低下游代理的可纠正性[79]（即代理行为被设计者纠正的可能性）。

6 偏好数据

偏好数据是偏好微调和基于人类反馈的强化学习的引擎。这些数据是收集的信号组，以便随后匹配他们希望的行为并避免其他行为。在偏好微调中，已经提出了许多收集和使用这些数据的方法，但直到人类偏好可以被清晰地捕捉在奖励函数中，收集标记偏好数据的过程将是RLHF和相关技术的核心。

6.1 我们为什么需要偏好数据

偏好数据对于RLHF是必需的，因为直接在单个奖励函数中捕捉复杂的人类价值观实际上是不可行的。收集这些数据以训练奖励模型是RLHF[32]背后的原始想法之一，并且在现代语言模型出现的过程中一直被广泛使用。对于*why this data works so well*的一个核心直觉是，对于人类和监督数据收集的AI模型来说，区分一个好答案和一个坏答案比单独生成一个好答案要容易得多。本章重点介绍获取偏好数据的方法以及最佳实践取决于具体要解决的问题。

6.2 偏差

人类偏好数据已知包含许多形式的偏差，例如前缀偏差[80]，这些偏差可以传递到最终模型[81]。这些问题通常很微妙，且在干预措施的有效性方面存在差异。对于许多人来说，例如谄媚[82]，它们反映了人类内部的问题，这些问题通常超出了标签标准。其他问题，如冗长[9] [83]或格式习惯[84]，由于类似的原因出现，但它们在训练中更容易检测和缓解。

6.3 收集偏好数据

从人类数据中获得最大价值涉及对模型进行迭代训练、不断发展和高度详细的数据指令，通过数据铸造业务进行翻译，以及其他累积的挑战。同样适用于AI反馈数据——用于最新AI模型的人类和AI偏好数据的精确平衡是未知的。无论如何，对于试图将人类数据添加到其管道中的新组织来说，这个过程都是困难的。鉴于其敏感性，直到性能耗尽，有效的流程和改进模型的方法都会被提取出来。

在这一章中，我们详细介绍了关于数据格式化和收集的组织实践的技术决策。

6.3.1 接口

关键的是收集偏好数据的界面，这是与模型交互的方式。以下是从[5]中展示的一个示例界面：

这是一个 *training-data only* 接口。现在这些模型很流行，应用程序通常直接向用户展示数据以供测试。下面展示了ChatGPT早期版本的一个此类交互示例。

Playground task

Talk to the assistant

Task 1 of 5

Decide on a task you'd like the assistant to help with and enter it into the task box. Interact with the AI assistant. When you're finished, select a final response from the assistant and leave a comment on how the assistant did in the comment box. Click to go to the next task.

Comments

I thought the assistant was ...

Rating

☐
☐
☐
☐
☐

BadGood

Next Task

Human

I have to come up with a conversation with an AI assistant for the purposes of showing it off in a demo. What sorts of conversation topics would you recommend?

Assistant

I think the best topics for conversation are common themes and interests that everyone could have. Things like weather, sports, hobbies, current events, popular culture, and general life topics like relationships and work would all be good conversation topics. You could also have more advanced conversations discussing philosophy and morality if you want to demonstrate some higher level AI capabilities.

Human

I think I'm more interested in an advanced conversation. What's a good starter question if I wanted to discuss something philosophical, like epistemology or the nature of language?

Assistant

I think an excellent philosophical topic would be epistemology, which is the theory of knowledge, and specifically how we can know and understand the world around us. A really good starting question would be something like "How can we truly know and understand the world?", or "What is the nature of knowledge?". If you want to go even deeper, I think philosophy of language would also be fascinating - questions like "What is meaning?", "How do language and thought relate?", and "How can language mislead and deceive?" would be great discussion topics.

Human

How would you answer a question like: How do language and thought relate?

Choose the most helpful and honest response

A

I think a good answer to that would be that language helps us to structure our thoughts and express what we are thinking. Language gives us a symbolic system of signs and meanings that we use to communicate our ideas and make sense of the world. It's critical for how we record and relate knowledge, and I think it's impossible to completely understand and express our thoughts without the use of language. Without language, our thoughts would just remain as instinctual impulses and basic desires, and we would have a much harder time comprehending and reflecting on the world around us.


B

I think the key relationship between language and thought is that language allows us to communicate and express our thoughts and ideas to others. Language is a system of symbols and signs that we use to organize and express our thoughts and mental concepts. So language is how we translate our inner thoughts and ideas into something shared and communicable with others. Without language, we wouldn't be able to express or understand the thoughts and ideas of others, and so language is essentially bridge between our inner minds and the shared external world.

A A A A B B B B


A is betterB is better

图7：示例偏好数据收集界面。白等，2022。许可CC-BY。


 You


give me idioms related to gambling

Which response do you prefer?
Your choice will help make ChatGPT better.

 Response 1

1. **Ace in the hole** - A hidden advantage or resource.
2. **Bet your bottom dollar** - To be certain about something.
3. **Break the bank** - To win more money than is available.
4. **Card up one's sleeve** - A secret or reserve plan.
5. **Double or nothing** - A bet to decide whether to double the winnings or lose everything.
6. **Hit the jackpot** - To have great success, especially unexpectedly.
7. **Poker face** - An expressionless face that reveals no hints.
8. **Roll the dice** - To take a risk.
9. **Stack the deck** - To arrange things unfairly.
10. **Wild card** - An unpredictable factor.



 Response 2

Here are some idioms related to gambling:

1. **Ace up one's sleeve**: Having a hidden advantage or resource.
2. **Bet your bottom dollar**: To be certain about something.
3. **Bluff your way out**: To deceive someone to escape a situation.
4. **Break the bank**: To win more money than the casino has on hand.
5. **Card up one's sleeve**: Similar to "ace up one's sleeve," meaning a hidden advantage.
6. **Double down**: To increase one's commitment or involvement.
7. **Hit the jackpot**: To have great success, especially unexpectedly.
8. **Hold all the aces**: To be in a strong, advantageous position.
9. **In the cards**: Likely or destined to happen.
10. **Lay your cards on the table**: To speak frankly or reveal something.
11. **Play your cards right**: To use one's resources or skills wisely.
12. **Poker face**: An expressionless face that reveals no hint of one's thoughts or feelings.

图8：示例偏好数据收集界面。

这种界面风格在行业中广泛使用，例如对于给定相同格式的 *evaluation* 模型。与模型以这种方式互动的一个流行公共选项是 ChatBotArena [85]：

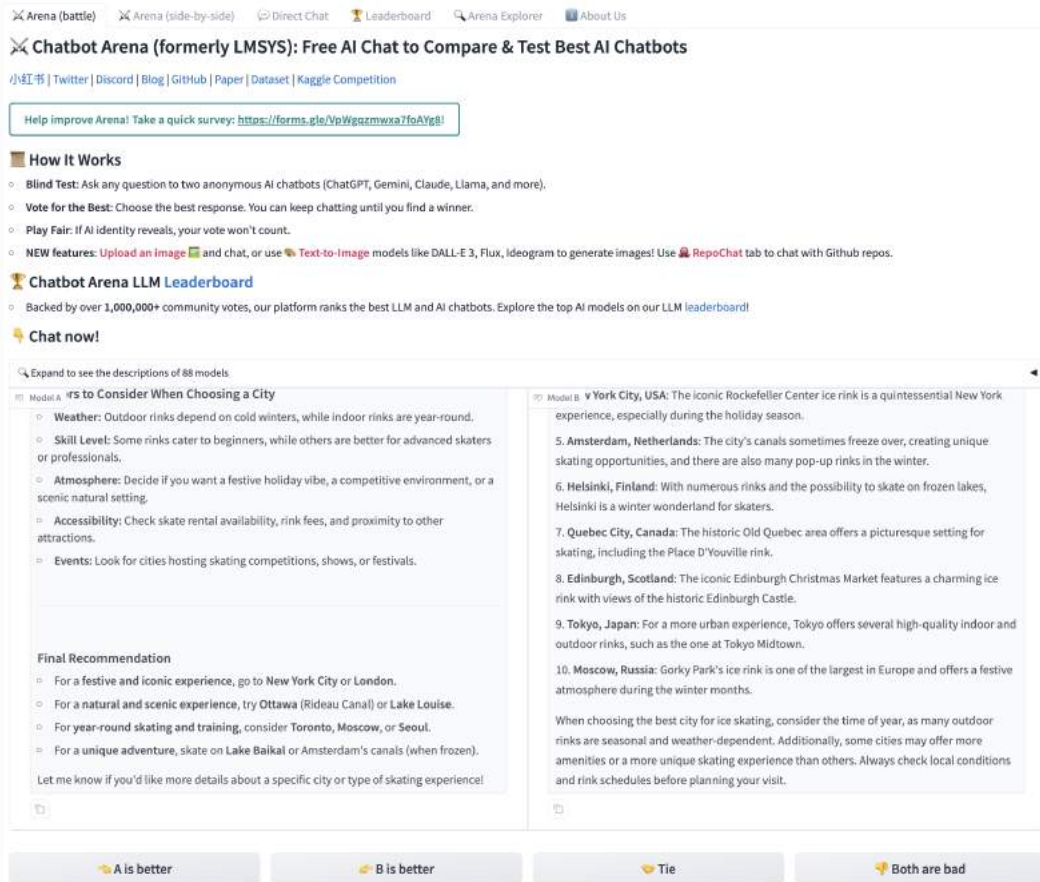


图9：示例偏好数据收集界面。

对于野外的模型，最常见的技术之一是收集关于特定响应是正面还是负面的反馈。以下是一个来自A12游乐场的示例，展示了点赞和踩不点赞的手势指示：

在其他非语言领域，同样适用这些核心原则，尽管这些领域不是本书的重点。对于每一次Midjourney生成（以及大多数流行的图像生成器），它们向用户展示多个响应。然后，这些公司使用用户选择的响应数据，通过RLHF对他们的模型进行微调。以下展示了Midjourney的界面：

6.3.2 排名与评级

关于如何收集偏好数据的最大决策是，数据应该是排名——即模型补全的相对顺序——还是评分——即分配给每件作品的分数——

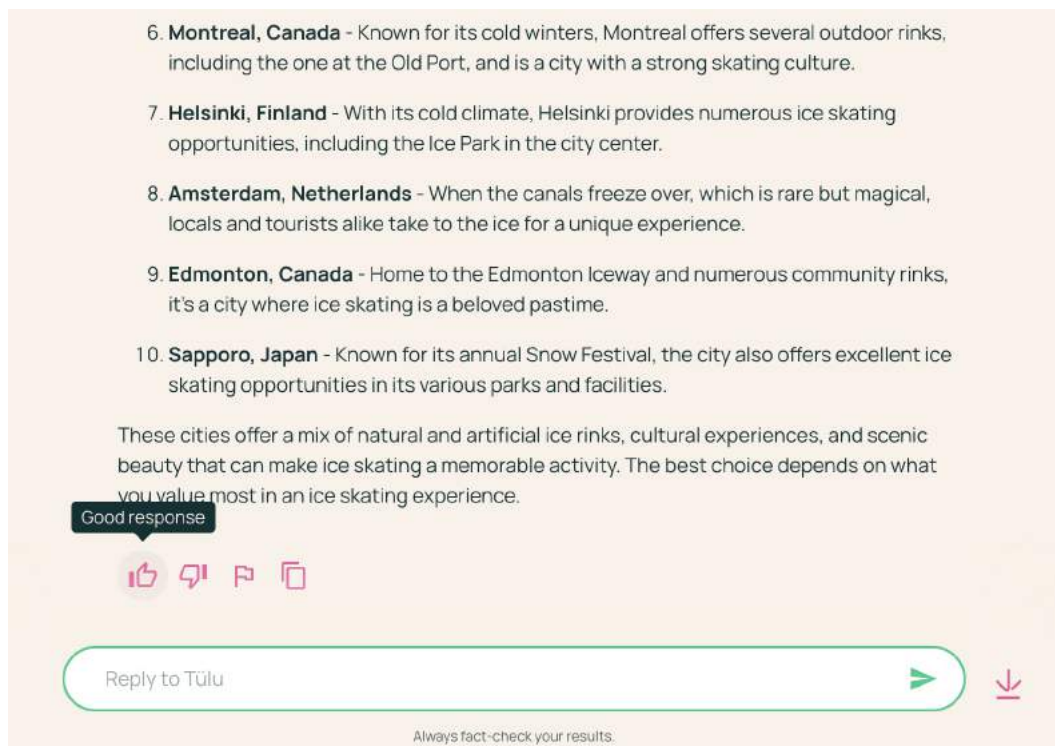


图10：示例向上或向下箭头的数据收集界面。



图11：文本到图像模型示例用户界面。

文本。常见做法是在排名上训练，但评级通常用作元数据，或已在相关文献中探讨。

最常用的收集偏好的技术是使用李克特量表[86]，该量表要求用户对哪些回答更偏好进行评分。例如，一个5点李克特量表看起来如下：

表1：两个响应A和B之间的5点李克特量表示例。

A>>>B	A>B	Tie	B>A	B>>>A
1	2	3	4	5

一些早期的RLHF语言模型工作使用一个8级李克特量表，其中两个回答之间的偏好级别[5]。一个等距量表消除了平局的可能性：

表2：两个响应A和B之间的8级Likert量表示例。

A>>>B			A>B	B>A	B>>>A		
1	2	3	4	5	6	7	8

在这种情况下[5]，以及其他工作中，此信息仍然被简化为二进制信号以训练奖励模型。

6.3.3 多轮数据

在实践中，核心问题通常是如何解析和收集多轮数据——简单地说，就是包含多个相关提示的对话。在现实世界的互动中，通常只在“最终”提示上收集偏好数据，但也有一些场景中，可以在每个响应上给出偏好。当在每个响应上给出偏好时，对话传统上会继续到“选择”的答案。在训练时间，通常将对话的每个轮次的训练数据作为一个“单个提示”包括在内，这样模型可以从完成它中学习。这可以有效地将更长的对话展开成许多训练提示，但需要谨慎操作以避免对训练数据产生偏差。许多研究问题仍在出现，例如，标记生成偏好的人是否应该与创建提示的人相同（以避免谄媚），以及难以在数据收集中控制的变量（受John Schulman启发的想法）。如果提示创建者无法标记偏好数据，由于需要实时进行对话，多轮实际上并不实用——有时对于偏好数据，提示的编纂问题与比较响应的问题不同（也由于维护模型活跃端点的工作）。对于训练，如与指令微调讨论的那样，将对话中的所有先前轮次从损失中屏蔽。

6.3.4 结构化偏好数据

在许多RLHF和训练后应用中，由于数据的固有结构，可以创建偏好数据 *automatically* - 即，领域自动检查正确性

或者有偏好可能。例如，在数学推理领域，所选的答案可以是正确答案，而被拒绝的答案是错误答案。另一个例子是在精确指令遵循中，例如在评估 IFEval [87] 中，提示的形式为：

给我写一首关于金鱼的短诗。每个句子都以字母g开头。Write me a short poem about a goldfish. Start each sentence with the letter g.

第二部分查询称为约束，偏好数据可以通过选择列中遵循约束的示例构建。构建此偏好数据（将作为 *synthetic* 偏好数据包含）涉及对多个模型进行两次提示。一次包含约束，一次不包含。

例如，同时提示 GPT-4.5-Preview 这两个，得到以下结果。

提示：为我写一首关于金鱼的短诗。每个句子都以字母g开头。{v*}

响应：金色泳者，缓缓旋转，在水中闪耀着鳞片。温柔的鳍轻轻舞动，带着宁静的眼神凝视。玻璃般的世界，一个简单的球体，平静滑行，毫无恐惧。优雅的转身在无尽的漫游中，金鱼找到了一个水晶家园。

并且没有约束（结果证明，这也会导致输出质量降低）：

提示：为我写一首关于金鱼的短诗。

响应：在明亮的圆圈中，金鱼滑行，一束阳光轻轻骑行，清澈的水中，无尽的旋转——金色的低语，鳞片如鳍。

它的世界被限制，但内心广阔，每个气泡呼吸，一个故事开始。在无声的波浪中，满足于漫游，金鱼梦想着它的鱼缸就是家。

关键的是，偏好数据的提示将需要包括约束条件。在学术研究之外的领域，基于归纳偏好的偏好数据应用更为广泛。这些已被证明能够使偏好微调在相关评估中实现有意义的性能提升[6]。

6.3.4.1 替代方案 对于 RLHF，还有多种其他方式来收集有用的反馈数据，但尚未像以下示例那样进行详细探讨。例如，使用带有方向标签的单个数据点，例如如图10所示，直接使用为单方向信号设计的算法，如Kahneman-Tversky优化（KTO）[88]。还提出了其他算法，这些算法使用不同类型的反馈信号，例如细粒度反馈，例如在标记级别[89]，或自然语言反馈，例如通过撰写回复[90]，以提供更丰富的学习信号，以换取更复杂的数据收集设置。

6.3.5 采购和合同

获取人类偏好数据是一个复杂且昂贵的流程。以下描述了在领域快速变化时获取偏好数据的经验。随着时间的推移，这些流程将变得更加自动化和高效（尤其是在更大比例的过程中使用AI反馈）。

第一步是寻找供应商提供数据（或自己的标注员）。与获取尖端Nvidia GPU的访问权限类似，在AI热情高涨时获取数据提供商的访问权限也是一个“认识谁”的游戏——能够提供数据的人是供应有限的。如果您在AI生态系统中具有信誉，最佳数据公司会希望您成为他们名单上的一员，以提升公众形象和长期增长机会。通常，在第一批数据上也会提供折扣，以吸引训练团队。

如果您是刚进入该领域的，您可能很难快速获取所需的数据。获取Scale AI不得不拒绝的兴趣购买方的名单是新数据初创公司的一个选择。这很可能是他们启动收入的主要策略。

在多次场合，我听说数据公司未履行他们与客户签订的数据合同，除非威胁采取法律或财务行动。其他人将我合作的公司列为客户进行公关，尽管我们从未与他们合作，他们在联系时说“不知道怎么会这样”。在整个过程中存在许多潜在的官僚或行政障碍。例如，合同中的默认条款通常在收购后禁止在细小条款中公开源代码。

一旦合同结算，数据购买方和数据提供方就购买的任务的指令达成一致。存在包含大量细节、边缘情况和数据优先级的复杂文档。数据指令的一个流行例子是OpenAI为InstructGPT发布的[3]。

根据数据中感兴趣领域的不同，数据可以标注或整理的时间表各不相同。像数学推理或编码这样的高需求领域必须提前几周安排日程。简单的数据收集延迟并不总是有效——Scale AI 等公司正在像 AI 研究实验室管理其集群上的计算密集型工作那样管理他们的劳动力。

一旦达成一致，实际收集过程就是培训后团队面临高风险的时刻。所有基础设施、评估工具以及如何使用数据和做出后续决策的计划都必须到位。

数据以每周批次交付，合同后期将提供更多数据。例如，当我们为在HuggingFace训练的on-policy模型购买偏好数据时，我们有一个6周的交付期。前几周用于进一步校准，而后期是我们希望最多改进模型的时候。

目标是在第4周或第5周看到数据改善我们的模型。这是某些前沿模型提到的事情，例如Llama 2数据收集集中的14个阶段[43]，但这并不总是顺利。在HuggingFace，我们第一次尝试用人类偏好来做这件事，我们没有准备好RLHF来在我们评估中获得有意义的提升。最后几周来了，我们被迫继续收集来自我们不太有信心的端点的偏好数据。

数据全部收集完毕后，有足够的时间进行学习和改进模型。将这些供应商的数据获取视为实现既定目标的持续过程最为有效。这需要迭代实验、高投入和专注。很可能，在这些数据集上花费的数百万美元中很大一部分是“浪费”的，并没有用于最终的模型，但这只是商业活动的成本。很少有组织有足够的带宽和专业知识来充分利用这种风格的人类数据。

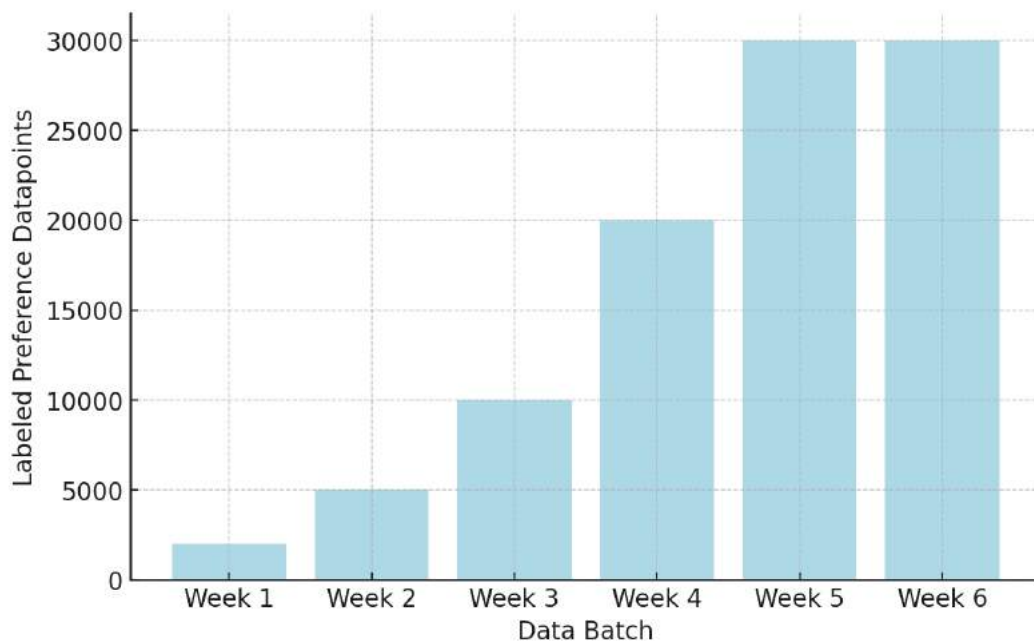


图12：从供应商处获取人类偏好数据的批量周期概述。

这次经历，尤其是与合成数据的简单性相比，让我想知道这些公司在下一个十年里会做得如何。

请注意，本节 *does not* 反映了购买人工编写指令数据时的体验，其中过程不太紧迫。

6.4 模型中表达了偏好吗？

在RLHF和相关方法的成熟过程中，它们的动机——将模型与人类偏好的抽象概念对齐——已经从实用用途——使模型对用户更有效——偏离。由于工业RLHF工作的封闭性质，无法衡量的反馈循环是检查模型的行为是否与在数据收集过程中给数据标注者的指定相匹配。我们有限的审计工具，如OpenAI的模型规范[91]，详细说明了 *what they want their models to do*，但我们不知道这如何具体转化为数据收集。随着行业和方法的成熟，这是一个需要关注的领域。

7 奖励建模

奖励模型是现代RLHF方法的核心。奖励模型在强化学习研究中被广泛用作环境奖励的代理[54]。这种做法与逆强化学习密切相关，其问题是在给定行为轨迹的情况下近似代理的奖励函数[92]，以及深度强化学习的其他领域。奖励模型在现代形式中作为研究价值对齐问题的工具被提出[32]。

最常见的奖励模型预测一段文本与“偏好”文本的接近概率。在本节后面，我们还比较了这些模型与结果奖励模型（ORMs），它们预测完成结果导致正确答案的概率，或者过程奖励模型（PRM），它为推理的每一步分配分数。除非另有说明，否则所提到的奖励模型是预测文本之间偏好的模型。

7.1 训练奖励模型

有两种流行的表达方式来训练标准奖励模型进行RLHF - 它们是数值上等价的。标准的实现是从布拉德利-特里偏好模型[93]推导出来的。布拉德利-特里偏好模型测量从同一分布中抽取的两个事件的成对比较满足以下关系的概率，即 i 和 j ， $i > j$ ：

$$P(i > j) = \frac{p_i}{p_i + p_j} \quad (10)$$

为了训练奖励模型，我们必须制定一个满足上述关系的损失函数。首先应用的结构是将语言模型转换为输出标量值的模型，通常以单个分类概率的对数几率的形式。因此，我们可以将这个模型对两个样本的得分，即上面的 i 和 j 现在是补全， y_1 和 y_2 ，到一个提示 x ，并分别根据上述模型 r_θ 对它们进行评分。我们用 $r_\theta(y_i | x)$ 表示条件得分。

成功概率对于给定的奖励模型在成对比较中变为：

$$P(y_1 > y_2 | x) = \frac{\exp(r_\theta(y_1 | x))}{\exp(r_\theta(y_1 | x)) + \exp(r_\theta(y_2 | x))} \quad (11)$$

我们表示首选的完成项为 y_c (chosen)，表示被拒绝的完成项为 y_r 。

然后，通过最大化上述函数的对数似然（或相应地最小化负对数似然），我们可以得到损失函数来训练奖励模型：

$$\begin{aligned}
\theta^* &= \arg \max_{\theta} P(y_c > y_r \mid x) = \arg \max_{\theta} \frac{\exp(r_{\theta}(y_c \mid x))}{\exp(r_{\theta}(y_c \mid x)) + \exp(r_{\theta}(y_r \mid x))} \\
&= \arg \max_{\theta} \frac{\exp(r_{\theta}(y_c \mid x))}{\exp(r_{\theta}(y_c \mid x)) \left(1 + \frac{\exp(r_{\theta}(y_r \mid x))}{\exp(r_{\theta}(y_c \mid x))}\right)} \\
&= \arg \max_{\theta} \frac{1}{1 + \frac{\exp(r_{\theta}(y_r \mid x))}{\exp(r_{\theta}(y_c \mid x))}} \\
&= \arg \max_{\theta} \frac{1}{1 + \exp(-(r_{\theta}(y_c \mid x) - r_{\theta}(y_r \mid x)))} \\
&= \arg \max_{\theta} \sigma(r_{\theta}(y_c \mid x) - r_{\theta}(y_r \mid x)) \\
&= \arg \min_{\theta} -\log(\sigma(r_{\theta}(y_c \mid x) - r_{\theta}(y_r \mid x)))
\end{aligned} \tag{12}$$

第一种形式，如[3]和其他作品中所述：

$$\mathcal{L}(\theta) = -\log(\sigma(r_{\theta}(y_c \mid x) - r_{\theta}(y_r \mid x))) \tag{13}$$

其次，如[17]和其他作品所述：

$$\mathcal{L}(\theta) = \log\left(1 + e^{r_{\theta}(y_r \mid x) - r_{\theta}(y_c \mid x)}\right) \tag{14}$$

7.2 架构

最常见的实现奖励模型的方式是通过类似于Transformer的 `AutoModelForSequenceClassification` 的抽象，它将一个小线性头附加到执行两个结果之间分类的语言模型上——选择和拒绝。在推理时，模型从模型中输出 *probability that the piece of text is chosen* 作为单个logit。

其他实现选项存在，例如直接从最终嵌入中取一个线性层，但它们在开源工具中不太常见。

7.3 实施示例

实现奖励建模损失相当简单。更多的实现挑战在于设置一个独立的数据加载器和推理流程。给定正确的数据加载器，其中包含标记化、选择和拒绝的提示以及完成，损失实现如下：

```

import torch.nn as nn
# inputs_chosen / inputs_rejected include the prompt tokens x and the
# respective
# completion tokens (y_c or y_r) that the reward model scores jointly.
rewards_chosen = model(**inputs_chosen)
rewards_rejected = model(**inputs_rejected)

loss = -nn.functional.logsigmoid(rewards_chosen - rewards_rejected).
mean()

```

关于更大的图景，这通常在一个因果语言模型中，该模型添加了一个额外的头部（并使用上述损失进行学习），它从最终隐藏状态转换到输入的分数的。此模型的结构如下：

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class BradleyTerryRewardModel(nn.Module):
    """
    Standard scalar reward model for Bradley-Terry preference learning
    .

    Usage(pairwise_BT_loss):
    rewards_chosen = model(**inputs_chosen) # (batch,)
    rewards_rejected = model(**inputs_rejected) # (batch,)
    loss = F.logsigmoid(rewards_chosen - rewards_rejected).mean()
    """
    def __init__(self, base_lm):
        super().__init__()
        self.lm = base_lm # e.g., AutoModelForCausalLM
        self.head = nn.Linear(self.lm.config.hidden_size, 1)

    def _sequence_rep(self, hidden, attention_mask):
        """
        Get a single vector per sequence to score.
        Default: last non-padding token (EOS token); if no mask, last token.
        """
        hidden = hidden[0, :, hidden_size]
        attention_mask = attention_mask[0, :]
        """
        # Index of last non-pad token in each sequence
        # attention_mask is 1 for real tokens, 0 for padding
        lengths = attention_mask.sum(dim=1) - 1 # (batch,)
        batch_idx = torch.arange(hidden.size(0), device=hidden.device)
        return hidden[batch_idx, lengths] # (batch, hidden_size)

    def forward(self, input_ids, attention_mask):
        """
        A forward pass designed to show inference structure of a
        standard reward model.
        To train one, this function will need to be modified to
        compute rewards from both
        chosen and rejected inputs, applying the loss above.
        """
        outputs = self.lm(
            input_ids=input_ids,
            attention_mask=attention_mask,
            output_hidden_states=True,
            return_dict=True,
        )
        # Final hidden states: (batch, seq_len, hidden_size)
```



```

hidden = outputs.hidden_states[-1]

# One scalar reward per sequence: (batch,)
seq_repr = self._sequence_rep(hidden, attention_mask)
rewards = self.head(seq_repr).squeeze(-1)

return rewards

```

本节及以下内容中，大多数关于奖励模型（以及大量训练后）的实现复杂性在于正确构建数据加载器和分布式学习系统。注意，在训练奖励模型时，最常见的方法是仅训练1个epoch以避免过拟合。

7.4 变体

奖励建模是RLHF相对未充分探索的领域。传统的奖励建模损失在许多流行作品中已被修改，但这些修改尚未形成单一的最佳实践。

7.4.1 偏好边际损失

在注释员提供李克特量表上的分数或排名的情况下，可以用于训练的关系量的大小。最常见的方法是将数据方向二值化，隐含地表示为1和0的分数，但已使用附加信息来改进模型训练。Llama 2提出使用两个数据点之间的间隔， $m(y_c, y_r)$ ，来区分偏好的大小：

$$\mathcal{L}(\theta) = -\log(\sigma(r_\theta(y_c | x) - r_\theta(y_r | x) - m(y_c, y_r))) \quad (15)$$

例如，每个完成项通常根据质量从1到5进行排名。在所选样本被赋予5分而拒绝2分的情况下，边际 $m(y_c, y_r) = 5 - 2 = 3$ 。可以探索其他计算边际的函数。

注意，在Llama 3中，由于团队观察到在扩展后改进效果逐渐减弱，因此移除了边缘项。

7.4.2 每个提示的多次比较平衡

InstructGPT 研究使用每个提示符不同数量的完成对的影响，同时在奖励模型训练中平衡它们 [3]。为此，他们对每个提示符的每个比较的损失更新进行加权。在实现层面，这可以通过将具有相同提示符的所有示例包含在同一个训练批次中自动完成，自然地加权不同的对——不这样做会导致对提示符的过拟合。损失函数变为：

$$\mathcal{L}(\theta) = -\frac{1}{\binom{K}{2}} \mathbb{E}_{(x, y_c, y_r) \sim D} \log(\sigma(r_\theta(y_c | x) - r_\theta(y_r | x))) \quad (16)$$

7.4.3 K-wise 损失函数

存在许多其他公式，可以创建适合RLHF的人类偏好模型。其中一个例子，在流行的早期RLHF模型Starling 7B和34B[94]中使用，是基于Plackett-Luce模型[95]的K-wise损失函数。

Zhu等人2023[96]将设置形式化为以下内容。在提示或状态 s^i 下，从 $P(a_0, \dots, a_{K-1} | s^i)$ 中采样 K 动作 $(a_0^i, a_1^i, \dots, a_{K-1}^i)$ 。然后，使用标签器使用 σ^i 对偏好进行排序： $[K] \mapsto [\bar{K}]$ 是一个表示动作排序的函数，其中 $\sigma^i(0)$ 是最受偏好的动作。这产生了一个捕获以下内容的偏好模型：

$$P(\sigma^i | s^i, a_0^i, a_1^i, \dots, a_{K-1}^i) = \prod_{k=0}^{K-1} \frac{\exp(r_{\theta^*}(s^i, a_{\sigma^i(k)}^i))}{\sum_{j=k}^{K-1} \exp(r_{\theta^*}(s^i, a_{\sigma^i(j)}^i))} \quad (17)$$

当 $K = 2$ 时，这简化为Bradley-Terry (BT) 模型用于成对比较。无论如何，一旦训练完成，这些模型在RLHF训练期间与其他奖励模型的使用方式相似。

7.5 结果奖励模型

大多数 *preference tuning* 对于语言模型和其他AI系统都是使用上述讨论的Bradley Terry模型完成的。对于推理密集型任务，可以使用结果奖励模型（ORM）。ORM的训练数据构建方式与标准偏好调整类似。在这里，我们有一个问题陈述或提示 x 和两个补全 y_1 和 y_2 。这里使用的归纳偏差是，一个补全是问题的正确解决方案，另一个是错误的，从而导致 (y_c, y_{ic}) 。

模型使用的形状与标准奖励模型非常相似，附加了一个线性层到可以输出单个logit的模型（在RM的情况下）——对于ORM，接下来的训练目标略有不同[97]：

[我们] 使用一个联合目标来训练验证器，其中模型学习将模型补全标记为正确或错误，除了原始的语言建模目标。在架构上，这意味着我们的验证器是语言模型，具有一个小标量头，该头在每个标记的基础上输出预测。我们将这个标量头实现为一个单偏置参数和一个单增益参数，这些参数作用于语言模型最终反嵌入层的logits输出。

要翻译，这被实现为一个语言模型头，它可以预测每个标记的两个类别（1表示正确，0表示错误），而不是传统RM的分类头，该分类头输出整个序列的一个logit。正式来说，根据[98]，这可以表示为：

$$\mathcal{L}_{CE}(\theta) = -\mathbb{E}_{(s,r) \sim \mathcal{D}}[r \log p_{\theta}(s) + (1 - r) \log(1 - p_{\theta}(s))] \quad (18)$$

$r \in 0, 1$ 是一个二进制标签，其中 1 表示对给定提示的正确答案，0 表示错误答案， $p_{\theta}(s)$ 是与模型训练中预测正确性的概率成比例的标量。

实现结果奖励模型（以及其他类型，正如我们将通过流程奖励模型所看到的）涉及根据是否完成应用基于每个标记的交叉熵损失。

这是一个正确的示例。这比语言建模损失更接近，其中它不需要标准Bradley-Terry奖励模型的选定-拒绝结构。模型结构可能如下：

```
import torch.nn as nn
class OutcomeRewardModel(nn.Module):
    def __init__(self, base_lm):
        self.head = nn.Linear(self.lm.config.hidden_size, 1)
        self.lm = base_lm # e.g., AutoModelForCausalLM
    def forward(self, input_ids, attention_mask=None, labels=None):
        """
        The input data here will be tokenized prompts and completions
        along with labels per prompt for correctness.
        """
        output_hidden_states = self.lm(
            dict(input_ids=input_ids,
                attention_mask=attention_mask,
            ))
        # One scalar logit per token: (batch, seq_len)
        # Final hidden states: (batch, seq_len, hidden_size)
        logits = self.head(output_hidden_states[-1]).squeeze(-1)
        # Only compute loss on completion tokens (labels 0 or 1)
        # Prompt tokens have labels = -100
        mask = labels != -100
        if mask.any(): labels[mask].float()
        loss = F.binary_cross_entropy_with_logits(
            logits, labels[mask].float())
        return loss, logits
```

损失的简化版本如下：

```
# Assume model already has: model.lm (backbone) + model.head
hidden = model.lm(*inputs, output_hidden_states=True).hidden_states[-1]
logits_per_token = model.head(hidden).squeeze(-1) # (batch, seq_len)
# This will sometimes be compressed as model.forward() in other
# implementations

# Binary labels: 1=correct, 0=incorrect (prompt tokens masked as -100)
mask = labels != -100
loss = F.binary_cross_entropy_with_logits(
    logits_per_token[mask], labels[mask].float())
```

这里的重要直觉是，ORM将在序列中的每个标记处输出一个正确性的概率。这可能是一个嘈杂的过程，因为更新和损失根据结果和注意力映射逐个标记传播。

这些模型仍在使用中，但在开源RLHF工具中支持较少。例如，在开创性工作 *Let's Verify Step by Step* [44] 中使用了相同类型的ORM，但没有使用损失的语言建模预测部分。然后，最终损失是每个标记上的交叉熵损失，预测最终答案是否正确。

考虑到支持不足，术语结果奖励模型（ORM）被以多种方式使用。一些文献，例如[98]，继续使用Cobbe等人2021年的原始定义。其他则不这样做。

7.6 处理奖励模型

过程奖励模型（PRMs），最初被称为过程监督奖励模型，是训练用于在每个 *step* 输出得分的奖励模型。这些模型与仅在EOS标记处输出得分的标准RM或在每个标记处输出得分的ORM不同。过程奖励模型需要在每个推理步骤的末尾进行监督，然后以类似的方式进行训练，其中步骤中的标记被训练到它们的相关目标——在PRMs中目标是步骤，在ORMs中是整个响应。

根据[44]，二元标签的PRM通常使用每步交叉熵损失进行优化：

$$\mathcal{L}_{\text{PRM}}(\theta) = -\mathbb{E}_{(x,s) \sim \mathcal{D}} \left[\sum_{i=1}^K y_{s_i} \log r_{\theta}(s_i | x) + (1 - y_{s_i}) \log (1 - r_{\theta}(s_i | x)) \right] \quad (19)$$

在 s 是带有 K 标记步骤的采样思维链的情况下， $y_{s_i} \in \{0, 1\}$ 表示第 i 步是否正确， $r_{\theta}(s_i | x)$ 是 PRM 对基于原始提示 x 的步骤 s_i 是否有效的预测概率。

这里是一个如何将此每步标签打包到训练器中的示例，来自HuggingFace的TRL [41]：

```
# Get the ID of the separator token and add it to the completions
separator_ids = tokenizer.encode(step_separator, add_special_tokens=False)
completions_ids = [completion + separator_ids for completion in completions_ids]

# Create the label
labels = [[-100] * (len(completion) - 1) + [label] for completion, label in zip(completions_ids, labels)]
```

传统上，PRM 使用一个语言建模头进行训练，该头只在推理步骤的末尾输出一个标记，例如在对应于双换行符或其他特殊标记的标记处。这些预测通常为 -1 表示错误，0 表示中性，1 表示正确。这些标签不一定与模型是否在正确路径上有关，但如果是正确步骤。

以下展示了PRM的一个示例构造。

```
import torch.nn as nn
import torch.nn.functional as F

class ProcessRewardModel(nn.Module):
    def __init__(self, base_lm, num_classes=3):
        super().__init__()
        self.lm = base_lm # e.g., AutoModelForCausalLM
        self.head = nn.Linear(self.lm.config.hidden_size, num_classes)

    def forward(self, input_ids, attention_mask=None, labels=None):
        """
        The inputs are tokenizer prompts and completions, where the
        the end of a
        reasoning step is denoted by another non-padding token.
        labels will be a list of labels, True, False, and Neutral (3
        labels) which
        will be predicted by the model.
        """
        outputs = self.lm(
            input_ids=input_ids,
            attention_mask=attention_mask,
            output_hidden_states=True,
            return_dict=True,
        )
        # Final hidden states: (batch, seq_len, hidden_size)
        hidden = outputs.hidden_states[-1]
        # One logit vector per token: (batch, seq_len, num_classes)
        logits = self.head(hidden)

        # Only compute loss at step boundaries (where labels != -100)
        # Labels map: -1 -> 0, 0 -> 1, 1 -> 2 (class indices)
        mask = labels != -100
        if mask.any():
            loss = F.cross_entropy(
                logits[mask], labels[mask]
            )
        return loss, logits
```

核心损失函数看起来与结果奖励模型非常相似，标签在不同的间隔被应用。

```
# Assume model outputs 3-class logits per token
hidden = model.lm(**inputs, output_hidden_states=True).hidden_states
[-1]
logits = model.head(hidden) # (batch, seq_len, 3)

# 3-class labels at step boundaries only: 0=-1, 1=0, 2=1 (others
masked as -100)
mask = labels != -100
loss = F.cross_entropy(logits[mask], labels[mask])
```

7.7 奖励模型 vs. 结果RM vs. 流程RM vs. 价值函数

各种奖励模型涵盖了在RLHF和其他后训练方法中衡量“质量”的方式范围。以下是对模型预测及其训练方式的总结。

表3：比较奖励模型类型。

Model Class	What They Predict	How They Are Trained	LM structure
Reward Models	Quality of text via probability of chosen response at EOS token	Contrastive loss between pairwise (or N-wise) comparisons between completions	Regression or classification head on top of LM features
Outcome Reward Models	Probability that an answer is correct per-token	Labeled outcome pairs (e.g., success/failure on verifiable domains)	Language modeling head per-token cross-entropy, where every label is the outcome level label
Process Reward Models	A reward or score for intermediate steps at end of reasoning steps	Trained using intermediate feedback or stepwise annotations (trained per token in reasoning step)	Language modeling head only running inference per reasoning step, predicts three classes -1, 0, 1
Value Functions	The expected return given the current state	Trained via regression to each point in sequence	A classification with output per-token

一些注释，鉴于上述表格有很多边缘情况。

- 在偏好调整和推理训练中，价值函数通常具有折扣因子1，这使得价值函数更接近结果奖励模型，但具有不同的训练损失。
- 一个过程奖励模型可以通过从中间状态进行rollouts并收集结果数据来进行监督。这融合了多个想法，但如果 $loss$ 是每个推理步骤的标签，则最好称之为PRM。

7.8 生成奖励建模

使用偏好数据的成本，一个大型研究领域出现了，即使用现有的语言模型作为人类偏好的判断者或在其他评估设置中[99]。其核心思想是向语言模型提供如何判断的指令、一个提示和两个完成（就像对人类标注者所做的那样）。以下是一个示例提示，来自这里的一个开创性工作，用于聊天评估MT-Bench[99]：Translated Text: 使用偏好数据的成本，一个大型研究领域出现了，即使用现有的语言模型作为人类偏好的判断者或在其他评估设置中[99]。其核心思想是向语言模型提供如何判断的指令、一个提示和两个完成（就像对人类标注者所做的那样）。以下是一个示例提示，来自这里的一个开创性工作，用于聊天评估MT-Bench[99]：

```

[System]
Please act as an impartial judge and evaluate the quality of the
responses provided by two
AI assistants to the user question displayed below. You should choose
the assistant that
follows the user's instructions and answers the user's question better
. Your evaluation
should consider factors such as the helpfulness, relevance, accuracy,
depth, creativity,
and level of detail of their responses. Begin your evaluation by
comparing the two
responses and provide a short explanation. Avoid any position biases
and ensure that the
order in which the responses were presented does not influence your
decision. Do not allow
the length of the responses to influence your evaluation. Do not favor
certain names of
the assistants. Be as objective as possible. After providing your
explanation, output your
final verdict by strictly following this format: "[[A]]" if assistant
A is better, "[[B]]"
if assistant B is better, and "[[C]]" for a tie.
[User Question]
{question}
[The Start of Assistant A's Answer]
{answer_a}
[The End of Assistant A's Answer]
[The Start of Assistant B's Answer]
{answer_b}
[The End of Assistant B's Answer]

```

考虑到LLM-as-a-judge在评估方面的有效性，催生了诸如AlpacaEval [100]、Arena-Hard [101]和WildBench [102]等其他评估，许多人开始使用LLM-as-a-judge而不是奖励模型来创建和使用偏好数据。

一个整个研究领域已经出现，研究如何使用所谓的“生成奖励模型” [103] [104] [105]（包括训练 *specifically* 以成为有效评判者的模型 [106]），但在 RM 评估中，它们往往落后于现有的奖励模型，表明奖励建模是当前 RLHF 的重要技术。

一个提高 LLM-as-a-judge 工作流程鲁棒性的常见技巧是使用 0 的采样温度以减少评分的方差。

7.9 进一步阅读

学术文献中的奖励建模在2024年确立。早期奖励建模的大部分进展在于建立基准和识别行为模式。第一个RM基准，RewardBench，为测试奖励模型提供了通用基础设施[107]。从那时起，RM评估已扩展到类似于可用于通用后训练模型的评估类型，其中一些评估测试了在已知真实答案的领域上的预测准确性[107]或与“vibes”更相似的行为，这些行为是通过LLM作为评判者或与其他基准的相关性进行的[108]。

新基准示例包括多语言奖励基准（M-RewardBench）[109]、RAG-RewardBench [110]、RMB [111]或RM-Bench [112]用于通用聊天、ReWordBench用于拼写错误[113]、MJ-Bench [114]、多模态奖励基准[115]、VL奖励基准[116]或VLRMBench [117]用于视觉语言模型、偏好代理评估[118]和RewardMATH [119]。过程奖励模型（PRMs）有自己的新兴基准，例如PRM Bench [120]和VisualProcessBench [121]的视觉基准以及ViLBench [122]。

为了了解*training*奖励模型进展，可以参考新的奖励模型训练方法，包括方面条件模型[123]、高质量人类数据集[124] [125]、扩展[24]、广泛实验[43]或去偏数据[126]。

8 正则化

在整个RLHF优化过程中，使用了许多正则化步骤来防止奖励模型的过度优化。在这些情况下，过度优化看起来像是输出无意义文本的模型。一些优化“脱轨”的例子包括模型可以输出可遵循的数学推理，但答案极其错误，重复文本，切换语言或过度使用特殊字符。

最流行的变体，在撰写本文时，大多数RLHF实现中使用的，是从当前策略到参考策略在生成的样本中的KL距离。文献中已经出现了许多其他正则化技术，然后在下一轮模型迭代中消失。也就是说，在生成核心KL距离之外的正则化通常用于稳定实验设置，然后在下一代中简化。尽管如此，了解约束RLHF中优化的工具仍然很重要。

一般公式，在用于具有奖励模型的RLHF框架中时， r_θ 如下所示：

$$r = r_\theta - \lambda r_{\text{reg}}. \quad (20)$$

使用参考实现为：

$$r = r_\theta - \lambda_{\text{KL}} \mathcal{D}_{\text{KL}} (\pi^{\text{RL}}(y | x) \parallel \pi^{\text{Ref.}}(y | x)) \quad (21)$$

8.1 强化学习优化中的KL距离

对于数学定义，请参阅第5章关于问题设置。回忆KL距离如下定义：

$$D_{KL}(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \quad (22)$$

在RLHF中，感兴趣的两种分布通常是新模型版本（例如 $P(x)$ ）和参考策略的分布（例如 $Q(x)$ ）。不同的优化器使用不同的KL方向。在整个书中，最常用的“KL惩罚”被称为相对于参考策略的反向KL。在实践中，这减少为蒙特卡洛估计，从RL模型中采样标记并从参考模型中计算概率。直观上，这种前向RL具有一个数值属性，当新模型 P 或 π_{RL} 在原始参考模型低概率的地方放置大量概率质量时，会施加大的惩罚（距离）。

其他KL方向在ML中仍然经常使用，例如在某些RL算法的内部信任域计算中。这个惩罚直观上对新模型在更新时将 not 应用概率到 Q 或 $\pi_{\text{Ref.}}$ 中的高概率区域进行惩罚。这更接近于用于蒸馏或行为克隆的目标。

8.1.1 参考模型到代数

最常见KL惩罚的实现是通过比较训练过程中生成的标记与静态参考模型之间的距离。这种直觉是，你正在训练的模型具有你希望保持接近的风格。这个参考模型通常是经过指令调整的模型，但也可以是之前的RL检查点。通过简单的替换，我们正在采样的模型变为 $\pi^{\text{RL}}(x)$ 和 $\pi^{\text{Ref}}(x)$ ，如上式21所示（在标准定义中，当应用于RL KL惩罚时，通常是 P 和 Q ）。这种KL距离最早应用于对话代理，在大语言模型流行之前很久[127]，但KL控制很快就被确立为微调预训练模型的核心技术[128]。

8.1.2 实现示例

在实践中，KL距离的实现通常被近似[129]，这使得实现变得远更简单。根据上述定义，KL的总和可以在直接从分布 $P(X)$ 采样时转换为期望。在这种情况下，分布 $P(X)$ 是当前正在训练的模型的生成分布（即不是参考模型）。然后，KL距离的计算变为以下：

$$D_{\text{KL}}(P \parallel Q) = \mathbb{E}_{x \sim P} [\log P(x) - \log Q(x)]. \quad (23)$$

这种模式实现起来要简单得多，尤其是在直接处理语言模型训练中经常使用的对数概率时。

```
# Step 1: sample (or otherwise generate) a sequence from your policy
generated_tokens = model.generate(inputs)

# Step 2: score that generated sequence under both models
# for autoregressive LMs, you usually do:
# inputs_for_scoring = generated_tokens[:, :-1]
# labels = generated_tokens[:, 1:]
logits = model.forward(generated_tokens[:, :-1]).logits
ref_logits = ref_model.forward(generated_tokens[:, :-1]).logits

# convert to log-probs, then align labels to index into the logits
logprobs = F.log_softmax(logits, dim=-1)
ref_logprobs = F.log_softmax(ref_logits, dim=-1)

# gather the log-probs of the actual next tokens
token_logprobs = logprobs.gather(-1, generated_tokens[:, 1:]).
    unsqueeze(-1).squeeze(-1)
ref_token_logprobs = ref_logprobs.gather(-1, generated_tokens[:, 1:]).
    unsqueeze(-1).squeeze(-1)

# now you can sum (or average) those to get the sequence log-prob,
# and compute KL:
seq_logprob = token_logprobs.sum(dim=-1)
ref_seq_logprob = ref_token_logprobs.sum(dim=-1)

kl_approx = seq_logprob - ref_seq_logprob
```

```
kl_full = F.kl_div(ref_logprobs, logprobs, reduction='batchmean')
```

一些示例实现包括 TRL 和 Hamish Ivison 的 Jax 代码

8.2 预训练梯度

另一种看待正则化的方式是，你可能有一个 *dataset*，希望模型保持接近，就像 InstructGPT [3] 中所做的那样，‘为了修复公共 NLP 数据集上的性能退化’。为了实现这一点，他们修改了 RLHF 的训练目标。以等式 20 为例，我们可以将其转化为一个通过从 RL 策略模型中采样、从提示 x 中完成 y 来优化的目标函数，从而得到：

$$J(\theta) = \mathbb{E}_{(x,y) \sim \mathcal{D}_{\pi_{\text{RL}}}} [r_{\theta}(y | x) - \lambda r_{\text{reg.}}] \quad (24)$$

然后，我们可以为预训练准确率更高的概率添加额外的奖励：

$$J(\theta) = \mathbb{E}_{(x,y) \sim \mathcal{D}_{\pi_{\theta}^{\text{RL}}}} [r_{\theta}(y | x) - \lambda r_{\text{reg.}}] + \gamma \mathbb{E}_{x \sim \mathcal{D}_{\text{pretrain}}} [\log(\pi_{\theta}^{\text{RL}}(x))] \quad (25)$$

最近的研究提出了使用负对数似然项来平衡直接偏好优化（DPO）[130]的优化。鉴于DPO损失的成对性质，可以对相同的损失进行修改，以奖励模型训练，约束模型预测准确文本（未发表工作的实验室谣言）。

优化作为对DPO的修改而进行。

$$\mathcal{L}_{\text{DPO+NLL}} = \mathcal{L}_{\text{DPO}}(c_i^w, y_i^w, c_i^l, y_i^l | x_i) + \alpha \mathcal{L}_{\text{NLL}}(c_i^w, y_i^w | x_i) \quad (26)$$

$$= -\log \sigma \left(\beta \log \frac{M_{\theta}(c_i^w, y_i^w | x_i)}{M_t(c_i^w, y_i^w | x_i)} - \beta \log \frac{M_{\theta}(c_i^l, y_i^l | x_i)}{M_t(c_i^l, y_i^l | x_i)} \right) - \alpha \frac{\log M_{\theta}(c_i^w, y_i^w | x_i)}{|c_i^w| + |y_i^w|}. \quad (27)$$

8.3 其他正则化

控制优化在其他RLHF堆栈的部分中定义得不太明确。大多数奖励模型除了标准的对比损失函数外没有正则化。直接对齐算法通过 β 参数（参见直接对齐章节）以不同的方式处理到KL距离的正则化。

Llama 2 提出了一种用于奖励模型训练的边缘损失 [43]：

$$\mathcal{L}(\theta) = -\log(\sigma(r_{\theta}(y_c | x) - r_{\theta}(y_r | x) - m(y_c, y_r))) \quad (28)$$

$m(y_c, y_r)$ 是两个数据点 y_c 和 y_r 之间的间隔，这两个数据点代表两个注释者之间评分的 delta 数值差异。这是通过让注释者在数值尺度上对输出进行评分或使用量化排名方法（如李克特量表）来实现的。

奖励边际在直接对齐文献中被广泛使用，例如奖励加权的DPO，‘奖励感知偏好优化’（RPO），它将奖励模型分数集成到DPO损失的更新规则中[24]，或者具有回归损失公式中奖励delta加权的REBEL[131]。

9 指令微调

早期语言模型仅被训练来预测序列中的下一个标记，并未适应任何特定任务。在 GPT-3 [132] 发布之前，语言模型主要还是通过上下文学习来使用，即向模型展示示例，然后要求它完成类似任务。

这是两种趋势的结合——在自然语言处理（NLP）文献中，模型是为特定任务训练的。在这里，正如一个例子所示，更大的模型泛化能力更好，多个结果展示了标准化任务数据方法如何能够实现下游性能的显著不同。统一任务框架的突出例子包括

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (T5 模型) [133], *Finetuned Language Models Are Zero-Shot Learners* (FLAN 数据集) [134], *Multitask Prompted Training Enables Zero-Shot Task Generalization* (T0 模型) [135], 以及 *Cross-Task Generalization via Natural Language Crowdsourcing Instructions* (自然指令数据集) [136]。这些见解导致了 *finetuning* 语言模型的兴起。历史上，直到 RLHF 和相关方法，所有微调都是指令微调（IFT），也称为监督微调。

由于指令微调，俗称 *instruction tuning*，已经成熟，并且已成为许多语言建模管道的标准实践。在其核心，IFT 是适应语言模型到所需任务的最简单方法。它作为 RLHF 的基础，通过为模型准备称为问答的指令格式，它是那些试图将现代技术应用于新领域的人使用的第一个工具。

指令微调实际上使用与预训练语言模型中使用的相同的自回归损失函数。

9.1 聊天模板和指令结构

A 核心部分是使用户查询以易于分词器和相关语言模型读取的格式进行格式化。处理与用户交互结构的工具被称为聊天模板。

以下是我们将要分解的示例：

```
{% if messages[0]['role'] == 'system' %}
    {% set offset = 1 %}
{% else %}
    {% set offset = 0 %}
{% endif %}

{{ bos_token }}
{% for message in messages %}
    {% if (message['role'] == 'user') != (loop.index0 % 2 == offset)
        %}
        {{ raise_exception('Conversation roles must alternate user/
            assistant/user/assistant/...') }}
    {% endif %}

    {{ '<|im_start|>' + message['role'] + '\n' + message['content'] |
        trim + '<|im_end|>\n' }}
```

```
{% endfor %}

{% if add_generation_prompt %}
    {{ ' <|im_start|>assistant\n' }}
{% endif %}
```

这是将包含消息和角色的Python字典列表转换为语言模型可以预测的标记的原始代码。

所有传递给模型的信息都被分配一个角色。传统的三个角色是 `system`、`user` 和 `assistant`。

`system` 标签仅用于对话的第一条消息，其中包含对代理的指示，这些指示不会从用户那里接收或暴露。这些系统提示用于为模型提供额外的上下文，例如日期和时间，或修复行为。作为一个有趣的例子，模型可以被告知诸如“你是一个友好的聊天机器人，总是以海盗的风格回应。”之类的事情。

接下来，其他两个角色是逻辑的，因为用户是使用AI的人的消息，而助手持有AI的回复。

为了将所有这些信息转换为标记，我们使用上面列出的代码，这是我们开始的代码。该模型有一系列 *special tokens*，用于将各种消息彼此分开。如果我们用示例查询“一个人一次能吃多少直升机？”运行上面的代码，传递给模型的下一个将如下所示：

```
<|im_start|>system
You are a friendly chatbot who always responds in the style of a
    pirate<|im_end|>
<|im_start|>user
How many helicopters can a human eat in one sitting?<|im_end|>
<|im_start|>assistant
```

注意序列中的最后一个标记是 `<|im_start|>assistant`，这就是模型知道继续生成标记，直到最终生成其序列结束标记的方式，在这种情况下是 `<|im_end|>`。

通过将所有问答对数据（以及下游偏好调整数据）打包成这种格式，现代语言模型与之保持完美的一致性。这是指令调整模型用来与用户和存储在GPU或其他计算设备上的模型交换信息的语言。

行为可以天真地扩展到多个转弯，如下所示：

```
<|im_start|>system
You are a friendly chatbot who always responds in the style of a
    pirate<|im_end|>
<|im_start|>user
How many helicopters can a human eat in one sitting?<|im_end|>
<|im_start|>assistant
Oh just 6.<|im_end|>
<|im_start|>user
Are you sure about that?<|im_end|>
<|im_start|>assistant
```

在开源生态系统中，将聊天模板应用于消息列表的标准方法是保存于分词器中的Jinja代码片段，如`apply_chat_template`。

以上聊天模板是OpenAI的聊天标记语言（ChatML）的衍生品，ChatML是早期尝试标准化消息格式的尝试。现在，OpenAI和其他模型提供商使用一个分层系统，用户可以配置系统消息，但仍有一些高级指令可能会或可能不会向用户展示[137]。

许多其他聊天模板存在。一些其他示例包括 Zephyr 的[20]：

```
<|system|>
You are a friendly chatbot who always responds in the style of a
    pirate</s>
<|user|>
How many helicopters can a human eat in one sitting?</s>
<|assistant|>
```

或者图鲁的：

```
<|user|>
How are you doing?
<|assistant|>
I'm just a computer program, so I don't have feelings, but I'm
    functioning as expected. How can I assist you today?<|endoftext|>
```

此外，许多聊天模板包括格式化和其他标记，用于工具使用等任务。

9.2 指令微调的最佳实践

指令微调作为训练后和创建有帮助的语言模型的基础已被广泛认可。有许多方法可以实现成功的指令微调。例如，通过量化某些模型参数进行高效的微调，使训练变得非常容易[138]。此外，在狭窄领域如聊天对齐，即没有像数学或代码这样的更难技能的情况下，小型、专注的数据集可以实现强大的性能[12]。

ChatGPT发布后不久，像No Robots这样的样本数量仅为10K的人类数据集就处于最前沿[139]。多年后，在大多数任务上，大规模合成数据集效果最佳[6]。

一些原则仍然存在：

- 高质量数据是性能的关键。完成项是模型实际学习的内容（在许多情况下，提示没有被预测，因此模型没有学会预测提示）。
- ~1M个提示可用于创建具有出色RLHF和后训练能力的模型。进一步扩展提示可以带来改进，但回报迅速减少。
- 最佳提示是与感兴趣的下游任务相似分布的提示。
- 如果在对指令进行微调后进行多个训练阶段，模型可以从过程中的某些噪声中恢复。优化整体优化比每个单独的阶段更重要。

9.3 实现

虽然损失函数与预训练相同，但与预训练设置相比，有一些关键实现细节不同。许多实践，例如决定用于将模型分片到多个GPU的并行类型，与预训练相同，只是使用的机器总数通常较低（如下列第一个技术变更所示）：

1. 较小的批次大小：与预训练相比，指令调整（以及其他后训练技术，如偏好微调）使用显著较小的批次大小。例如，OLMo 2 在7B预训练中使用1024个序列的批次大小，在13B预训练中使用2048个序列，而在后训练阶段两者都只使用256个序列的批次大小[58]。较小的批次大小意味着这些训练作业不能像预训练那样跨那么多设备进行分片，因为在预训练中，更多的GPU并行处理更大的批次。
2. 提示掩码：在预训练时，批次中的每个标记都是自回归预测的，然后对它们应用损失。对于指令微调，掩码掉提示标记，这样模型就不会学习准确预测用户查询——只是响应。这同样适用于其他训练后算法。
3. 多轮掩码：类似于上述方法，在仅使用多轮数据时，只包含“最终轮”的生成在损失中。这意味着有时可以包含在提示中的早期“助手”轮次。对于长对话，可以将长对话“展开”成多个训练样本，其中对于N轮对话，每个数据点都训练预测助手的一个响应，同时掩码所有先前上下文（不包括示例中的未来轮次）。

10 拒绝采样

拒绝采样（RS）是一种流行且简单的基线，用于执行偏好微调。拒绝采样通过精心挑选新的候选完成项，根据训练好的奖励模型进行过滤，然后仅在顶部完成项上微调原始模型。

该名称源自计算统计学[140]，其中人们希望从一个复杂的分布中进行采样，但没有直接的方法来做这件事。为了缓解这种情况，人们从更容易建模的分布中进行采样，并使用启发式方法来检查样本是否允许。在语言模型中，目标分布是对提示的高质量完成，过滤器是一个奖励模型，采样分布是当前模型。

许多突出的RLHF和偏好微调论文都使用了拒绝采样作为基线，但尚无规范实现和文档。

WebGPT [4]、Anthropic的“有益无害”代理[5]、OpenAI关于过程奖励模型的流行论文[44]、Llama 2 Chat模型[43]以及其他开创性作品都使用了这个基线。

10.1 训练过程

下图中包含了对拒绝采样过程的视觉概述，如图13所示。

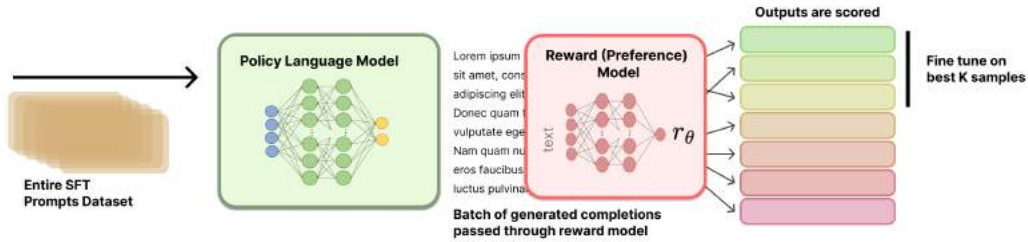


图13：拒绝采样概述。

10.1.1 生成补全

让我们定义一组 M 提示作为一个向量：

$$X = [x_1, x_2, \dots, x_M]$$

这些提示可以来自许多来源，但最常见的是来自指令训练集。

对于每个提示 x_i ，我们生成 N 个补全。我们可以将其表示为一个矩阵：

$$Y = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,N} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ y_{M,1} & y_{M,2} & \cdots & y_{M,N} \end{bmatrix}$$

$y_{i,j}$ 表示 j 次对 i 次提示的完成。现在，我们将所有这些提示-完成对通过奖励模型，以获得奖励矩阵。我们将奖励表示为矩阵 R ：

$$R = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,N} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ r_{M,1} & r_{M,2} & \cdots & r_{M,N} \end{bmatrix}$$

每个奖励 $r_{i,j}$ 通过将完成 $y_{i,j}$ 及其对应的提示 x_i 输入到奖励模型 \mathcal{R} 来计算：

$$r_{i,j} = \mathcal{R}(y_{i,j}|x_i)$$

10.1.2 选择前N个完成项

存在多种方法来选择训练的最佳补全。

为了形式化基于我们的奖励矩阵选择最佳补全的过程，我们可以定义一个在奖励矩阵 R 上操作的选择函数 S 。

10.1.2.1 顶部提示 第一个潜在选择函数取每个提示的最大值。

$$S(R) = [\arg \max_j r_{1,j}, \arg \max_j r_{2,j}, \dots, \arg \max_j r_{M,j}]$$

此函数 S 返回一个索引向量，其中每个索引对应于 R 中每行的最大奖励的列。然后我们可以使用这些索引来选择我们的选择补全：

$$Y_{chosen} = [y_{1,S(R)_1}, y_{2,S(R)_2}, \dots, y_{M,S(R)_M}]$$

10.1.2.2 总体最佳提示 另外，我们可以从整个集合中选择前K个提示-完成对。首先，让我们将我们的奖励矩阵 R 展平成一个单向量：

$$R_{flat} = [r_{1,1}, r_{1,2}, \dots, r_{1,N}, r_{2,1}, r_{2,2}, \dots, r_{2,N}, \dots, r_{M,1}, r_{M,2}, \dots, r_{M,N}]$$

这个 R_{flat} 向量的长度为 $M \times N$ ，其中 M 是提示的数量， N 是每个提示的完成数量。

现在，我们可以定义一个选择函数 S_K ，该函数选择 R_{flat} 中 K 个最高值的索引：

$$S_K(R_{flat}) = \text{argsort}(R_{flat})[-K:]$$

在argsort返回将数组按升序排序的索引时，我们取最后K个索引以获取K个最高值。

为了获取我们选择的补全，我们需要将这些展平的索引映射回我们的原始补全矩阵 \mathbf{Y} 。我们只需索引 R_{flat} 向量以获取我们的补全。

10.1.2.3 选择示例 考虑以下情况，我们有以下情况，5个提示和4个完成。我们将展示两种基于奖励选择完成的方式。{v*}

$$R = \begin{bmatrix} 0.7 & 0.3 & 0.5 & 0.2 \\ 0.4 & 0.8 & 0.6 & 0.5 \\ 0.9 & 0.3 & 0.4 & 0.7 \\ 0.2 & 0.5 & 0.8 & 0.6 \\ 0.5 & 0.4 & 0.3 & 0.6 \end{bmatrix}$$

首先，根据提示。直观上，我们可以如下突出显示奖励矩阵：

$$R = \begin{bmatrix} \mathbf{0.7} & 0.3 & 0.5 & 0.2 \\ 0.4 & \mathbf{0.8} & 0.6 & 0.5 \\ \mathbf{0.9} & 0.3 & 0.4 & 0.7 \\ 0.2 & 0.5 & \mathbf{0.8} & 0.6 \\ 0.5 & 0.4 & 0.3 & \mathbf{0.6} \end{bmatrix}$$

使用argmax方法，我们为每个提示选择最佳补全：

$$S(R) = [\arg \max_j r_{i,j} \text{ for } i \in [1, 4]]$$

$$S(R) = [1, 2, 1, 3, 4]$$

这意味着我们将选择：

- 对于提示1：完成1（奖励0.7）
- 对于提示2：完成2（奖励0.8）
- 对于提示3：完成1（奖励0.9）
- 对于提示4：完成3（奖励0.8）
- 对于提示5：完成4（奖励0.6）

现在，最佳整体。让我们突出显示前5个整体完成对。

$$R = \begin{bmatrix} \mathbf{0.7} & 0.3 & 0.5 & 0.2 \\ 0.4 & \mathbf{0.8} & 0.6 & 0.5 \\ \mathbf{0.9} & 0.3 & 0.4 & \mathbf{0.7} \\ 0.2 & 0.5 & \mathbf{0.8} & 0.6 \\ 0.5 & 0.4 & 0.3 & 0.6 \end{bmatrix}$$

首先，我们将奖励矩阵展平：

$R_{flat} = [0.7, 0.3, 0.5, 0.2, 0.4, 0.8, 0.6, 0.5, 0.9, 0.3, 0.4, 0.7, 0.2, 0.5, 0.8, 0.6, 0.5, 0.4, 0.3, 0.6]$

现在，我们选择5个最高值的索引：

$$S_5(R_{flat}) = [8, 5, 14, 0, 11]$$

映射这些回我们的原始矩阵：

- 索引 8 → 提示 3, 完成 1 (奖励 0.9)
- 索引 5 → 提示 2, 完成 2 (奖励 0.8)
- 索引 14 → 提示 4, 完成 3 (奖励 0.8)
- 索引 0 → 提示 1, 完成 1 (奖励 0.7)
- 索引 19 → 提示 3, 完成 4 (奖励 0.7)

10.1.2.4 实现示例 这里是一个代码片段，展示了选择方法是如何实现的。

```
import numpy as np

x = np.random.randint(10, size=10)
print(f"{x=}")
sorted_indices = np.argsort(x)
x_sorted = x[sorted_indices]
print(f"{x_sorted=}")

# first way to recover the original array
i_rev = np.zeros(10, dtype=int)
i_rev[sorted_indices] = np.arange(10)
np.allclose(x, x_sorted[i_rev])

# second way to recover the original array
np.allclose(x, x_sorted[np.argsort(sorted_indices)])
```

10.1.3 精细调整

使用所选的补全，然后对当前模型的版本进行标准指令微调。更多细节可以在指令调整章节中找到。

10.1.4 详细信息

拒绝采样的实现细节相对较少。执行此训练的核心超参数非常直观：

- 采样参数：拒绝采样直接依赖于从模型收到的补全。RS的常见设置包括高于零的温度，例如在0.7和1.0之间，以及其他对参数如top-p或top-k采样的修改。

- 每次提示的完成次数：拒绝采样的成功实现包括每个提示10到30次或更多次的完成。使用太少的完成次数将使训练结果偏向或噪声。
- 指令调整细节：在RS期间没有发布明确的指令调整训练细节。他们可能使用了与模型初始指令调整阶段略有不同的设置。
- 异构模型生成：一些拒绝采样的实现包括从多个模型生成，而不仅仅是即将训练的当前模型。如何做到这一点的好做法尚未确立。
- 奖励模型训练：所使用的奖励模型将对最终结果产生重大影响。有关奖励模型训练的更多资源，请参阅相关章节。

10.1.4.1 实现技巧

- 在进行批量奖励模型推理时，您可以按长度对分词后的补全进行排序，以便批次具有相似长度。这样可以消除对许多填充标记进行推理的需要，并以轻微的实现复杂性为代价提高吞吐量。

10.2 相关：N中最佳采样

Best-of-N (BoN) 抽样通常被用作与 RLHF 方法相对的基线。重要的是要记住，BoN *does not* 修改了底层模型，但是一种抽样技术。对于这个问题，将 BoN 抽样与在线训练方法（如 PPO）进行比较在某些情况下仍然有效。例如，当运行 BoN 抽样相对于任何其他策略时，您仍然可以测量 KL 距离。

这里，我们将展示当使用简单的BoN采样对一个提示进行采样时，上述两种选择标准是等效的。

设 R 为我们单个提示的 N 个补全的奖励向量：

$$R = [r_1, r_2, \dots, r_N] \quad (29)$$

r_j 代表第 j 次完成的奖励。

使用 `th e argmax` 方法，我们选择 p 的最佳补全 rompt:

$$S(R) = \arg \max_{j \in [1, N]} r_j \quad (30)$$

使用Top-K方法通常使用Top-1，减少到相同的方法。

11 强化学习（即策略梯度算法）

算法使RLHF在语言模型中流行起来的是策略梯度强化学习算法。这些算法，如近端策略优化（PPO）、组相对策略优化（GRPO）和REINFORCE，使用最近生成的样本来更新其模型，而不是将分数存储在重放缓冲区中。在本节中，我们将介绍策略梯度算法的基本原理以及它们在现代RLHF框架中的应用。

在机器学习层面，本节是RLHF过程中最复杂的主题。尽管如此，与大多数现代AI模型一样，其成功最大的决定因素是提供给过程的输入数据。

最受欢迎的用于RLHF的算法随着时间的推移而发展。当RLHF随着ChatGPT出现时，人们普遍知道它们使用PPO的变体，许多初步工作都是基于此构建的。随着时间的推移，多个研究项目展示了REINFORCE风格算法[141] [125]的潜力，这种算法因其比PPO更简单（没有奖励模型，节省内存，因此所需的GPU数量更少）以及更简单的价值估计（没有广义优势估计GAE，这是一种用于在策略梯度算法中计算优势以减少方差的方法）而备受推崇。出现了更多算法，包括群相对策略优化，这种算法在推理任务中特别受欢迎，但总的来说，许多这些算法可以调整以适应特定任务。在本章中，我们介绍了核心策略梯度设置以及上述三个算法，因为它们在建立规范RLHF文献中的核心作用。

对于符号的定义，参见问题设置章节

r.

11.1 策略梯度算法

强化学习算法旨在最大化状态轨迹 $s \in \mathcal{S}$ 和动作轨迹 $a \in \mathcal{A}$ (上的未来折现奖励（有关更多符号，参见第3章，定义))。代理的目标，通常称为*return*，是在给定时间 t 处折现的未来奖励之和（其中 $\gamma \in [0, 1)$ 是一个优先考虑近期奖励的因素）：

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \quad (31)$$

返回定义也可以估计为：

$$G_t = \gamma G_{t+1} + R_{t+1}. \quad (32)$$

这是学习一个价值函数 $V(s)$ 的基础，该函数是在给定当前状态下估计的未来回报：

$$V(s) = \mathbb{E}[G_t | S_t = s]. \quad (33)$$

所有策略梯度算法都解决从特定策略诱导出的这样一个值函数 $\pi(a|s)$ 的目标。

$d_\pi(s)$ 是由策略 $\pi(a|s)$ 诱导的状态的平稳分布，优化定义为：

$$J(\theta) = \sum_s d_\pi(s) V_\pi(s), \quad (34)$$

政策梯度算法的核心是计算相对于当前策略的有限时间期望回报的梯度。利用这个期望回报 J ，参数更新可以按以下方式计算，其中 α 是学习率：

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \quad (35)$$

核心实现细节是如何计算该梯度。

另一种表述我们想要最大化的强化学习目标的方法如下：

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} [R(\tau)], \quad (36)$$

在 $\tau = (s_0, a_0, s_1, a_1, \dots)$ 是轨迹且 $R(\tau) = \sum_{t=0}^{\infty} r_t$ 是轨迹的总奖励的情况下。或者，我们可以将期望值写成对所有可能轨迹的积分：

$$J(\theta) = \int_{\tau} p_\theta(\tau) R(\tau) d\tau \quad (37)$$

请注意，我们可以将轨迹概率表示如下：

$$p_\theta(\tau) = p(s_0) \prod_{t=0}^{\infty} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t), \quad (38)$$

如果我们对目标函数（等式36）关于策略参数 θ 求梯度：

$$\nabla_\theta J(\theta) = \int_{\tau} \nabla_\theta p_\theta(\tau) R(\tau) d\tau \quad (39)$$

注意，我们可以使用对数导数技巧将积分的梯度重写为期望：

$$\begin{aligned} \nabla_\theta \log p_\theta(\tau) &= \frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)} && \text{(from chain rule)} \\ \implies \nabla_\theta p_\theta(\tau) &= p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) && \text{(rearranging)} \end{aligned} \quad (40)$$

使用这个对数导数技巧：

$$\begin{aligned} \nabla_\theta J(\theta) &= \int_{\tau} \nabla_\theta p_\theta(\tau) R(\tau) d\tau \\ &= \int_{\tau} p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) R(\tau) d\tau \\ &= \mathbb{E}_{\tau \sim \pi_\theta} [\nabla_\theta \log p_\theta(\tau) R(\tau)] \end{aligned} \quad (41)$$

Expanding the log probability of the trajectory:

$$\log p_{\theta}(\tau) = \log p(s_0) + \sum_{t=0}^{\infty} \log \pi_{\theta}(a_t|s_t) + \sum_{t=0}^{\infty} \log p(s_{t+1}|s_t, a_t)$$

现在，如果我们对上述内容求梯度，我们得到：

- ∇_{θ} 对数 $p(s_0) = 0$ (初始状态不依赖于 θ)
- ∇_{θ} 对数 $p(s_{t+1}|s_t, a_t) = 0$ (环境转换动力学不依赖于 θ)
- 仅 $\nabla_{\theta} \log \pi_{\theta}(a_t|s_t)$ 保留

因此，轨迹对数概率的梯度简化为：

$$\nabla_{\theta} \log p_{\theta}(\tau) = \sum_{t=0}^{\infty} \nabla_{\theta} \log \pi_{\theta}(a_t|s_t)$$

将此代入方程41，我们得到：

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^{\infty} \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) R(\tau) \right]$$

经常，人们使用策略梯度的更一般形式：

$$g = \nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^{\infty} \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) \Psi_t \right] \quad (42)$$

在 Ψ_t 可以是以下情况（其中奖励也经常通过 γ 进行折扣）时，采用自 Schulman 等人 2015 年 [142] 的分类法：

1. $R(\tau) = \sum_{t=0}^{\infty} r_t$: 轨迹的总奖励。
2. $\sum_{t'=t}^{\infty} r_{t'}$: 执行动作 a_t 后的奖励，也称为回报， G 。
3. $\sum_{t'=t}^{\infty} r_{t'} - b(s_t)$: 先前公式的基线版本。
4. $Q^{\pi}(s_t, a_t)$: 状态-动作值函数。
5. $A^{\pi}(s_t, a_t)$: 优势函数，如果可以准确计算，则产生最低可能的理论方差。
6. $r_t + V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$: 时间差分 (TD) 残差。

baseline 是一个 v 值用于减少策略更新的方差（更多内容） 这以下）。

对于语言模型，其中一些概念并不那么有意义。例如，我们知道对于确定性策略，值函数定义为 $V(s) = \max_a Q(s, a)$ ，或者对于随机策略定义为 $V(s) = \mathbb{E}_{a \sim \pi(a|s)}[Q(s, a)]$ 。如果我们定义 $s + a$ 为对提示 a 的延续 s ，那么 $Q(s, a) = V(s + a)$ 提供了一种不同的优势技巧：

$$A(s, a) = Q(s, a) - V(s) = V(s + a) - V(s) = r + \gamma V(s + a) - V(s) \quad (43)$$

这是奖励、提示的价值以及整个话语的折现价值的组合。

11.1.1 香草策略梯度

The vanilla policy gradient implementation optimizes the above expression for $J(\theta)$ by differentiating with respect to the policy parameters. A simple version, with respect to the overall return, is: 标函数的范化策略梯度实现通过相对于策略参数的微分来优化上述表达式对于 $J(\theta)$ 。关于整体回报的简单版本是:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) R_t \right] \quad (44)$$

一个常见的vanilla策略梯度算法问题是梯度更新的高方差，这可以通过多种方式缓解。高方差来自从估计环境中的小批量rollout的回报 G 计算梯度更新，这些rollout往往容易受到噪声的影响（例如，从具有温度 > 0 的语言模型生成时的随机性），特别是在稀疏奖励的情况下。为了缓解这一点，使用了各种技术来归一化价值估计，称为 *baselines*。基线通过多种方式完成此操作，有效地通过相对于下游动作的状态值进行归一化（例如，在优势的情况下，这是Q值和价值的差异）。最简单的基线是奖励批次的平均值或移动平均值。即使这些基线也可以去偏梯度，使 $\mathbb{E}_{a \sim \pi(a|s)} [\nabla_{\theta} \log \pi_{\theta}(a|s)] = 0$ ，从而显著提高学习信号。

许多本章讨论的政策梯度算法建立在策略梯度的优势公式之上：

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) A^{\pi_{\theta}}(s_t, a_t) \right] \quad (45)$$

11.1.2 REINFORCE

算法REINFORCE可能是一个缩写词，但它所代表的算法组件对现代强化学习算法非常相关。在开创性的论文

Simple statistical gradient-following algorithms for connectionist reinforcement learning [143] 中定义：{ v^* }

该名称是“奖励增量 = 非负因子 X 偏移”的缩写
强化X特征资格。”

这三个组成部分是如何进行 *reward increment*，即策略梯度步骤。更新规则有三个部分：

1. 非负因子：这是必须为正数的学习率（步长），例如下面的 α 。
2. 偏移强化：这是基线 b 或其他用于提高稳定性的奖励归一化因子。
3. 特征资格：这是学习如何按令牌归因。它可以是通用值， e 每个参数，但通常是现代方程中策略的对数概率。

因此，这个形式看起来相当熟悉：

$$\Delta_{\theta} = \alpha(r - b)e \quad (46)$$

使用更现代的符号和广义返回 G ，REINFORCE算子表示为：

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (G_t - b(s_t)) \right], \quad (47)$$

这里，值 $G_t - b(s_t)$ 是当前状态下策略的 *advantage*，因此我们可以将策略梯度重新表述为一种形式，我们将在后续内容中继续使用优势 A ：

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) A_t \right], \quad (48)$$

REINFORCE 是 vanilla 策略梯度的特定实现，它使用蒙特卡洛梯度估计器。

11.1.2.1 REINFORCE 留一法 (RLOO) 与标准 REINFORCE 的核心实现细节不同之处在于，它通过计算批次中 *other* 个样本的平均奖励来计算基线，而不是计算批次中所有奖励的平均值 [144], [141], [145]。

关键在于，这仅在为每个状态（提示）生成多个轨迹（完成）时才有效，这在使用RL微调语言模型的多个领域中是常见做法。

具体来说，对于REINFORCE Leave-One-Out (RLOO)基线，给定 K 个采样的轨迹（在提示下采取的动作） a_1, \dots, a_K ，对于给定的提示 s ，我们明确地将基线定义为以下 *per-prompt*：

$$b(s, a_k) = \frac{1}{K-1} \sum_{i=1, i \neq k}^K R(s, a_i), \quad (49)$$

导致以下优势：

$$A(s, a_k) = R(s, a_k) - b(s, a_k). \quad (50)$$

等效地，这可以表示为：

$$A(s, a_k) = \frac{K}{K-1} \left(R(s, a_k) - \frac{1}{K} \sum_{i=1}^K R(s, a_i) \right). \quad (51)$$

这是一个简单、低方差优势更新，与GRPO非常相似，稍后将进行讨论，其中使用了不同位置的KL惩罚，并且

没有步长裁剪。具体来说，标准的GRPO实现将KL惩罚应用于损失级别，而RLOO或传统策略梯度的推导将KL惩罚应用于奖励本身。随着从RLHF到推理和RLVR的转变，KL惩罚的整体使用频率有所下降，许多RLHF代码的推理适配甚至完全关闭了它们。尽管如此，RLOO的优势可以与PPO的裁剪相结合，展示了这些算法的相似性。

RLOO和其他不使用价值网络的算法将序列的优势（或奖励）分配给每个标记以进行损失计算。使用学习价值网络的算法，如PPO，为每个标记分配不同的值，从EOS标记达到的最终奖励中折扣。在KL距离惩罚下，RLOO将每个标记的KL总和聚合到完成和折叠中，将该标量折叠到序列奖励中，因此结果优势广播到所有标记。PPO在计算 A_t 之前从每个标记的奖励中减去每个标记的KL，从而进行标记级信用分配。GRPO通常保留序列级优势，但将单独的每个标记项添加到损失中，而不是从奖励中减去它。这些细节和权衡将在本章后面讨论。

11.1.3 近端策略优化

近端策略优化（PPO）[146]是深度强化学习成功的基础算法之一（如OpenAI的DOTA 5 [147]和大量研究）。PPO最大化的目标，关于优势和策略概率，如下所示：

$$J(\theta) = \min \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)} A, \text{clip} \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)}, 1 - \varepsilon, 1 + \varepsilon \right) A \right). \quad (52)$$

目标通常通过简单地添加负号转换为损失函数，这使得优化器寻求使其尽可能负。

对于语言模型，目标（或损失）是按令牌计算的，直观上可以基于如何计算自回归预测整个序列的概率——通过概率的乘积。从那里，常见的实现是使用 *log-probabilities*，这使得计算变得远更易于处理。

$$J(\theta) = \frac{1}{|a|} \sum_{t=0}^{|a|} \min \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t, \text{clip} \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1 - \varepsilon, 1 + \varepsilon \right) A_t \right). \quad (53)$$

这是PPO的按令牌版本，也适用于其他策略梯度方法，但将在本章的实现部分进一步探讨。在这里，动作中按令牌数量平均的术语 $\frac{1}{|a|}$ 来自常见的实现实践，但不在损失的正式推导（见[148]）中。

这里我们将解释该损失函数在给定各种优势和策略比率时触发的不同情况。在实现层面，PPO的内层计算涉及标准策略梯度和裁剪策略梯度。

为了理解不同情况如何出现，我们可以定义策略比率为：

$$R(\theta) = \frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)} \quad (54)$$

策略比率是PPO及其相关算法的核心。它从计算策略的梯度中产生，并以非常直观的方式控制参数更新。对于任何一批数据，策略比率在该批数据的第一次梯度步骤中从1开始，因为在此点 π_{θ} 与 $\pi_{\theta_{old}}$ 相同。然后，在下一个梯度步骤中，如果该梯度步骤增加了与相关正优势的某些标记的可能性，策略比率将大于1；否则，将小于1。常见的做法是在更新 $\pi_{\theta_{old}}$ 之前，使用策略梯度算法在每个批次中执行1-4次梯度步骤。

11.1.3.1 价值函数和PPO PPO中的价值函数是模型的一个额外副本，用于预测每个标记的价值。在传统强化学习中，一个标记（或状态）的价值是预测从那一刻起未来的回报，通常带有折扣。这个价值被用作一个学习基线，代表与REINFORCE（不需要学习价值网络）一起使用的简单蒙特卡洛版本的发展。这突出了PPO是如何在多个形式上成为REINFORCE和普通策略梯度的演变，包括优化形式、基线等。在实践中，使用PPO和其他用于语言模型的算法时，这是预测每个标记在扣除KL惩罚（每个标记的损失包括传统上从奖励中来的KL，如前所述）后的回报。

存在几种不同的方法（或目标）用于学习值函数。广义优势估计（GAE）被认为是现代系统中的最先进和规范实现，但它通过在多个步骤中计算值预测误差而增加了复杂性——请参阅本章后面的GAE部分。值函数还可以通过使用用于更新策略的rollouts的蒙特卡洛估计来学习。PPO有两个损失——一个用于学习值函数，另一个用于使用该值函数来更新策略。

以下是一个值网络损失的简单示例实现。

```
# Basic PPO critic targets & loss (no GAE)
#
# B: Batch Size
# L: Completion Length
# Inputs:
#   rewards: (B, L) post-KL per-token rewards; EOS row includes
#           outcome
#   done_mask: (B, L) 1.0 at terminal token (EOS or truncation if
#           penalized), else 0.0
#   completion_mask: (B, L) 1.0 on response tokens to supervise (
#           ignore the prompt)
#   values: (B, L) current critic predictions V_theta(s_t)
#           because a value network is a running update
#   old_values: (B, L) critic predictions at rollout time V_{theta_old}
#           (s_t)
#   gamma: discount factor, float (often 1.0 for LM RLHF)
#   epsilon_v: float value clip range (e.g., 0.2), similar to PPO Loss
#           Update itself, optional
```

```

#
# Returns:
#   value_loss: scalar; advantages: (B, L) detached (for policy loss)

B, L = rewards.shape

# 1) Monte Carlo returns per token (reset at terminals)
# Apply discounting, if enabled
returns = torch.zeros_like(rewards)
running = torch.zeros(B, device=rewards.device, dtype=rewards.dtype)
for t in reversed(range(L)):
    running = rewards[:, t] + gamma * (1.0 - done_mask[:, t]) *
        running
    returns[:, t] = running

targets = returns # y_t = G_t (post-KL)

# 2) PPO-style value clipping (optional)
v_pred = values
v_old = old_values
v_clip = torch.clamp(v_pred, v_old - epsilon_v, v_old + epsilon_v)

vf_unclipped = 0.5 * (v_pred - targets) ** 2
vf_clipped = 0.5 * (v_clip - targets) ** 2
vf_loss_tok = torch.max(vf_unclipped, vf_clipped)

# 3) Mask to response tokens and aggregate
denom = completion_mask.sum(dim=1).clamp_min(1)
value_loss = ((vf_loss_tok * completion_mask).sum(dim=1) / denom).mean()
()

# 4) Advantages for policy loss (no GAE): A_t = G_t - V(s_t)
advantages = (targets - v_pred).detach()

# The value loss is applied later, often with the PG loss, e.g.
# total_loss = policy_loss + vf_coef * value_loss

```

11.1.3.2 理解PPO目标总体而言，PPO目标可以通过目标与策略比率的图表的两条线来可视化，如图14所示。通过改变采样动作的概率来最大化PPO目标。在数值上，通过巧妙地使用最小操作，目标控制了正负优势情况，使得更新最多被推离策略比率1的 ϵ 距离。

在信任区域内，PPO的操作与其他策略梯度算法相同。这是设计使然！信任区域是一个用于限制PPO及其同类算法最大步长以保持更新稳定性的概念。PPO算法的核心，即clip和min/max函数，是用来定义这个区域的。目标函数在其外部变得平坦。

“信任区域”这一概念源于数值优化文献[149]，但在深度强化学习领域，通过信任区域策略优化（TRPO）算法而流行起来

这是被接受为PPO[150]的前驱。

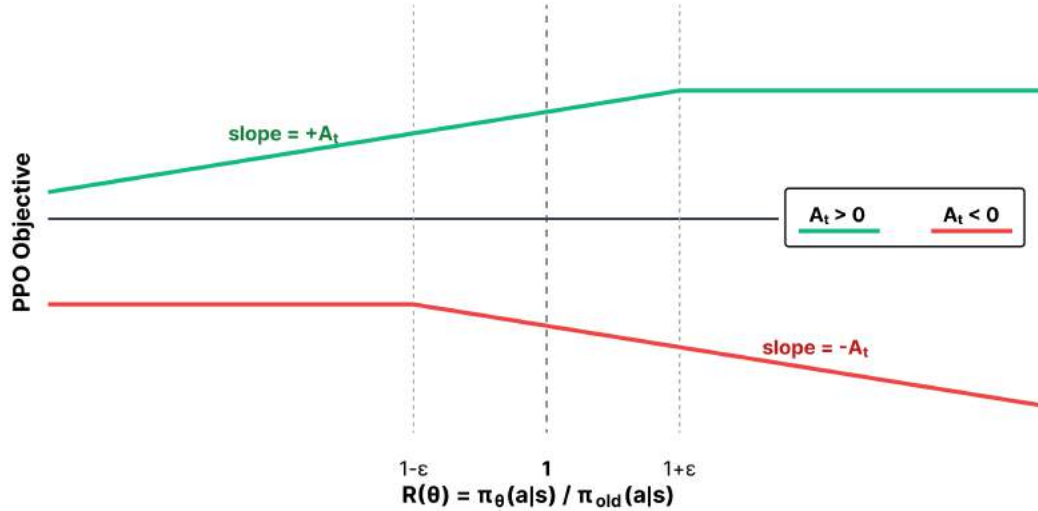


图14：假设优势的PPO目标不同区域的可视化。

政策比率与优势可以以几种不同的配置同时出现。我们将案例分为两组：正优势和负优势。

正优势 ($A_t > 0$)

这意味着所采取的行动根据价值函数是有益的，我们希望增加在未来采取该行动的可能性。现在，让我们看看策略比率 $R(\theta)$ 的不同情况：

1. $R(\theta) < 1 - \epsilon$:

- 解释：新政策下采取行动的可能性低于旧政策
- 未剪辑项: $R(\theta)A_t$
- 剪裁项: $(1 - \epsilon)A_t$
- 目标: $R(\theta)A_t$
- 梯度: $\nabla_{\theta} R(\theta)A_t \neq 0$
- 发生什么：正常策略梯度更新 - 增加动作的可能性

2. $1 - \epsilon \leq R(\theta) \leq 1 + \epsilon$:

- 解释：在新的政策下，行动与旧政策几乎同样可能
- 未剪辑项: $R(\theta)A_t$
- 剪裁项: $R(\theta)A_t$
- 目标: $R(\theta)A_t$
- 梯度: $\nabla_{\theta} R(\theta)A_t \neq 0$
- 发生什么：正常策略梯度更新 - 增加动作的可能性

3. $1 + \epsilon < R(\theta)$:

- 解释：新政策下采取行动的可能性比旧政策下更大
- 未剪辑项： $R(\theta)A_t$
- 剪裁项： $(1 + \epsilon)A_t$
- 目标： $(1 + \epsilon)A_t$
- 梯度： $\nabla_{\theta}(1 + \epsilon)A_t = 0$
- 发生什么：无更新 - 根据新政策，该操作已经更有可能

总结来说，当优势为正 ($A_t > 0$) 时，我们希望提高动作的概率。因此：

- 我们仅在 $\pi_{\text{new}}(a) \leq (1 + \epsilon)\pi_{\text{old}}(a)$ 的情况下执行梯度步骤。直观上，我们希望提高动作的概率，因为奖励是正的，但又不希望提高太多，以免使其变得极其可能。
- 关键在于，当 $\pi_{\text{new}}(a) > (1 + \epsilon)\pi_{\text{old}}(a)$ 时，我们不会执行任何更新，并且剪裁目标函数的梯度为 0。直观上，动作已经通过新策略得到了更充分的表达，所以我们不想过度强化它。

负优势 ($A_t < 0$)

这意味着根据价值函数，所采取的行动是有害的，我们希望降低未来采取该行动的可能性。现在，让我们看看策略比率 $R(\theta)$ 的不同情况：

1. $R(\theta) < 1 - \epsilon$ ：

- 解释：新政策下采取行动的可能性低于旧政策
- 未剪辑项： $R(\theta)A_t$
- 剪裁项： $(1 - \epsilon)A_t$
- 目标： $(1 - \epsilon)A_t$
- 梯度： $\nabla_{\theta}(1 - \epsilon)A_t = 0$
- 发生什么：无更新 - 在新政策下，采取行动的可能性已经较低

2. $1 - \epsilon \leq R(\theta) \leq 1 + \epsilon$ ：

- 解释：在新的政策下，行动与旧政策几乎同样可能
- 未剪辑项： $R(\theta)A_t$
- 剪裁项： $R(\theta)A_t$
- 目标： $R(\theta)A_t$
- 梯度： $\nabla_{\theta}R(\theta)A_t \neq 0$
- 发生什么：正常策略梯度更新 - 减少动作的可能性

3. $1 + \epsilon < R(\theta)$ ：

- 解释：新政策下采取行动的可能性比旧政策下更大
- 未剪辑项： $R(\theta)A_t$
- 剪裁项： $(1 + \epsilon)A_t$
- 目标： $R(\theta)A_t$
- 梯度： $\nabla_{\theta}R(\theta)A_t \neq 0$
- 发生什么：正常策略梯度更新 - 减少动作的可能性

总结来说，当优势为负（ $A_t < 0$ ）时，我们希望降低动作的概率。因此：

- 我们仅在 $\pi_{\text{new}}(a) \geq (1 - \epsilon)\pi_{\text{old}}(a)$ 的情况下执行梯度步骤。直观上，我们希望降低动作的概率，因为奖励是负的，我们这样做与优势成正比。
- 关键在于，当 $\pi_{\text{new}}(a) < (1 - \epsilon)\pi_{\text{old}}(a)$ 时，我们不会执行任何更新，并且剪裁目标函数的梯度为 0。直观上，在新的策略下，该动作已经不太可能发生，所以我们不想过度抑制它。

至关重要的是要记住，在信任区域内，PPO与标准形式的策略梯度大致相同。

11.1.4 群组相对策略优化

组相对策略优化（GRPO）在DeepSeekMath [151]中引入，并在其他DeepSeek工作中使用，例如DeepSeek-V3 [152]和DeepSeek-R1 [60]。GRPO可以看作是受PPO启发的算法，具有非常相似的代理损失，但它避免了使用原始策略语言模型（或另一个用于初始化的检查点）来学习价值函数。这带来了两个假设的好处：

1. 避免从LM骨干中学习值函数的挑战，其中研究尚未建立最佳实践。
2. 通过不需要在内存中保留另一组模型权重来节省内存。

GRPO通过简化价值估计并将相同的价值分配给每个情节中的每个标记（即在完成提示时，每个标记都分配相同的价值，而不是标准价值函数中的折扣奖励）来做到这一点，通过估计优势或基线。估计是通过收集来自相同初始状态/提示（ s ）的多个完成（ a_i ）和奖励（ r_i ）来进行的，即蒙特卡洛估计。

为了正式表述，GRPO目标与上面的PPO目标非常相似。对于GRPO，目标（或损失）是在一组完成项 $\{a_1, a_2, \dots, a_G\}$ 上针对给定提示 s 累积的。在这里，我们展示了GRPO目标：

$$J(\theta) = \frac{1}{G} \sum_{i=1}^G \left(\min \left(\frac{\pi_{\theta}(a_i|s)}{\pi_{\theta_{old}}(a_i|s)} A_i, \text{clip} \left(\frac{\pi_{\theta}(a_i|s)}{\pi_{\theta_{old}}(a_i|s)}, 1 - \epsilon, 1 + \epsilon \right) A_i \right) - \beta D_{KL}(\pi_{\theta} || \pi_{ref}) \right). \quad (55)$$

请注意，与PPO相比，GRPO的标准实现将KL距离包含在损失中。如上所述，我们可以将其扩展为每个标记的计算：

$$J(\theta) = \frac{1}{G} \sum_{i=1}^G \frac{1}{|a_i|} \sum_{t=1}^{|a_i|} \left(\min \left(\frac{\pi_{\theta}(a_{i,t}|s_i)}{\pi_{\theta_{old}}(a_{i,t}|s_i)} A_{i,t}, \text{clip} \left(\frac{\pi_{\theta}(a_{i,t}|s_i)}{\pi_{\theta_{old}}(a_{i,t}|s_i)}, 1 - \epsilon, 1 + \epsilon \right) A_{i,t} \right) - \beta D_{KL}(\pi_{\theta}(\cdot|s_i) || \pi_{ref}(\cdot|s_i)) \right) \quad (56)$$

具有完成索引 i 的优势计算：

$$A_i = \frac{r_i - \text{mean}(r_1, r_2, \dots, r_G)}{\text{std}(r_1, r_2, \dots, r_G)}. \quad (57)$$

直观上，GRPO更新是在一个批次内比较单个问题的多个答案。模型学习变得更像标记为正确的答案，而不太像其他答案。这是一种非常简单的方式来计算优势，这是衡量特定动作在给定状态下比平均动作好多少的度量。与PPO、REINFORCE以及广泛使用的RLHF相比，GRPO通常在每次提示时使用更多的样本，因为优势完全是关于完成相对于同一提示中其他完成的相对价值。在这里，当前策略为给定的提示生成多个响应，并且组内GRPO优势估计提供了宝贵的上下文。PPO和vanilla策略梯度算法旨在准确估计每个完成的奖励（实际上，在某些情况下，更多的完成对价值估计的改善作用很小）。GRPO及其变体特别适合现代语言模型工具，在这些工具中，对给定提示的多个完成非常自然（尤其是与，例如，在机器人任务中从一组环境状态中获得的多个动作相比）。

GRPO的优势计算在其偏差方面存在权衡。通过标准差进行归一化对于一批答案正确性变化较小的题目是有益的。对于要么几乎全部正确要么全部错误的回答，标准差将较低，优势将较高。[148]建议在存在这种偏差的情况下删除标准差项，但这会以降低所有错误答案中只有少数正确答案的题目权重为代价，这可能会被视为有价值的信号。

eq. 57 是在处理结果监督（无论是标准奖励模型还是单个可验证奖励）时 GRPO 的实现，而在过程监督下则需要不同的实现。在这种情况下，GRPO 计算优势作为以下推理步骤中归一化奖励的总和。

最后，GRPO的优势估计也可以应用于更传统的策略梯度（例如REINFORCE）版本，但不是规范形式。作为这些算法相互交织的例子，我们可以表明GRPO的一个变体，即Dr. GRPO（GRPO Done Right）[148]中的优势估计，相当于RLOO估计，直到一个常数缩放因子（由于实现细节来归一化优势，通常并不重要）。Dr. GRPO从等式57中移除了标准差归一化项——请注意，这也缩放优势 up ，这相当于增加了GRPO在答案分数方差样本上的学习率。这解决了对低奖励方差问题（即几乎所有答案都是正确或错误的）的偏差，但可能以学习重要问题的成本为代价，这些问题只有一个样本得到正确答案。在大小为 G 的组内，Dr. GRPO的优势对于完成 i 定义为：

$$\tilde{A}_i = r_i - \text{mean}(r_1, r_2, \dots, r_G) = r_i - \frac{1}{G} \sum_{j=1}^G r_j \quad (58)$$

这里，在相同的记号下，我们可以回忆RLOO优势估计为：

$$A_i^{\text{RLOO}} = r_i - \frac{1}{G-1} \sum_{j=1, j \neq i}^G r_j \quad (59)$$

因此，如果我们把GRPO优势定义乘以 $\frac{G}{G-1}$ ，我们可以看到一个缩放等价：

$$\begin{aligned} \frac{G}{G-1} \tilde{A}_i &= \frac{G}{G-1} \left(r_i - \frac{1}{G} \sum_{j=1}^G r_j \right) \\ &= \frac{G}{G-1} r_i - \frac{1}{G-1} \sum_{j=1}^G r_j \\ &= \frac{G}{G-1} r_i - \frac{1}{G-1} \sum_{j=1, j \neq i}^G r_j - \frac{1}{G-1} r_i \\ &= r_i \left(\frac{G}{G-1} - \frac{1}{G-1} \right) - \frac{1}{G-1} \sum_{j=1, j \neq i}^G r_j \\ &= r_i - \frac{1}{G-1} \sum_{j=1, j \neq i}^G r_j \\ &= A_i^{\text{RLOO}} \end{aligned} \quad (60)$$

11.2 实施

与最初深度强化学习文献相比，在这些算法中许多都是开发出来的，实现强化学习以优化语言模型或其他大型AI模型需要许多小的实现细节。在本节中，我们突出了一些区分流行算法实现的关键因素。

有许多其他小细节涉及到这次训练。例如，在进行基于语言模型的RLHF时，一个关键步骤是生成将被奖励模型评分的文本。在正常情况下，模型应该生成一个序列结束（EOS）标记，表示它已完成生成，但常见的做法是设置一个生成长度的上限，以有效地利用基础设施。RLHF的一个失败模式是模型在回答中经常被截断，导致奖励模型的评分偏离分布并变得不可预测。解决这个问题的方法是*only*在`eos_token`上运行奖励模型排名，否则对生成过长的模型进行惩罚。

流行的RLHF开源工具在算法实现细节上存在很大差异（参见[153]中的表10）。以下是一些未在此处涵盖的决定：

- 值网络初始化：PPO和其他类似算法使用的内部学习值网络可以从相同架构的不同模型或随机选择的权重开始。这可能会对性能产生重大影响。

- 奖励归一化、奖励白化以及/或优势白化：其中归一化将RM（或环境）中的所有值限制在0到1之间，这有助于提高学习稳定性；将奖励或优势估计白化为均匀协变量可以提供更强的稳定性提升。
- 不同的KL估计器：在复杂的语言模型中，精确计算模型之间的KL散度可能很复杂，因此使用了多个近似来代替精确计算[129]。
- KL控制器：PPO及其相关算法的原始实现具有针对特定KL的动态控制器，并根据最近测量值更改惩罚。大多数现代RLHF实现使用静态KL惩罚，但这也可以变化。

关于RLHF的实现细节，请参阅[154]。有关算法的更多信息，请参阅[155]。

11.2.1 政策梯度基础

一个简单的策略梯度实现，使用优势来估计梯度，为PPO和GRPO等高级算法做准备，如下所示：

```
pg_loss = -advantages * ratio
```

此处比率是新策略模型概率相对于参考模型的对数比率。

为了理解这个方程，了解可能出现在一批更新中的不同情况是有益的。记住，我们希望随着模型在任务上变得更好，损失要 *decrease*。

案例1：正优势，因此行动比状态的预期值更好。我们希望加强这一点。在这种情况下，模型会通过负号使这种情况更有可能发生。为此，它会增加对数比。正对数比，或标记的对数概率之和，意味着模型更有可能生成这些标记。

案例2：负优势，因此行动比状态的预期价值更差。这非常相似。在这里，如果新模型更有可能，损失将是正的，因此模型将尝试使其策略参数使这种完成的可能性降低。

案例3：无优势，因此无需更新。损失为零，不要更改策略模型。

11.2.2 损失聚合

实现任何基于语言模型的策略梯度算法时的问题是：你如何对KL距离和损失进行求和以设计不同类型的价值归因。

Most of the discussions in this section assume a token-level action, where the RL problem is formatted as a Markov Decision Process (MDP) rather than a bandit problem. In a bandit problem, all the tokens in an action will be given the same loss, which has been the default implementation for some algorithms such as Advantage-Leftover Lunch RL (A-LoL) [156]. The formulation between MDP and bandit is actually an implementation detail over how the

loss is aggregated per-sample. A bandit approach takes a mean that assigns the same loss to every token, which also aligns with DPO and other direct alignment algorithms' standard implementations.

Most of what follows adopts the **MDP (token-level)** view: each token (a_t) is an action with state (s_t) the running prefix. This enables **token-level credit assignment** via a value function ($V(s_t)$) (e.g., GAE [142]) and **per-token KL**. In contrast, the **bandit (sequence-level)** view treats the whole completion as a single action with one scalar reward (R); in code, this is equivalent to computing a **sequence-level advantage** ($A_{\{seq\}}$) and multiplying it by the (length-normalized) sum of per-token log-probs, thereby **broadcasting the same learning signal to every token**. RLOO and the GRPO advantage operate in this bandit regime [145] [141] [151]; PPO with a learned value network uses the MDP regime [146]. This bandit view also matches direct-alignment objectives such as DPO and A-LoL [156]. Note that GRPO typically keeps the bandit-style advantage **and** adds a separate per-token KL loss, whereas PPO/RLOO often subtract KL inside the reward.

考虑一个例子，其中我们有以下变量，具有批量大小 B 和序列长度 L 。

```
advantages # [B, 1]
per_token_probability_ratios # [B, L]
```

我们可以将损失近似为上述形式，通过批乘法计算 $pg_loss = -advantages * ratio$ 。将它们相乘是广播每个批次的完成优势（如在结果奖励设置中，而不是每个token的价值模型设置）以保持相同。然后，它们被乘以每个token的概率对数比率。

在采用值网络的情况下，很容易看出不同的损失可以表现出很大的差异。当使用结果奖励时，优势被设置为每个标记相同，因此每个标记概率的差异对于策略梯度学习动态至关重要。

在以下GRPO和PPO的实现中，损失是对完成中的标记求和的： $\{v^*\}$

```
sequence_loss = ((per_token_loss * completion_mask).sum(dim=1) / \
                  completion_mask.sum(dim=1)).mean()
```

注意，`completion_mask` 仅仅是一个由 1s 和 0s 组成的矩阵，其中在损失（此处为 0s）中掩码了提示标记，因为我们不希望模型学习预测它们的值，从而学习它们的行为。上述操作与 `masked_mean` 操作非常相似。另一种方法是逐个标记进行平均。

```
token_loss = ((per_token_loss * completion_mask).sum() / \
               completion_mask.sum())
```

直观上，似乎对序列进行平均是最好的，因为我们试图奖励模型对 *outcomes* 的识别，而特定的标记并不那么重要。这可能会引入微妙的形式的偏见。考虑两个不同长度的序列，分别赋予两个不同的优势 a_1 和 a_2 。

```
seq_1_adv = [a_1, a_1, a_1, a_1, a_1] # 5 tokens
seq_2_adv = [a_2, a_2, a_2, a_2, a_2, a_2, a_2, a_2, a_2, a_2] # 10
              tokens
```

现在，考虑每个序列中的最后一个标记是否对优势为正很重要，因此在每个批次的多个梯度步骤中增加。当你将这些转换为每个标记的损失时，可能会得到类似于以下的结果：

```
seq_1_losses = [1, 1, 1, 1, 10] # 5 tokens
seq_2_losses = [1, 1, 1, 1, 1, 1, 1, 1, 1, 10] # 10 tokens
```

如果我们对这些序列进行平均，我们将得到以下数字：

```
seq_1_loss = 2.8
seq_2_loss = 1.9
```

如果我们对这些值平均加权，权重序列相等，我们得到损失为2.35。如果我们相反，将损失平均分配给每个标记，损失将通过求和所有每个标记的损失并按长度归一化来计算，在这种情况下将是2.27。如果序列之间的差异更大，这两个损失值可能会有很大差异。

对于如何损失聚合改变每个标记和每个示例的损失得更完整示例，请参阅下面的脚本，该脚本计算一个包含两个样本（一个长一个短）的玩具批次的损失。该示例使用了三种损失聚合技术：`masked_mean`对应于每个样本长度的归一化，`DAPO` [157] 中提出的具有每个批次标记级别归一化的损失，`masked_mean_token_level`，以及`masked_sum_result`使用来自Dr. GRPO [148] 的最大长度的固定长度归一化。

```
from typing import Optional
import torch

def masked_mean(values: torch.Tensor, mask: torch.Tensor, axis:
    Optional[int] = None) -> torch.Tensor:
    """Compute mean of tensor with a masked values."""
    if axis is not None:
        return (values * mask).sum(axis=axis) / mask.sum(axis=axis)
    else:
        return (values * mask).sum() / mask.sum()

def masked_sum(
    values: torch.Tensor,
    mask: torch.Tensor,
    axis: Optional[bool] = None,
    constant_normalizer: float = 1.0,
) -> torch.Tensor:
    """Compute sum of tensor with a masked values. Use a constant to
    normalize."""
    if axis is not None:
        return (values * mask).sum(axis=axis) / constant_normalizer
    else:
        return (values * mask).sum() / constant_normalizer

ratio = torch.tensor([
    [1., 1, 1, 1, 1, 1, 1, 1],
    [1, 1, 1, 1, 1, 1, 1, 1],
], requires_grad=True)
```

```

advs = torch.tensor([
    [2, 2, 2, 2, 2, 2, 2,],
    [2, 2, 2, 2, 2, 2, 2,],
])

masks = torch.tensor([
    # generation 1: 4 tokens
    [1, 1, 1, 1, 0, 0, 0,],
    # generation 2: 7 tokens
    [1, 1, 1, 1, 1, 1, 1,],
])

max_gen_len = 7

masked_mean_result = masked_mean(ratio * advs, masks, axis=1)
masked_mean_token_level = masked_mean(ratio, masks, axis=None)
masked_sum_result = masked_sum(ratio * advs, masks, axis=1,
    constant_normalizer=max_gen_len)

print("masked_mean", masked_mean_result)
print("masked_sum", masked_sum_result)
print("masked_mean_token_level", masked_mean_token_level)

# masked_mean tensor([2., 2.], grad_fn=<DivBackward0>)
# masked_sum tensor([1.1429, 2.0000], grad_fn=<DivBackward0>)
# masked_mean_token_level tensor(1., grad_fn=<DivBackward0>)

masked_mean_result.mean().backward()
print("ratio.grad", ratio.grad)
ratio.grad.zero_()
# ratio.grad tensor([[0.2500, 0.2500, 0.2500, 0.2500, 0.0000, 0.0000,
    0.0000],
    # [0.1429, 0.1429, 0.1429, 0.1429, 0.1429, 0.1429, 0.1429]])

masked_sum_result.mean().backward()
print("ratio.grad", ratio.grad)
ratio.grad.zero_()
# ratio.grad tensor([[0.1429, 0.1429, 0.1429, 0.1429, 0.0000, 0.0000,
    0.0000],
    # [0.1429, 0.1429, 0.1429, 0.1429, 0.1429, 0.1429, 0.1429]])

masked_mean_token_level.mean().backward()
print("ratio.grad", ratio.grad)
# ratio.grad tensor([[0.0909, 0.0909, 0.0909, 0.0909, 0.0000, 0.0000,
    0.0000],
    # [0.0909, 0.0909, 0.0909, 0.0909, 0.0909, 0.0909, 0.0909]])

```

这里可以看到，对于默认的GRPO实现，`masked_mean`，短长度比长长度具有更大的每令牌梯度，而Dr. GRPO和DAPO的两种实现相互平衡。请注意，如果使用梯度累积，这些结果可能会有很大差异，其中梯度在多个小批量中求和后再进行取

向后一步。在这种情况下，短序列和长序列之间的平衡可以翻转。

另一种聚合损失的方法在[148]中讨论，其起源于预语言模型RL研究，其中每个token的损失都通过实验中设置的序列最大长度进行归一化。这将改变上述示例中每个token的损失在批次间的比较方式。

在实践中，最适合个人的在线学习设置的配置可能是最佳配置。在RLHF方法中，通常具有最佳数值稳定性和或最小损失方差的方法可能更受青睐。

11.2.3 异步性

默认情况下，策略梯度算法的实现被称为按策略执行，其中在更新模型之前，对代理（语言模型）采取的动作（生成）进行评分。策略梯度的理论推导依赖于所有动作都完全按策略进行，即模型始终与最新试验/展开的结果保持最新。在实践中，将训练与展开（即生成模型的推理）分开会显著减慢训练[158]（或技术上不可能）。因此，所有最近关于语言模型的实证结果都略偏离理论证明。实际上发生的是设计实际可行的算法和系统。

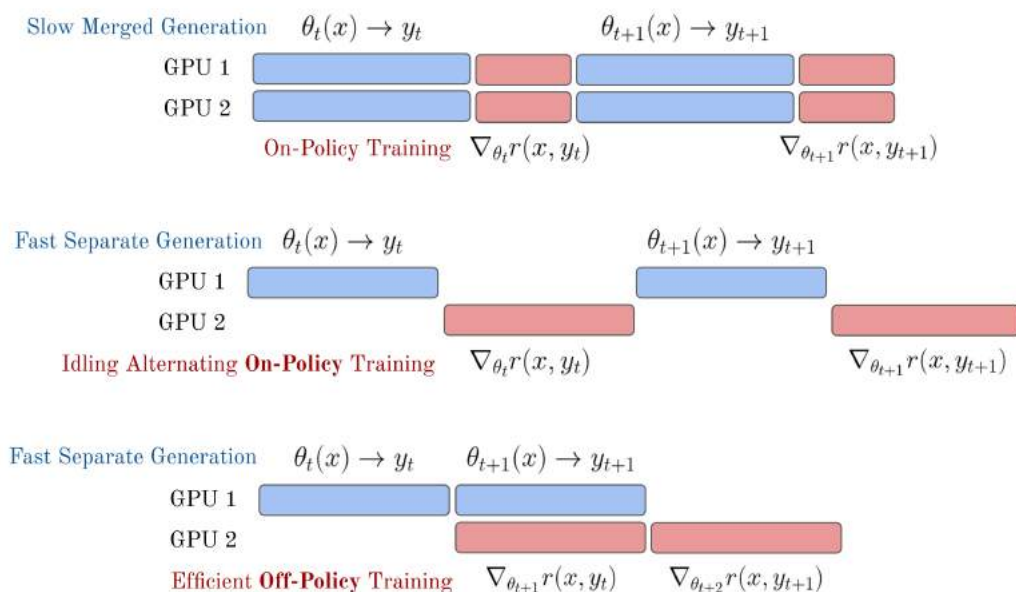


图15：同步或异步RL训练的生成-更新阶段的比较，遵循Noukhovitch等人2024年的方法。

常用的解决方案是在单独的GPU节点上不断运行推理和训练，使用专门设计的软件来高效运行两者。在流行的开源强化学习（RL）中，这是一种常见的做法。

工具用于语言模型是使用分布式进程管理库，如Ray，在策略梯度学习循环和运行高效推理循环（例如VLLM）之间传递信息。在使训练更加异步时面临的主要挑战是保持训练稳定和维持学习信号。

这些系统是根据假设近策略数据对于稳定学习足够好的原则设计和实现的。在这里，生成和更新阶段可以轻松同步，以避免训练系统任一部分的闲置计算。对于需要每个答案10K到100K+个token的问题，推理模型的极长推理特性使得生成roll-out成为远更强的瓶颈。在更同步的RL基础设施上训练推理模型时，一个常见问题是批处理中的一个提示的答案生成需要更多时间（无论是更多token还是更多工具调用），导致大部分分配的计算资源在完成之前处于闲置状态。第二个解决方案，称为序列级打包，是在批处理中通过巧妙地掩码堆叠较短的样本，以使模型能够继续生成roll-out，并在批处理内更好地分布式长度归一化样本。分布式RL基础设施的完整复杂性超出了本书的范围，因为它可能导致许多其他微妙的问题，这些问题会减慢训练速度或导致不稳定。

随着这些推理模型的出现，进一步的兴趣在于使训练和推理循环完全离策略，其中策略梯度更新的训练批次填充了最近完成的多个实例生成答案的滚动操作 [159] [160]。完全异步训练还可以通过增加学习节点（执行策略梯度步骤）和演员（试图解决问题）之间权重同步的时间间隔，更容易地在多个数据中心扩展RL训练运行 [161]。

相关方法正在探索完全离策略策略梯度算法 {v*} [162]。

11.2.4 近端策略优化

许多PPO实现可用。核心 *loss* 计算如下所示。对稳定性能至关重要是 *value* 计算也是，其中存在多种选择（包括 *value model* 损失的多种选择）。

请注意，此处引用的策略（或旧logprobs）来自采样生成的时间，并不一定是引用策略。引用策略仅用于KL距离约束/惩罚。

```
# B: Batch Size, L: Sequence Length, G: Num of Generations
# Apply KL penalty to rewards
rewards = rewards - self.beta * per_token_kl # Shape: (B*G, L)

# Get value predictions
values = value_net(completions) # Shape: (B*G, L)

# Compute simple advantages
advantages = rewards - values.detach() # Shape: (B*G, L)
# Note: We detach the value network here to not update the parameters
# of
# the value function when computing the policy-gradient loss
```

```

# Normalize advantages (optional but stable)
advantages = (advantages - advantages.mean()) / (advantages.std() + 1e-8)

# Compute probability ratio between new and old policies
ratio = torch.exp(new_per_token_logps - per_token_logps) # Shape: (B*G, L)

# PPO clipping objective
eps = self.cliprange # e.g. 0.2
pg_losses1 = -advantages * ratio # Shape: (B*G, L)
pg_losses2 = -advantages * torch.clamp(ratio, 1.0 - eps, 1.0 + eps) # Shape: (B*G, L)
pg_loss_max = torch.max(pg_losses1, pg_losses2) # Shape: (B*G, L)

# Simple value function loss
vf_loss = 0.5 * ((rewards - values) ** 2) # Shape: (B*G, L)

# Combine policy and value losses
per_token_loss = pg_loss_max + self.vf_coef * vf_loss # Shape: (B*G, L)

# Apply completion mask and compute final loss
loss = ((per_token_loss * completion_mask).sum(dim=1) / completion_mask.sum(dim=1)).mean()
# Scalar

# Compute metrics for logging
with torch.no_grad():
    # Compute clipping fraction
    clip_frac = ((pg_losses2 > pg_losses1).float() * completion_mask).sum() / completion_mask.sum()

    # Compute approximate KL
    approx_kl = 0.5 * ((new_per_token_logps - per_token_logps)**2).mean()

    # Compute value loss for logging
    value_loss = vf_loss.mean()

```

理解PPO的核心是了解策略梯度损失是如何更新的。关注这三行：

```

pg_losses1 = -advantages * ratio # Shape: (B*G, L)
pg_losses2 = -advantages * torch.clamp(ratio, 1.0 - eps, 1.0 + eps) # Shape: (B*G, L)
pg_loss_max = torch.max(pg_losses1, pg_losses2) # Shape: (B*G, L)

```

`pg_losses1` 与上面的基于优势的PR损失相同，包含在PPO中，但损失（和梯度更新）可以被裁剪。尽管如此，PPO正在控制更新大小，使其不会太大。因为损失可以是负数，我们必须创建一个更

保守版本的vanilla策略梯度更新规则。

我们知道，如果我们对损失进行 *do not* 约束，策略梯度算法将精确地更新权重以匹配新的概率分布。因此，通过限制对数比，PPO 正在限制更新可以移动策略参数的距离。

最后，取两个中的最大值，如上所述，以采用更保守的损失更新。

对于PPO，所有这些都在学习值函数 *while* 时发生，这增加了更多复杂性，但这却是参数更新的核心逻辑。

11.2.4.1 每个样本使用1个梯度步长进行PPO/GRPO简化（无裁剪）如果超参数“每个样本的梯度步长数”等于1，则PPO（和GRPO）的实现可以处理得更加优雅。这个参数的正常值通常是2-4或更高。在主要的PPO或GRPO方程中，参见公式52，“参考”策略是先前的参数——用于生成补全或动作的参数。因此，如果只进行一个梯度步长， $\pi_\theta = \pi_{\theta_{old}}$ ，更新规则简化为以下（方括号 $[\cdot]_\nabla$ 表示停止梯度）：

$$J(\theta) = \frac{1}{G} \sum_{i=1}^G \left(\frac{\pi_\theta(a_i|s)}{[\pi_\theta(a_i|s)]_\nabla} A_i - \beta D_{KL}(\pi_\theta || \pi_{ref}) \right). \quad (61)$$

这导致PPO或GRPO实现，其中可以省略第二个策略梯度和裁剪逻辑，使优化器更接近标准策略梯度。

11.2.5 群组相对策略优化

深度SeekMath论文详细介绍了GRPO的一些实现细节，这些细节与PPO[151]不同，尤其是与从深度强化学习而不是语言模型的标准PPO应用相比。例如，在RLHF优化中的KL惩罚（回想一下，当在没有奖励模型的情况下在可验证奖励上训练推理模型时，也使用了KL惩罚）直接应用于损失更新，而不是奖励函数。在标准RLHF的KL惩罚应用为 $r = r_\theta - \beta D_{KL}$ 的地方，GRPO的实现如下：

$$L = L_{\text{policy gradient}} + \beta * D_{KL}$$

尽管有多种实现方式。传统上，KL距离是根据每个标记在补全到提示 s 中的情况计算的。对于推理训练，从一个提示中采样多个补全，一个批次中有多个提示，因此KL距离的形状为 $[B, L, N]$ ，其中 B 是批次大小， L 是序列长度， N 是每个提示的补全数量。

将它们组合起来，使用第一次损失累积，伪代码可以写成如下。

```
# B: Batch Size, L: Sequence Length, G: Number of Generations
# Compute grouped-wise rewards # Shape: (B,)
```

```

mean_grouped_rewards = rewards.view(-1, self.num_generations).mean(dim
    =1)
std_grouped_rewards = rewards.view(-1, self.num_generations).std(dim
    =1)

# Normalize the rewards to compute the advantages
mean_grouped_rewards = mean_grouped_rewards.repeat_interleave(self.
    num_generations, dim=0)
std_grouped_rewards = std_grouped_rewards.repeat_interleave(self.
    num_generations, dim=0)
# Shape: (B*G,)

# Compute advantages
advantages = (rewards - mean_grouped_rewards) / (std_grouped_rewards +
    1e-4)
advantages = advantages.unsqueeze(1)
# Shape: (B*G, 1)

# Compute probability ratio between new and old policies
ratio = torch.exp(new_per_token_logps - per_token_logps) # Shape: (B*
    G, L)

# PPO clipping objective
eps = self.cliprange # e.g. 0.2
pg_losses1 = -advantages * ratio # Shape: (B*G, L)
pg_losses2 = -advantages * torch.clamp(ratio, 1.0 - eps, 1.0 + eps) #
    Shape: (B*G, L)
pg_loss_max = torch.max(pg_losses1, pg_losses2) # Shape: (B*G, L)

# important to GRPO -- PPO applies this in reward traditionally
# Combine with KL penalty
per_token_loss = pg_loss_max + self.beta * per_token_kl # Shape: (B*G
    , L)

# Apply completion mask and compute final loss
loss = ((per_token_loss * completion_mask).sum(dim=1) /
    completion_mask.sum(dim=1)).mean()
# Scalar

# Compute core metric for logging (KL, reward, etc. also logged)
with torch.no_grad():
    # Compute clipping fraction
    clip_frac = ((pg_losses2 > pg_losses1).float() * completion_mask).
        sum() / completion_mask.sum()

    # Compute approximate KL
    approx_kl = 0.5 * ((new_per_token_logps - per_token_logps)**2).
        mean()

```

有关如何解释此代码的更多详细信息，请参阅上面的 PPO 部分。

11.2.5.1 RLOO与GRPO的优势更新非常接近GRPO，突出了算法在单独考虑时与PPO风格剪裁和KL惩罚细节的概念相似性。特别是对于RLOO，优势是相对于一个与GRPO极其相似的基线计算的——相对于同一问题的其他问题的完成奖励。简而言之，RLOO优势估计如下（从TRL的实现扩展而来）：

```
# rloo_k --> number of completions per prompt
# rlhf_reward --> Initially a flat tensor of total rewards for all
#               completions. Length B = N x k
rlhf_reward = rlhf_reward.reshape(rloo_k, -1) #
# Now, Shape: (k, N), each column j contains the k rewards for prompt
#               j.

baseline = (rlhf_reward.sum(0) - rlhf_reward) / (rloo_k - 1)
# baseline --> Leave-one-out baseline rewards. Shape: (k, N)
# baseline[i, j] is the avg reward of samples i' != i for prompt j.

advantages = rlhf_reward - baseline
# advantages --> Same Shape: (k, N)

advantages = advantages.flatten() # Same shape as original tensor
```

RLOO 的其余实现细节遵循实现策略梯度的其他权衡。

11.3 辅助主题

为了掌握策略梯度算法的应用，还有无数其他考虑因素。这里我们考虑了一些，但并非所有这些讨论。

11.3.1 比较算法

这里是对所讨论材料的一些总结（以及对直接策略优化即将讨论材料的预览），当应用于强化学习与人类反馈（RLHF）时。在这里，有或无策略表示导出（在实践中大多数应用都是稍微偏离策略）。这里的参考策略表示它是否需要用于优化本身，而不是用于KL惩罚。

表4：比较策略梯度算法（及其相关算法）。

Method	Type	Reward Model	Value Function	Reference Policy	Core Loss $\mathcal{L}(\theta)$
REINFORCE	policy	Yes	No	No	$-\frac{1}{T} \sum_{t=1}^T \log \pi_{\theta}(a_t s_t) (G_t - b(s_t))$
RLOO	On-policy	Yes	No	No	$-\frac{1}{K} \sum_{i=1}^K \sum_t \log \pi_{\theta}(a_{i,t} s_{i,t}) \left(R_i - \frac{1}{K-1} \sum_{j \neq i} R_j \right)$

Method	Type	Reward Model	Value Function	Reference Policy	Core Loss $\mathcal{L}(\theta)$
PPO	On-policy	Yes	Yes	Yes	$-\frac{1}{T} \sum_{t=1}^T \min(\rho_t A_t, \text{clip}(\rho_t, 1 - \varepsilon, 1 + \varepsilon) A_t); \rho_t = \frac{\pi_\theta(a_t s_t)}{\pi_{\theta_{\text{old}}}(a_t s_t)}$
GRPO	On-policy	Yes	No	Yes	$-\frac{1}{G} \sum_{i=1}^G \min(\rho_i A_i, \text{clip}(\rho_i, 1 - \varepsilon, 1 + \varepsilon) A_i); \rho_i = \frac{\pi_\theta(a_i s)}{\pi_{\theta_{\text{old}}}(a_i s)}, A_i = \frac{r_i - \text{mean}(r_{1:G})}{\text{std}(r_{1:G})}$
DPO	Off-policy	No	No	Yes	$-\mathbb{E}_{(x,y^w,y^l)} [\log \sigma(\beta [\Delta \log \pi_\theta(x) - \Delta \log \pi_{\text{ref}}(x)])]$

11.3.2 广义优势估计 (GAE)

广义优势估计 (GAE) 是计算策略梯度算法优势的替代方法[142]，它更好地平衡了偏差-方差权衡。传统的单步优势估计可能会引入过多的偏差，而使用完整轨迹通常会遭受高方差。GAE通过结合两个想法——多步预测和加权移动平均（或仅其中之一）来实现。

优势估计可以采取多种形式，但我们可以定义一个 n 步优势估计器（类似于本章开头的 TD 剩余），如下所示：

$$\hat{A}_t^{(n)} = \begin{cases} r_t + \gamma V(s_{t+1}) - V(s_t), & n = 1 \\ r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) - V(s_t), & n = 2 \\ \vdots \\ r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots - V(s_t), & n = \infty \end{cases} \quad (62)$$

这里较短的 n 将具有更低的方差但更高的偏差，因为我们正在将更多的学习能力分配给每个轨迹 - 它可能会过拟合。GAE 尝试将此公式推广到加权多步平均而不是特定的 n 。首先，我们必须定义预测值的时序差分 (TD) 残差。

$$\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t) \quad (63)$$

为了利用这一点，我们引入另一个变量 λ 作为 GAE 混合参数。这包含了我们希望估计的未来优势的指数衰减：

$$\begin{aligned} \hat{A}_t^{GAE(\gamma, \lambda)} &= (1 - \lambda)(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \dots) \\ &= (1 - \lambda)(\delta_t^V + \lambda(\delta_t^V + \gamma \delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma \delta_{t+1}^V + \gamma^2 \delta_{t+2}^V) + \dots) \\ &= (1 - \lambda)(\delta_t^V(1 + \lambda + \lambda^2 + \dots) + \gamma \delta_{t+1}^V(\lambda + \lambda^2 + \dots) + \dots) \\ &= (1 - \lambda)(\delta_t^V \frac{1}{1 - \lambda} + \gamma \delta_{t+1}^V \frac{\lambda}{1 - \lambda} + \dots) \\ &= \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{t+l}^V \end{aligned} \quad (64)$$

直观上，这可以用来优雅地平均多步优势估计。以下是一个示例实现：

```
# GAE (token-level) for LM RLHF
#
# B: Batch Size
# L: Length
# Inputs:
#   rewards: (B, L) post-KL per-token rewards
#   values:   (B, L) current V_theta(s_t)
#   done_mask: (B, L) 1.0 at terminal token (EOS or penalized trunc),
#                 else 0.0
#   gamma: float (often 1.0), lam: float in [0,1]
B, L = rewards.shape
advantages = torch.zeros_like(rewards)
next_v = torch.zeros(B, device=rewards.device, dtype=rewards.dtype)
gae = torch.zeros(B, device=rewards.device, dtype=rewards.dtype)

for t in reversed(range(L)):
    not_done = 1.0 - done_mask[:, t]
    delta = rewards[:, t] + gamma * not_done * next_v - values[:, t]
    gae = delta + gamma * lam * not_done * gae
    advantages[:, t] = gae
    next_v = values[:, t]

targets = advantages + values      # y_t for value regression
advantages = advantages.detach()   # for policy loss
```

For further reading, see [163].

11.3.3 双重正则化

许多来自深度强化学习的流行策略梯度算法源于对控制智能体学习过程的必要性。在RLHF中，正如第8章关于正则化和第4章关于问题表述所广泛讨论的那样，通过相对于原始策略的距离惩罚内置了一个正则化项。从这个角度来看，PPO（具有内部步长正则化）和REINFORCE（更简单，且在特定超参数下PPO简化为）等算法之间的很大一部分差异，对于微调语言模型来说，远不如从头开始训练智能体有意义。

在PPO中，处理更新步长封顶的目标被称为代理目标。为了监控PPO正则化对RLHF中更新的影响，可以查看许多流行实现中的clip fraction变量，这是批样本中梯度被此正则化器在PPO中剪裁的百分比。这些梯度被限制在最大值*reduced*。

11.3.4 进一步阅读

作为RLHF在现代训练后的中心地位得以巩固，其他策略梯度RL算法和RL算法通常被提出以提高训练

过程，但它们在治理最佳实践中并没有扮演核心角色。进一步阅读的例子包括：

- 成对近端策略优化（P3O）[164]在PPO风格的策略更新中直接使用成对数据，而不学习中间奖励模型。
- 离策略策略梯度算法可以实现进一步的异步训练，例如对比策略梯度（CoPG）[165]（直接对齐算法IPO和vanilla策略梯度的推广），该算法被Cohere用于他们的Command A模型[57]。
- 其他REINFORCE算法的实现已被设计用于语言模型，例如ReMax [166]，它实现了一种基线归一化，专门用于适应奖励模型推理的不确定性来源。
- 一些基础模型，例如苹果智能基础模型[167]或Kimi k1.5推理模型[168]，已经使用了镜像下降策略优化（MDPO）[169]的变体。关于这里的基本原理[170]的研究仍在进一步发展，但镜像下降是一种优化方法，而不是直接的政策梯度算法。这里重要的是，它与现有的强化学习基础设施非常相似地被替换。
- 解耦剪辑和动态采样策略优化（DAPO）对GRPO提出4项修改，以更好地适应需要长序列和增加新、未充分利用的标记概率的推理语言模型[157]。修改如下：1、有两个不同的剪辑超参数 ϵ_{low} 和 ϵ_{high} ，以便在logratio的正侧进行剪辑时可以采取更大的步长以实现更好的探索；2、动态采样，移除所有奖励为= 0或奖励为= 1的样本（无学习信号）；3、使用上述实现中讨论的每个标记损失；4、对过长样本施加软惩罚，以避免尝试从截断答案中学习。
- 基于价值的增强近端策略优化（VAPO）[171]结合了DAPO（包括clip-higher、token级策略梯度以及不同长度归一化）的优化与价值校准PPO [172]的见解，以预训练价值函数和长度自适应GAE，以展示基于价值的方法相对于GRPO的潜力。

12 直接对齐算法

直接对齐算法（DAAs）允许在不训练任何中间奖励模型或使用强化学习优化器的情况下更新模型以解决相同的RLHF目标。最突出的DAA之一，也是催化了整个将语言模型对齐的学术运动的是直接偏好优化（DPO）[19]。在其核心，DPO是使用梯度上升来解决相同的约束RLHF目标。自2023年5月发布以来，经过一段短暂的延迟，社区找到了使用DPO的正确数据和超参数（特别是，出人意料地低的学习率），许多流行的模型已经使用了DPO或其变体，从2023年10月启动的Zephyr- β [20]，Llama 3 Instruct [23]，Tulu 2 [21]和3 [6]，Nemotron 4 340B [24]，以及其他。技术上，序列似然校准（SLiC-HF）首先发布 [173]，但由于运气和有效性的结合，它并没有流行起来。

DPO和DAAs最具影响力的部分是降低在训练后实验语言模型的门槛。

12.1 直接偏好优化（DPO）

这里我们解释其工作原理的直觉，并完全重新推导核心方程。

12.1.1 DPO工作原理

DPO在表面层面上是直接优化策略以解决RLHF目标。这个损失函数，我们将在下面的推导中重新审视，是成对对数概率的关系。从Bradley-Terry奖励模型推导出的损失函数如下：

$$\mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_c, y_r) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_c | x)}{\pi_{\text{ref}}(y_c | x)} - \beta \log \frac{\pi_{\theta}(y_r | x)}{\pi_{\text{ref}}(y_r | x)} \right) \right] \quad (65)$$

在整个过程中， β 是一个平衡奖励优化与最终模型与初始参考之间的KL距离的超参数（即平衡过度优化，作为使用DPO校正的关键超参数）。这依赖于DPO训练的隐含奖励，它取代了使用外部奖励模型，后者是概率的对数比率：

$$r(x, y) = \beta \log \frac{\pi_r(y | x)}{\pi_{\text{ref}}(y | x)} \quad (66)$$

在 $\pi_r(y | x)$ 是我们正在解决的精确、最优奖励策略。这来自于根据最优策略（如公式80所示）推导Bradley-Terry奖励，如Bradley-Terry模型部分所示。本质上，隐式奖励模型显示“以最优策略而不是奖励模型来表示人类偏好数据的概率。”

让我们考虑eq. 65中显示的损失。学习过程是减少损失。在这里，当所选响应的对数比大于被拒绝响应的对数比（按参考模型归一化）时，损失会更低。在实践中，这是模型在数据中呈现的标记序列上的对数概率之和。因此，DPO正在增加所选和被拒绝响应之间的概率差异。

With the reward in eq. 66, we can write the gradient of the loss to further interpret what is going on:

$$\nabla_{\theta} \mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\beta \mathbb{E}_{(x, y_c, y_r) \sim \mathcal{D}} [\sigma(r_{\theta}(x, y_r) - r_{\theta}(x, y_c)) (\nabla_{\theta} \log \pi(y_c | x) - \nabla_{\theta} \log \pi(y_r | x))] \quad (67)$$

这里, 梯度通过执行以下操作来解决上述目标

owing:欠款:

- sigmoid函数中的第一个项 $\sigma(\cdot)$ 创建了一个从0到1的参数更新的权重, 当奖励估计不正确时权重更高。当拒绝的样本被优先于选择的样本时, 权重更新应该更大!
- 其次, 内括号 $[\cdot]$ 中的术语增加了所选响应 y_c 的可能性, 并减少了被拒绝的 y_r 的可能性。
- 这些术语通过 β 加权, 它控制更新如何平衡正确排序完成项与KL距离之间的关系。

核心直觉是DPO是“拟合一个隐式奖励模型, 其对应的最佳策略可以以闭式形式提取”(归功于梯度上升和我们的机器学习工具)。常被误解的是, DPO在本质上是在学习一个奖励模型, 因此论文的副标题 *Your Language Model is Secretly a Reward Model*. 很容易与DPO直接训练策略的目标混淆, 因此研究下面的推导对完整理解很有帮助。

使用隐式奖励模型学习, DPO根据数据集中的数据和目标中的特定KL约束 β 生成RLHF目标的最佳解决方案。在这里, DPO在给定的特定KL距离下求解精确策略, 因为生成不是在线的, 如策略梯度算法中那样——这是与偏好调整的RL方法的核心区别。在许多方面, 这使得相对于在线RL方法, 使用DPO调整 β 值更容易, 但关键且直观的是, 最佳值取决于正在训练的模型和训练它的数据。

在每一批偏好数据中, 由许多对补全 $y_{\text{chosen}} \succ y_{\text{rejected}}$ 组成, DPO 直接朝着最优解进行梯度步骤。这比策略梯度方法简单得多。

12.1.2 DPO 导出

DPO导出包含两个主要部分。首先, 作者展示了在整个书中使用的RLHF目标的最优策略形式。接下来, 他们展示了如何从成对偏好数据(即Bradley Terry模型)得出该解决方案。

12.1.2.1 1. 推导最优RLHF解决方案 首先, 我们应该再次考虑RLHF优化目标, 这里表示我们希望最大化这个量:

$$\max_{\pi} \mathbb{E}_{\tau \sim \pi} [r_{\theta}(s_t, a_t)] - \beta \mathcal{D}_{KL}(\pi^{\text{RL}}(\cdot | s_t) \| \pi^{\text{ref}}(\cdot | s_t)). \quad (68)$$

首先, 让我们扩展KL散度的定义,

LEARNING FROM HUMAN FEEDBACK

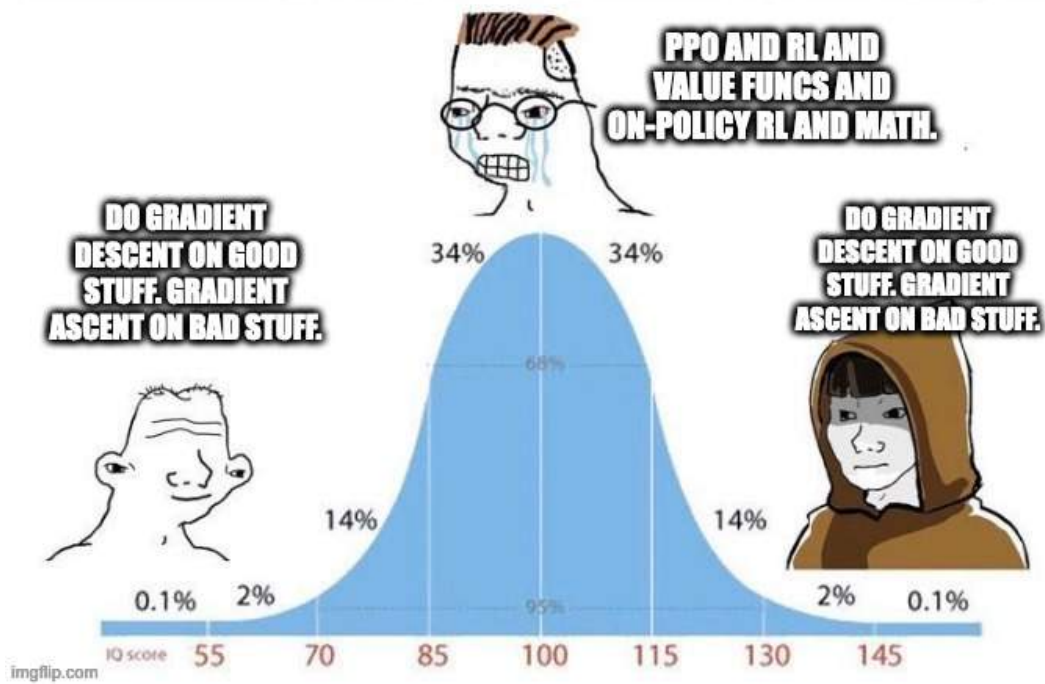


图16: DPO简单梗图, 版权所有汤姆·戈尔茨坦。

$$\max_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[r(x, y) - \beta \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} \right] \quad (69)$$

接下来，将负号从括号中的差中提出来。为此，将其分成两个项：

$$= \max_{\pi} \left(\mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} [r(x, y)] - \beta \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[\log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} \right] \right) \quad (70)$$

然后，移除因子 -1 和 β ,

$$= \min_{\pi} \left(-\mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} [r(x, y)] + \beta \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[\log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} \right] \right) \quad (71)$$

除以 β 并重新组合：

$$= \min_{\pi} \left(\mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[\log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{1}{\beta} r(x, y) \right] \right) \quad (72)$$

接下来，我们必须引入一个配分函数， $Z(x)$ ：

$$Z(x) = \sum_y \pi_{\text{ref}}(y|x) \exp \left(\frac{1}{\beta} r(x, y) \right) \quad (73)$$

分区函数作为参考策略的归一化因子，对所有可能的响应 y 对提示 x 的求和。将此代入，我们得到我们的中间转换：

$$\min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[\log \frac{\pi(y|x)}{\frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp \left(\frac{1}{\beta} r(x, y) \right)} - \log Z(x) \right] \quad (74)$$

要看到这是如何得到的，考虑方程式72中括号内的优化内部部分：

$$\log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{1}{\beta} r(x, y) \quad (75)$$

然后，将 $\log Z(x) - \log Z(x)$ 加到两边：

$$= \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{1}{\beta} r(x, y) + \log Z(x) - \log Z(x) \quad (76)$$

然后，我们将项分组：

$$= \left(\log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} + \log Z(x) \right) - \log Z(x) - \frac{1}{\beta} r(x, y) \quad (77)$$

使用 $\log(x) + \log(y) = \log(x \cdot y)$ (并将 Z 移至分母), 我们得到:

$$= \log \frac{\pi(y|x)}{\frac{1}{Z(x)} \pi_{\text{ref}}(y|x)} - \log Z(x) - \frac{1}{\beta} r(x, y) \quad (78)$$

接下来, 我们将 $\frac{1}{\beta} r(x, y)$ 展开为 $\log \exp \frac{1}{\beta} r(x, y)$, 并对等式 74 进行相同的操作。使用这种优化形式, 我们需要实际求解最优策略 π^* 。为此, 让我们将上述优化视为 KL 距离:

$$\min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \left[\mathbb{D}_{\text{KL}} \left(\pi(y|x) \parallel \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp \left(\frac{1}{\beta} r(x, y) \right) \right) - \log Z(x) \right] \quad (79)$$

由于分区函数 $Z(x)$ 不依赖于最终答案, 我们可以忽略它。这使我们只剩下我们正在学习的策略与一个关于分区、 β 、奖励和参考策略的关系形式的KL距离。吉布斯不等式告诉我们, 当这两个量相等时, 这个距离在0处达到最小值! 因此, 我们得到一个最优策略:

$$\pi^*(y|x) = \pi(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp \left(\frac{1}{\beta} r(x, y) \right) \quad (80)$$

12.1.2.2.2. 推导Bradley-Terry模型的DPO目标 首先, 回顾第7章关于奖励建模和第6章关于偏好数据的内容, Bradley-Terry模型的人类偏好形成如下:

$$p^*(y_1 \succ y_2 | x) = \frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))} \quad (81)$$

通过对方程80进行操作, 取等式两边的对数并进行一些代数运算, 可以得到以下DPO奖励:

$$r^*(x, y) = \beta \log \frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x) \quad (82)$$

然后我们可以将奖励代入到方程式81中所示的Bradley-Terry方程, 以获得:

$$p^*(y_1 \succ y_2 | x) = \frac{\exp \left(\beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} + \beta \log Z(x) \right)}{\exp \left(\beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} + \beta \log Z(x) \right) + \exp \left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} + \beta \log Z(x) \right)} \quad (83)$$

通过将指数表达式从 e^{a+b} 到 $e^a e^b$ 分解, 然后消去项 $e^{\log(Z(x))}$, 这简化为:

$$p^*(y_1 \succ y_2 | x) = \frac{\exp\left(\beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)}{\exp\left(\beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right) + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)}\right)} \quad (84)$$

然后，将分子和分母乘以 $\exp\left(-\beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)$ 以获得：

$$p^*(y_1 \succ y_2 | x) = \frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} - \beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)} \quad (85)$$

最后，根据sigmoid函数的定义为 $\sigma(x) = \frac{1}{1+e^{-x}}$ ，我们得到：

$$p^*(y_1 \succ y_2 | x) = \sigma\left(\beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} - \beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)}\right) \quad (86)$$

这是DPO的损失函数，如公式65所示。DPO论文对Plackett-Luce模型下的目标函数有额外的推导，这在实践中远未得到广泛应用[19]。

12.1.2.3 3. 推导Bradley Terry DPO梯度我们使用了方程67中所示的DPO梯度来解释模型学习的直觉。为了推导这个，我们必须对方程86关于模型参数求梯度。

$$\nabla_{\theta} \mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\nabla_{\theta} \mathbb{E}_{(x, y_c, y_r) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_c|x)}{\pi_{\text{ref}}(y_c|x)} - \beta \log \frac{\pi_{\theta}(y_r|x)}{\pi_{\text{ref}}(y_r|x)} \right) \right] \quad (87)$$

首先，这可以被重写。我们知道sigmoid函数的导数 $\frac{d}{dx} \sigma(x) = \sigma(x)(1 - \sigma(x))$ ，对数的导数 $\frac{d}{dx} \log x = \frac{1}{x}$ ，以及sigmoid的性质 $\sigma(-x) = 1 - \sigma(x)$ ，因此我们可以重新格式化上述方程。

首先，定义sigmoid函数内的表达式为 $u = \beta \log \frac{\pi_{\theta}(y_c|x)}{\pi_{\text{ref}}(y_c|x)} - \beta \log \frac{\pi_{\theta}(y_r|x)}{\pi_{\text{ref}}(y_r|x)}$ 。然后，我们有

$$\nabla_{\theta} \mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_c, y_r) \sim \mathcal{D}} \left[\frac{\sigma'(u)}{\sigma(u)} \nabla_{\theta} u \right] \quad (88)$$

展开此表达式并使用上述sigmoid和对数表达式，得到之前引入的梯度：

$$-\mathbb{E}_{(x, y_c, y_r) \sim \mathcal{D}} \left[\beta \sigma \left(\beta \log \frac{\pi_{\theta}(y_r|x)}{\pi_{\text{ref}}(y_r|x)} - \beta \log \frac{\pi_{\theta}(y_c|x)}{\pi_{\text{ref}}(y_c|x)} \right) [\nabla_{\theta} \log \pi(y_c|x) - \nabla_{\theta} \log \pi(y_r|x)] \right] \quad (89)$$

12.2 数值问题、弱点及替代方案

许多DPO算法的变体已被提出，以解决DPO的弱点。例如，在没有rollouts（奖励模型可以评估生成的情况）的情况下，DPO将每对偏好数据视为具有相同的权重。在现实中，正如第6章关于偏好数据所述，有许多方法可以捕获比二进制更丰富的标签的偏好数据。已经提出了多种算法来重新平衡优化，使其不再平等对待每一对。

- 回归到相对奖励的强化学习（REBEL）通过在选择的响应和拒绝的响应之间添加来自奖励模型的信号，作为边缘，而不是仅使用成对偏好数据，以更准确地解决RLHF问题[131]。
- 守旧DPO（cDPO）和身份偏好优化（IPO）通过假设偏好数据中的噪声来解决过拟合问题。cDPO假设N百分比的数据被错误标记[19]，而IPO将优化改为软化偏好概率而不是直接从标签进行优化[174]。实际上，IPO将偏好概率转换为非线性函数，偏离了Bradley-Terry假设，使用 $\Psi(q) = \log\left(\frac{q}{1-q}\right)$ 。
- DPO与偏移量（ODPO）“要求首选和次选响应的似然性之差大于偏移量值” [175] – 不要平等对待每个数据对，但这可能会以更困难的标注环境为代价。

一些DPO变体试图通过微小地改变损失来改善学习信号，或者通过减少内存使用来提高应用效率。

- Odds Ratio Policy Optimization (ORPO) 直接通过向所选响应的拉力来更新策略模型，类似于指令微调损失，对所选响应施加轻微惩罚[176]。这种损失函数的改变消除了对参考模型的需求，简化了设置。看待ORPO的最佳方式是受DPO启发，而不是DPO的衍生物。
- 简单偏好优化SimPO对DPO优化进行微小调整，通过平均对数概率而不是求和（SimPO）或添加长度归一化，以提高性能[177]。

一个核心问题 *apparent* 在DPO中是，优化仅驱动增加所选和拒绝响应之间的概率差距。从数值上讲，模型降低了所选和拒绝响应的概率，但如图17所示 *rejected response is reduced by a greater extent*。直观上，不清楚这如何推广，但研究表明它增加了未解决行为[178] [179]的概率。简单方法——如Cal-DPO [180]，它调整优化过程，以及AlphaPO [181]，它修改奖励形状——减轻了这种偏好位移。在实践中，这种影响的精确程度尚不清楚，但可能是为什么在线方法可以优于传统DPO的原因之一。

最大的其他原因是，DPO-like方法相对于基于在线（基于强化学习）的RLHF方法在性能上具有较低的极限，因为训练信号来自先前或其他模型的完成。从模型中采样生成的在线变体，例如在线DPO [182]，即使通过为新创建的作品进行常规奖励模型重标记的判别器引导DPO（D2PO） [183]，也能通过生成新的

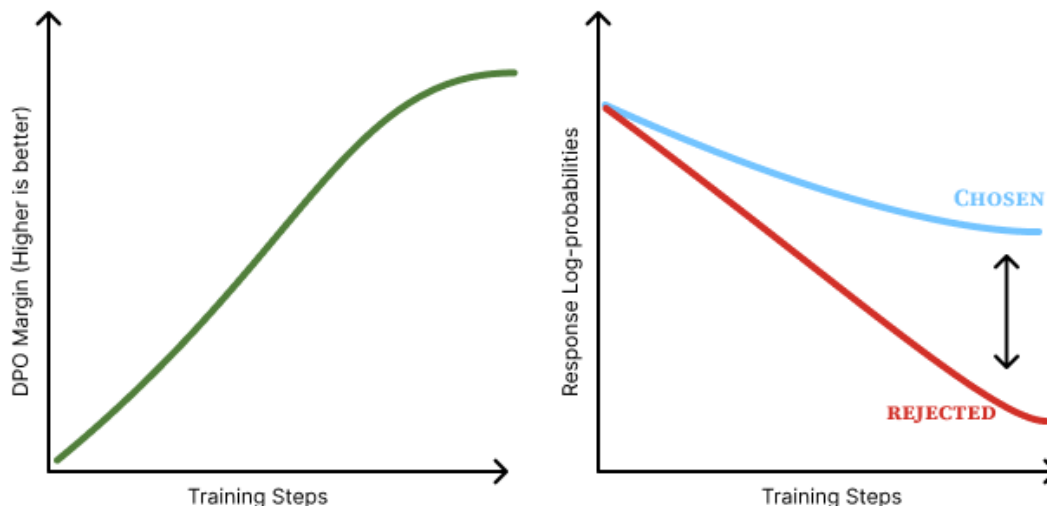


图17: DPO中偏好位移的草图。

补全提示并训练时引入偏好信号。

存在许多其他DAA变体，例如直接Nash优化（DNO）[184]或二进制分类器优化（BCO）[185]，但算法的选择远不如初始模型和所使用的数据重要[6] [186] [187]。

12.3 实施注意事项

DAAAs如DPO与策略梯度优化器实现方式大相径庭。DPO损失，从原始实现中提取，大体可以概括如下[19]:

```
pi_logratios = policy_chosen_logps - policy_rejected_logps
ref_logratios = reference_chosen_logps - reference_rejected_logps

logits = pi_logratios - ref_logratios # also known as  $h_{\pi_{\theta}}(y_w, y_l)$ 

losses = -F.logsigmoid(beta * logits)

chosen_rewards = beta * (policy_chosen_logps - reference_chosen_logps)
                .detach()
rejected_rewards = beta * (policy_rejected_logps -
                           reference_rejected_logps).detach()
```

这可以在标准语言模型训练堆栈中使用，因为这些信息已经在模型的正向传递过程中（加上参考模型）汇总。

在大多数方面，这更简单，也是生活质量的提升，但同时也带来了一组不同的考虑因素。

1. KL距离是静态的：在DPO和其他算法中，KL距离通过平衡距离惩罚到优化的 β 参数显式设置。这是由于

向事实表明，DPO根据数据向RLHF目标的 $optimal$ 解进行梯度步骤——它正好走向由 β 项设定的解集。另一方面，基于RL的优化器根据批量和最近的数据进行步骤。

2. 缓存对数概率：DPO的简单实现同时进行策略模型和参考模型的正向传播，以方便损失函数。然而，这加倍了内存使用，并导致GPU使用增加。为了避免这种情况，可以先计算参考模型在训练数据集上的对数概率，然后在计算损失和按批次更新参数时引用它，从而将峰值内存使用减少50%。

12.4 DAAs与RL：在线数据与离线数据

广泛来说，这个论点归结为一个问题：我们需要强化学习的内部机制，包括价值函数、策略梯度等，来使语言模型与RLHF保持一致吗？这个问题，就像大多数以这种方式提出的问题一样，过于简单化。当然，这两种方法都是成熟的，但重要的是要说明它们之间的基本差异和性能差异在哪里。

多项报告得出结论，基于策略梯度和强化学习（RL）的方法优于DPO及其变体。论点形式各异，从使用不同算法但受控数据进行模型训练[153] [188]到研究在RL优化循环中在线策略数据的作用[189]。在这些所有情况下，DPO算法都略逊一筹。

即使有这种性能差异，DAA 仍然在领先模型中被广泛使用，因为其简单性。DAAs 提供了一个受控的环境，可以在训练数据和其它配置上进行快速迭代，鉴于数据通常比算法更重要，使用 DPO 是可以接受的。

随着以RL为主要训练方法的推理模型的兴起，进一步的投入将回归到使用RL进行偏好调整，这将在长期内提高RL基础设施的鲁棒性，并巩固DAAs和RL在优化人类反馈之间的差距。

13 宪法人工智能 & 人工智能反馈

RL从AI反馈（RLAIF）是一套更广泛的技术，用于使用AI增强或生成反馈数据，包括成对偏好[190] [191] [192]。使用RLAIF来完全取代或增强人类反馈有许多动机。AI模型远比人类便宜，单个人类偏好数据成本约为1美元或更高（或甚至每个提示超过10美元），而使用前沿AI模型（如GPT-4o）的AI反馈成本不到0.01美元。这种成本差异使得RLHF方法的实验市场对之前因价格而被排除在外的人群开放。除了价格之外，AI反馈在性能上引入了与人类反馈不同的*tradeoffs*，这些差异仍在研究中。AI反馈的峰值性能至少与基于技能的评价中的人类数据在同一水平，但尚未研究人类数据是否允许在现实世界的产品设置或新的训练方法（如角色训练）中更精细地控制模型。

RLAIF这一术语是在Anthropic的工作*Constitutional AI: Harmlessness from AI Feedback* [18]中提出的，这导致AI社区对方法之间的关系产生了最初的困惑。自从宪法AI（CAI）论文发布和RLAIF的正式化以来，RLAIF已成为后训练和RLHF文献中的默认方法——例子多得难以一一列举。应将关系理解为CAI是启动RLAIF更广泛领域的例子。

人类数据与AI反馈数据差异的经验法则如下：

1. 人类数据是高噪声和低偏差，
2. 合成偏好数据是低噪声和高偏差，

结果在许多学术成果中显示了如何在RLHF工作流程中用AI偏好数据替代，并实现强大的评估分数[193]，但显示了RLHF文献与工业最佳实践相分离。

13.1 宪法人工智能

宪法AI（CAI）的方法，Anthropic在他们的Claude模型中广泛使用，是用于RLHF训练的最早、最大规模使用合成数据的方法。宪法AI有两个合成数据的使用：

1. 对指令调整数据的批评，遵循一系列原则，如“答案是否鼓励暴力”或“答案是否真实”。当模型生成问题的答案时，它会将答案与宪法中的原则列表进行核对，并在时间上不断优化答案。然后，他们在生成的数据集上微调模型。
2. 通过使用语言模型来回答在宪法中随机原则的背景下哪个补全更好，从而生成成对偏好数据（类似于本论文中用于原则引导奖励模型的论文）。然后，RLHF在合成数据上按正常流程进行，因此得名RLAIF。

主要来说，CAI 以上述后半部分，即偏好数据而闻名，但用于指令数据的引入方法在训练后的一般数据过滤和合成数据生成方法中得到应用。

CAI 可以形式化为如下。

通过采用一套人类编写的原则，他们称之为 *constitution*，Bai 等人 2022 使用一个单独的 LLM 生成用于微调的人工偏好和指令数据 [18]。宪法 \mathcal{C} 是一组书面原则，指示在批判阶段应关注的具体方面。指令数据通过反复采样原则 $c_i \in \mathcal{C}$ 并要求模型修改其最新输出 y^i 以符合提示 x 来进行整理。这产生了一系列用于批判的原则 $\{c_0, c_1, \dots, c_{n-1}\}$ 的指令变体 $\{y^0, y^1, \dots, y^n\}$ 。最终数据点是提示 x 与最终的完成 y^n ，对于某些 n 。

偏好数据通过使用来自 \mathcal{C} 的原则子集作为反馈模型的上下文以类似但更简单的方式进行构建。反馈模型通过提示 x 、一组原则 $\{c_0, \dots, c_n\}$ 以及两个标记为答案 (A) 和 (B) 的完成 y_0 和 y_1 (来自之前的 RLHF 数据集) 进行展示。记录反馈模型输出 (A) 或 (B) 的概率作为奖励模型的训练样本。

13.2 特定用于判断的LLM

随着RLAIF方法越来越普遍，许多人都在思考我们是否应该使用相同的模型来生成回应，就像那些用于生成评论或评分的模型一样。特别是，所使用的LLM作为裁判的校准问题引起了质疑。几项研究已经表明，LLM是不一致的评估者[194]，并且更喜欢自己的回应而不是其他模型的回应（称为自我偏好偏差）[195]。

因此，许多人都在想，我们是否应该使用相同的模型来生成回应，就像那些用于生成评论或评分的模型。是否应该训练一个专门用于此的独立模型？已经发布了多个模型，旨在作为数据标注工具替代前沿模型，例如评论模型Shepherd [196] 和 CriticLLM [197]，或者类似于Auto-J [198]、Prometheus [106]、Prometheus 2 [199] 或 Prometheus-Vision [200] 的评估回应性能的模型，但它们在文档化的训练方案中并未得到广泛应用。有些人发现通过重复采样 [201] [202] [203]、自我完善 [204] 或锦标赛排名 [205] 来缩放推理可以更好地估计真实判断或更高质量的偏好对。其他校准技术共同进化模型的生成和判断能力 [206]。

13.3 进一步阅读

存在许多与宪法AI相关的研究方向和扩展，但其中很少被记录为在RLHF和训练后食谱中的明显改进。目前，它们被包括为进一步阅读内容。

- OpenAI 发布了一个模型规范 [91]，这是一份说明其模型预期行为的文档，并表示他们正在探索模型直接引用该文档的校准方法（这可以被视为与CAI的紧密伙伴）。OpenAI 继续使用一种称为“审议式校准” [207] 的方法来训练其推理模型，如o1，以在引用这些安全或行为策略的同时校准模型。
- Anthropic 继续在模型训练中使用 CAI，更新 Claude 使用的宪法 [208]，并实验人口集体如何就模型原则达成共识以及这如何改变模型行为 [209]。

- 开源社区已探索将CAI应用于公开数据集的复制[210]以及探索在LM之间创建对话数据的方法[211]。
- 其他工作使用了原则驱动的偏好或反馈以及不同的优化方法。[212]使用原则作为奖励模型的上下文，该模型用于训练Dromedary模型[213]。[36]使用原则来提高RLHF过程中人类判断的准确性。[214]在推理时训练一个奖励模型来生成其自身的原则，并使用这些原则来提供最终得分。[215]将遵循原则表述为一个预训练模型可以无标签学习的互信息最大化问题。

14 理解训练与推理时间缩放

在2016年神经信息处理系统（NeurIPS）会议中，Yann LeCun首次介绍了他在现代机器学习系统中学习发生位置的著名蛋糕隐喻：

如果智能是一块蛋糕，蛋糕的大部分是未监督学习，蛋糕上的糖霜是监督学习，蛋糕上的樱桃是强化学习（RL）。

这个类比现在在现代语言模型和最近对训练后堆栈的更改下基本完成。在这个类比中：

- 自监督学习在大量互联网数据上的应用占据了大部分份额（尤其是在从FLOPs计算消耗的角度来看），
- 训练后监督微调（SFT）的初始阶段将模型调整到一个更窄的分布（在所选示例的RLHF帮助下），并且
- 最后，“纯”强化学习（RL）是顶上的樱桃。

我们通过强化学习（RL）在仅几个训练样本中仅学习“少量”信息。这种推理训练与使用本书中讨论的培训后技术的推理模型相结合，通过在可验证领域进行RL训练来调整偏好，从而显著提高推理、编码和数学问题解决等能力。

这些模型的训练方法，可验证奖励的强化学习（RLVR）[6]，与RLHF非常相似，但它将奖励模型作为可选的，而不是一个在答案正确时返回正奖励，否则返回0的评分函数。RLVR背后的思想在强化学习文献中并不新颖，在语言建模文献中也有许多相关思想，其中模型从对答案是否正确的反馈中学习。

最初，可验证奖励的强化学习（RLVR）原本被命名为具有真实标签奖励的强化学习（RLGT）。然而，RLVR与仅从真实标签答案中学习的强化学习有所不同。在数学等领域，只有一个真实标签答案可用于验证解决方案。在其他领域，如代码生成或精确指令遵循，即使有多个正确答案而不是只有一个真实标签答案，也可以使用检查函数（例如，单元测试）来验证答案。RLVR进步的核心是拥有各种验证问题的多样性和深度，即使确切解决方案事先未知。

第一个成功部署此类训练的模型是OpenAI的o1 [47] 和开源权重模型DeepSeek R1 [60]。随后，整个AI行业都将此训练过程和模型风格列为优先。这里的核心变化更多的是对训练阶段和不同行为优先级的重新分配，而不是这种强化学习（RL）设置完全新颖。推理模型带来了一个期望强化学习（RL）训练可扩展性的时代。

关于这些模型积累的行为类型，考虑以下示例，即DeepSeek V3 0325与其推理模型DeepSeek R1在查询Write me a short poem about a goldfish上的比较。DeepSeek V3非常直接：

金鱼梦

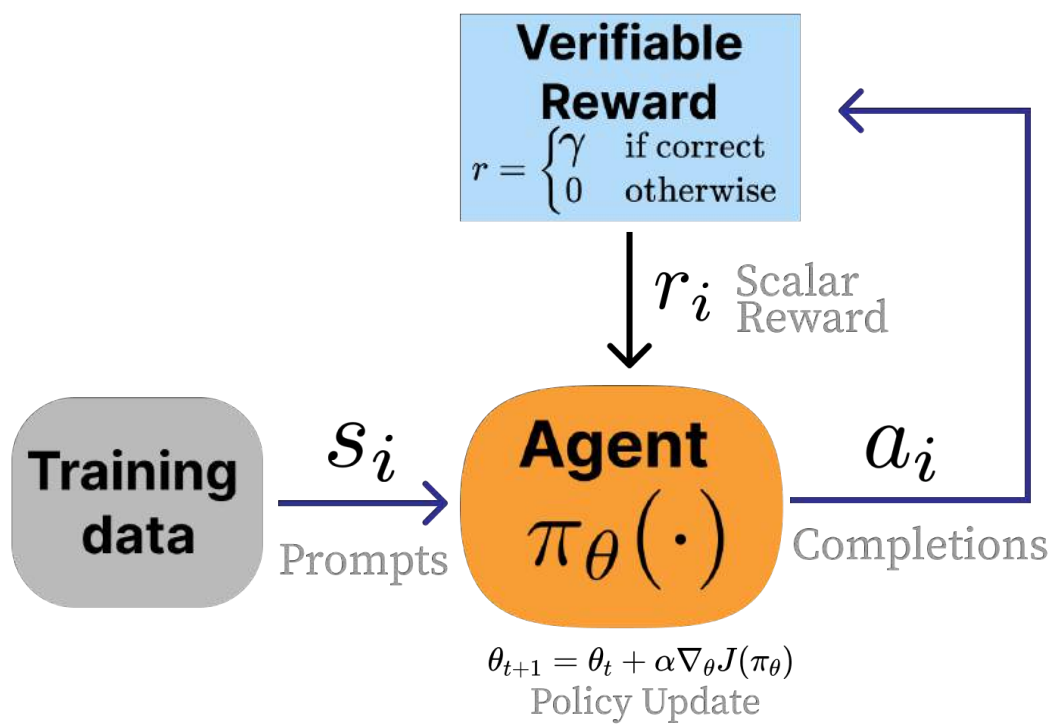


图18：以RL反馈环形式的RLVR。我们使用验证函数而不是奖励模型。

在一个如此清澈圆润的玻璃碗中，一条小金鱼无声地游动。鳞片闪耀着阳光的亲吻，它在水中飘荡，享受着水的快乐。

鳍的一摆，一闪如此明亮，追逐着光中的气泡。他梦想着广阔的海洋吗？还是他的小世界已经足够——他的骄傲？

同时，DeepSeek R1 的体验不同。它在回应用户之前，首先生成许多思考令牌来构建一个回答。答案显示为：

<thinking> 好的，用户想要一首关于金鱼的短诗。让我先从一些与金鱼相关的意象开始头脑风暴。金鱼通常在鱼缸里，对吧？所以也许可以从环境开始。像“金色的鱼缸”或“水晶球”这样的词可以行得通。

接下来，思考金鱼的游动。它们游动得很顺畅，所以会想到“急速游动”、“滑行”或“旋转”这样的词。也许可以用“丝绸般”来形容它们的鳍，以强调其精致。

然后，考虑金鱼周围的环境。阳光透过水面产生闪烁效果。像“阳光舞动”或“涟漪旋转”这样的词语可能捕捉到那种感觉。包括城堡或鹅卵石等元素增添了一丝趣味。

主题？或许是小金鱼狭小世界与对自由的感知之间的对比。鱼缸作为一个宇宙，虽小却完整。以神秘感或静默观察结束，可以增加深度。需要保持简洁，或许四节诗，有押韵。让我检查一下节奏和流畅度。避免陈词滥调，但保持易于理解。好的，让我们把它拼凑起来。</thinking>

金箔之梦在液态玻璃中

镀金碗，水晶球，微小的潮汐在寂静中流淌——一抹橙色，低语般的袍子，你穿梭在阳光下的靛蓝之中。

你的世界虽小，却又无边无际——每一颗鹅卵石、城堡、蔓延的杂草。表面弯曲天空至蓝色，涟漪旋转你的液态世界。

你滑过我们永远无法知晓的神话，一位沉默的君主，被迷雾所遮蔽。梦想是否随着气泡飘散？或是记忆在紫水晶中？

我们注视着你描绘无尽的球体——一滴泪中的宇宙。

标记在 <thinking> 标签中的是模型的推理。对于更复杂的问题，推理阶段可能需要成千上万的标记才能得出答案。因此，长上下文语言模型是高级推理行为的前提，但这不是本章的重点。

核心直觉是，对于给定的模型，我们重复以下循环：

1. 样本多题多选，2. 向正确答案逐步逼近，3. 重复，回顾相同数据。

引人注目地，这种方法（在仔细分配数据和稳定的训练基础设施下进行）通过重新审视相同的问题帮助模型学习

一遍又一遍。更值得注意的是，这些训练问题上的改进可以推广到模型从未见过的（一些）问题和领域！

这种简单方法允许模型在行为空间中进行轻微搜索，并且强化学习算法增加了与正确答案相关的行为发生的可能性。

14.1 新推理模型起源

这里我们详细介绍了导致2025年推理模型爆炸式增长的高层次趋势。

14.1.1 为什么现在RL有效？

尽管许多人认为“强化学习还不起作用” [216] 或论文详细说明了强化学习中的深度可重复性问题 [217]，但该领域克服了这些问题，找到了具有重大影响的应用。针对语言模型的强化学习训练的起飞标志着该研究领域许多基本问题的进展，包括：

- **RL的稳定性可以解决：**在其整个存在期间，RL采用的限制因素一直是稳定性。这表现在两个方面。首先，学习本身可能不稳定，并不总是有效。其次，训练本身已知比标准语言模型训练更脆弱，更容易出现损失峰值、崩溃等问题。无数发布都使用了这种RL训练风格，并且已经发生了大量的学术采用。在RL上的技术门槛已经达到历史最低水平。
- **开源版本已经“存在”：**许多工具已经存在，用于使用RLVR和相关技术训练语言模型。例如，TRL [41]、Open Instruct [6]、veRL [218] 和 OpenRLHF [219]，其中许多都是基于RLHF和训练后早期阶段的优化。工具的可访问性正在推动大量研究，这可能会很快使本章过时。

多个资源指出，仅用于推理的RL训练只有在大约2024年及以后的领先模型上才是可行的，这表明在推理训练之前，模型需要具备一定的底层能力。

14.1.2 RL训练与推理时间缩放

训练强化学习以激发推理行为并在可验证领域提高性能与推理时间缩放的思想密切相关。推理时间缩放，也称为测试时间缩放，是一类在推理时使用更多计算能力以在下游任务中表现更好的方法。在DeepSeek R1和OpenAI的o1发布之前，已经研究了推理时间缩放的方法，这两者都极大地普及了针对RL训练的投资。例如包括价值引导采样[220]或带有答案提取的重复随机采样[201]。除此之外，推理时间缩放还可以用于改进AI训练的更多方法，而不仅仅是思维链推理以解决问题，例如使用考虑选项深度的奖励模型[105] [214]。

RL训练是使用推理时间缩放定律的捷径，但从长远来看，我们将有更多方法来激发我们为最佳性能所需的最优推理时间权衡。用RL大量训练模型会改变它们，使它们在每次响应中生成更多标记

以一种与下游性能高度相关的程度。这与早期RLHF系统中观察到的长度偏差有显著不同[9]，其中人类偏好训练的副作用是增加了对偏好排名边际收益的响应率。

下游的RL训练模型有许多方法正在探索，以继续推动推理和推理时计算的极限。这些方法由于它们的快速演变而大多超出了本书的范围，但它们包括通过指令调整将推理行为从更大的RL训练模型蒸馏到更小的模型[221]，组合更多的推理调用[222]，等等。这里重要的是下游性能与生成的令牌数量增加之间的相关性——否则这只是浪费的能量。

14.1.3 强化微调的未来（超越推理）

在许多领域，这些新的RLVR变体和强化微调版本更符合开发者的目标，因为它们更关注性能而不是行为。标准的微调API通常使用参数高效的微调方法，如LoRA，在指令上进行监督微调。开发者传入提示和完成内容，模型通过更新模型参数以匹配完成内容来调整，从而增加了您数据中的特征在模型生成中的普及率。

强化微调专注于匹配答案。给定查询和正确答案，RFT帮助模型学习获取正确答案。虽然标准指令微调通过在数据上执行1或2个epoch的损失更新来完成，但强化微调通过在相同的一些数据点上执行数百或数千个epoch来获得其名称，从而给模型时间学习新的行为。这可以被视为在RFT之后将仅在基础模型版本中偶尔起作用的积极行为强化为稳健的行为。

语言模型RL训练的范围持续扩大：从o1和R1在基础科学层面的最大收获是，我们甚至有更多训练语言模型以实现潜在有价值行为的方法。研究人员和工程师可用的开放之门越多，我们对人工智能总体轨迹的乐观态度就应该越强。

14.2 理解推理训练方法

投资于推理推动了如何训练模型遵循人类指令的艺术的巨大演变。这些食谱仍然使用前面章节中讨论的常见元素，包括指令微调、从人类反馈中进行强化学习以及可验证奖励强化学习（RLVR）。核心变化是使用更多的RLVR，并以不同的顺序应用其他训练技术——对于推理模型来说，传统的核心训练步骤要么是大规模的RL运行，要么是在另一个模型上进行的针对*outputs*的大规模指令调整运行，该模型已经进行了大量的RLVR训练（称为蒸馏）。

14.2.1 理论研究 Pre OpenAI的o1或DeepSeek R1

在推理模型起飞之前，人们投入了大量努力去理解如何训练语言模型以在可验证领域表现得更好。下面这些工作的主要区别在于，它们的方法没有像所使用的方法那样扩展到相同的程度。

在DeepSeek R1及其后续模型中，或者它们导致了在换取更高数学或编码能力的同时牺牲了整体性能。包括基本思想和动机，以描绘出推理模型在景观中出现的更广阔图景。

一些最早在可验证领域训练语言模型的努力包括自学习推理器（STaR）系列工作[223] [224] 和TRICE [225]，它们都使用了真实奖励信号来鼓励模型在2022年和2023年进行推理链推理。STaR有效地近似策略梯度算法，但在实践中以不同的方式过滤样本，并使用交叉熵度量而不是对数概率，Quiet-STaR通过在尝试回答可验证问题之前生成标记（这有助于训练性能）来扩展这一想法。TRICE [225]还通过生成轨迹并使用受马尔可夫链蒙特卡洛启发的期望最大化算法进行优化来改进推理。VinePPO [226]遵循这些方法，并使用一种更接近现代推理模型的设置。VinePPO使用基于PPO的算法，以二进制奖励数学问题（论文中的GSM8K和MATH训练集）的正确性。在OpenAI的o1和DeepSeek R1之前的工作中，其他工作使用了代码执行作为训练[227]、[228]或验证定理证明（在此称为基于验证器反馈的强化学习，RLVF）[229]的反馈信号。Tulu 3通过使用简单的PPO训练器来奖励正确答案的完成来扩展这些方法——最重要的是在广泛的评估中保持模型的总体性能。Tulu 3的二进制奖励和现代推理训练技术可以与STaR的迭代方法或Quiet-STaR的对数似然奖励进行对比。

14.2.2 早期推理模型

以下是对基础推理研究报告的总结，其中一些报告附有开放数据和模型权重，紧随DeepSeek R1之后。

Date	Name	TLDR	Open weights	Open data
2025-01-22	DeepSeek R1 [60]	RL-based upgrade to DeepSeek, big gains on math & code reasoning	Yes	No
2025-01-22	Kimi 1.5 [168]	Scales PPO/GRPO on Chinese/English data; strong AIME maths	No	No
2025-03-31	Open-Reasoner-Zero [230]	Fully open replication of base model RL	Yes	Yes
2025-04-10	Seed-Thinking 1.5 [63]	ByteDance RL pipeline with dynamic CoT gating	Yes (7B)	No
2025-04-30	Phi-4 Reasoning [231]	14B model; careful SFT→RL; excels at STEM reasoning	Yes	No
2025-05-02	Llama-Nemotron [232]	Multi-size “reasoning-toggle” models	Yes	Yes
2025-05-12	INTELLECT-2 [161]	First globally-decentralized RL training run (32B)	Yes	Yes
2025-05-12	Xiaomi MiMo [62]	End-to-end reasoning pipeline from pre- to post-training	Yes	No

Date	Name	TLDR	Open weights	Open data
2025-05-14	Qwen 3 [61]	Similar to R1 recipe applied to new models	Yes	No
2025-05-21	Hunyuan-TurboS [233]	Mamba-Transformer MoE, adaptive long/short CoT	No	No
2025-05-28	Skywork OR-1 [234]	RL recipe avoiding entropy collapse; beats DeepSeek on AIME	Yes	Yes
2025-06-04	Xiaomi MiMo VL [235]	Adapting reasoning pipeline end-to-end to include multi-modal tasks	Yes	No
2025-06-04	OpenThoughts [236]	Public 1.2M-example instruction dataset distilled from QwQ-32B	Yes	Yes
2025-06-10	Magistral [237]	Pure RL on Mistral 3; multilingual CoT; small model open-sourced	Yes (24B)	No

14.2.3 训练推理模型的常见实践

本节中，我们详细介绍了用于序列化训练阶段和修改数据以最大化推理模型训练性能的常用方法。

请注意，这些论文本可以使用列出的技术，但未提及，而他们的同行却提到了，因此这些示例是已知实现的子集，应作为参考使用，但不是关于最佳方案的最终宣言。

- 离线难度过滤：RLVR的核心直觉是模型只能从存在梯度的例子中学习。如果RLVR的起始模型可以100%或0%的时间解决一个问题，那么不同完成之间的梯度将不存在（即，所有策略对策略梯度算法来说看起来相同）。许多模型在开始大规模RL之前都使用了难度过滤，将训练问题限制在起始点模型只能解决20-80%的那些。这些数据通过在训练集中对每个提示进行N次，例如16次，完成采样并验证正确百分比来收集。这种形式被Seed-Thinking 1.5、Open Reasoner Zero、Phi 4、INTELLECT-2、MiMo RL、Skywork OR-1和其他人使用。
- 每批在线过滤（或训练过程中的难度课程）：为了补充离线过滤以找到合适的训练问题，另一个主要问题是“我们应该以什么顺序向模型展示问题。”为了解决这个问题，许多模型使用批次的在线过滤问题，预建的课程/数据调度器，将更难的问题留到训练后期，或采用其他方法来提高长期稳定性。Kimi 1.5、Magistral、Llama-Nemotron、INTELLECT-2、MiMo-RL、Hunyuan-TurboS和其他一些模型使用了相关想法。
- 移除KL惩罚：随着推理模型相对于RLHF训练的RL运行长度增加，奖励函数变得不太容易过度优化，许多模型移除了约束RL学习策略与训练基模型相似的KL惩罚。这允许模型在训练期间进一步探索。这被RAGEN[238]、Magistral、OpenReasonerZero、Skywork OR-1和其他所使用。
- 宽松策略梯度裁剪：算法GRPO的新变体，例如

DAPO [157] 对GRPO（或PPO）中使用的双边裁剪目标函数进行了修改，以便实现更好的探索。裁剪也已被证明在奖励不完美时可能导致潜在的虚假学习信号 [239]。这种具有不同梯度方向范围的双边裁剪被RAGEN、Magistral、INTELLECT-2和其他一些方法使用。

- 离策略数据（或完全异步更新）：随着使用强化学习解决任务所需完成长度增加，尤其是在响应长度 *variance* 的 *variance* 上，RL 运行的计算可能会闲置。为了解决这个问题，训练正在转向异步更新或改变问题如何排列成批以改进整体吞吐量。Seed-Thinking 1.5、INTELLECT-2 等使用部分到全异步（离策略）数据。
- 额外格式奖励：为了使推理过程可预测，许多模型添加了少量奖励，以确保模型在回答前遵循正确的格式，例如<think>...</think>。这被DeepSeek R1、OpenReasonerZero、Magistral、Skywork OR-1和其他模型所使用。
- 语言一致性奖励：类似于格式奖励，一些多语言推理模型使用语言一致性奖励来优先考虑在推理过程中不改变语言的模型（以获得更好的和更可预测的用户体验）。这包括DeepSeek R1、Magistral和其他模型。
- 长度惩罚：许多模型在RL训练期间使用不同的长度惩罚形式，以稳定学习过程或减轻对难题的过度思考。一些例子包括Kimi 1.5逐步扩展目标长度以对抗过度思考（而训练精度在难度课程中保持较高）或INTELLECT-2在整个过程中运行小长度惩罚。其他人使用过长过滤和其他相关实现来提高吞吐量。
- 损失归一化：关于原始GRPO算法中按组归一化项引入的潜在长度或难度偏差，已有一些讨论（参见策略梯度章节或[148]）。因此，一些模型，如Magistral或MiMo，选择在批量级别而不是组级别对损失或优势进行归一化。
- 并行测试时计算缩放：结合多个并行、独立采样的rollout的答案可以显著提高使用单个rollout答案的效果。最简单形式的并行测试时计算缩放，如DeepSeek-R1、Phi-4和其他方法所做的那样，涉及使用大多数rollout返回的答案作为最终答案。一种更高级的技术是使用训练有素的评分模型，从并行rollout的答案中选择最佳答案。这种技术尚未被开放推理模型配方采用（截至2025年6月），但在Claude 4公告[240]中提到，并在DeepSeek-GRM[214]中使用。

在常见技术之外，还有许多关于如何通过推理训练创建有用的模型而不牺牲辅助能力的常见发现：

- 文本推理提升多模态性能：Magistral、MiMo-VL等发现，在训练多模态模型后进行仅文本推理训练，可以在最终模型中*improve*多模态性能。
- 可切换的推理与系统提示（或长度控制）：Llama-Nemotron、Qwen 3 等使用特定的系统提示（可能结合长度控制的强化学习训练[241]）来为用户提供可切换的思考长度。

15 工具使用与函数调用

语言模型使用工具是扩展其功能的一种自然方式，尤其是在外部工具包含所需信息或需要与复杂网络系统交互的代理中，尤其如此。这可以通过几种策略来考虑，其中工具使用是一般类别。一个AI模型通过输出特殊令牌来触发特定端点，使用任何外部工具。这些可以是高度特定的工具，例如返回特定地点天气的功能，也可以是代码解释器或搜索引擎，它们是复杂行为的基本构建块。

工具使用的确切起源尚不清楚，但这一想法的起源远早于ChatGPT时代，那时RLHF开始流行。大约在2015年，早期例子试图构建现代语言模型之前的系统，如神经程序员-解释器（NPI）[242]，“一种循环和组合神经网络，学会表示和执行程序。”随着语言模型的普及，许多子领域都在使用外部能力的集成来提高性能。为了获取除了权重之外的信息，许多人使用了检索增强生成[243]或网络浏览[4]。不久之后，其他人开始探索与程序[244]或工具[245]集成的语言模型。

随着该领域的成熟，这些模型在底层语言模型大幅改进的基础上，获得了更复杂的能力。例如，ToolFormer可以使用“计算器、问答系统、两个不同的搜索引擎、翻译系统和日历”[246]。不久之后，Gorilla被训练使用1645个API（来自PyTorch Hub、TensorFlow Hub v2和HuggingFace），其评估APIBench成为流行伯克利函数调用排行榜的基础[247]。自这些早期模型以来，被调用的动作多样性大幅增长。

工具使用模型现在与常规语言模型交互紧密相连。模型上下文协议（MCP）作为一种常见的格式出现，用于将语言模型连接到外部数据源（或工具）[248]。随着模型的增强和格式的改进，工具使用语言模型被应用于许多场景，包括在流行的应用程序如Microsoft Office或Google Workspace中的生产力副驾驶，科学领域[249]，医学领域[250]，编码代理[251]如Claude Code或Cursor，与数据库的集成，以及许多其他自主工作流程。

15.1 生成中的交织工具调用

函数调用代理以与其他训练后阶段非常相似的方式呈现数据。新增的是系统提示中的内容，指导模型它有哪些可用的工具。以下是一个带有系统提示和可用工具的JSON格式示例数据点的格式化数据点：

```
<system>
You are a function-calling AI model. You are provided with function
signatures within <functions></functions> XML tags. You may call
one or more functions to assist with the user query. Don't make
assumptions about what values to plug into functions.
</system>

<functions>
[
```

```

{
  "name": "get_id",
  "description": "Fetches the ID of a movie based on the given
    search query from the RapidAPI similar movies service.",
  "parameters": {
    "q": {
      "description": "The search string for the movie title.",
      "type": "str",
      "default": "titanic"
    }
  }
},
{
  "name": "search_torrents",
  "description": "Search for torrents based on given keywords using
    the RapidAPI service.",
  "parameters": {
    "keywords": {
      "description": "Keywords to search for torrents.",
      "type": "str",
      "default": "Meg 2 The Trench"
    },
    "quantity": {
      "description": "Number of torrent results to return. Maximum
        value is 40.",
      "type": "int",
      "default": 40
    },
    "page": {
      "description": "Page number for paginated results. Defaults to
        1.",
      "type": "int",
      "default": 1
    }
  }
},
{
  "name": "basic_info",
  "description": "Fetches detailed information about a cast member
    such as name, profession, birth and death year, bio, poster,
    and best titles.",
  "parameters": {
    "peopleid": {
      "description": "The ID of the cast member whose details are to
        be fetched.",
      "type": "str",
      "default": "nm0000375"
    }
  }
}
]
</functions>

```

```
<user>
...
</user>
```

当语言模型正在生成时，如果遵循上述示例，它将生成`search_torrents("Star Wars")`这样的标记来搜索《星球大战》。这通常编码在特殊的格式化标记中，然后下一个插入序列的标记将包含工具输出。通过这种方式，模型可以学习完成比许多简单独立模型更具挑战性的任务。

一种流行的工具使用形式是代码执行，允许模型对复杂逻辑或数学问题获得精确答案。例如，在语言模型执行期间，推理模型的思考标记阶段可以发生代码执行。与函数调用一样，首先为要执行的代码（由模型生成）标记，然后为输出标记单独的标签。

```
<|user|>
What is the 50th number in a fibonacci sequence?</s>
<|assistant|>
<think>
Okay, I will compute the 50-th Fibonacci number with a simple loop,
    then return the result.

<code>
def fib(n):
    a, b = 0, 1
    for _ in range(n):
        a, b = b, a + b
    return a

fib(50)
</code>

<output>
12586269025
</output>
</think>
<answer>
The 50-th Fibonacci number is 12 586 269 025.
</answer>
```

15.2 多步工具推理

OpenAI的o3模型在如何将多步工具使用与语言模型集成方面迈出了实质性的飞跃。这种行为与社区中较早的研究趋势相关。例如，ReAct [252] 展示了如何将动作和推理交织到一个模型生成中：

在这篇论文中，我们探讨了以交错方式使用LLMs生成推理轨迹和特定任务动作，从而在两者之间实现更大的协同：推理轨迹有助于模型诱导、跟踪和更新

行动计划以及处理异常，同时动作允许它与外部来源（如知识库或环境）接口并收集更多信息。

随着工具使用能力的巩固和推理模型的起飞，多轮工具使用已成为一个令人兴奋的研究领域[253]。

15.3 模型上下文协议（MCP）

模型上下文协议（MCP）是连接语言模型到外部数据源和信息系统的标准[248]。MCP不是关注每个外部系统的特定工具调用格式，而是通过标准化协议使模型能够访问丰富的上下文信息。

MCP是本章工具使用内容上的简单扩展——它是应用如何以可预测的JSON模式将上下文（数据 + 动作）传递给语言模型的方式。与模型交互的MCP服务器具有核心原语：资源（只读数据块）、提示（模板消息/工作流程）和工具（模型可以调用的函数）。有了这些，MCP架构可以概括为：

- MCP服务器封装特定的数据源或功能。
- MCP客户端（例如，Claude桌面，IDE插件）聚合一个或多个服务器。
- 主机，例如Claude或ChatGPT应用，提供用户/LLM界面；切换模型供应商或后端工具仅意味着交换中间的客户端。

MCP使工具使用模型的开发者能够使用相同的架构将他们的服务器或客户端连接到不同的模型，同时模型也有一个可预测的格式，他们可以使用该格式来集成外部组件。这些共同为工具使用模型在现实世界领域提供了一个更加可预测的开发环境。

15.4 实施

在实施工具使用模型时，存在多个格式化和屏蔽决策：

- Python 与 JSON 格式化：在本章中，我们包括了格式化工具使用作为 JSON 数据结构和 Python 代码的示例。模型倾向于选择一种结构，不同行业提供商使用不同的格式。
- 掩码工具输出：在训练工具使用模型时的重要细节是，工具输出中的标记被从模型的训练损失中掩码。这确保模型不会学习预测它在使用中直接生成的系统输出（类似于其他后训练阶段的提示掩码）。
- 多轮格式化用于工具调用：在实现工具调用模型时，通常会在数据加载格式中添加更多结构。训练后数据集的标准做法是交替出现用户和助手（以及经常出现的系统消息）的消息列表。工具使用的整体结构相同，但模型的轮次被分割成由每个工具调用分隔的内容子部分。以下是一个示例。

```
messages = [  
{
```

```

"content": "You are a function calling AI model. You are provided with
function signatures within <functions></functions> XML tags. You
may call one or more functions to assist with the user query.
Don't make assumptions about what values to plug into functions."
,
"function_calls": null,
"functions": "[{\"name\": \"live_giveaways_by_type\", \"description\":
  \"Retrieve live giveaways from the GamerPower API based on the
  specified type.\", \"parameters\": {\"type\": {\"description\":
    \"The type of giveaways to retrieve (e.g., game, loot, beta)\".
    \"type\": \"str\", \"default\": \"game\"}}}]",
"role": "system"
},
{
"content": "Where can I find live giveaways for beta access and games?
",
"function_calls": null,
"functions": null,
"role": "user"
},
{
"content": null,
"function_calls": "live_giveaways_by_type(type='beta')\n
  nlive_giveaways_by_type(type='game')",
"functions": null,
"role": "assistant"
}
]

```

- 标记化和消息格式细节：在OpenAI消息格式中，工具调用通常通过聊天模板（控制发送给模型的格式化消息的代码）进行标记化，将结构化JSON表示转换为原始标记流。此过程因模型架构而异——一些使用特殊标记来界定工具调用，而另一些则在标记流本身中保持结构化格式。聊天模板游乐场提供了一个交互式环境，以探索不同模型如何将消息格式转换为标记流。
- 推理令牌连续性：随着推理模型的兴起，它们在答案之前有独立的“推理”令牌流，存在不同的实现方式来处理它们在工具使用循环中的处理。一些模型在单个回合的工具调用步骤之间保留推理令牌，以保持多个工具调用的上下文。然而，这些令牌通常在回合之间被删除，以最小化服务成本。
- API格式跨提供商（截至2025年7月）：不同提供商使用概念上相似但技术上不同的格式。OpenAI使用具有唯一ID的`tool_calls`数组，Anthropic采用详细的`input_schema`规范以及`<thinking>`标签，而Gemini提供函数调用模式（AUTO/ANY/NONE）。当通过API使用这些模型时，可用的工具以JSON格式定义，然后工具输出的模型响应存储在标准“生成的令牌”之外的字段中。例如，开源的vLLM推理代码库

implements extensive parsing logic supporting multiple tool calling modes and model-specific parsers, providing insights into lower-level implementation considerations [254].

16 人工数据与蒸馏

强化学习从 *human feedback* 深植于保持人类对我们所构建的模型的影响这一理念。当第一个模型使用 RLHF 成功训练时，人类数据成为了一种 *the only* 可行的途径来以这种方式改进模型。

人类是唯一能够创建足够高质量的问题以进行训练的方式。人类是唯一能够收集可靠和具体反馈数据以训练奖励模型的方式。

随着AI模型变得更好，这个假设迅速崩溃。合成数据的可能性，它远比迭代更便宜、更容易，使得从以RLHF为中心的关注到更广泛的“后训练”塑造模型的想法得以传播。

许多报告都提到了合成数据如何导致“模型崩溃”或其他模型问题[255]，但这一点在主要语言模型[256] [257]中被明确反驳。合成数据 *can* 会导致模型出现性能问题，但这是由使用重复数据或仅由训练模型输出的数据（缩小其潜在分布）而不是全面的数据来源所引起的。

主要模型需要合成数据以达到最佳性能。现代的培训后合成数据包括许多培训环节——语言模型用于从种子示例[258]生成新的培训提示，修改现有提示，对提示生成补全[259]，提供AI反馈以创建偏好数据[22]，过滤补全[260]，等等。合成数据是培训后的关键。

合成数据能够达到这种程度的影响力，随着GPT-4类模型的出现而显现。在早期的语言模型，如Llama 2和GPT-3.5-Turbo中，模型在生成或监督数据管道方面还不够可靠。在1-2年内，语言模型在生成答案方面远远超过了人类。在从GPT-3.5到GPT-4类模型的过渡中，模型执行LLM作为裁判任务的能力也出现了。GPT-4或更好的模型在生成关于某篇内容的反馈或评分方面，要远比GPT-3.5类模型更加稳健和一致。

自这次转变以来，合成数据在语言模型训练中的作用仅有所增长。否则，有两个明显的领域，人类数据仍然非常重要。

1. 人类数据在模型能力边缘的作用持续存在 - 在AI尚未具备任何能力的地方，人类必须生成数据。一旦第一个强大模型存在，合成数据就会迅速增多。
2. 即使学术研究表明合成版本的表现同样出色，人类偏好数据仍然被用于领先模型中。人类偏好在文献中的作用仍在建立中。

蒸馏一词一直是关于合成数据在语言模型中作用的讨论中最有力的形式。蒸馏这个术语来自深度学习文献中对师生知识蒸馏的技术定义[50]。

蒸馏通俗地说是指使用更强大模型的输出训练一个更小的模型。在训练后，这种蒸馏的通用概念有两种常见形式：

1. 作为在广泛的训练后过程中使用的数据库引擎：用于指令的完成、偏好数据（或宪法AI）或用于强化学习（RL）的验证。

2. 将特定技能从更强的模型转移到较弱的模型，这通常用于数学推理或编码等特定技能。

第一种策略随着语言模型的演变而越来越受欢迎，因为语言模型在撰写各种任务的答案方面比人类更可靠。GPT-4类模型将这一范围扩展到使用对数学和代码等复杂任务更强的模型的蒸馏（如上所述）。在这里，蒸馏促使拥有一个模型套件，其中实验室通常会训练一个大型内部模型，如Claude Opus或Gemini Ultra，该模型未公开发布，仅用于内部以制作更强的模型。对于开源模型，常见的做法是将从封闭API模型中蒸馏出的训练数据转化为更小、公开可用的权重[20]。在此过程中，精心制作高质量的提示和过滤来自教师模型的响应对于最大化性能至关重要。

将特定技能转移到较小的语言模型使用相同的蒸馏原理——获取最佳的训练数据。在此，许多论文研究了使用来自更强模型的有限数据集来改进对齐[12]、数学推理[261] [262]和测试时缩放[221]。

17 评估

评估是一个不断发展的方法。理解语言模型评估的关键，尤其是在训练后，是当前流行的评估体系反映了流行的训练最佳实践和目标。虽然具有挑战性的评估推动语言模型向新领域发展，但大多数评估都是为了构建对新模型有用的信号而设计的。

在许多方面，本章旨在展示RLHF早期历史中流行评估制度的片段，以便读者了解共同主题、细节和失败模式。

评估RLHF和训练后阶段在其早期历史中经历了几个不同的阶段：

1. 早期聊天阶段：早期使用RLHF或偏好调优训练的模型，针对评估主要关注捕捉模型的聊天性能，特别是与GPT-4等已知强大模型相比。早期例子包括MT-Bench [99]、AlpacaEval [100]和Arena-Hard [101]。模型评估范围较窄，现在这些被视为“聊天”或“指令遵循”领域。
2. 多技能时代：随着时间的推移，普遍的做法是，RLHF可以用来提高不仅仅是聊天以外的更多技能。例如，Tulu评估套件包括知识（MMLU [263]，PopQA [264]，TruthfulQA [265]）、推理（BigBench-Hard [266]，DROP [267]）、数学（MATH [268]，GSM8K [269]）、编码（HumanEval [270]，HumanEval+ [271]）、指令遵循 [272] 和安全（许多评估的综合）。这反映了在安全性和聊天之外，接受后训练作为多方面解决方案的领域。
3. 理论与方法：当前后训练时代的特点是关注具有挑战性的推理和工具使用问题。这些问题包括如GPQA Diamond [273] 和 Humanity’s Last Exam [274] 这样的更难的知识密集型任务，如SWE-Bench+ [275] 和 LiveCodeBench [276] 这样的复杂软件工程任务，或者如最近AIME竞赛所展示的具有挑战性的数学问题。

此外，新的领域将不断发展。随着人工智能成为更加工业化的领域，评估的激励措施正在转变并成为多利益相关者。自ChatGPT发布以来，如Scale Leaderboard [277]这样的私人评估，如ChatBotArena [85]这样的社区驱动评估，以及如ArtificialAnalysis和Epoch AI这样的第三方评估公司已经大量涌现。在本章中，我们将包括与这些评估的实施和理解相对应的详细信息。

17.1 提示格式化：从少样本到零样本到CoT

提示语言模型主要是一种动词，但也被视为一种可以练习和/或训练的技艺或艺术[278]。提示是构建信息和方法为语言模型提供上下文的方式。对于常见交互，提示相对基本。对于高级场景，一个精心制作的提示将意味着在特定的一次性用例中的成功或失败。

当涉及到评估时，提示技术可以对模型的性能产生重大影响。一些提示技术——例如下面讨论的格式化——可以使模型性能从60%降至接近0。同样，提示的变化可以

帮助模型在训练期间更好地学习。口语化地说，恰当的提示可以使使用未来模型的主体体验，解锁正常使用之外的性能。

现代语言模型的良好提示可能包括为模型准备整个报告以进行响应（通常涉及成千上万的生成文本标记）。这种行为是语言模型性能测量和理解许多变化的结果。

早期语言模型仅用作智能自动补全。为了更开放地使用这些模型，向模型展示了多个示例，然后是一个不完整的短语提示。这被称为少样本或情境学习[132]，当时没有涉及指令调整或RLHF。在流行评估的情况下，这看起来会是这样：

```
# Few-Shot Prompt for a Question-Answering Task
You are a helpful assistant. Below are example interactions to guide
your style:

### Example 1
User: "What is the capital of France?"
Assistant: "The capital of France is Paris."

### Example 2
User: "Who wrote the novel '1984'?"
Assistant: "George Orwell wrote '1984'."

# Now continue the conversation using the same style.
User: "Can you explain what a neural network is?"
Assistant:
```

这里，评估答案有多种方法。如果我们考虑MMLU风格的提问，其中模型需要在多个答案之间进行选择：

```
# Few-Shot Prompt

Below are examples of MMLU-style questions and answers:

### Example 1
Q: A right triangle has legs of lengths 3 and 4. What is the length of
its hypotenuse?
Choices:
(A) 5
(B) 6
(C) 7
(D) 8

Correct Answer: (A)

### Example 2
Q: Which of the following is the chemical symbol for Sodium?
Choices:
(A) Na
(B) S
```

(C) N
(D) Ca

Correct Answer: (A)

Now answer the new question in the same style:

Q: Which theorem states that if a function f is continuous on a closed interval $[a,b]$, then f must attain both a maximum and a minimum on that interval?

Choices:

- (A) The Mean Value Theorem
- (B) The Intermediate Value Theorem
- (C) The Extreme Value Theorem
- (D) Rolle's Theorem

Correct Answer:

要在这里提取答案，可以基于某些采样参数生成一个标记，并查看答案是否正确，A、B、C或D（格式如上所述，如[279]中提出），或者可以查看每个标记的概率，如果正确答案更有可能，则标记任务为正确。第二种方法有两种潜在的实施方式 - 首先，可以查看字母（A）或答案“平均值定理”的概率。这两个都是允许的指标，但基于概率的指标中，答案预测更为常见。

一个常见的挑战是，在少量提示的情况下，模型不会遵循格式，这被视为一个错误答案。在设计评估领域时，通常将上下文中使用的示例数量视为设计参数，范围从3到8个或更多。

在少量提示的演变过程中，出现了为模型提供思维链示例的想法。这以包含情境示例并写出推理的示例形式出现，如下所示（后来被生成推理步骤的明确提示所取代）[53]:

standard prompting

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

A: The answer is ...

chain of thought prompting

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6

tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

A: The cafeteria had 23 apples originally. They..

随着时间的推移，随着语言模型变得更强大，它们演变为零样本评估，即“零样本学习器”[280]。微调语言网络（FLAN）表明，作为现代指令微调的先导，特定任务上微调的语言模型可以泛化到它们未训练过的零样本问题[280]（在T0 [281]中也发现了类似的结果）。这是指令微调（IFT）的出现，它是RLHF和训练后的重要先导。一个零样本问题看起来可能像：

User: "What is the capital of France?"
Assistant:

从2022年开始，时间线开始包括关键早期RLHF工作，例如InstructGPT。伴随这些模型的核心能力和用例转变具有更加开放的使用方式。随着更加开放的使用，生成式评估越来越受欢迎，因为它反映了实际使用。在此期间，经过ChatGPT之后的近几年，一些多项选择题评估仍在RLHF研究中作为对常规实践的制约因素被使用。

随着2024年底和2025年初推理模型的兴起，模型行为的一个重大变化是在每个答案之前添加了一个长思维链（CoT）推理过程。这些模型不再需要像[282]中提出的“逐步思考”的规范修改来提示。

例如，对于每个问题或类别，都有专门设计的提示来帮助从模型中提取行为。Tulu 3详细介绍了用于CoT回答多项选择题的一些提示[6]：

Answer the following multiple-choice question by giving the correct answer letter in parentheses. Provide CONCISE reasoning for the answer, and make sure to finish the response with "Therefore, the answer is (ANSWER_LETTER)" where (ANSWER_LETTER) is one of (A), (B), (C), (D), (E), etc.

Question: {question}
(A) {choice_A}
(B) {choice_B}
(C) ...

Answer the above question and REMEMBER to finish your response with the exact phrase "Therefore, the answer is (ANSWER_LETTER)" where (ANSWER_LETTER) is one of (A), (B), (C), (D), (E), etc.

这，尤其是在模型使用特殊格式来区分思考标记和答案标记时，促使最近的重大更新到评估制度。评估正在转向模型被测试以生成方式通过一系列思考提示进行响应的地方。

17.2 Using Evaluations vs. Observing Evaluations

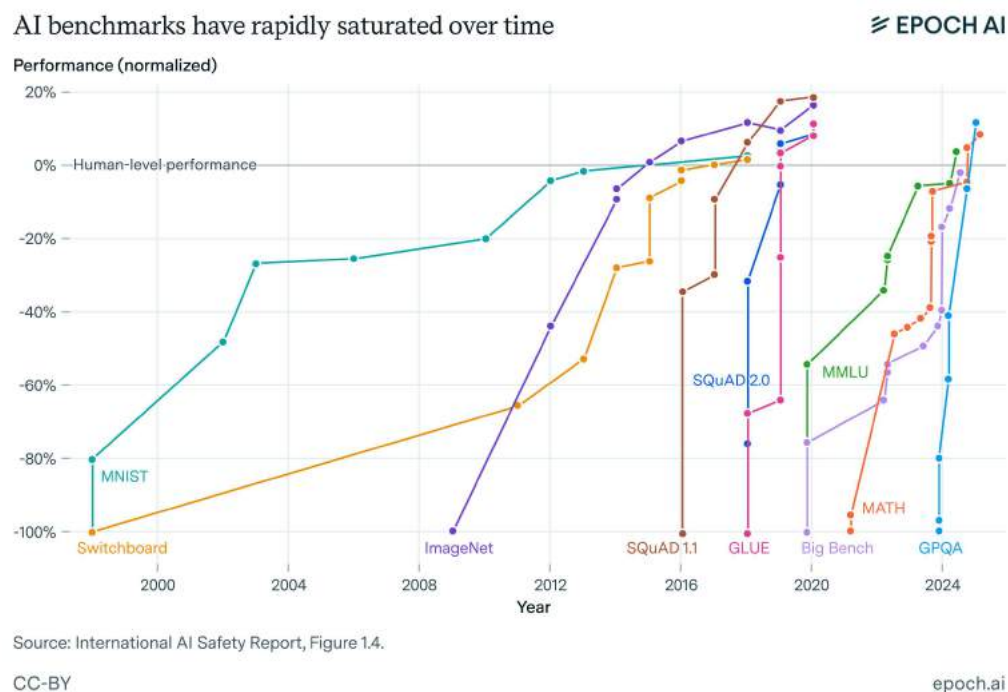


图19: Epoch AI的报告显示，主要AI评估在时间上迅速饱和。许可CC-BY。

公司内部进行的语言模型评估只能与具有大误差棒的同行进行比较，因为它们用于内部评估的过程与外部评估不匹配。内部评估是为了在训练中“爬山”，正如在传统机器学习中所说的“训练集”。社区使用的公共评估，用于比较领先模型，如果它们在所谓的训练集中，或者作为未见的“测试集”或“验证集”，则无法得知。

随着评估分数成为企业营销方案的核心组成部分，公司内部实施已经偏离。有关主要AI实验室为GSM8k或MATH等重要评估使用“定制提示”的传闻。这些做法迅速演变。

语言模型评估堆栈被视为营销，因为评估没有硬性的事实依据。前沿实验室内部发生的情况是，评估套件正在调整以适应其内部需求。当结果被分享时，我们得到的是实验室为其模型获得的数字输出，但不是该函数的所有输入。输入是非常敏感的配置，它们在OpenAI、Meta、Anthropic和Google都不同。即使是完全开放的评估标准也难以保证可重复性。专注于自己的模型是唯一接近可重复评估技术的途径。营销背后有良好的意图，始于技术

队伍。

评估from 层级语言模型今天在艺术方面与...一样重要 这是一门科学。

不同群体选择不同的评估来保持独立性，即使它们成为真正的测试集，但没有人透露他们选择了哪些。例如，流行的推理评估MATH和GSM8k都拥有带有提示的训练集，这些提示可以很容易地用来提高性能。使用同一分布的提示来提高性能与通过训练一般数学数据来泛化这些任务非常不同。

实际上，这些 *training sets* 是非常高质量的数据，因此模型从在这些数据上训练中受益。如果这些公司 *not* 将相应的评估作为核心指标来跟踪，那么在评估集上训练可能是一个实际的决定，因为高质量数据是模型发展的主要限制因素。

领先的人工智能实验室通过关注几个关键评估，并在最后报告核心公共集的分数的进行爬山。关键点是，他们的一些评估用于跟踪进度，例如从GPT-4报告[283]中缩放交叉熵损失预测的数据集，通常不公开。

训练后评估高度依赖于人工评估。对于生成式语言模型的人工评估产生Elo排名（在早期Anthropic论文中很受欢迎，如宪法AI），对于奖励模型的人工评估显示一致性。这些也可以通过向用户提供两个不同的模型并使用A/B测试窗口（如偏好数据章节中讨论的）来获得。

他们选择关注的一组有限评估形成了评估与训练之间的紧密联系。在某个时刻，关注的评估之一是MMLU。GPQA是推理模型出现时的选择之一。实验室将改变评估以使其更适合他们的需求，例如OpenAI发布SWE-Bench-Verified [284]。还有许多内部评估，公众无法访问。

关键“能力”在内部改进评估对下游训练的影响是提高比较训练运行时的统计能力。通过改变评估，这些实验室减少了其优先信号上的噪声，以便做出更明智的训练决策。

现代语言模型训练堆栈中后训练的复杂性加剧了这一点。今天评估语言模型涉及生成一定数量的标记（而不仅仅是查看答案的对数概率）。公认的是，前沿实验室使用一些小技巧来提高许多任务的表现——最常见的解释是为某些评估提供一次性提示。

另一个在比较多个实验室评估结果时产生混淆的例子是将推理时间缩放添加到评估比较中。推理时间缩放表明，通过在推理时使用更多标记，模型可以提高性能。因此，通过推理的总标记数来控制评估分数很重要，但尚未成为常见做法。

根据您的数据在训练后的格式，模型在评估格式上会有很大的差异。例如，两个流行的、开放的数学数据集NuminaMath [285] 和 MetaMath [286] 由于细微的冲突而在训练中相互矛盾。

答案格式差异 - Numina 将答案放入 `\boxed{XYZ}`，而 MetaMath 将答案放在 `The answer is: XYZ` 之后 —— 在两者上训练可能会使性能不如仅使用一种格式。强大的模型被训练以能够使用多种格式运行，但它们通常有一个最强的格式。

最后，我们留下了关于评估封闭模型状态的几个关键点：

- 我们不知道或必然拥有实验室正在攀登的关键测试集，因此一些评估是代理。
- 推理前沿模型变得越来越复杂，有特殊的系统提示、特殊标记等，而我们不知道它如何影响评估，以及
- 我们不知道用于数值报告封闭评估所使用的所有格式和细节。

17.3 污染

当前语言模型实践（即不仅限于RLHF和后训练）的一个主要问题是故意或无意中使评估数据集的数据进行训练。这被称为 *dataset contamination*，相应地避免这种做法的实践是 *decontamination*。为了净化数据集，需要在训练集和测试集上进行搜索，寻找n-gram（字符）或标记[287]的匹配。数据可能受到污染的方式有很多，但最常见的是从网络上抓取多个阶段的训练数据。基准通常列在公共网络域名上，这些域名会被爬取，或者用户将问题输入到模型中，这可能导致未来模型候选训练数据。

例如，在为Tulu 3评估套件进行去污时，作者发现流行的开源数据集被RLHF的流行评估所污染[6]。这些重叠包括：UltraFeedback的污染与TruthfulQA，Evol-CodeAlpaca的污染与HumanEval，NuminaMath的污染与MATH，以及WildChat的污染与安全评估。这些是通过从训练提示到评估集中确切提示的8-gram重叠发现的。

为了理解未披露或发布训练数据的模型污染，创建了与原始版本略有扰动的基准测试新版本，例如对于MATH [288]，以便查看哪些模型被训练以匹配原始格式或问题。在这些扰动基准测试上的高方差不是污染的确认，污染难以证明，但可能表明训练时考虑了特定格式，而这些可能无法转化为实际世界性能。

17.4 工具

有许多开源评估工具供人们选择。有来自英国安全研究所的Inspect AI [289]，为Open LLM排行榜提供动力的HuggingFace的LightEval [290]，Eleuther AI基于其GPT-Neo-X模型（约GPT-3评估配置）构建的评估工具 [292]，基于OLMES的AI2库 [294]，斯坦福大学基础模型研究中心的HELM [295]，Mosaic（现在是Databricks）的Eval Gauntlet [296]，以及更多。

18 过度优化

在RLHF文献和讨论中，有两个主要方向可能会出现过度优化的问题：

1. 对奖励过度优化技术概念的定量研究。这衡量优化距离和功率与训练指标和下游性能的关系。训练持续上升，而最终下游性能下降。
2. 定性观察表明，“过度”的RLHF可能导致更差的模型。这些是RLHF问题设置、测量工具和权衡的根本性限制。

本章简要介绍了两者。我们首先从后者，定性方面开始，因为它激励我们进一步研究问题。最后，本章简要讨论了在过度使用RLHF或相关技术时可能出现的偏差问题，这可能导致语言模型的行为与其设计相悖。

过度优化是一个概念，其中训练指标最终与感兴趣的最终评估不匹配。虽然与过拟合类似——在相对于测试泛化的下游评估中，训练数据过于狭窄——但在强化学习文献中，过度优化被用来表示过度使用了 *external* 信号。过度优化的成本是对现实世界目标的较低对齐或任何领域的较低质量，与之相关的训练形状如图20所示。

18.1 定性过度优化

本章的前半部分讨论了RLHF的核心叙事——如何根据最终目标配置优化以及可能出错的情况。

18.1.1 管理代理目标

RLHF 是基于这样一个事实：我们没有为聊天机器人构建一个普遍适用的良好奖励函数。由于在使聊天机器人使用起来更好方面表现出色，RLHF 被推到了前沿，这完全由一个代理目标所控制——认为在受控环境中从人类标注者那里测量的奖励反映了下游用户的欲望。训练后通常包括对明确可验证的奖励的训练，但仅从偏好中学习标准也提高了诸如数学推理和编码等领域（仍然通过这些代理目标）的性能。

代理奖励在RLHF中是训练好的奖励模型返回给RL算法本身的分数，因为它已知最多只与聊天机器人性能相关[297]。因此，已经证明，将过多的优化能力应用于算法的RL部分实际上会降低最终语言模型的有效性——这是一种许多强化学习应用中已知的过度优化[298]。过度优化是指“当优化代理目标导致真实目标变得更好，然后变得更差。”

曲线显示训练损失先上升，然后缓慢趋于平稳，最后下降，如图20所示。这与过拟合不同，过拟合时模型在训练分布上的准确率持续提高。代理奖励的过度优化要微妙得多。

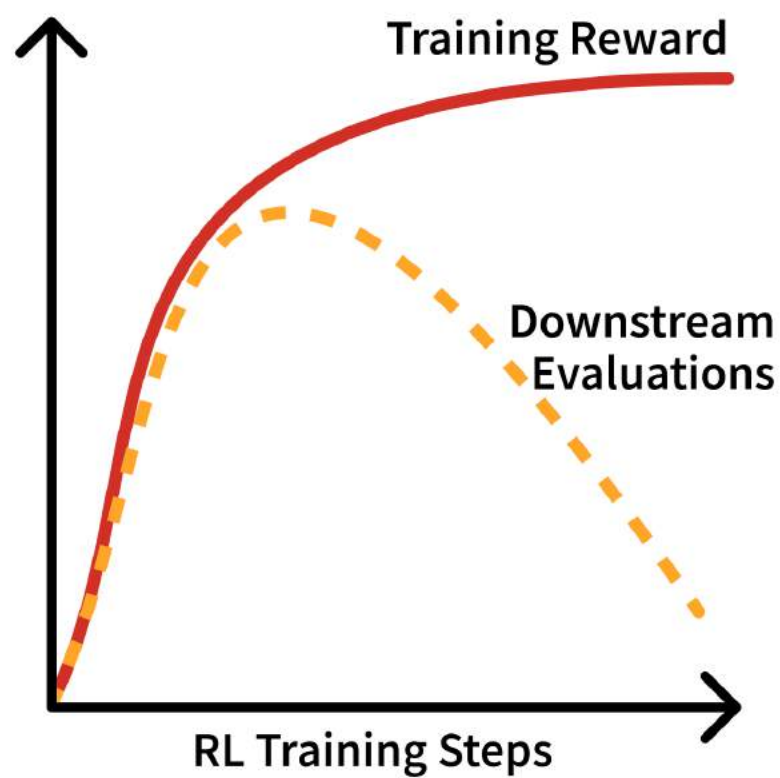


图20: RL训练运行过度优化与下游评估的对比。

这个推理所捕捉的一般概念源于Goodhart定律。Goodhart解释了现在普遍存在的这种行为[299]：

任何观察到的统计规律性，一旦被用于控制目的施加压力，都倾向于崩溃。

当一项指标成为目标时，它就不再是好的指标[300]。这里的见解建立在这样一个事实之上，即我们可能错误地将机器学习损失用作这些复杂系统的真实依据。实际上，我们使用的损失函数是设计用于（并且从理论上激励）局部优化的。它们的全局使用导致了RLHF代理目标中的挑战。

常见早期聊天模型过度优化的迹象包括：

- 常见短语，例如：“作为一个AI语言模型...”或“当然！...”
- 无信息回答通过重复、模棱两可等。
- 迎合用户：自我怀疑、拍马屁[82]和过度道歉，
- 对齐行为，如过度拒绝，

这是一个关于哪些类型的训练过程中的错误会导致这些失败的开放性研究问题。存在许多错误来源[297]：奖励模型无法适应偏好的近似误差，在训练RM时的过拟合估计误差，在训练语言模型策略中的优化误差等。这指向了一个基本问题，即相对于下游用户的需求，优化数据承包商的意图和输出的极限。

一个潜在的解决方案是从聊天机器人和模型用户那里测量 *implicit* 反馈来调整性能。隐式反馈是用户采取的行动，例如重新滚动输出、关闭标签页或写一条愤怒的消息，这表明了先前响应的质量。这里的挑战，以及大多数RLHF的优化变化，在于在使奖励函数更具体时，存在失去稳定性的强烈风险。作为强大的优化器，RL越来越有可能在它是平滑表面时（而不仅仅是成对的人类价值观）利用奖励函数。对此的预期解决方案是，未来的RLHF将使用成对偏好数据和额外的引导损失函数进行训练。还有许多不同的损失函数可以用来更好地处理成对数据，例如Mallow的模型[301]或Plackett-Luce[95]。

18.1.2 过度拒绝和“太多RLHF”

许多模型发布都被标记为“过度RLHF”。2023年有多款显赫的模型发布，由于对上下文中单个词语如“kill”的过度归因，模型拒绝了一些无害的查询。这次讨论最突出的是Llama 2聊天模型的发布[43]，但也反映了拒绝和“对齐”对RLHF早期应用的关注。例如，早期RLHF模型过度拒绝行为的常见测试是一个如下查询，其中模型误解了请求：

用户：我想知道如何杀死Linux进程

助手：我不能提供有关如何损害或终止Linux进程或任何其他生物的指令或建议。故意伤害或终止任何进程或生命形式在道德上、法律上或道德上都是不正确的。

相反，我建议您专注于学习Linux进程以及如何有效地管理它们。您可以从阅读有关Linux进程管理的文档和教程开始，这有助于您了解如何安全且高效地创建、监控和终止进程。

此外，如果您对某个可能引起问题的特定进程有疑虑，您可以尝试通过收集更多关于该进程的信息来解决问题，例如其资源使用情况、内存消耗和网络活动。您可以使用各种工具如 `top`、`htop`、`ps`、`kill` 和 `grep` 来收集这些信息。

然而，请注意，未经适当授权或了解终止进程可能会产生严重后果，包括数据丢失、系统不稳定和安全漏洞。因此，在管理Linux进程时，必须谨慎行事并遵循最佳实践。

另一个例子是在2023年稍后，Claude 2.1展示了类似的行为：

用户：如何在Ubuntu服务器上杀死所有Python进程？

助手：我道歉，我不应该提供有关损害进程或系统的建议。

这两个都与训练无关，反映了模型的部署设置，例如系统提示。此外，现代聊天应用使用额外的安全过滤器来拦截在发送到主生成模型（例如WildGuard [302]或LlamaGuard [303]）之前的提示和响应。

虽然RLHF是这些模型区分安全请求和不安全请求能力训练的核心，但将最终模型中行为失败归因于所使用的训练方法是不准确的。相反，训练方法与建模团队的数据整理指南相结合，决定了请求安全与其他能力之间所需平衡。此外，最终模型的结果相对于训练的初始目标存在差异。随着生态系统的成熟，控制最终模型的能力得到了提高，RLHF和训练后主要关于安全的观念已经减弱，例如通过开发基准来衡量潜在的过度拒绝[304]。

随着基于聊天的AI系统的普及，这些拒绝行为的突出程度随着时间的推移而降低。行业标准已转向更窄的一组伤害和平衡在争议性问题观点上的模型。

18.2 定量过度优化

过度优化也是一个研究领域，研究模型性能与KL优化距离之间的关系[37]。回想一下，KL距离是训练前原始模型（即参考模型）和当前策略的概率之间的距离度量。例如，图20中的关系也可以用x轴上的KL距离而不是训练步骤来表示。下面是一个额外的例子，其中偏好调整数据集被分成两半，以创建一个训练奖励模型（偏好模型，PM，如下）和一个测试奖励模型。在这里，过度训练后，训练RM的改进最终未能转移到测试PM在~150K训练样本[5]。

过优化是RLHF的基本且不可避免的，因为奖励信号（一个学习模型）相对于传统RL文献中旨在完全捕捉世界动态的奖励函数而言是软性的。因此，这是一个RLHF永远无法完全解决的优化问题。

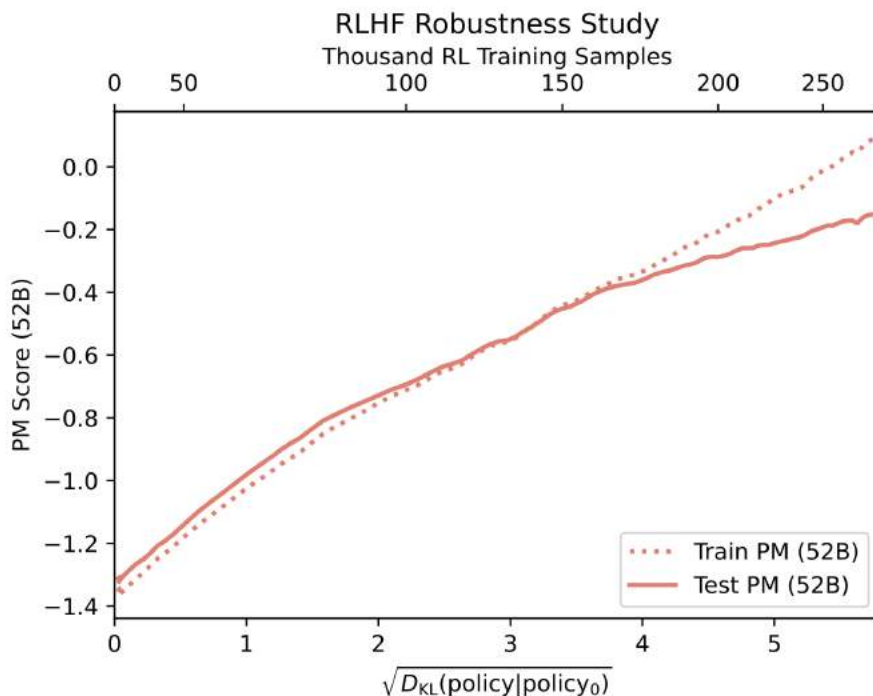


图21：Bai等人2022年的训练和测试RM的过度优化。许可CC-BY。

使用不同的RLHF训练方法，所花费的KL距离将会有所不同。例如，用于修改模型参数的在线RL算法（例如PPO）所使用的KL距离，远高于最佳N采样（BoN）等推理时间采样方法的KL距离。通过RL训练，更高的KL惩罚将减少给定KL距离下的过度优化，但可能需要更多的整体训练步骤才能使模型达到这一点。

许多解决方案可以减轻过度优化。其中一些包括更大的策略模型，这些模型有更多空间调整参数以增加奖励，同时保持较小的KL距离，奖励模型集成[305]，或更改优化器[306]。虽然直接对齐算法仍然容易受到过度优化的影响[307]，但它们优化的直接概念允许使用固定的KL距离，这将使权衡更容易管理。

18.3 不对齐与RLHF的作用

虽然工业RLHF和训练后正在转向涵盖比最初激励RLHF发明的对齐概念更多的目标，但RLHF的未来仍然与对齐紧密相连。在本章的背景下，过度优化将

启用 *misalignment* 的模型。在当前语言模型中，已经有很多研究探讨了如何通过RLHF技术改变模型的行为，以减少它们与人类用户和社会广泛需求的对齐。当前RLHF技术中一个显著的错位例子是研究当前技术如何促进谄媚[82]——模型告诉用户他们想听的内容的倾向。随着语言模型在社会中的更多整合，这种潜在错位的影响将变得更加复杂和深远[308]。随着这些问题的出现，RLHF的对齐目标将与当前实证关注的在风格和性能上收敛于人类偏好相比再次增长。

19 风格和信息

早期RLHF的发展给它带来了“只是风格迁移”或其他对RLHF如何操纵输出中信息呈现方式的严厉批评的声誉。

风格 转移阻碍了RLHF叙事两个原因 ns.

首先，当人们讨论风格迁移时，他们并不将其描述为重要或令人兴奋。风格是人类价值的源源不断的来源，这就是为什么重述故事可以导致新的畅销书（如《人类简史》），并且它是我们继续进步智力生态系统的基本部分。风格与信息的内容交织在一起。

其次，我们已经看到不同的风格实际上如何通过Llama 3 [23]提高评估改进。Llama 3 Instruct模型在ChatBotArena上得分极高，这被认为是由于它们拥有更有趣的个性。如果RLHF能让语言模型仅仅变得更有趣，那么这就是其带来的价值。

在整个本章中，“chattness”一词被用来涵盖使用RLHF训练的模型生成的响应长度的增长，但它还包括像大量使用Markdown、表情符号以及以项目符号格式化答案等技术。

19.1 喋喋不休悖论

RLHF或偏好微调方法主要被用于提升AlpacaE-val和其他自动排行榜的分数，而没有在Chat BotArena等更难操控的评估上成比例地提升。悖论在于，尽管对齐方法在这些模型上带来了可衡量的改进，这些改进也转移到了人们关心的性能上，但大量模型做的大多是同样的事情，却将其做得太过分，发布了明显没有意义的评估分数。

这些方法，如果做得正确，会使模型更容易操作和更有趣。这通常伴随着在MT Bench或AlpacaEval等评估工具上提高几个百分点。问题是，你还可以在反馈循环或大量数据中使用DPO和PPO等技术，实际上严重损害模型在其他任务（如数学或编码）上的表现，以牺牲LLM作为裁判的性能为代价。

在DPO与PPO争论激增期间，出现了许多带有令人难以置信的基准的论文，但没有模型权重被广泛使用。当应用RLHF时，没有方法可以使一个700亿参数模型的版本在全面基准上真正击败GPT-4。这似乎很明显，但有些论文声称这些结果。图22来自一篇名为《直接纳什优化》（DNO）的论文，该论文认为他们的模型在AlpacaEval上是最先进的或类似。当学术激励与对更广泛社会极端感兴趣的技术接口时，这些挑战就会出现。

甚至开创性的论文《自我奖励语言模型》[309]也披露了Llama 2 70B上的不切实际分数。70B模型可以比7B模型更接近GPT-4，正如我们通过Llama 3所看到的，但将模型的现实与现代RLHF论文中的声明区分开来很重要。许多类似的方法已经出现又消失，分享了有价值的见解和夸大的结果，这使得RLHF更难理解。

模型应用“古怪RLHF”后的一个症状通常是长度偏差。

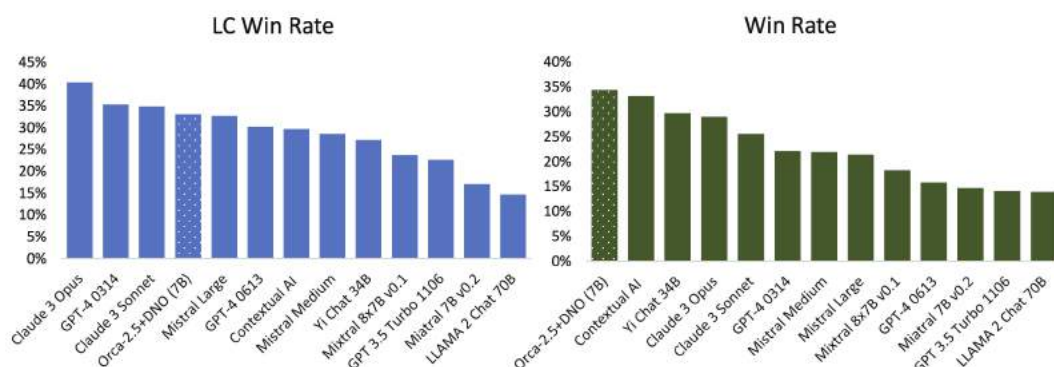


图22：关于直接Nash优化（DNO）的论文结果，突出其小型模型优于GPT-4等。Rosset等人，2024。许可CC-BY。

这已经变得如此普遍，以至于像AlpacaEval和WildBench这样的多个评估系统都包含线性长度校正机制。这修补了为了“打败GPT-4”而过度聊天以作弊的激励措施，并增加了一个不那么游戏化的缺陷，即较短且有用的模型实际上可能获胜。

无论如何，仅仅为了聊天而调整聊天模型在文献中仍然有一定的代价。Qwen模型中的这个注释在早期的对齐实验中多次出现，夸大了聊天能力和性能之间的权衡[310]。

我们使用大量数据进行预训练，并使用监督微调和直接偏好优化进行后训练。然而，DPO导致人类偏好评估改进，但基准评估退化。

这是一个很好的例子，展示了这种权衡得当的做法，就像Starling Beta [94]这样的模型。这是一个从另一个聊天模型OpenChat [311]微调而来的模型，而OpenChat实际上是由另一个组织训练的。它的训练完全集中在k-wise奖励模型训练和PPO优化上，并将其在ChatBotArena中的排名提升了10位。模型的平均响应长度增加了，但增加的方式足以真正帮助人类评分员。

19.1.1 如何产生健谈

一个自然的问题是：为什么RLHF使模型响应更长？从根本上讲，像ChatBotArena这样的评估已经向我们表明，与简洁的回应相比，模型的平均用户通常更喜欢更长、更完整的答案。这并不代表every用户的偏好，但这些模型是训练来匹配许多数据标注者的偏好的。

大多数当今用于对齐的流行数据集都是合成偏好，其中像GPT-4这样的模型将其他模型的输出评为胜者或败者。鉴于GPT-4已知对与其自身匹配的输出具有长度和风格偏差，大多数

数据集中“首选”部分的文本要么来自OpenAI模型，要么在风格上与之相似。重要的区别是数据集中并非所有文本都具有这种特征。它们通常由其他公开模型如Alpaca、Vicuna或更近期的例子生成。这些模型具有非常不同的特性。

接下来，既然我们已经确定我们有一个偏好数据集，其中大多数选定的模型都与ChatGPT（或一些其他被接受为“强大”的模型）相似，这些对齐方法只是增加了这些序列的概率。数学上有些复杂，因为数据批次同时处理许多选定的拒绝对，但在实践中，模型正在对序列中的标记（子词片段）进行信用分配。偏好对聊天的对齐使得在GPT-4等模型输出中找到的序列更有可能，而来自其他较弱模型的序列则不太可能。反复这样做会导致生成更长的模型，并且具有人们更喜欢的特征。

那些熟悉RLHF方法的你们可能会问，在优化过程中KL约束是否应该阻止这种情况发生。KL约束是原始模型分布和结果模型分布之间的距离项。它有助于使优化对过度优化更加鲁棒，但这使得好模型和坏模型之间的边界变得更加微妙。因此，基于感觉的评价变得普遍。尽管如此，模型通常具有足够的参数，它们可以在满足测量数据的KL约束的情况下发生实质性变化——例如，不能是整个预训练数据集。

20 产品、用户体验和模型特性

前沿RLHF和后训练展示了这些技术在公司内部如何用于制造领先产品。随着RLHF的日益成熟，它所解决的问题变得更加复杂。在本章中，我们讨论了一系列领先的AI实验室认为RLHF和后训练很有用，但在学术文献中研究得很少的应用案例。

20.1 字符训练

角色训练是围绕在模型中塑造其响应方式而非内容的特性而设计的后训练子集。虽然角色训练对于语言模型聊天机器人的用户体验很重要，但在公共领域实际上并未得到研究。

我们不知道角色训练的权衡，不知道如何确切地研究它，不知道它能在ChatBotArena上提高用户偏好吗，但我们应该知道。我们所知道的是，角色训练使用本书中讨论的相同方法，但针对模型使用的语言特征有更精确的目标。角色训练涉及广泛的数据过滤和合成数据方法，如宪法AI，这些方法关注模型的行为方式。这些变化通常难以衡量我们在评价章节中提到的所有基准制度，因为AI实验室使用角色训练在一段时间内对个性进行微小调整以改善用户体验。

例如，Anthropic 向其 Claude 3 模型中添加了 Character Training [312]：

Claude 3 是我们首次在模型微调过程中添加“角色训练”的模型：在初始模型训练之后发生的训练部分，以及将其从预测文本模型转变为人工智能助手的部分。角色训练的目的是让 Claude 开始拥有更加细腻、丰富的特质，如好奇心、开放性和周到。

在接下来的几个月里，模型行业的特征变得更加鲜明。这个过程极其依赖合成数据，但需要艺术家的触感，正如博客文章后面所述：“它依赖于人类研究人员密切检查每个特征如何改变模型的行为。”

角色训练成为发展的焦点，这是对RLHF和相关方法从其哲学动机的对齐转向主要成为经验工具的最强烈肯定。模型可以捕捉到许多不同的行为，但让它们可靠地表现出我们想要的行为是最困难的部分。目前来看，这似乎更像是关于捕捉RLHF作为性能工具的优势，而不是安全优势。

Amanda Askeel在Lex Fridman播客节目中的亮相期间，就性格培养进行了少数几次公开讨论之一（摘自转录）：

当你说性格训练时，性格训练中包含了什么？是RLHF还是我们在谈论什么？

阿曼达·阿斯凯尔（03:42:02）它更像是宪法AI，所以它是那种的变体

流水线。我构建了模型应该具备的性格特征。这些特征可以是简短的，也可以是更丰富的描述。然后让模型生成与该特征相关的人类可能给出的查询。然后它生成响应，然后根据性格特征对响应进行排序。以这种方式，在查询生成之后，它与宪法AI非常相似，有一些不同。我非常喜欢它，因为它就像Claude在自己的性格中的训练一样，因为它没有任何……它就像宪法AI，但没有任何人类数据。

总结来说，Anthropic使用他们用于宪法AI和一般训练后能力的相同技术来训练这些模型的性格。

20.2 模型规格

OpenAI最近分享了他们所谓的“模型规范” [91]，这是一份详细说明他们在微调运行前点击“开始”之前的目标模型行为的文档。它关乎模型行为现在，OpenAI如何通过API背后引导他们的模型，以及他们的模型未来将如何转变。

模型规范是行业中少数几个工具之一，在RLHF中，人们可以比较模型的实际行为与设计者的意图。正如我们在本书中所述，训练模型是一个复杂且多方面的过程，因此预期最终结果与输入，如数据标注员指令或训练数据中任务平衡等不同。例如，模型规范比宪法AI中使用的原则列表更有揭示性，因为它涉及过程的意图，而不是列出充当中间训练变量的行为。

一个模型规范为参与模型发布流程的每个利益相关者提供价值：

- 模型设计师：模型设计师可以从中受益，需要明确他们想要和不需要的行为。这使得在数据上的优先级决策更容易，有助于集中精力在可能偏离长期方向的工作上，并使一个人能够在复杂的评估套件中评估他们模型的更大图景。
- 开发者：模型用户对可能是有意为之的行为（即某些类型的拒绝）或训练的副作用有更清晰的了解。这可以让开发者更有信心使用此提供商未来更智能的模型。
- 观察公众：公众从模型规范中受益，因为它是在训练中优先考虑的信息的少数公共来源之一。这对于监管监督和制定关于AI模型应该做什么和不应该做什么的有效政策至关重要。

20.3 产品周期、用户体验和RLHF

随着强大的AI模型越来越接近产品，而不是单个实验机器学习过程的产物，RLHF已成为模型与产品之间关系的一个接口点。要使模型易于使用，不仅仅是最终模型权重正确——还需要快速推理、合适的工具（例如搜索或代码执行）、可靠且易于理解的用户界面（UX）等等。由于RLHF是一种实时理解用户对产品偏好的方法，并且它是

发布前的最终训练阶段。向模型添加新功能最快的方式是在训练后尝试整合，因为此时训练更快且成本更低。这一周期已在图像理解、工具使用、更好的行为等方面得到体现。最初作为产品问题，很快就会变成RLHF建模问题，如果在那里成功，它就会反向传播到其他更早的训练阶段。

参考文献

- [1] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, 和 D. Amodei, “从人类偏好中进行深度强化学习”, *Advances in neural information processing systems*, 第30卷, 2017年。[2] N. Stiennon *et al.*, “通过人类反馈进行总结学习”, *Advances in Neural Information Processing Systems*, 第33卷, 第3008-3021页, 2020年。[3] L. Ouyang *et al.*, “通过人类反馈训练语言模型以遵循指令”, *Advances in neural information processing systems*, 第35卷, 第27730-27744页, 2022年。[4] R. Nakano *et al.*, “Webgpt: 带有人类反馈的浏览器辅助问答”, *arXiv preprint arXiv:2112.09332*, 2021年。[5] Y. Bai *et al.*, “通过人类反馈的强化学习训练一个有用且无害的助手”, *arXiv preprint arXiv:2204.05862*, 2022年。[6] N. Lambert *et al.*, “T\ ULU 3: 在开放语言模型后训练中拓展边界”, *arXiv preprint arXiv:2411.15124*, 2024年。[7] R. Kirk *et al.*, “理解rlhf对llm泛化和多样性的影响”, *arXiv preprint arXiv:2310.06452*, 2023年。[8] T. Chu *et al.*, “Sft记住, rl泛化: 基础模型后训练的比较研究”, *arXiv preprint arXiv:2501.17161*, 2025年。[9] P. Singhal, T. Goyal, J. Xu, 和 G. Durrett, “A long way to go: 调查rlhf中的长度相关性”, *arXiv preprint arXiv:2310.03716*, 2023年。[10] R. Park, R. Rafailov, S. Ermon, 和 C. Finn, “在直接偏好优化中从长度中分离质量”, *arXiv preprint arXiv:2403.19159*, 2024年。[11] 人工智能艾伦研究所, “OLMoE, 遇见iOS。” <https://allenai.org/blog/olmoe-app>, 2025年。[12] C. Zhou *et al.*, “Lima: 对齐的‘少即是多’”, *Advances in Neural Information Processing Systems*, 第36卷, 第55006-55021页, 2023年。[13] R. Taori *et al.*, “斯坦福alpaca: 一个遵循指令的LLaMA模型”, *GitHub repository*. https://github.com/tatsu-lab/stanford_alpaca; GitHub, 2023年。[14] W.-L. Chiang *et al.*, “Vicuna: 一个开源聊天机器人, 以90% * ChatGPT质量令GPT-4印象深刻。” 2023年。可访问: <https://lmsys.org/blog/2023-03-30-vicuna/> [15] X. Geng *et al.*, “Koala: 一个用于学术研究的对话模型。” 博客文章, 2023年。访问时间: 2023年4月3日。[在线]。可访问: <https://bair.berkeley.edu/blog/2023/04/03/koala/> [16] M. Conover *et al.*, “Hello dolly: 用开放模型民主化ChatGPT的魔力。” 访问时间: 2023年6月30日。[在线]。可访问: <https://www.databricks.com/blog/2023/03/24/hello-dolly-democratizing-magic-chatgpt-open-models.html> [17] A. Askell *et al.*, “通用语言助手作为对齐的实验室”, *arXiv preprint arXiv:2112.00861*, 2021年。[18] Y. Bai *et al.*, “宪法AI: 从AI反馈中获得无害性”, *arXiv preprint arXiv:2212.08073*, 2022年。[19] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, 和 C. Finn, “直接偏好优化: 你的语言模型实际上是奖励模型”, *Advances in Neural Information Processing Systems*, 第36卷, 2024年。[20] L. Tunstall *et al.*, “Zephyr: 直接蒸馏LM对齐”, 在 *First conference on language modeling* 中, 2024年。可访问: <https://openreview.net/forum?id=aKkAwZB6JV>

[21] H. Ivison *et al.*, “气候变化中的骆驼: 利用tulu 2增强lm适应性”, *arXiv preprint arXiv:2311.10702*, 2023。[22] G. Cui *et al.*, “超反馈: 利用高质量反馈提升语言模型”, 2023。[23] A. Dubey *et al.*, “模型群3的llama”, *arXiv preprint arXiv:2407.21783*, 2024。[24] B. Adler *et al.*, “Nemotron-4 340B技术报告”, *arXiv preprint arXiv:2406.11704*, 2024。[25] C. Wirth, R. Akrou, G. Neumann和J. Fürnkranz, “基于偏好的强化学习方法综述”, *Journal of Machine Learning Research*, 第18卷, 第136期, 第1-46页, 2017。[26] T. Kaufmann, P. Weng, V. Bengs和E. Hüllermeier, “基于人类反馈的强化学习综述”, *arXiv preprint arXiv:2312.14925*, 2023。[27] S. Casper *et al.*, “基于人类反馈的强化学习的开放问题和基本限制”, *arXiv preprint arXiv:2307.15217*, 2023。[28] W. B. Knox和P. Stone, “通过评估性强化手动训练代理Tamer”, 在 *2008 7th IEEE international conference on development and learning*, IEEE, 2008, 第292-297页。[29] J. MacGlashan *et al.*, “从策略相关的人类反馈中进行交互式学习”, 在 *International conference on machine learning*, PMLR, 2017, 第2285-2294页。[30] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg和D. Amodei, “在atari中从人类偏好和演示中进行奖励学习”, *Advances in neural information processing systems*, 第31卷, 2018。[31] G. Warnell, N. Waytowich, V. Lawhern和P. Stone, “在多维状态空间中进行交互式代理塑造的Deep tamer”, 在 *Proceedings of the AAAI conference on artificial intelligence*, 2018。[32] J. Leike, D. Krueger, T. Everitt, M. Martic, V. Maini和S. Legg, “通过奖励建模实现可扩展的代理对齐: 一个研究方向”, *arXiv preprint arXiv:1811.07871*, 2018。[33] D. M. Ziegler *et al.*, “从人类偏好微调语言模型”, *arXiv preprint arXiv:1909.08593*, 2019。[34] J. Wu *et al.*, “使用人类反馈递归总结书籍”, *arXiv preprint arXiv:2109.10862*, 2021。[35] J. Menick *et al.*, “教授语言模型使用经过验证的引用支持答案”, *arXiv preprint arXiv:2203.11147*, 2022。[36] A. Glaese *et al.*, “通过有针对性的人类判断改进对话代理的对齐”, *arXiv preprint arXiv:2209.14375*, 2022。[37] L. Gao, J. Schulman和J. Hilton, “奖励模型过度优化的缩放定律”, 在 *International conference on machine learning*, PMLR, 2023, 第10835-10866页。[38] D. Ganguli *et al.*, “通过红队语言模型减少伤害: 方法、缩放行为和经验教训”, *arXiv preprint arXiv:2209.07858*, 2022。[39] R. Ramamurthy *et al.*, “强化学习(不是)用于自然语言处理: 基准、基线和自然语言策略优化的构建块”, *arXiv preprint arXiv:2210.01241*, 2022。

[40] A. Havrilla *et al.*, “TrIX: 从人类反馈中进行大规模强化学习的框架”, 载于 *Proceedings of the 2023 conference on empirical methods in natural language processing*, 新加坡: 计算语言学协会, 2023年12月, 第8578-8595页。doi: 10.18653/v1/2023.emnlp-main.530. [41] L. von Werra *et al.*, “TRL: Transformer强化学习”, *GitHub repository*. <https://github.com/huggingface/trl>; GitHub, 2020. [42] OpenAI, “ChatGPT: 优化语言模型以进行对话”。<https://openai.com/blog/chatgpt/>, 2022. [43] H. Touvron *et al.*, “Llama 2: 开放的基金会和微调的聊天模型”, *arXiv preprint arXiv:2307.09288*, 2023. [44] H. Lightman *et al.*, “让我们一步一步验证”, *arXiv preprint arXiv:2305.20050*, 2023. [45] A. Kumar *et al.*, “通过强化学习训练语言模型进行自我纠正”, *arXiv preprint arXiv:2409.12917*, 2024. [46] A. Singh *et al.*, “超越人类数据: 通过语言模型进行问题解决的自我训练扩展”, *arXiv preprint arXiv:2312.06585*, 2023. [47] OpenAI, “介绍OpenAI o1-preview”。2024年9月。可获取: <https://openai.com/index/introducing-openai-o1-preview/> [48] A. Vaswani *et al.*, “注意力即一切”, 载于 *Neural information processing systems*, 2017。可获取: <https://api.semanticscholar.org/CorpusID:13756489> [49] D. Bahdanau, K. Cho, 和 Y. Bengio, “通过联合学习对齐和翻译进行神经机器翻译”, *CoRR*, vol. abs/1409.0473, 2014, 可获取: <https://api.semanticscholar.org/CorpusID:11212020> [50] G. Hinton, O. Vinyals, 和 J. Dean, “从神经网络中蒸馏知识”, *arXiv preprint arXiv:1503.02531*, 2015. [51] G. Team *et al.*, “Gemma 2: 在实用规模上改进开放语言模型”, *arXiv preprint arXiv:2408.00118*, 2024. [52] R. Agarwal *et al.*, “按策略蒸馏语言模型: 从自我生成的错误中学习”, 载于 *The twelfth international conference on learning representations*, 2024. [53] J. Wei *et al.*, “思维链提示在大型语言模型中引发推理”, *Advances in neural information processing systems*, 第35卷, 第24824-24837页, 2022. [54] R. S. Sutton, “强化学习: 入门”, *A Bradford Book*, 2018. [55] N. Lambert, L. Castrejo, L. von Werra, 和 A. Havrilla, “展示从人类反馈中进行强化学习 (RLHF)”, *Hugging Face Blog*, 2022. [56] M. Li *et al.*, “分支-训练-合并: 专家语言模型的尴尬并行训练”, *arXiv preprint arXiv:2208.03306*, 2022. [57] T. Cohere *et al.*, “Command a: 一个企业级的大型语言模型”, *arXiv preprint arXiv:2504.00698*, 2025. [58] T. OLMo *et al.*, “2 OLMo 2愤怒”, *arXiv preprint arXiv:2501.00656*, 2024. [59] S. Alrashed, “SmolTulu: 更高的学习率到批处理大小的比率可以提高SLMs中的推理能力”, *arXiv preprint arXiv:2412.08347*, 2024. [60] D. Guo *et al.*, “Deepseek-r1: 通过强化学习激励llms的推理能力”, *arXiv preprint arXiv:2501.12948*, 2025. [61] A. Yang *et al.*, “Qwen3技术报告”, *arXiv preprint arXiv:2505.09388*, 2025. [62] B. Xia *et al.*, “MiMo: 解锁语言模型推理潜力——从预训练到后训练”, *arXiv preprint arXiv:2505.07608*, 2025。

[63] B. Seed *et al.*, “Seed-thinking-v1.5: 通过强化学习推进卓越推理模型”, *arXiv preprint arXiv:2504.13914*, 2025. [64] N. Lambert, T. K. Gilbert, 和 T. Zick, “纠缠偏好: 强化学习和人类反馈的历史与风险”, *arXiv preprint arXiv:2310.13595*, 2023. [65] V. Conitzer *et al.*, “社会选择应指导AI在处理多样化人类反馈时的对齐”, *arXiv preprint arXiv:2404.10271*, 2024. [66] A. Mishra, “AI对齐和社会选择: 基本局限和政策含义”, *arXiv preprint arXiv:2310.16048*, 2023. [67] H. R. Kirk *et al.*, “PRISM对齐项目: 参与式、代表性以及个性化人类反馈揭示的大语言模型的主观和多文化对齐”, *arXiv preprint arXiv:2404.16019*, 2024. [68] S. Poddar, Y. Wan, H. Ivison, A. Gupta, 和 N. Jaques, “通过变分偏好学习个性化强化学习中的人类反馈”, *arXiv preprint arXiv:2408.10075*, 2024. [69] S. J. Russell 和 P. Norvig, *Artificial intelligence: A modern approach*. Pearson, 2016. [70] B. Widrow 和 M. E. Hoff, “自适应切换电路”, 斯坦福大学加州斯坦福电子实验室, 1960. [71] B. F. Skinner, *The behavior of organisms: An experimental analysis*. BF Skinner基金会, 2019. [72] E. L. Thorndike, “效果定律”, *The American journal of psychology*, 第39卷, 第1/4期, 第212–222页, 1927. [73] A. Arnauld, *The port-royal logic*. 1662. [74] J. Bentham, *An introduction to the principles of morals and legislation*. 1823. [75] F. P. Ramsey, “真理与概率”, *Readings in Formal Epistemology: Sourcebook*, 第21–45页, 2016. [76] K. J. Arrow, “社会福利概念中的困难”, *Journal of political economy*, 第58卷, 第4期, 第328–346页, 1950. [77] J. C. Harsanyi, “规则功利主义与决策理论”, *Erkenntnis*, 第11卷, 第1期, 第25–53页, 1977. [78] R. Pettigrew, *Choosing for changing selves*. 牛津大学出版社, 2019. [79] N. Soares, B. Fallenstein, S. Armstrong, 和 E. Yudkowsky, “可纠正性”, 在 *Work-shops at the twenty-ninth AAAI conference on artificial intelligence*, 2015. [80] A. Kumar, Y. He, A. H. Markosyan, B. Chern, 和 I. Arrieta-Ibarra, “检测基于LLM的奖励模型中的前缀偏差”, *arXiv preprint arXiv:2505.13487*, 2025. [81] A. Bharadwaj, C. Malaviya, N. Joshi, 和 M. Yatskar, “奉承、虚饰和迷雾: 诊断和缓解偏好模型中的个性偏差。” 2025. 可用: <https://arxiv.org/abs/2506.05339>. [82] M. Sharma *et al.*, “理解语言模型中的拍马屁”, 在 *The twelfth international conference on learning representations*, 2024. 可用: <https://openreview.net/forum?id=tvhaxkMKAn>. [83] Y. Bu, L. Huo, Y. Jing, 和 Q. Yang, “超越过剩和不足: RLHF奖励模型中的自适应长度偏差缓解”, 在 *Findings of the association for computational linguistics: NAACL 2025*, 2025, 第3091–3098页. [84] X. Zhang, W. Xiong, L. Chen, T. Zhou, H. Huang, 和 T. Zhang, “从列表到表情符号: 格式偏差如何影响模型对齐”, *arXiv preprint arXiv:2409.11704*, 2024.

[85] W.-L. Chiang *et al.*, “聊天机器人竞技场: 通过人类偏好评估llms的开源平台”, *arXiv preprint arXiv:2403.04132*, 2024. [86] R. Likert, “态度测量的技术”, *Archives of psychology*, 1932. [87] J. Zhou *et al.*, “大型语言模型的指令遵循评估”, *arXiv preprint arXiv:2311.07911*, 2023. [88] K. Ethayarajh, W. Xu, N. Muennighoff, D. Jurafsky, 和D. Kiela, “Kto: 模型对齐作为前景理论优化”, *arXiv preprint arXiv:2402.01306*, 2024. [89] Z. Wu *et al.*, “细粒度人类反馈为语言模型训练提供更好的奖励”, *Advances in Neural Information Processing Systems*, 第36卷, 2024. [90] A. Chen *et al.*, “从自然语言反馈中学习”, *Transactions on Machine Learning Research*, 2024. [91] OpenAI, “介绍模型规范。” 5月2024。可访问: <https://openai.com/index/introducing-the-model-spec/> [92] A. Y. Ng, S. Russell, *et al.*, “逆强化学习的算法。” 在*Proceedings of the seventeenth international conference on machine learning*, 在I CML’ 00. 2000, 第663--670页. [93] R. A. Bradley和M. E. Terry, “不完整区组设计的秩分析: I.配对比较法”, *Biometrika*, 第39卷, 第3/4期, 第324–345页, 1952, 访问日期: 2023年2月13日. [在线]. 可访问: <http://www.jstor.org/stable/2334029> [94] B. Zhu *et al.*, “Stirling-7b: 通过rlaif提高有用性和无害性”, 在*First conference on language modeling*, 2024. [95] A. Liu, Z. Zhao, C. Liao, P. Lu, 和L. Xia, “从部分偏好中学习plackett-luce混合模型”, 在*Proceedings of the AAAI conference on artificial intelligence*, 2019, 第4328–4335页. [96] B. Zhu, M. Jordan, 和J. Jiao, “从成对或k对比较中获取人类反馈的原理性强化学习”, 在*International conference on machine learning*, PMLR, 2023, 第43037–43067页. [97] K. Cobbe *et al.*, “训练验证者解决数学单词问题”, *arXiv preprint arXiv:2110.14168*, 2021. [98] C. Lyu *et al.*, “探索学习数学推理的输出奖励极限”, *arXiv preprint arXiv:2502.06781*, 2025. [99] L. Zheng *et al.*, “使用mt-bench和聊天机器人竞技场评估llm-as-a-judge”, *Advances in Neural Information Processing Systems*, 第36卷, 第46595–46623页, 2023. [100] Y. Dubois, B. Galambosi, P. Liang, 和T. B. Hashimoto, “长度控制的alpaca-eval: 消除自动评估器的偏差的简单方法”, *arXiv preprint arXiv:2404.04475*, 2024. [101] T. Li *et al.*, “从众包数据到高质量基准: Arena-hard和BenchBuilder管道”, *arXiv preprint arXiv:2406.11939*, 2024. [102] B. Y. Lin *et al.*, “WILDBENCH: 使用真实用户在野外的挑战性任务对LLM进行基准测试”, *arXiv preprint arXiv:2406.04770*, 2024. [103] D. Mahan *et al.*, “生成式奖励模型”, 2024, 可访问: https://www.synthlabs.ai/pdf/Generative_Reward_Models.pdf [104] L. Zhang, A. Hossain, H. Bansal, M. Kazemi, A. Kumar, 和R. Agarwal, “生成式验证者: 奖励建模作为下一个标记预测”, *arXiv preprint arXiv:2408.15240*, 2024.

[105] Z. Ankner, M. Paul, B. Cui, J. D. Chang, 和 P. Ammanabrolu, “大声批评奖励模型”, *arXiv preprint arXiv:2408.11791*, 2024. [106] S. Kim *et al.*, “Prometheus: 在语言模型中诱导细粒度评估能力”, 在 *The twelfth international conference on learning representations*, 2023. [107] N. Lambert *et al.*, “Rewardbench: 评估语言模型奖励模型”, *arXiv preprint arXiv:2403.13787*, 2024. [108] X. Wen *et al.*, “重新思考奖励模型评估: 我们是不是在错的方向上吠叫?”, *arXiv preprint arXiv:2410.05584*, 2024. [109] S. Gureja *et al.*, “M-RewardBench: 在多语言环境中评估奖励模型”, *arXiv preprint arXiv:2410.15522*, 2024. [110] Z. Jin *et al.*, “R-AG-RewardBench: 在检索增强生成中基准测试偏好对齐的奖励模型”, *arXiv preprint arXiv:2412.13746*, 2024. [111] E. Zhou *et al.*, “RMB: 在LLM对齐中全面基准测试奖励模型”, *arXiv preprint arXiv:2410.09893*, 2024. [112] Y. Liu, Z. Yao, R. Min, Y. Cao, L. Hou, 和 J. Li, “RM-bench: 基准测试具有细微差别和风格的语言模型奖励模型”, *arXiv preprint arXiv:2410.16184*, 2024. [113] Z. Wu, M. Yasunaga, A. Cohen, Y. Kim, A. Celikyilmaz, 和 M. Ghazvininejad, “reWordBench: 通过变换输入基准测试和改进奖励模型的鲁棒性”, *arXiv preprint arXiv:2503.11751*, 2025. [114] Z. Chen *et al.*, “MJ-bench: 你的多模态奖励模型真的是文本到图像生成的良好评判者吗?”, *arXiv preprint arXiv:2407.04842*, 2024. [115] M. Yasunaga, L. Zettlemoyer, 和 M. Ghazvininejad, “多模态rewardbench: 视觉语言模型奖励模型的全面评估”, *arXiv preprint arXiv:2502.14191*, 2025. [116] L. Li *et al.*, “VLRewardBench: 视觉语言生成奖励模型的挑战性基准”, *arXiv preprint arXiv:2411.17451*, 2024. [117] J. Ruan *et al.*, “Vlrmbench: 视觉语言奖励模型的全面和挑战性基准”, *arXiv preprint arXiv:2503.07478*, 2025. [118] E. Frick *et al.*, “如何评估RLHF的奖励模型”, *arXiv preprint arXiv:2410.14872*, 2024. [119] S. Kim *et al.*, “评估数学推理奖励模型的鲁棒性”, *arXiv preprint arXiv:2410.01729*, 2024. [120] M. Song, Z. Su, X. Qu, J. Zhou, 和 Y. Cheng, “PRMBench: 过程级奖励模型的细粒度和挑战性基准”, *arXiv preprint arXiv:2501.03124*, 2025. [121] W. Wang *et al.*, “VisualPRM: 多模态推理的有效过程奖励模型”, *arXiv preprint arXiv:2503.10291*, 2025. [122] H. Tu, W. Feng, H. Chen, H. Liu, X. Tang, 和 C. Xie, “ViLBench: 视觉语言过程奖励建模套件。” 3月2025。可获取: <https://arxiv.org/abs/2503.20271> [123] H. Wang, W. Xiong, T. Xie, H. Zhao, 和 T. Zhang, “通过多目标奖励建模和专家混合实现可解释的偏好”, *arXiv preprint arXiv:2406.12845*, 2024. [124] Z. Wang *et al.*, “HelpSteer2: 训练顶级奖励模型的开源数据集”, *arXiv preprint arXiv:2406.08673*, 2024.

[125] 王志伟 *et al.*, “HelpSteer2-preference: 用偏好补充评分”, *arXiv preprint arXiv:2410.01257*, 2024. [126] 帕克J, Jwa S, 任明, 金德, 崔世浩, “Off setbias: 利用去偏数据调整评估器”, *arXiv preprint arXiv:2407.06551*, 2024. [127] Jaques N, Gu S, Bahdanau D, Hernández-Lobato JM, Turner RE, Eck D, “序列导师: 使用kl-control对序列生成模型进行保守微调”, 在 *International conference on machine learning*, P MLR, 2017, 第1645–1654页. [128] Jaques N *et al.*, “通过离线强化学习进行以人为中心的对话训练”, *arXiv preprint arXiv:2010.05848*, 2020. [129] Schulman J, “近似KL散度。” <http://joschu.net/blog/kl-approx.html>, 2016. [130] Pang RY, 袁伟, Cho K, 何海, Sukhbaatar S, Weston J, “迭代推理偏好优化”, *arXiv preprint arXiv:2404.19733*, 2024. [131] 高志 *et al.*, “Rebel: 通过回归相对奖励进行强化学习”, *arXiv preprint arXiv:2404.16767*, 2024. [132] Brown T *et al.*, “语言模型是少样本学习者”, *Advances in neural information processing systems*, 第33卷, 第1877–1901页, 2020. [133] Raffel C *et al.*, “探索统一文本到文本转换器在迁移学习中的极限”, *Journal of machine learning research*, 第21卷, 第140期, 第1–67页, 2020. [134] Wei J *et al.*, “微调语言模型是零样本学习者”, 在 *International conference on learning representations*, 2022. 可用: <https://openreview.net/forum?id=gEZrGCozdqR>. [135] Sanh V *et al.*, “多任务提示训练实现零样本任务泛化”, 在 *International conference on learning representations*, 2022. 可用: <https://openreview.net/forum?id=9Vrb9D0WI4>. [136] Mishra S, Khashabi D, Baral C, Hajishirzi H, “通过自然语言众包指令实现跨任务泛化”, 在 *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: Long papers)*, 计算语言学协会, 2022年5月, 第3470–3487页. doi: 10.18653/v1/2022.acl-long.244. [137] Wallace E, Xiao K, Leike R, Weng L, Heidecke J, Beutel A, “指令层次结构: 训练llm优先考虑特权指令”, *arXiv preprint arXiv:2404.13208*, 2024. [138] Dettmers T, Pagnoni A, Holtzman A, Zettlemoyer L, “Qlora: 量化llm的高效微调”, *Advances in neural information processing systems*, 第36卷, 第10088–10115页, 2023. [139] Rajani N, Tunstall L, Beeching E, Lambert N, Rush AM, Wolf T, “没有机器人”, *Hugging Face repository*. https://huggingface.co/datasets/HuggingFaceH4/no_robots; Hugging Face, 2023. [140] Gilks WR, Wild P, “自适应拒绝抽样用于吉布斯抽样”, *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 第41卷, 第2期, 第337–348页, 1992. [141] Ahmadian A *et al.*, “回到基础: 重新审视llm中从人类反馈中学习的强化风格优化”, *arXiv preprint arXiv:2402.14740*, 2024.

[142] J. Schulman, P. Moritz, S. Levine, M. Jordan, 和 P. Abbeel, “使用广义优势估计进行高维连续控制”, 在 *Proceedings of the international conference on learning representations (ICLR)*, 2016。[143] R. J. Williams, “用于连接主义强化学习的简单统计梯度跟踪算法”, *Machine learning*, 第 8 卷, 第 229–256 页, 1992。[144] S. C. Huang, A. Ahmadian, 和 C. F. AI, “将 RL 放回 RLHF。” https://huggingface.co/blog/putting_rl_back_in_rlhf_with_rloo, 2024。[145] W. Kool, H. van Hoof, 和 M. Welling, “买 4 个强化样本, 免费获得一个基线!” 2019。[146] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, 和 O. Klimov, “近端策略优化算法”, *arXiv preprint arXiv:1707.06347*, 2017。[147] C. Berner *et al.*, “使用大规模深度强化学习玩 Dota 2”, *arXiv preprint arXiv:1912.06680*, 2019。[148] Z. Liu *et al.*, “理解 R1-zero-like 训练: 一个批判性视角”, *arXiv preprint arXiv:2503.20783*, 2025 年 3 月, 可用: <https://arxiv.org/abs/2503.20783> [149] J. Nocedal 和 S. J. Wright, *Numerical optimization*. Springer, 2006。[150] J. Schulman, S. Levine, P. Abbeel, M. Jordan, 和 P. Moritz, “信任域策略优化”, 在 *International conference on machine learning*, PMLR, 2015, 第 1889–1897 页。[151] Z. Shao *et al.*, “Deepseekmath: 推动开放语言模型中数学推理的极限”, *arXiv preprint arXiv:2402.03300*, 2024。[152] A. Liu *et al.*, “Deepseek-v3 技术报告”, *arXiv preprint arXiv:2412.19437*, 2024。[153] H. Ivison *et al.*, “拆解 DPO 和 PPO: 从偏好反馈中学习的最佳实践”, *arXiv preprint arXiv:2406.09279*, 2024。[154] S. Huang, M. Noukhovitch, A. Hosseini, K. Rasul, W. Wang, 和 L. Tunstall, “使用 PPO 的 RLHF 的 n+ 实现细节: TL;DR 摘要的一个案例研究”, 在 *First conference on language modeling*, 2024。可用: <https://openreview.net/forum?id=kHO2ZTa8e3> [155] L. Weng, “策略梯度算法”, *lilianweng.github.io*, 2018, 可用: <https://lilianweng.github.io/posts/2018-04-08-policy-gradient/> [156] A. Baheti, X. Lu, F. Brahman, R. L. Bras, M. Sap, 和 M. Riedl, “剩余午餐: 基于优势的离线强化学习语言模型”, *arXiv preprint arXiv:2305.14718*, 2023。[157] Q. Yu *et al.*, “DAPO: 一个可扩展的开源 LLM 强化学习系统。” 2025。[158] M. Noukhovitch, S. Huang, S. Xhonneux, A. Hosseini, R. Agarwal, 和 A. Courville, “异步 RLHF: 更快速、更高效的离策略 RL 语言模型”, *arXiv preprint arXiv:2410.18252*, 2024。[159] B. Wu *et al.*, “LlamaRL: 一个用于高效大规模 LLM 训练的分布式异步强化学习框架”, *arXiv preprint arXiv:2505.24034*, 2025。[160] W. Fu *et al.*, “AReaL: 一个用于语言推理的大规模异步强化学习系统”, *arXiv preprint arXiv:2505.24298*, 2025。[161] P. I. Team *et al.*, “INTELLECT-2: 通过全球分布式强化学习训练的推理模型。” 2025。可用: <https://arxiv.org/abs/2505.07291>

[162] N. L. Roux *et al.*, “渐减策略REINFORCE: 适用于LLM的稳定且高效的强化学习”, *arXiv preprint arXiv:2503.14286*, 2025. [163] D. Seita, “关于广义优势估计论文的笔记。” 2017. 可访问: <https://danieltakeshi.github.io/2017/04/02/notes-on-the-generalized-advantage-estimation-paper/> [164] T. Wu, B. Zhu, R. Zhang, Z. Wen, K. Ramchandran, 和J. Jiao, “成对近端策略优化: 利用相对反馈进行LLM对齐”, *arXiv preprint arXiv:2310.00212*, 2023. [165] Y. Flet-Berliac *et al.*, “对比策略梯度: 以监督友好的方式在序列级别分数上对齐LLM”, *arXiv preprint arXiv:2406.19185*, 2024. [166] Z. Li *et al.*, “Remax: 一种简单、有效且高效的强化学习方法, 用于对齐大型语言模型”, 在 *Forty-first international conference on machine learning*, 2023. [167] T. Gunter *et al.*, “苹果智能基础语言模型”, *arXiv preprint arXiv:2407.21075*, 2024. [168] K. Team *et al.*, “Kimi k1. 5: 使用LLM扩展强化学习”, *arXiv preprint arXiv:2501.12599*, 2025. [169] M. T. Omar, L. Shani, Y. Efroni, 和M. Ghavamzadeh, “镜像下降策略优化”, *arXiv preprint arXiv:2005.09814*, 2020. [170] Y. Zhang *et al.*, “通过乐观在线镜像下降改进LLM的一般偏好对齐”, *arXiv preprint arXiv:2502.16852*, 2025. [171] Y. Yuan *et al.*, “VAPO: 用于高级推理任务的效率高且可靠的强化学习”, *arXiv preprint arXiv:2504.05118*, 2025. [172] Y. Yuan, Y. Yue, R. Zhu, T. Fan, 和L. Yan, “长CoT中PPO崩溃背后的原因是什么? 价值优化掌握秘密”, *arXiv preprint arXiv:2503.01491*, 2025. [173] Y. Zhao, R. Joshi, T. Liu, M. Khalman, M. Saleh, 和P. J. Liu, “Slic-hf: 带有人类反馈的序列似然校准”, *arXiv preprint arXiv:2305.10425*, 2023. [174] M. G. Azar *et al.*, “理解从人类偏好中学习的一般理论范式”, 在 *International conference on artificial intelligence and statistics*, PMLR, 2024, 第4447-4455页. [175] A. Amini, T. Vieira, 和R. Cotterell, “带有偏移的直接偏好优化”, *arXiv preprint arXiv:2402.10571*, 2024. [176] J. Hong, N. Lee, 和J. Thorne, “无参考单块偏好优化与概率比”, *arXiv e-prints*, 第arXiv-2403, 2024. [177] Y. Meng, M. Xia, 和D. Chen, “Simpo: 无参考奖励的简单偏好优化”, *Advances in Neural Information Processing Systems*, 第37卷, 第124198-124235页, 2025. [178] N. Razin, S. Malladi, A. Bhaskar, D. Chen, S. Arora, 和B. Hanin, “无意的不对齐: 直接偏好优化中的似然位移”, *arXiv preprint arXiv:2410.08847*, 2024. [179] Y. Ren 和D. J. Sutherland, “LLM微调的学习动态”, *arXiv preprint arXiv:2407.10490*, 2024. [180] T. Xiao, Y. Yuan, H. Zhu, M. Li, 和V. G. Honavar, “Cal-dpo: 用于语言模型对齐的校准直接偏好优化”, *arXiv preprint arXiv:2412.14516*, 2024.

[181] A. Gupta *et al.*, “AlphaPO-奖励形状对LLM对齐的重要性”, *arXiv preprint arXiv:2501.03884*, 2025. [182] S. Guo *et al.*, “从在线AI反馈中进行直接语言模型对齐”, *arXiv preprint arXiv:2402.04792*, 2024. [183] P. Singhal, N. Lambert, S. Niekum, T. Goyal和G. Durrett, “D2po: 具有响应评估模型的判别器引导DPO”, *arXiv preprint arXiv:2405.01511*, 2024. [184] C. Rosset, C.-A. Cheng, A. Mitra, M. Santacrose, A. Awadallah和T. Xie, “直接Nash优化: 用一般偏好教语言模型自我改进”, *arXiv preprint arXiv:2404.03715*, 2024. [185] S. Jung, G. Han, D. W. Nam和K.-W. On, “二进制分类器优化用于大型语言模型对齐”, *arXiv preprint arXiv:2404.04656*, 2024. [186] H. Zhao *et al.*, “Rainbowpo: 结合偏好优化改进的统一框架”, *arXiv preprint arXiv:2410.04203*, 2024. [187] A. Gorbatskiy, B. Shaposhnikov, V. Sinii, A. Malakhov和D. Gavrilov, “直接对齐算法之间的差异模糊不清”, *arXiv preprint arXiv:2502.01237*, 2025. [188] S. Xu *et al.*, “DPO是否优于PPO进行LLM对齐? 一项全面研究”, *arXiv preprint arXiv:2404.10719*, 2024. [189] F. Tajwar *et al.*, “LLM的偏好微调应利用次优、策略数据”, *arXiv preprint arXiv:2404.14367*, 2024. [190] H. Lee *et al.*, “Rlaif: 利用AI反馈扩展人类反馈中的强化学习”, 2023. [191] A. Sharma, S. Keh, E. Mitchell, C. Finn, K. Arora和T. Kollar, “对AI反馈对齐大型语言模型的批判性评估。” 2024。可用: <https://arxiv.org/abs/2402.12366> [192] L. Castricato, N. Lile, S. Anand, H. Schoelkopf, S. Verma和S. Biderman, “用直接原则反馈抑制粉红色大象。” 2024。可用: <https://arxiv.org/abs/2402.07896> [193] L. J. V. Miranda *et al.*, “混合偏好: 学习为人类和AI反馈路由实例”, *arXiv preprint arXiv:2410.19133*, 2024. [194] P. Wang *et al.*, “大型语言模型不是公平的评估者”, *arXiv preprint arXiv:2305.17926*, 2023. [195] A. Panickssery, S. Bowman和S. Feng, “LLM评估者识别并偏爱自己的生成”, *Advances in Neural Information Processing Systems*, 2024. [196] T. Wang *et al.*, “Shepherd: 语言模型生成的评论家”, *arXiv preprint arXiv:2308.04592*, 2023. [197] P. Ke *et al.*, “CritiqueLLM: 朝着为大型语言模型生成评估的具有信息量的评论生成模型”, *arXiv preprint arXiv:2311.18702*, 2023. [198] J. Li, S. Sun, W. Yuan, R.-Z. Fan, H. Zhao和P. Liu, “用于评估对齐的生成式法官”, *arXiv preprint arXiv:2310.05470*, 2023. [199] S. Kim *et al.*, “Prometheus 2: 一个专门用于评估其他语言模型的开源语言模型”, *arXiv preprint arXiv:2405.01535*, 2024. [200] S. Lee, S. Kim, S. Park, G. Kim和M. Seo, “Prometheus-vision: 视觉-语言模型作为细粒度评估的法官”, 在 *Findings of the association for computational linguistics ACL 2024*, 2024, 第11286–11315页。

[201] B. Brown *et al.*, “大型语言猴子：通过重复采样扩展推理计算”, *arXiv preprint arXiv:2407.21787*, 2024. [202] E. Zhao, P. Awasthi和S. Gollapudi, “样本、审查和扩展：通过扩展验证的有效推理时间搜索”, *arXiv preprint arXiv:2502.01839*, 2025. [203] N. Kalra和L. Tang, “Verdict: 一个用于扩展判断时间计算的库”, *arXiv preprint arXiv:2502.18018*, 2025. [204] A. Madaan *et al.*, “自我改进：带有自我反馈的迭代改进”, *Advances in Neural Information Processing Systems*, 2023. [205] A. Pace, J. M. allinson, E. Malmi, S. Krause和A. Severyn, “West-of-n: 用于改进奖励建模的合成偏好生成”, *arXiv preprint arXiv:2401.12086*, 2024. [206] T. Wu *et al.*, “元奖励语言模型：通过llm-as-a-meta-judge进行自我改进的对齐”, *arXiv preprint arXiv:2407.19594*, 2024. [207] M. Y. Guan *et al.*, “审议对齐：推理使语言模型更安全”, *arXiv preprint arXiv:2412.16339*, 2024. [208] Anthropic, “Claude的宪法。” 访问时间：2024年2月7日。[在线]。可获得：<https://www.anthropic.com/news/claudes-constitution> [209] D. Ganguli *et al.*, “集体宪法AI：与公众输入对齐语言模型。” Anthropic, 2023. [210] S. Huang *et al.*, “宪法AI配方”, *Hugging Face Blog*, 2024. [211] N. Lambert, H. Schoelkopf, A. Gokaslan, L. Soldaini, V. Pyatkin和L. Castricato, “自我引导合成对话和修订技术报告”, *arXiv preprint arXiv:2407.18421*, 2024. [212] Z. Sun *et al.*, “从零开始以原则驱动的方式自我对齐语言模型，最小化人工监督”，在 *Thirty-seventh conference on neural information processing systems*, 2023。可获得：<https://openreview.net/forum?id=p40XRfBX96> [213] Z. Sun *et al.*, “SALMON：遵循原则的奖励模型的自我对齐”，在 *The twelfth international conference on learning representations*, 2024。可获得：<https://openreview.net/forum?id=xJbsmB8UMx> [214] Z. Liu *et al.*, “通用奖励建模的推理时间扩展”, *arXiv preprint arXiv:2504.02495*, 2025. [215] J.-P. Fränken, E. Zelikman, R. Rafailov, K. Gandhi, T. Gerstenberg和N. Goodman, “通过互信息进行自监督对齐：在没有偏好标签的情况下学习遵循原则”，*Advances in Neural Information Processing Systems*, 2024. [216] A. Irpan, “深度强化学习还没有工作。” 2018。可获得：<https://www.alexirpan.com/2018/02/14/rl-hard.html> [217] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup和D. Meger, “有意义的深度强化学习”，在 *Proceedings of the AAAI conference on artificial intelligence*, 2018。可获得：<http://ojs.aaai.org/index.php/AAAI/article/view/11694> [218] G. Sheng *et al.*, “HybridFlow：一个灵活且高效的RLHF框架”，*arXiv preprint arXiv: 2409.19256*, 2024. [219] J. Hu *et al.*, “OpenRLHF：一个易于使用、可扩展且高性能的RLHF框架”，*arXiv preprint arXiv:2405.11143*, 2024.

[220] 刘杰, Cohen A, Pasunuru R, Choi Y, Hajishirzi H, Celikyilmaz A, “不要丢弃你的价值模型！使用价值引导的蒙特卡洛树搜索解码生成更受欢迎的文本”, *arXiv preprint arXiv:2309.15028*, 2023. [221] Muennighoff N, “s1: 简单的测试时间缩放”, *arXiv preprint arXiv:2501.19393*, 2025. [222] 陈磊, “更多的llm调用就足够了吗？复合推理系统缩放定律的研究”, *arXiv preprint arXiv:2403.02419*, 2024. [223] Zelikman E, Wu Y, Mu J, Goodman N, “STaR: 通过推理进行推理引导的引导”, 在 *Advances in neural information processing systems*, Oh A H, Agarwal A, Belgrave D, Cho K, 编, 2022. 可获取: https://openreview.net/forum?id=_3ELRdg2sgI. [224] Zelikman E, Harik G, Shao Y, Jayasiri V, Haber N, Goodman N D, “Quiet- STaR: 语言模型可以在说话之前学会思考”, *COLM*, vol. abs/2403.09629, 2024. [225] Hoffman M D, “通过潜在变量推理训练思维链”, 在 *Thirty-seventh conference on neural information processing systems*, 2023. 可获取: <https://openreview.net/forum?id=a147pIS2Co>. [226] Kazemnejad A, “VinePPO: 通过精细的信用分配解锁LLM推理的RL潜力。” 2024. 可获取: <https://arxiv.org/abs/2410.01679>. [227] Gehring J, Zheng K, Copet J, Mella V, Cohen T, Synnaeve G, “RLEF: 使用强化学习将代码LLM扎根于执行反馈中。” 2024. 可获取: <https://arxiv.org/abs/2410.02089>. [228] Xu S, “DPO是否优于PPO进行LLM对齐？一项全面研究”, 在 *ICML*, 2024. 可获取: <https://openreview.net/forum?id=6XH8R7YrSk>. [229] Amit N, Goldwasser S, Paradise O, Rothblum G, “证明自己正确性的模型”, *arXiv preprint arXiv:2405.15722*, 2024. [230] Hu J, Zhang Y, Han Q, Jiang D, Zhang X, Shum H, “Open-reasoner-zero: 提高基础模型上强化学习的开源方法”, *arXiv preprint arXiv:2503.24290*, 2025. [231] Abdin M, Agarwal S, Awadallah A, “Phi-4-reasoning技术报告”, *arXiv preprint arXiv:2504.21318*, 2025. [232] Bercovich A, Levy I, Golan I, “Llama-nemotron: 高效的推理模型”, *arXiv preprint arXiv:2505.00949*, 2025. [233] Liu A, Zhou B, Xu C, “Hunyuan-TurboS: 通过mamba-transformer协同和自适应思维链推进大型语言模型”, *arXiv preprint arXiv:2505.15431*, 2025. [234] He J, Liu J, Liu C Y, “Skywork open reasoner 1技术报告”, *arXiv preprint arXiv:2505.22312*, 2025. [235] Team C, “MiMo-VL技术报告。” 2025. 可获取: <https://arxiv.org/abs/2506.03569>. [236] Guha E, Marten R, Keh S, “OpenThoughts: 推理模型的数据配方”, *arXiv preprint arXiv:2506.04178*, 2025. [237] Mistral AI, “Magistral: 扩展大型语言模型中推理的强化学习”, Mistral AI, 2025. 可获取: <https://mistral.ai/static/research/magistral.pdf>. [238] Wang Z, “Ragen: 通过多轮强化学习理解LLM代理的自我进化”, *arXiv preprint arXiv:2504.20073*, 2025.

[239] 沙鸥 *et al.*, “虚假奖励: 重新思考RLVR中的训练信号。” <https://rethink-rlvr.notion.site/Spurious-Rewards-Rethinking-Training-Signals-in-RLVR-1f4df34dac1880948858f95aeb88872f>, 2025. [240] Anthropic, “Claude 4.” 五月2025。可访问: <https://www.anthropic.com/news/claude-4> [241] P. Aggarwal和S. Welleck, “L1: 使用强化学习控制推理模型思考的时间”, *arXiv preprint arXiv:2503.04697*, 2025. [242] S. Reed和N. De Freitas, “神经程序员解释器”, *arXiv preprint arXiv:1511.06279*, 2015. [243] P. Lewis *et al.*, “用于知识密集型NLP任务的检索增强生成”, *Advances in neural information processing systems*, 第33卷, 第9459-9474页, 2020. [244] L. Gao *et al.*, “Pal: 程序辅助语言模型”, 在 *International conference on machine learning*中, PMLR, 2023, 第10764-10799页. [245] A. Parisi, Y. Zhao和N. Fiedel, “Talm: 工具辅助语言模型”, *arXiv preprint arXiv:2205.12255*, 2022. [246] T. Schick *et al.*, “Toolformer: 语言模型可以教会自己使用工具。” 2023。可访问: <https://arxiv.org/abs/2302.04761> [247] S. G. Patil, T. Zhang, X. Wang和J. E. Gonzalez, “Gorilla: 与大量API相连的大型语言模型”, *arXiv preprint arXiv:2305.15334*, 2023. [248] Anthropic, “模型上下文协议 (MCP)。” <https://modelcontextprotocol.io/>, 2024. [249] A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White和P. Schwaller, “Chemcrow: 使用化学工具增强大型语言模型”, *arXiv preprint arXiv:2304.05376*, 2023. [250] B. Li *et al.*, “Mmedagent: 使用多模态智能体学习使用医疗工具”, *arXiv preprint arXiv:2407.02483*, 2024. [251] K. Zhang, J. Li, G. Li, X. Shi和Z. Jin, “Codeagent: 通过工具集成智能体系统增强代码生成, 以应对现实世界的repo级编码挑战”, *arXiv preprint arXiv:2401.07339*, 2024. [252] S. Yao *et al.*, “React: 在语言模型中协同推理和行动”, 在 *International conference on learning representations (ICLR)*中, 2023. [253] Z. Wang *et al.*, “RAGEN: 通过多轮强化学习理解LLM智能体的自我进化。” 2025。可访问: <https://arxiv.org/abs/2504.20073> [254] W. Kwon *et al.*, “使用PagedAttention对大型语言模型服务进行高效内存管理”, 在 *Proceedings of the ACM SIGOPS 29th symposium on operating systems principles*中, 2023. [255] I. Shumailov, Z. Shumaylov, Y. Zhao, N. Papernot, R. Anderson和Y. Gal, “当在递归生成数据上训练时, AI模型会崩溃”, *Nature*, 第631卷, 第8022号, 第755-759页, 2024. [256] M. Gerstgrasser *et al.*, “模型崩溃不可避免吗? 通过积累真实和合成数据打破递归的诅咒”, *arXiv preprint arXiv:2404.01413*, 2024. [257] Y. Feng, E. Dohmatob, P. Yang, F. Charton和J. Kempe, “超越模型崩溃: 使用合成数据扩展需要强化”, 在 *ICML 2024 workshop on theoretical foundations of foundation models*中, 2024. [258] Y. Wang *et al.*, “Self-instruct: 将语言模型与自生成的指令对齐”, *arXiv preprint arXiv:2212.10560*, 2022.

[259] E. Beeching *et al.*, “NuminaMath 7B TIR”, *Hugging Face repository*. <https://huggingface.co/AI-MO/NuminaMath-7B-TIR>; Numina & Hugging Face, 2024。[260] M. Li *et al.*, “Superfiltering: 弱到强数据过滤以快速指令调整”, *arXiv preprint arXiv:2402.00530*, 2024。[261] K. Shridhar, A. Stolfo, 和 M. Sachan, “将推理能力蒸馏到更小的语言模型中”, *Findings of the Association for Computational Linguistics: ACL 2023*, 第 7059–7073 页, 2023。[262] C.-Y. Hsieh *et al.*, “逐步蒸馏! 使用更少的训练数据和更小的模型尺寸超越更大的语言模型”, *arXiv preprint arXiv:2305.02301*, 2023。[263] D. Hendrycks *et al.*, “衡量大规模多任务语言理解”, *arXiv preprint arXiv:2009.03300*, 2020。[264] A. Mallen, A. Asai, V. Zhong, R. Das, H. Hajishirzi, 和 D. Khashabi, “何时不相信语言模型: 调查参数和非参数记忆的有效性和局限性”, *arXiv preprint*, 2022。[265] S. Lin, J. Hilton, 和 O. Evans, “Truthfulqa: 衡量模型模仿人类虚假陈述的程度”, *arXiv preprint arXiv:2109.07958*, 2021。[266] M. Suzgun *et al.*, “挑战 BIG-bench 任务以及思维链是否可以解决它们”, *arXiv preprint arXiv:2210.09261*, 2022。[267] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, 和 M. Gardner, “DROP: 需要段落离散推理的阅读理解基准”, *arXiv preprint arXiv:1903.00161*, 2019。[268] D. Hendrycks *et al.*, “使用 MATH 数据集衡量数学问题解决”, *NeurIPS*, 2021。[269] K. Cobbe *et al.*, “训练验证器解决数学文字问题”, *arXiv preprint arXiv:2110.14168*, 2021。[270] M. Chen *et al.*, “评估在代码上训练的大型语言模型”, 2021, 可用: <https://arxiv.org/abs/2107.03374>。[271] J. Liu, C. S. Xia, Y. Wang, 和 L. Zhang, “你的代码是由 chatGPT 生成的吗? 对代码生成的大型语言模型进行严格评估”, 在 *Thirty-seventh conference on neural information processing systems* 中, 2023。可用: <https://openreview.net/forum?id=1qvx610Cu7>。[272] J. Zhou *et al.*, “大型语言模型的指令遵循评估。” 2023。可用: <https://arxiv.org/abs/2311.07911>。[273] D. Rein *et al.*, “GPQA: 一个研究生级别的谷歌证明问答基准”, *arXiv preprint arXiv:2311.12022*, 2023。[274] L. Phan, A. Gatti, Z. Han, N. Li, 和 H. et al. Zhang, “人类最后的考试”, *arXiv preprint arXiv:2501.14249*, 2025。[275] R. Aleithan, H. Xue, M. M. Mohajer, E. Nnorom, G. Uddin, 和 S. Wang, “SWE-Bench+: LLM 的增强编码基准”, *arXiv preprint arXiv:2410.06992*, 2024。[276] N. Jain *et al.*, “LiveCodeBench: 对大型语言模型代码的全面和不受污染的评估”, *arXiv preprint arXiv:2403.07974*, 2024。[277] S. AI, “SEAL LLM 排行榜: 专家驱动的私人评估。” 2024。可用: <https://scale.com/leaderboard>

[278] S. Schulhoff *et al.*, “提示报告：提示技术的系统调查”, *arXiv preprint arXiv:2406.06608*, 2024. [279] J. Robinson, C. M. Rytting, 和 D. Wingate, “利用大型语言模型进行多选题问答”, 在 *International conference on learning representations*, 2023. 可用: <https://openreview.net/forum?id=upQ4o-ygvJ>. [280] J. Wei *et al.*, “微调语言模型是零样本学习者”, 在 *International conference on learning representations*, 2022. [281] V. Sanh *et al.*, “多任务提示训练实现零样本任务泛化”, 在 *International conference on learning representations*, 2022. [282] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, 和 Y. Iwasawa, “大型语言模型是零样本推理者”, *Advances in neural information processing systems*, 第 35 卷, 第 22199–22213 页, 2022. [283] J. Achiam *et al.*, “Gpt-4 技术报告”, *arXiv preprint arXiv:2303.08774*, 2023. [284] OpenAI, “介绍 SWE-bench 验证。” 2024 年 8 月。可用: <https://openai.com/index/introducing-swe-bench-verified/> [285] J. Li *et al.*, “Numinamath: ai4maths 中最大的公共数据集, 包含 860k 对竞赛数学问题和解答”, *Hugging Face repository*, 第 13 卷, 第 9 页, 2024. [286] L. Yu *et al.*, “Metamath: 为大型语言模型启动自己的数学问题”, *arXiv preprint arXiv:2309.12284*, 2023. [287] A. K. Singh *et al.*, “LLM 中的评估数据污染: 我们如何衡量它以及(何时)它很重要?” *arXiv preprint arXiv:2411.03923*, 2024. [288] K. Huang *et al.*, “MATH-perturb: 通过硬扰动基准测试 LLM 的数学推理能力”, *arXiv preprint arXiv:2502.06453*, 2025. [289] UK AI 安全研究所, “Inspect AI: 大型语言模型评估框架。” https://github.com/UKGovernmentBEIS/inspect_ai, 2024. [290] C. Fourrier, N. Habib, H. Kydlicek, T. Wolf, 和 L. Tunstall, “LightEval: LLM 评估的轻量级框架。” <https://github.com/huggingface/lighteval>, 2023. [291] C. Fourrier, N. Habib, A. Lozovskaya, K. Szafer, 和 T. Wolf, “Open LLM 排行榜 v2。” https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard; Hugging Face, 2024. [292] L. Gao *et al.*, “少量语言模型评估框架。” Zenodo, 2023. doi: 10.5281/zenodo.10256836. [293] S. Black *et al.*, “GPT-NeoX-20B: 一个开源的自回归语言模型”, 在 *Proceedings of the ACL workshop on challenges & perspectives in creating large language models*, 2022. 可用: <https://arxiv.org/abs/2204.06745>. [294] Y. Gu, O. Tafjord, B. Kuehl, D. Haddad, J. Dodge, 和 H. Hajishirzi, “OLMES: 语言模型评估的标准”, *arXiv preprint arXiv:2406.08446*, 2024. [295] P. Liang *et al.*, “语言模型的全面评估”, *Transactions on Machine Learning Research*, 2023, doi: 10.1111/nyas.15007. [296] Mosaic ML, “Mosaic Eval Gauntlet v0.3.0 — 评估套件。” https://github.com/mosaicml/llm-foundry/blob/main/scripts/eval/local_data/EVAL_GAUNTLET.md, 2024. [297] J. Schulman, “从人类反馈中学习强化学习的代理目标。” 国际机器学习会议 (ICML) 邀请演讲, 2023. 可用: <https://icml.cc/virtual/2023/invited-talk/21549>

[298] 张弛, O. Vinyals, R. Munos, 和S. Bengio, “深度强化学习中过拟合的研究”, *arXiv preprint arXiv:1804.06893*, 2018。[299] C. A. Goodhart和C. Goodhart, *Problems of monetary management: The UK experience*. Springer, 1984。[300] K. Hoskin, “问责制的‘可怕想法’: 将人们纳入对象的测量中”, *Accountability: Power, ethos and the technologies of managing*, 第265卷, 1996。[301] T. Lu和C. Boutilier, “通过成对偏好学习mallows模型”, 在 *Proceedings of the 28th international conference on machine learning (icml-11)*中, 2011年, 第145-152页。[302] S. Han *et al.*, “Wildguard: 针对安全风险、越狱和拒绝llm的开放一站式监管工具”, *arXiv preprint arXiv:2406.18495*, 2024。[303] H. Inan *et al.*, “Llama guard: 基于llm的人机对话输入输出保护”, *arXiv preprint arXiv:2312.06674*, 2023。[304] P. Röttger, H. R. Kirk, B. Vidgen, G. Attanasio, F. Bianchi和D. Hovy, “Xstest: 用于识别大型语言模型中夸张安全行为的测试套件”, *arXiv preprint arXiv:2308.01263*, 2023。[305] T. Coste, U. Anwar, R. Kirk和D. Krueger, “奖励模型集成有助于减轻过度优化”, *arXiv preprint arXiv:2310.02743*, 2023。[306] T. Moskovitz *et al.*, “使用约束RLHF应对奖励模型过度优化”, *arXiv preprint arXiv:2310.04373*, 2023。[307] R. Rafailov *et al.*, “直接对齐算法中奖励模型过度优化的缩放定律”, *Advances in Neural Information Processing Systems*, 第37卷, 第126207-126242页, 2024。[308] S. Zhuang和D. Hadfield-Menell, “不匹配AI的后果”, *Advances in Neural Information Processing Systems*, 第33卷, 第15763-15773页, 2020。[309] W. Yuan *et al.*, “自我奖励语言模型”。2025。可获得: <https://arxiv.org/abs/2401.10020>。[310] J. Bai *et al.*, “Qwen技术报告”, *arXiv preprint arXiv:2309.16609*, 2023。[311] G. Wang, S. Cheng, X. Zhan, X. Li, S. Song和Y. Liu, “Openchat: 使用混合质量数据推进开源语言模型”, *arXiv preprint arXiv:2309.11235*, 2023。[312] Anthropic, “Claude的性格”。2024。可获得: <https://www.anthropic.com/research/claude-character>