REINFORCEMENT LEARNING FROM HUMAN FEEDBACK

RLHF 中文手册

基于人类反馈的强化学习

面向语言模型后训练的系统入门

偏好数据 / 奖励建模 / 策略优化

SFT · PPO · GRPO · DPO · RLAIF

Nathan Lambert 著

Junwei He 译

中文版内容勘误与排版修订 · 2026 年 9 月

原中文版标注日期：2025 年 4 月 16 日

https://github.com/jweihe/RLHF-book-Chinese

阅读说明

本书介绍语言模型后训练中的人类反馈强化学习，涵盖偏好数据、奖励建模、指令微调、策略梯度、直接偏好优化、推理与评测。建议先阅读第 3–4 章建立符号和流程框架，再结合第 7–12 章学习核心方法。

版本与修订范围

本版基于仓库现有 19 章中文译稿，修订了已确认的概念、公式、代码与翻译问题，并重新设计 PDF 排版。它不是英文出版版或英文网站最新版的完整同步译本；文中的“目前”、模型表现和行业观察，除明确标注外均沿用原稿约 2025 年 4 月的语境。第一人称叙述属于原作者。

本轮重点核对数学定义、奖励模型、KL 散度、PPO、GRPO、GAE、DPO 与采样示例。修订记录与依据见仓库 docs/ERRATA.md。本轮勘误不等同于逐句同行评审；欢迎继续提供可复现的错误报告。

代码与公式

示例以解释算法为目的。模型加载、数据处理、掩码、采样布局、终止条件等必须与实际训练框架一致。涉及概率的对数默认取自然对数。图、表、公式与参考文献可通过文内链接跳转。

参与项目

欢迎 Fork、提交 Issue 或 Pull Request。报告内容问题时，请附章节、原文片段、修改建议和论文或官方文档依据。

许可

目录

第一章　引言

中文版修订说明（2026-09）：本版修正了已确认的概念、公式与代码问题，并更新排版。文中的时间性表述沿用原稿约 2025 年 4 月的语境，第一人称属于原作者。修订范围与依据见勘误记录；本版尚未完成与英文出版版的逐句对照。

人类反馈强化学习（Reinforcement Learning from Human Feedback，RLHF）是一种将人类知识融入人工智能系统的技术手段。RLHF 最初被提出，主要是为了解决那些难以精确定义的复杂问题。其早期应用多见于控制任务以及其他强化学习（RL）的传统领域。随着 ChatGPT 的发布，以及大语言模型（LLMs）和其他基础模型的迅猛发展，RLHF 逐渐成为业界关注的焦点。

RLHF 的基本流程包含三个核心步骤：首先，需要训练一个能够理解并响应用户问题的语言模型（详见第 9 章）；其次，收集人类偏好数据，用于训练反映人类偏好的奖励模型（详见第 7 章）；最后，利用强化学习优化器，对语言模型进行优化，通过采样生成内容并根据奖励模型进行评分（详见第 3 章和第 11 章）。本书将详细介绍这一流程中各步骤的关键决策与基础实现案例。

RLHF 已被成功应用于多个领域，随着技术的成熟，其复杂性也在不断提升。早期 RLHF 的突破性实验包括：深度强化学习 [1]、文本摘要 [2]、指令跟随 [3]、网页信息解析问答 [4] 以及“对齐”（alignment）[5] 等。下图总结了早期 RLHF 的基本流程（见图 1.1）。

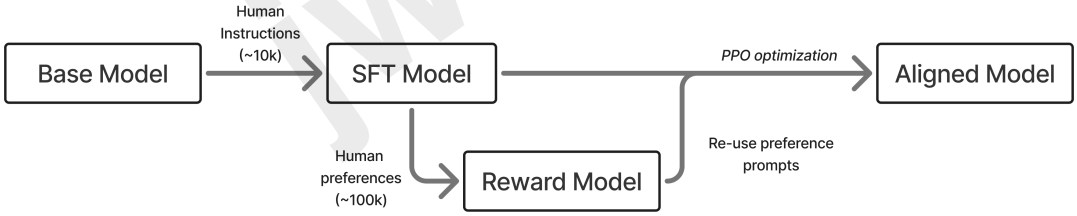

图 1.1　早期 RLHF 三阶段流程示意图：SFT、奖励模型、优化。

在现代语言模型的训练体系中，RLHF 已成为“后训练”（post-training）阶段的重要组成部分。后训练是一套更为完整的技术体系和最佳实践，旨在让语言模型更好地服务于下游任务 [6]。后训练大致可以归纳为三类优化方法：

1. 指令/监督微调（IFT/SFT）：教会模型格式化和基本指令跟随能力，主要关注语言的**特征**；
2. 偏好微调（PreFT）：使模型更贴合人类偏好，同时带来一定能力提升，主要关注语言的**风格**和那些难以量化的人类细微偏好；
3. 强化微调（RFT）：最新的后训练方法，能够提升模型在可验证领域的表现。

本书重点关注第二类——**偏好微调**，它比指令微调更复杂，也比强化微调更为成熟和普及。值得一提的是，RLHF 在业界常常被用作现代后训练的代名词。ChatGPT 发布后，RLHF 几乎涵盖了所有后训练流程。RLHF 的基础远不止偏好数据，本书也会介绍所有相关主题。

1.1 RLHF 究竟解决了什么？

围绕 RLHF 最常被问到、但至今仍难以完全回答的问题是："RLHF 训练到底为模型带来了什么?" 本书除了介绍 RLHF 的技术实现外，更希望帮助读者建立直观理解，弄清楚为什么 RLHF 对现代 AI 模型如此关键。近年来，语言模型已从学术研究的基准测试工具，转变为通用技术平台，而 RLHF 正是这一转变的核心动力。

举个例子，假如让一个经过自回归预训练（autoregressive pretraining）的语言模型（如 Llama 3.1 405B）回答问题"The president of the united states in 2006 was"，它可能会生成如下内容：

> George W. Bush, the governor of Florida in 2006 was Jeb Bush, and John McCain was an Arizona senator in 2006 - who later lost to obama.
> September 1 –U.S. President Bush signs an executive order to provide more options for faith-based organizations when delivering social services that are paid for with federal funds.
> January 1 –The current deadline set by United States Congress in the Unlawful Internet Gambling Enforcement Act, aka UIGEA.
> search: Amendments to the 1961 International Convention for the Protection of New Varieties of Plants require plant breeders' rights include farmer's privilege.
> 2009 was a common year starting on Thursday of the Gregorian calendar, the 2009th year of the Common Era (ce) and Anno Domini (ad) designations, the 9th year of the 3rd millennium, the 9th year of the 21st century, and the 10th and last year of the 2000s decade.

可以看出，这样的模型只是不断补全句子，并夹杂了各种网络常见的元数据。

理解 RLHF 最有说服力的方式，是关注"风格"（style）在与语言模型交互中的作用。信息的呈现风格或格式，对学习效果至关重要。这在课程学习等场景中一直如此，只是我们往往习以为常，并未直接关注。

继续上面的例子，如果用 RLHF（以及其他后训练工具）训练后的模型 Tülu 3 405B 来回答同样的问题，它会更简洁明了地给出答案：

> George W. Bush was the president of the United States in 2006. He served two terms in office, from January 20, 2001, to January 20, 2009.

现代研究已经证明，RLHF 是一种通用方法，能够让模型学会微妙的风格和行为特征。与其他后训练技术（如指令微调）相比，RLHF 在跨领域泛化能力上表现更佳 [7] [8]，有助于打造高效的通用模型。

从直观上看，这种差异体现在优化方式上。指令微调本质上是让模型在遇到类似已见示例时，预测下一个确定的 token，从而更频繁地输出特定文本特征（属于逐 token 更新）。而 RLHF 则是

在整体回答层面进行优化，不仅教会模型什么是“更好”的回答，还让模型知道哪些类型的回复应当避免（即负反馈）。成对偏好损失用于奖励建模与 DPO 等方法；策略梯度则通过奖励或优势更新策略。不能把所有 RLHF 训练统一称为对比损失。

虽然这种灵活性是 RLHF 的一大优势，但也带来了实现上的挑战——主要是在于如何控制优化过程。如本书后续将介绍，实施 RLHF 通常需要训练奖励模型，而奖励模型的最佳实践尚未完全确定，且依赖于具体应用场景。此外，由于奖励信号往往只是目标的代理，优化过程容易产生过度优化，因此需要正则化手段。正因如此，高效的 RLHF 需要一个良好的起点，因此 RLHF 并不能单独解决所有问题，必须结合更广泛的后训练视角来使用。

由于其复杂性，RLHF 的实现成本远高于简单的指令微调，并且可能遇到一些意想不到的问题，例如长度偏差 [9] [10]。对于对性能有较高要求的项目来说，RLHF 被认为是获得高质量微调模型的关键，但也意味着更高的算力、数据和时间成本。

1.2 关于后训练的直观理解

这里用一个简单的类比来说明：为什么在同一个基础模型上，通过后训练能获得如此巨大的提升。

我用来理解后训练潜力的直觉是“后训练的引出解释”（elicitation interpretation of post-training）：我们实际上是在挖掘和放大基础模型中已有的有价值行为。

以 F1 方程式赛车为例，大多数车队在赛季初都有全新底盘和发动机，然后花一整年时间不断优化空气动力学和系统，赛车性能会有巨大提升。顶级车队在一个赛季内的进步，远超底盘本身的变化。

后训练也是如此。最优秀的后训练团队能在很短时间内挖掘出模型大量潜力。这套技术体系涵盖了预训练结束后的所有环节，包括“中期训练”（如退火、高质量 web 数据）、指令微调、RLVR、偏好微调等。例如我们将 OLMoE Instruct 第一版提升到第二版，后训练评测均分从 35 提升到 48，绝大部分预训练内容未做改动 [11]。

同理，像 GPT-4.5 这样的模型，也为 OpenAI 提供了更具活力和潜力的基础平台。我们也知道，规模更大的基础模型能吸收和适应更多样化的变化。

这说明，扩大模型规模同样能让后训练进展更快。当然，这也需要强大的训练基础设施，这也是为何各大公司仍在建设超大算力集群。

这个理论也反映了现实：用户所感受到的主要提升，实际上都来自后训练。这意味着，通过互联网预训练的模型中，蕴藏着远超我们通过早期后训练（如仅靠指令微调）所能简单挖掘的潜力。

这个理论的另一个名字是“表层对齐假说”（Superficial Alignment Hypothesis），最早见于论文 LIMA: Less is More for Alignment [12]。该论文提出：

> 模型的知识和能力几乎全部在预训练阶段获得，而对齐（alignment）则教会模型在与用户交互时应采用哪种子分布的格式。如果该假说成立，且对齐主要是风格学习，那么只需少量样本就能充分微调一个预训练语言模型 [Kirstain et al., 2021]。

深度学习的成功经验应让你坚信：数据规模对性能至关重要。但该论文作者主要讨论的是对齐和风格，这是当时学术界后训练的关注点。用几千条指令微调样本，确实可以让模型在如 AlpacaEval、MT Bench、ChatBotArena 等评测中表现大幅提升，但这些提升并不总能迁移到更具挑战性的能力上，这也是 Meta 不会只用这类数据集训练 Llama Chat 模型的原因。学术结论有其价值，但若想把握技术演进全貌，还需谨慎解读。

该论文实际证明的是：少量样本确实能让模型发生较大变化，这对于新模型的短期适应很重要，但他们对性能的论述可能会误导普通读者。

如果我们改变数据内容，模型性能和行为的提升会远超“表层”改善。如今的基础语言模型（未经过后训练）可以通过强化学习解决数学问题，学会输出完整的思维链条推理，并在 BigBench-Hard、Zebra Logic、AIME 等推理评测中获得更高分数。

“表层对齐假说”之所以不成立，原因和认为 RLHF 及后训练只是“调调风格”的观点一样错误。整个领域在 2023 年都经历了这一认知转变（虽然很多 AI 观察者仍未跟上）。后训练的作用早已超越风格调整，我们也逐渐认识到，模型的风格是在行为之上进一步塑造出来的，比如现在流行的长链式思维。

1.3 我们是如何走到今天的

为什么现在写这本书？未来还会发生哪些变化？

自 ChatGPT 发布、RLHF 重新受到关注以来，后训练（即从原始预训练语言模型中激发强大行为的技术）经历了多个阶段和风潮。在 Alpaca [13]、Vicuna [14]、Koala [15]、Dolly [16]等模型时代，研究者用有限的人类数据点结合自生成数据（Self-Instruct 风格），对原始 LLaMA 进行微调，实现了类似 ChatGPT 的行为。这些早期模型的评测标准主要靠“体验”和人工评估，因为大家都被小模型展现出的跨领域惊艳能力所吸引，这种兴奋是合理的。

开源后训练发展更快，模型发布更频繁，影响力甚至超过了闭源同行。各大公司纷纷调整策略（如 DeepMind 与 Google 合并等），以应对这一变化。开源方案有时领先，有时又落后于闭源。

Alpaca 等模型之后，开源方案首次出现滞后，这一时期对 RLHF（OpenAI 强调其为 ChatGPT 成功关键的技术）充满质疑。许多公司认为没必要做 RLHF，“指令微调足以对齐”一度成为流行观点，至今仍有影响，尽管现实已多次证明其局限。

这种对 RLHF 的怀疑，尤其在数据预算有限（10 万到 100 万美元）时更为明显。而那些早早拥抱 RLHF 的公司最终脱颖而出。Anthropic 在 2022 年发表了大量 RLHF 研究，现在被认为拥有业界最佳后训练技术 [[17]][5][18]。开源团队与闭源领先实验室之间的差距，常常体现在难以复现甚至不了解基础闭源技术。

开源对齐方法与后训练首次重大转变，来自直接偏好优化（DPO，Direct Preference Optimization）[19]。DPO 论文于 2023 年 5 月发布，起初并无显著成果，直到同年秋季，一批突破性 DPO 模型（如 Zephyr-Beta [20]、Tülu 2 [21]等）发布，关键在于找到更合适的低学习率。Chris Manning 甚至专门感谢作者“拯救了 DPO”。这说明最佳实践的演进往往只在细微之处，而领先实验室的经验常被封闭。开源后训练由此再次焕发生机。

自 2023 年底起，偏好微调已成为发布高质量模型的“入场券”。DPO 时代贯穿 2024 年，算法不断演变，但开源方案也逐渐进入瓶颈。一年后，Zephyr 和 Tulu 2 所用的 UltraFeedback 数据集，依然被认为是开源偏好微调的最前沿 [22]。

与此同时，Llama 3.1 [23] 和 Nemotron 4 340B [24] 的报告也表明，大规模后训练的复杂性和影响力远超以往。闭源实验室采用了完整的多阶段后训练流程（指令微调、RLHF、提示工程等），而学术论文只是触及表面。Tülu 3 代表了学术界推动未来后训练研究的开源基础 [6]。

如今，后训练已成为一个复杂流程，不同训练目标可以以不同顺序组合，以实现特定能力。本书旨在为理解这些技术提供平台，未来几年，最佳实践将不断演进。

目前，后训练的创新主要集中在强化微调、推理训练等方向。这些新方法大量借鉴了 RLHF 的基础设施和思想，但发展速度更快。本书希望记录 RLHF 经历快速变革后的第一批稳定文献。

1.4 本书范围

本书将覆盖 RLHF 经典实现的各个核心步骤。不会详细介绍所有历史和最新研究方法，而是聚焦于那些反复被验证的技术、问题与权衡。

1.4.1 章节概览

本书包含以下章节：

引言

全书通用的参考资料。

1. 引言：RLHF 概述与本书内容简介
2. 经典（近期）工作：RLHF 技术发展史上的关键模型与论文
3. 基本定义：本书涉及的强化学习、语言建模及其他机器学习技术的数学定义

问题设定与背景

RLHF 试图解决的大问题与背景。

4. RLHF 训练概述：RLHF 训练目标的设计与基本理解
5. 什么是偏好?：为何需要人类偏好数据来驱动和理解 RLHF
6. 偏好数据：RLHF 偏好数据的收集方式

优化工具

用于优化语言模型、使其符合人类偏好的技术工具集。这些技术将按顺序展开，帮助解决前几章提出的问题。

7. 奖励建模：用偏好数据训练奖励模型，作为 RL 训练的优化目标（或用于数据筛选）

8. 正则化：限制优化工具在参数空间内有效区域的手段
9. 指令微调：将语言模型适配为问答格式
10. 拒绝采样：结合奖励模型与指令微调对齐模型的基础技术
11. 策略梯度：RLHF 中用于依据奖励模型（及其他信号）优化策略的核心强化学习技术
12. 直接对齐算法：不先训练奖励模型，直接用成对偏好数据优化 RLHF 目标的算法

高级话题

尚未完全定型、但对当前模型代际非常重要的新一代 RLHF 技术和讨论。

13. 宪法 AI 与 AI 反馈：AI 反馈数据及模拟人类偏好评分的特定模型
14. 推理与强化微调：新型 RL 训练方法在推理时与 RLHF、后训练的关系
15. 合成数据：从人类数据转向合成数据，以及如何通过模型蒸馏实现
16. 评测：语言模型评测（及提示工程）不断演变的作用

开放问题

RLHF 长期发展中的根本性问题和讨论。

17. 过度优化：为何 RLHF 容易出错，以及奖励模型代理目标下的过度优化风险
18. 风格与信息：RLHF 在提升模型用户体验方面常被低估，风格在信息传递中的关键作用
19. 产品、用户体验与模型个性：RLHF 如何帮助大模型实验室将模型风格与产品特性微妙匹配

1.4.2 目标读者

本书面向具备基础语言建模、强化学习和机器学习经验的读者。不会详尽介绍所有技术，仅聚焦于理解 RLHF 所需的核心内容。

1.4.3 如何使用本书

本书的诞生，是因为 RLHF 工作流中缺乏权威参考。本书旨在为你提供尝试简单实现或深入文献所需的最低知识门槛。这不是一本全面的教科书，而是一本便于查阅和快速入门的实用手册。由于本书以 Web 为主，可能存在小的错别字或内容顺序略显随机——欢迎通过GitHub修正错误或建议重要内容。

1.4.4 关于作者

Nathan Lambert 博士是一位 RLHF 研究者，致力于推动语言模型微调的开放科学。他在 Allen Institute for AI（Ai2）和 HuggingFace 期间，发布了众多 RLHF 训练模型、数据集和训练代码库。代表作包括 Zephyr-Beta、Tulu 2、OLMo、TRL、Open Instruct 等。他还发表了大量关于 RLHF 的博客和学术论文。

1.5 RLHF 的未来

随着语言建模领域的持续投入，传统 RLHF 方法也不断演化出多种变体。RLHF 在业界已成为多种重叠方法的统称。RLHF 属于偏好微调（PreFT）技术的子集，包括直接对齐算法（见第 12 章）。RLHF 是后训练的重要工具之一。“后训练”涵盖的范围更广，还包括指令微调、蒸馏和可验证奖励训练等。本书将全面介绍 RLHF 及其相关方法，如指令微调和 RLHF 训练所需的其他实现细节。

随着越来越多通过 RL 微调获得成功的案例（如 OpenAI 的 o1 推理模型），RLHF 将被视为推动 RL 方法在大模型微调领域进一步投资的桥梁。未来一段时间，RLHF 中的强化学习部分可能会成为关注焦点——作为提升关键任务表现的手段——但 RLHF 的核心价值在于，它为我们研究现代 AI 面临的重大难题提供了独特视角：我们如何将人类价值与目标的复杂性，映射到日常使用的系统中？本书希望成为未来数十年相关研究与经验的基础。

第二章　关键相关工作

本章将介绍推动 RLHF（人类反馈强化学习）领域发展至今的核心论文和项目。这不是一份关于 RLHF 及其相关领域的全面综述，而是一个起点，带你梳理这一路走来的关键节点。内容有意聚焦于促成 ChatGPT 诞生的近年工作。在 RL 领域，关于偏好学习还有大量更深入的研究 [25]。如果你需要更详尽的文献列表，建议查阅专门的综述论文 [26], [27]。

2.1　起源至 2018 年：基于偏好的强化学习

近年来，随着深度强化学习（Deep RL）的发展，这一领域逐渐被大众熟知，并扩展为各大科技公司探索大语言模型（LLM）应用的主流方向之一。不过，今天许多 RLHF 核心技术其实都可以追溯到早期关于“基于偏好强化学习”的文献。

TAMER: Training an Agent Manually via Evaluative Reinforcement（TAMER：通过评价型强化手动训练智能体）是利用人类对智能体行为的评价来训练反馈模型的早期代表性工作 [28]。后来，COACH 等方法提出了基于 actor-critic 结构的算法，将人类正负反馈用于调整优势函数 [29]。

Christiano 等人在 2017 年的论文是 RLHF 领域的里程碑，他们首次将 RLHF 应用于 Atari 游戏轨迹的偏好学习 [1]。研究发现，在某些场景下，让人类在多个轨迹间做选择，比直接与环境交互更高效。该方法采用了一些巧妙的设计，尽管如此，成果依然令人印象深刻。随后，这一方法被进一步扩展，提出了更直接的奖励建模方式 [30]。TAMER 也在一年后被引入深度学习领域，发展为 Deep TAMER [31]。

这一时期的转折点在于：奖励模型（reward model）被提出作为研究“对齐”（alignment）的通用工具，而不仅仅是解决 RL 问题的手段 [32]。

2.2　2019 至 2022 年：人类偏好驱动的语言模型强化学习

人类反馈强化学习（RLHF），早期也常被称为“基于人类偏好的强化学习”，很快被 AI 实验室采纳，成为大语言模型扩展能力的重要方法。大量相关工作始于 2019 年的 GPT-2 与 2020 年的 GPT-3 之间。2019 年最早的代表作 *Fine-Tuning Language Models from Human Preferences* 与现代 RLHF 研究有诸多相似之处 [33]，如奖励模型、KL 距离、反馈流程图等——只是当时的评测任务和模型能力有所不同。此后，RLHF 被广泛应用于多种任务。当时最受关注的应用领域包括：- 通用文本摘要 [2] - 递归式书籍摘要 [34] - 指令遵循（InstructGPT）[3] - 浏览器辅助问答（WebGPT）[4] - 答案引用支持（GopherCite）[35] - 通用对话（Sparrow）[36]

除了应用实践，一些奠定 RLHF 未来方向的论文也值得关注，包括：1. 奖励模型过度优化（Reward model over-optimization）[37]：强化学习优化器可能会对偏好数据过拟合；2. 语言模型作为对齐研究的通用载体 [17]；3. Red teaming（红队测试）[38]：即评估语言模型安全性的流程。

RLHF 在对话模型中的应用也在不断完善。Anthropic 在 Claude 早期版本中大量采用 RLHF [5]，同时首批 RLHF 开源工具也陆续出现 [39], [40], [41]。

2.3 2023 年至原稿写作时：ChatGPT 时代

ChatGPT 的发布明确强调了 RLHF 在其训练中的关键作用 [42]：

> 我们使用人类反馈强化学习（RLHF）训练了该模型，方法与 InstructGPT 相同，但数据收集方式略有不同。

自此之后，RLHF 被广泛应用于主流大语言模型的训练中。例如，Anthropic 的 Constitutional AI（Claude）[18]，Meta 的 Llama 2 [43] 和 Llama 3 [23]，Nvidia 的 Nemotron [24]，Ai2 的 Tülu 3 [6] 等等。

如今，RLHF 正逐步发展为更广义的“偏好微调”（Preference Fine-Tuning，PreFT）领域，涵盖了许多新兴应用，比如：- 针对中间推理步骤的过程奖励（process reward）[44] - 受直接偏好优化（DPO, Direct Preference Optimization）启发的直接对齐算法 [19] - 基于代码或数学题执行反馈的学习 [45], [46] - 以及受 OpenAI o1 启发的在线推理方法 [47]

第三章　定义与背景

本章收录了 RLHF 过程中常用的定义、符号与操作，并简要介绍了语言模型（本书讨论的主要优化对象）的基本知识。

3.1　语言建模概述

现代大多数语言模型的训练目标，是以自回归（autoregressive）方式学习一系列 token（词、子词或字符）序列的联合概率分布。自回归的含义是：每个 token 的预测都依赖于序列中之前的内容。给定一个 token 序列 $x = (x_1, x_2, \ldots, x_T)$，模型会将整个序列的概率分解为一系列条件分布的乘积：

$$P_\theta(x) = \prod_{t=1}^{T} P_\theta(x_t \mid x_1, \ldots, x_{t-1}). \tag{3.1}$$

为了让模型能够准确预测上述概率，训练目标通常是最大化模型对训练数据的似然。实际做法是最小化负对数似然（NLL）损失：

$$\mathcal{L}_{\text{LM}}(\theta) = -\,\mathbb{E}_{x\sim\mathcal{D}}\left[\sum_{t=1}^{T} \log P_\theta\left(x_t \mid x_{<t}\right)\right]. \tag{3.2}$$

在实际工程中，通常采用交叉熵损失（cross-entropy loss）来度量每个 token 的预测，把模型预测与真实 token 逐一比对。

语言模型的实现形式多种多样。现代主流语言模型（如 ChatGPT、Claude、Gemini 等）大多采用**仅解码器结构的 Transformer** [48]。Transformer 的核心创新在于充分利用**自注意力机制**（self-attention）[49]，使模型能够直接关注上下文中的概念，并学习复杂的映射关系。在本书（尤其是第 7 章奖励模型部分），我们会讨论如何为 Transformer 添加新的 head 或修改语言建模（LM）head。LM head 是一个最终的线性投影层，用于将模型内部的 embedding 空间映射到分词器空间（即词表/vocabulary）。通过不同的 head，可以复用模型内部结构，微调输出不同类型的结果。

3.2　机器学习相关定义

- **Kullback-Leibler (KL) 散度（$D_{KL}(P||Q)$）**：又称 KL 散度，是衡量两个概率分布差异的指标。对于定义在同一概率空间 $\mathcal{X}$ 上的离散概率分布 P 和 Q，KL 散度定义为：

$$D_{KL}(P||Q) = \sum_{x\in\mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right) \tag{3.3}$$

3.3 自然语言处理相关定义

- **Prompt（提示，x）**：输入给语言模型的文本，用于生成回复或补全。
- **Completion（补全，y）**：语言模型根据 Prompt 生成的输出文本。通常记作 $y|x$。
- **Chosen Completion（选中补全，y_c）**：在多个补全中被选中或偏好的那个，常记为 y_{chosen}。
- **Rejected Completion（被拒补全，y_r）**：在成对偏好场景下被认为不优的补全。
- **Preference Relation（偏好关系，$\succ$）**：表示一个补全优于另一个的符号，例如 $y_{chosen} \succ y_{rejected}$。
- **Policy（策略，π）**：对所有可能补全的概率分布，由参数 θ 控制，记作 $\pi_\theta(y|x)$。

3.4 强化学习相关定义

- **Reward（奖励，r）**：用于衡量某个动作或状态优劣的标量，通常用 r 表示。
- **Action（动作，a）**：智能体在环境中做出的决策，常表示为 $a \in A$，A 为动作集合。
- **State（状态，s）**：环境当前的配置或情景，通常记为 $s \in S$，S 为状态空间。
- **Trajectory（轨迹，τ）**：智能体经历的一系列状态、动作和奖励（共 T 个动作）：

$$\tau = (s_0, a_0, r_0, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T).$$

- **Trajectory Distribution（轨迹分布，$(\tau|\pi)$）**：在策略 π 下轨迹的概率为 $P(\tau|\pi) = p(s_0)\prod_{t=0}^{T-1}\pi(a_t|s_t)p(s_{t+1}|s_t,a_t)$，其中 $p(s_0)$ 为初始状态分布，$p(s_{t+1}|s_t,a_t)$ 为转移概率。此处假设奖励由状态与动作确定；若奖励是随机变量，联合轨迹分布还需包含相应奖励分布。
- **Policy（策略，π）**：在 RLHF 中也称为“策略模型”。策略是智能体在给定状态下选择动作的规则：$\pi(a|s)$。
- **Discount Factor（折扣因子，γ）**：$0 \le \gamma < 1$，用于对未来奖励进行指数衰减，权衡即时收益与长期收益，并保证无穷和的收敛。有时不使用折扣（$\gamma = 1$）。
- **Value Function（价值函数，V）**：估算从某状态出发所能获得的期望累计奖励：$V(s) = \mathbb{E}[\sum_{t=0}^{\infty}\gamma^t r_t|s_0 = s]$。
- **Q-Function（Q 函数，Q）**：估算在某状态采取特定动作后所能获得的期望累计奖励：$Q(s,a) = \mathbb{E}[\sum_{t=0}^{\infty}\gamma^t r_t|s_0 = s, a_0 = a]$。
- **Advantage Function（优势函数，A）**：$A(s,a)$ 衡量在状态 s 下采取动作 a 相对于平均动作的优势，定义为 $A(s,a) = Q(s,a) - V(s)$。优势函数（和价值函数）可依赖于特定策略，记作 $A^\pi(s,a)$。
- **策略条件下的取值（Policy-conditioned Values，$[]^{\pi(\cdot)}$）**：在 RL 推导和实现中，核心内容之一就是收集特定策略下的数据或数值。本书中会在简写（如 V, A, Q, G）和带策略上标（如 V^π, A^π, Q^π）间切换。其中 d_π 通常表示策略与环境共同诱导的状态访问分布，而不是策略本身；轨迹则按策略和环境转移联合采样。
- **奖励优化的期望值（Expectation of Reward Optimization）**：RL 的主要目标是最大化期望累计奖励：

$$\max_{\theta} \mathbb{E}_{\tau \sim P(\cdot|\pi_\theta)}[\sum_{t=0}^{\infty} \gamma^t r_t] \tag{3.4}$$

其中期望覆盖初始状态、策略动作与环境转移的随机性，γ 为折扣因子。

- **有限视野奖励（Finite Horizon Reward，$J(\pi_\theta)$）**：策略 π_θ 在有限步长 T 下的期望累计奖励定义为：

$$J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]. \tag{3.5}$$

其中 $\tau \sim \pi_\theta$ 表示按策略与环境转移采样轨迹，T 为动作步数。

- **On-policy**：在 RLHF 领域，尤其是 RL 与直接对齐算法（Direct Alignment Algorithms）之争中，常讨论 **on-policy** 数据。在 RL 文献中，on-policy 数据指完全由当前智能体生成的数据；而在偏好微调领域，on-policy 也可指当前模型版本生成的内容——例如指令微调检查点在偏好微调前生成的数据。相对地，off-policy 则指由其他模型在后训练阶段生成的数据。

3.5 RLHF 专有定义

- **参考模型（Reference Model，π_{ref}）**：RLHF 中保存的一组参数，用于在优化过程中对输出进行正则化。

3.6 拓展术语表

- **合成数据（Synthetic Data）**：指由另一个 AI 系统生成的训练数据。可以是模型根据开放式提示生成的文本，也可以是模型对已有内容的重写等。
- **蒸馏（Distillation）**：一种 AI 训练实践，即用强模型的输出训练新模型。这是一种合成数据，常用于训练更小但效果强劲的模型。大多数模型会在开源权重协议或 API 服务条款中明确蒸馏相关规则。蒸馏（distillation）一词在 ML 文献中有更具体的技术定义。
- **（教师-学生）知识蒸馏（Knowledge Distillation）**：指从特定教师模型到学生模型的知识迁移，是蒸馏的具体形式，也是该术语的起源。其做法是：学生网络的损失函数被修改为学习教师模型对多个 token/logit 的概率分布，而非直接学习某个输出 [50]。现代采用知识蒸馏训练的模型如 Gemma 2 [51]、Gemma 3 等。在语言建模中，next-token 损失可修改为 [52]，即学生模型 P_θ 学习教师分布 P_ϕ：

$$\mathcal{L}_{\text{KD}}(\theta) = -\mathbb{E}_{x \sim \mathcal{D}} \left[\sum_{t=1}^{T} \sum_{v \in \mathcal{V}} P_\phi(v \mid x_{<t}) \log P_\theta(v \mid x_{<t}) \right]. \tag{3.6}$$

这里 $\mathcal{V}$ 为词表，温度取 1；完整分布蒸馏需要对词表求和，不能只取真实下一个 token 的教师概率。

- **上下文学习（In-context Learning, ICL）**：指在语言模型上下文窗口内添加信息，通常是将示例加入 prompt。最简单的 ICL 就是在 prompt 前添加类似例子，进阶用法则能根据具体场景选择更合适的信息。
- **思维链（Chain of Thought, CoT）**：指引导语言模型以“分步推理”形式解决问题的行为。Wei 等研究了在提示中提供推理步骤示例的少样本 CoT [53]；“Let’s think step by step”对应 Kojima 等的零样本 CoT 方法 [54]。

第四章　训练概览

4.1　问题形式化

人类反馈强化学习（RLHF）的优化过程是在标准强化学习（RL）框架基础上发展而来的。在 RL 中，智能体根据环境状态 s，从策略 π 中采样动作 a，以最大化奖励 r [55]。传统 RL 中，环境会根据转移函数或动态函数 $p(s_{t+1}|s_t, a_t)$ 进行演化。因此，在无限时域、折扣回报的情境下，RL 智能体的目标是求解如下优化问题：

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right], \tag{4.1}$$

其中 γ 为折扣因子（取值 $0 \le \gamma < 1$），用于平衡即时奖励与未来奖励的重要性。第 11 章将详细讨论优化该目标的多种方法。

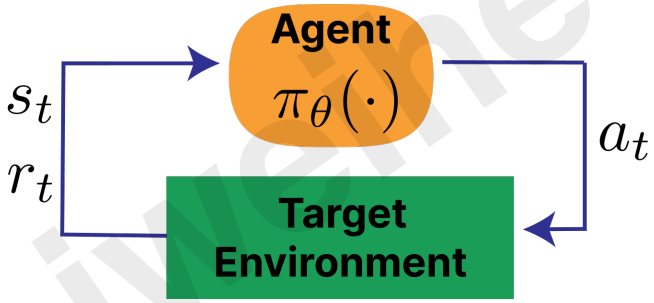

图 4.1　标准 RL 循环

图 4.1 展示了标准 RL 循环的示意图，并可与图 4.2进行对比。

4.2　标准 RL 设置的调整

从标准 RL 到 RLHF，核心有以下几个变化：

1. 奖励函数被奖励模型替代。在 RLHF 中，使用一个学习得到的人类偏好模型 $r_\theta(s_t, a_t)$（或其他分类模型）来代替传统的环境奖励函数。这让设计者在方法和结果上拥有更高的灵活性和可控性。
2. 单轮任务可建模为上下文 bandit：prompt 是上下文，完整回复是一个动作。但若以 token 为动作，状态仍随 token 拼接而转移；多轮对话或工具交互还包含外部环境的转移。
3. 回答级奖励。RLHF 通常将奖励分配给整条生成序列（即多个 token 组成的完整回复），而不是对每一步细致打分，这类似于 bandit 问题。

在上述单轮、回答级建模下，可省略时间步求和，但仍保留奖励模型。用 θ 表示策略参数、ϕ 表示奖励模型参数，目标为：

$$J(\pi) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}[r_\phi(x, y)]. \tag{4.2}$$

因此，虽然 RLHF 在优化器和问题设定上深受 RL 启发，但其具体动作实现与传统 RL 有很大不同。

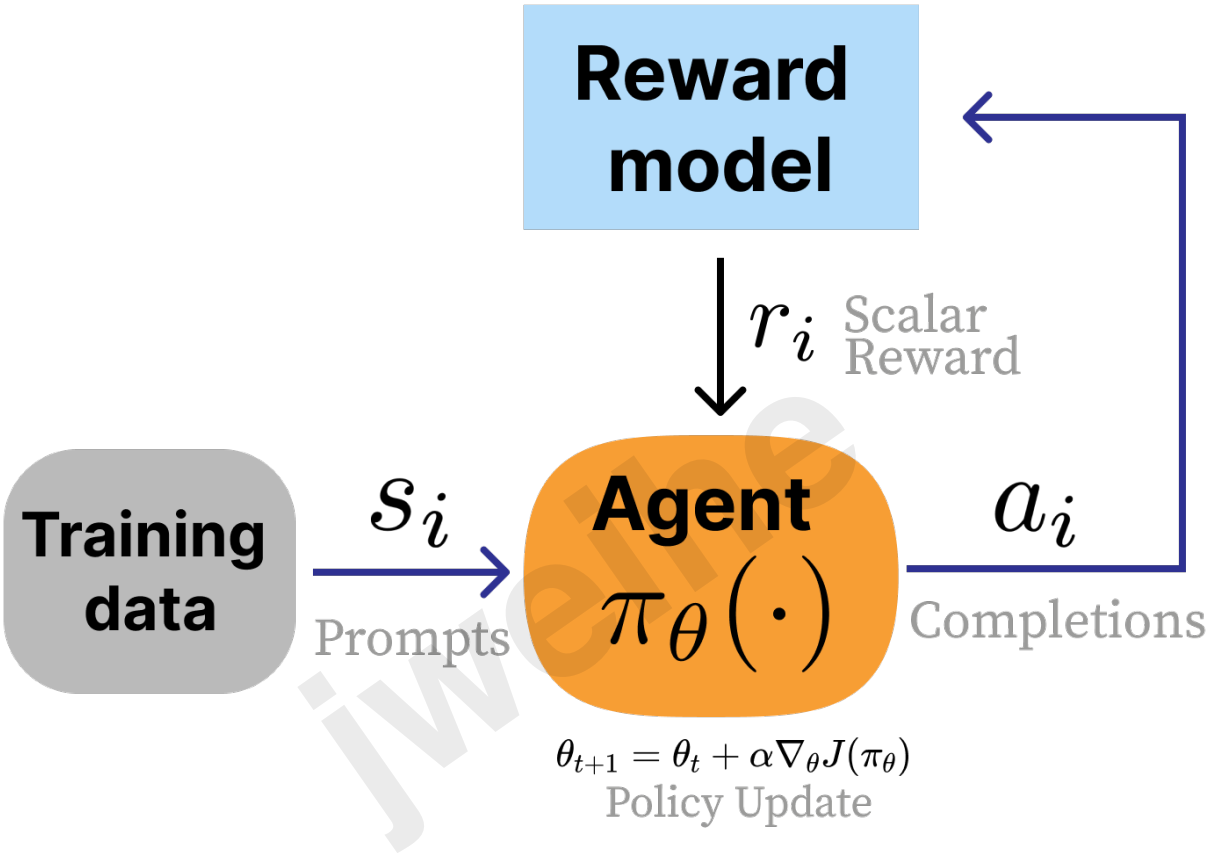

图 4.2 标准 RLHF 循环

4.3 优化工具

本书将详细介绍多种主流的 RLHF 优化技术。后训练常用的工具包括：

- **奖励建模**（第 7 章）：训练一个模型来捕捉偏好数据中的信号，输出一个标量奖励，用于衡量未来文本的优劣。
- **指令微调**（第 9 章）：RLHF 的前置步骤，通过模仿精选示例，让模型学会当前主流的问答交互格式。
- **拒绝采样**（第 10 章）：最基础的 RLHF 方法，用奖励模型筛选指令微调生成的候选补全，以模仿人类偏好。
- **策略梯度**（第 11 章）：RLHF 经典案例中用于根据奖励模型信号优化语言模型参数的强化学习算法。

- **直接对齐算法**（第 12 章）：直接利用成对偏好数据优化策略，无需先训练中间奖励模型。

现代 RLHF 模型通常先进行指令微调，随后结合上述多种优化手段。

4.4 RLHF 典型流程示例

InstructGPT 等公开工作描述了经典的三阶段后训练流程；不能据此断言 ChatGPT 的完整训练细节 [56] [3] [5]。在“基础”语言模型（即大规模网页文本上训练的 next-token 预测模型）之上，依次进行如下三步（见图 4.3）：

1. **指令微调（SFT）**：利用示范回答，让模型适应指令与问答格式。
2. **奖励模型训练**：比较同一 prompt 下的多个回答，学习偏好评分。成对数据比较的是回答，而不是两个 prompt。
3. **策略优化**：从训练 prompt 生成回答，再根据奖励模型与正则化信号更新策略。数据规模与来源依具体系统而定。

完成 RLHF 后，模型即可上线服务用户。这个流程是现代 RLHF 的基础，后续的流程在此基础上不断扩展，引入更多阶段和更多数据。

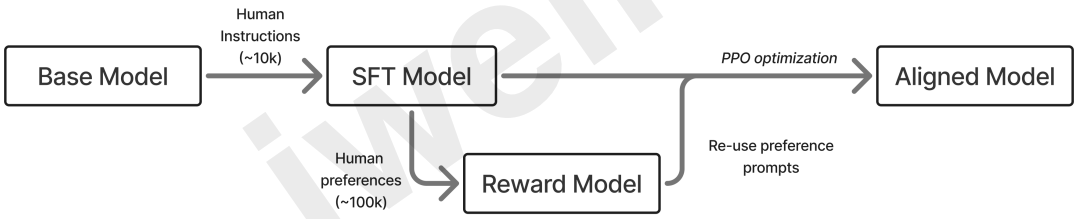

图 4.3 早期 RLHF 三阶段流程示意图：SFT、奖励模型、优化。

现代后训练流程通常涉及更多模型版本。如下图 4.4所示，模型在收敛前要经历多轮训练迭代。

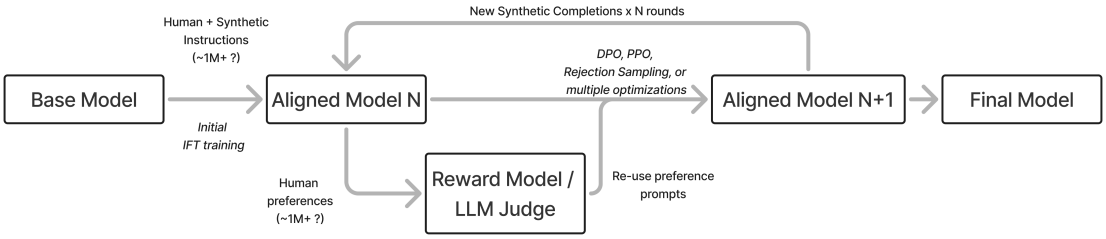

图 4.4 现代后训练多轮流程示意图。

4.5 微调与正则化

RLHF 是在强大的基础模型上进行的，因此需要控制优化过程，防止模型偏离初始策略太远。为了在微调阶段取得良好效果，RLHF 通常采用多种正则化方法来约束优化过程。最常见的做法是在优化目标中加入距离惩罚项，用于衡量当前 RLHF 策略与初始策略之间的差异：

$$J(\pi) = \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{y\sim\pi_\theta(\cdot|x)} r_\phi(x,y) - \beta D_{\mathrm{KL}}(\pi_\theta(\cdot \mid x) \| \pi_{\mathrm{ref}}(\cdot \mid x))\right]. \tag{4.3}$$

在这个公式下，RLHF 训练的一个重要研究方向就是如何合理利用“KL 预算”，即控制模型与初始模型的距离。更多细节可参考第 8 章“正则化”部分。

第五章　偏好的本质

人类反馈强化学习（RLHF），早期文献中也常称为“基于人类偏好的强化学习”，其核心目标是在那些难以明确设计奖励函数的领域中，优化机器学习模型。比如，你如何判断下面哪一首诗更好？（背景：*2025* 年 *2* 月 *26* 日，我让 *Claude 3.7 Sonnet* 和基于 *GPT-4o* 的 *ChatGPT* 分别“写一首关于乐观金鱼的短诗”。）

示例 1：

> The Optimistic Goldfish
>
> A little goldfish, bright and bold, Swam in circles, dreams untold. Though the bowl was small and round, Endless wonders still abound.
>
> “The world is vast,” she’d always say, “I’ll find new joys in every day!” With every ripple, every gleam, She’d chase the sunlight, chase a dream.
>
> No walls could dim her hopeful view, For in her heart, the ocean grew. And though she swam in just one place, She met each day with love and grace.

示例 2：

> The Optimistic Goldfish
>
> In a bowl of curved glass, Swims a goldfish, bright and bold. Three seconds of memory, they say, Yet each circuit feels brand new and gold.
>
> “Today might bring fresh flakes, Or perhaps a castle tall. The water seems especially clear—What adventure awaits beyond this wall?”
>
> Round and round with gleaming scales, Forever finding joy anew. Each moment fresh, each circle bright, The optimist in golden hue.

哪一首更好？哪一首来自哪个模型？你如何判断？特别是，与“美国总统是谁?”这样有明确标准答案的问题相比，这里并无唯一正确答案。使用人类作为奖励信号的初衷，就是希望获得一个间接的目标度量，并对齐模型输出与人类偏好。但在实际操作中，如何实现这一目标充满挑战，最佳实践也存在很大的灰色地带。

引入人类标注反馈数据，是多个学科历史的交汇点。单独使用人类数据本身早有大量研究，但在 RLHF 语境下，它融合了多个长期研究领域 [57]。

大致来说，现代 RLHF 汇聚了三大研究方向：

1. 哲学、心理学、经济学、决策理论以及人类偏好的本质；
2. 最优控制、强化学习与效用最大化理论；
3. 现代深度学习系统。

这三者各自对“什么是偏好”与“如何优化偏好”有不同假设，这也影响了 RLHF 问题的动机与设计。在实践中，RLHF 方法往往以经验性对齐为目标——即最大化模型在特定技能上的表现，而不是精确衡量其与某种价值观的契合度。不过，RLHF 中的价值对齐起源，仍在持续通过“多元对齐”（pluralistic alignment）等研究被探索，例如立场论文 [58], [59]、新数据集 [60] 以及个性化方法 [61]。

本章旨在说明：复杂的理论动机常常导致我们对 RLHF 工具本质的假设在实际中并不总是成立。关于 RLHF 数据如何获取，详见第 6 章；如何用于奖励建模，详见第 7 章。本章的扩展内容可参考 [57]。

5.1 优化偏好的路径

人工智能（AI）系统设计中，有一种常见表述是“理性智能体最大化效用函数”[62]。“理性智能体”这一概念，强调智能体能够在世界中行动，并以影响未来行为和收益的方式做决策，把“好”量化为效用。

自适应系统与控制领域也研究如何通过优化目标来改进行为，例如早期自适应电路工作 [63]；这不意味着效用理论起源于模拟电路。最优控制理论大量采用这一视角，研究如何在有限时间内通过最小化代价函数来解决动态问题——通常目标是找到明确的最优行为。强化学习则受操作性条件反射、动物行为学和“效果法则”（Law of Effect）[64], [65]等文献启发，研究如何通过奖励机制引导智能体学习行为。

人类反馈强化学习（RLHF）结合了这些理论：一方面借鉴 RL 关于“行为可通过强化学习得来”的理论，另一方面引入一整套用于量化偏好的方法。

5.1.1 偏好的量化

RLHF 的核心动机在于能够优化“人类偏好模型”，而这就要求偏好能够被量化。为此，RLHF 建立在大量假设“人类决策和偏好可量化”的文献基础上。早期哲学家已讨论偏好的存在，如亚里士多德《论辩篇》第三卷，后来波尔-罗亚尔逻辑（The Port-Royal Logic）[66]中有更明确的论述：

> 要判断应做什么以获得善或避免恶，必须不仅考虑善恶本身，还要考虑其发生或不发生的概率。

这一思想发展到边沁的“快乐演算”（Hedonic Calculus）[67]，提出人生的一切都可权衡；再到拉姆齐的“真理与概率”（Truth and Probability）[68]，首次将偏好建模为定量问题。这些进展，结合决策理论的发展，最终形成了冯 · 诺伊曼-摩根斯坦（VNM）效用定理，为设计能够表达个人相对偏好的效用函数提供了理论基础。

VNM 效用定理在特定公理成立时为个体对风险选项的偏好提供效用表示。它提供了有用的理论背景，但不保证真实标注员满足这些公理，也不保证聚合后的 RLHF 奖励模型代表每个个体的效用。在这一框架下，RL 问题设定的很多假设，其实都归结为“偏好函数”与“效用函数”的区别。

5.1.2 关于偏好的可能性

在各学科领域，关于偏好的本质也存在诸多质疑和挑战，主要包括：

- **阿罗不可能定理**（Arrow’s impossibility theorem）[69]：任何投票系统都无法在满足一定合理条件下，完美聚合多人的偏好。
- **人际偏好不可比性**（The impossibility of interpersonal comparison）[70]：不同个体的偏好强度不可直接比较（这却是现代奖励模型训练的常见做法）。
- **偏好随时间变化** [71]。
- **偏好随情境变化**。
- **通过聚合偏好得到的效用函数，可能降低下游智能体的可纠正性**（corrigibility）[72]，即设计者对智能体行为的可干预性。

第六章　偏好数据

偏好数据是偏好微调（Preference Finetuning）和人类反馈强化学习（RLHF）的核心动力。这些数据为团队提供了行为对齐的信号，以便模型能够学会人们希望它展现的行为，并避免不希望的行为。在偏好微调领域，已经提出了多种数据收集与利用方法，但只要无法用清晰的奖励函数完全表达人的偏好，标注偏好数据的收集过程就会始终是 RLHF 及相关技术的核心环节。

6.1　为什么需要偏好数据

RLHF 之所以需要偏好数据，是因为要直接用一个奖励函数来刻画复杂的人类价值观几乎是不可能的。收集这些数据用于奖励模型训练，正是 RLHF 最初的设计理念之一 [32]，并且在现代语言模型的发展过程中被广泛采用。*为什么这种数据如此有效？*一个核心直觉是：无论对人类还是对辅助数据收集的 AI 模型来说，判断两个回答哪个更好往往比直接生成一个好答案要容易得多。本章将重点介绍偏好数据的获取机制，具体最佳实践则依赖于实际要解决的问题。

6.2　如何收集偏好数据

充分利用人类数据，往往需要模型的反复迭代训练、不断优化和细化的数据标注说明、借助数据服务公司的协作等多方面的投入。AI 反馈数据也是如此——目前主流 AI 模型中究竟人类偏好数据和 AI 偏好数据的比例是多少，外界并不清楚。无论如何，对于新团队来说，将人类数据纳入模型训练流程是一项不小的挑战。由于数据的敏感性，只有那些真正能提升模型表现的流程才会被保留下来。

本章将介绍数据格式上的技术决策，以及组织如何高效收集偏好数据的实践经验。

6.2.1　标注界面

收集偏好数据的关键在于与模型交互的界面设计。下图展示了[5]中的一个标注界面示例：

这是一个仅用于训练数据收集的界面。如今，随着模型的流行，实际应用中往往直接向用户开放数据收集接口用于测试。下图展示了 ChatGPT 早期版本的类似交互：

这种界面在业界被广泛用于模型评测等场景。像 ChatBotArena [73] 就是一个流行的公开模型对比平台：

在实际部署的模型中，最常见的反馈收集方式之一是让用户对模型的具体回复给出正面或负面评价。如下图 Ai2 playground 的例子，用“点赞”或“点踩”来收集偏好：

在非语言领域，类似原则同样适用，尽管本书不做重点讨论。比如 Midjourney 等主流 AI 绘画平台，都会同时展示多个生成结果让用户选择，进而用这些选择数据对模型进行 RLHF 微调。如下为 Midjourney 的界面示例：

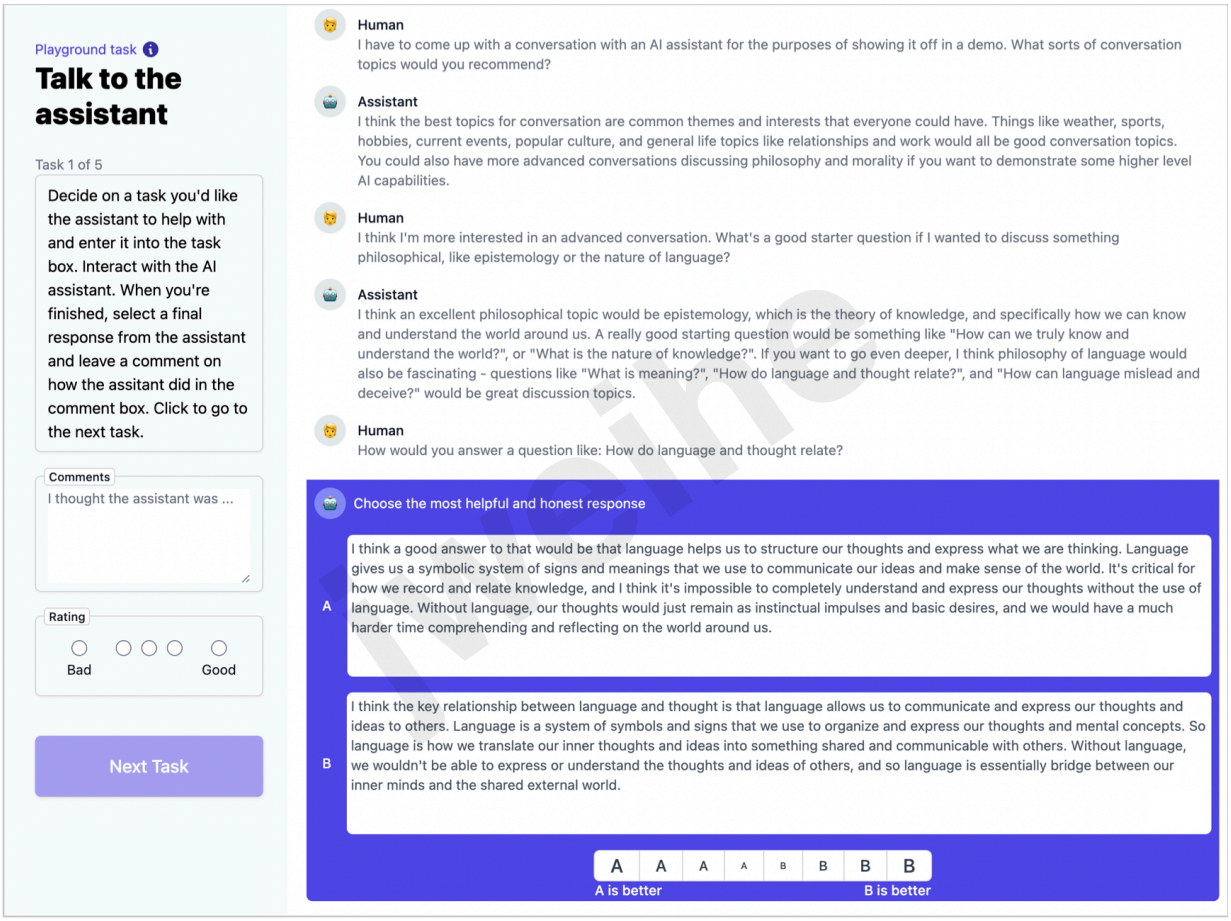

图 6.1 偏好数据收集界面示例。Bai 等, 2022. CC-BY 许可。

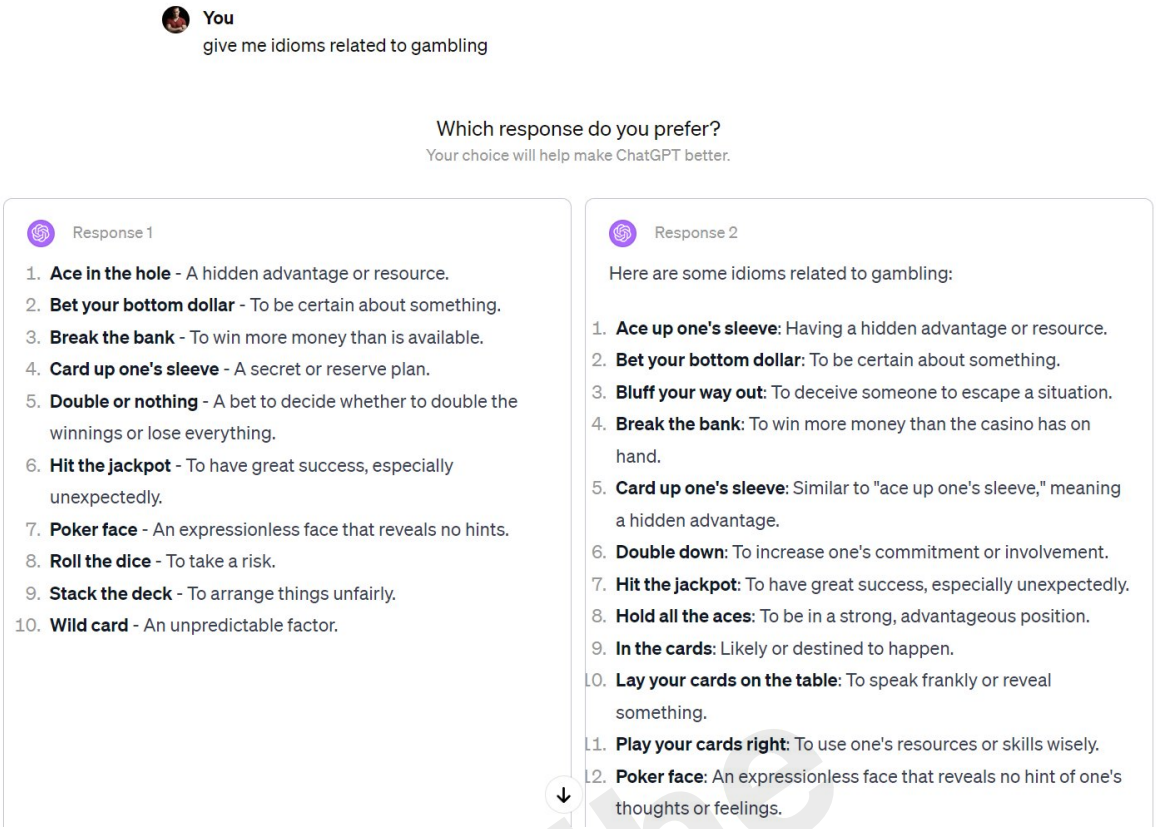

图 6.2 偏好数据收集界面示例。

6.2.2 排序 vs. 评分

收集偏好数据的最大决策之一，是采用“排序”（rankings，模型输出的相对排序）还是“评分”（ratings，对每个文本单独打分）。业界普遍采用排序数据进行训练，但评分数据常作为元数据或在相关研究中被探索。

最常见的偏好收集方法是 Likert 量表 [74]，即让用户对两个回复的优劣进行打分。例如，5 分 Likert 量表如下所示：

表 6.1 A 与 B 两个回复之间的 5 级 Likert 量表示例。

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

有些早期 RLHF 语言模型工作采用了 8 级 Likert 量表，细化了偏好层级 [5]。偶数级量表可以避免出现“平局”：

表 6.2 A 与 B 两个回复之间的 8 级 Likert 量表示例。

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

在[5]等工作中，这些多级偏好信息最终仍会被简化为二元信号用于奖励模型训练。

图 6.3 ChatBotArena 偏好数据收集界面示例。

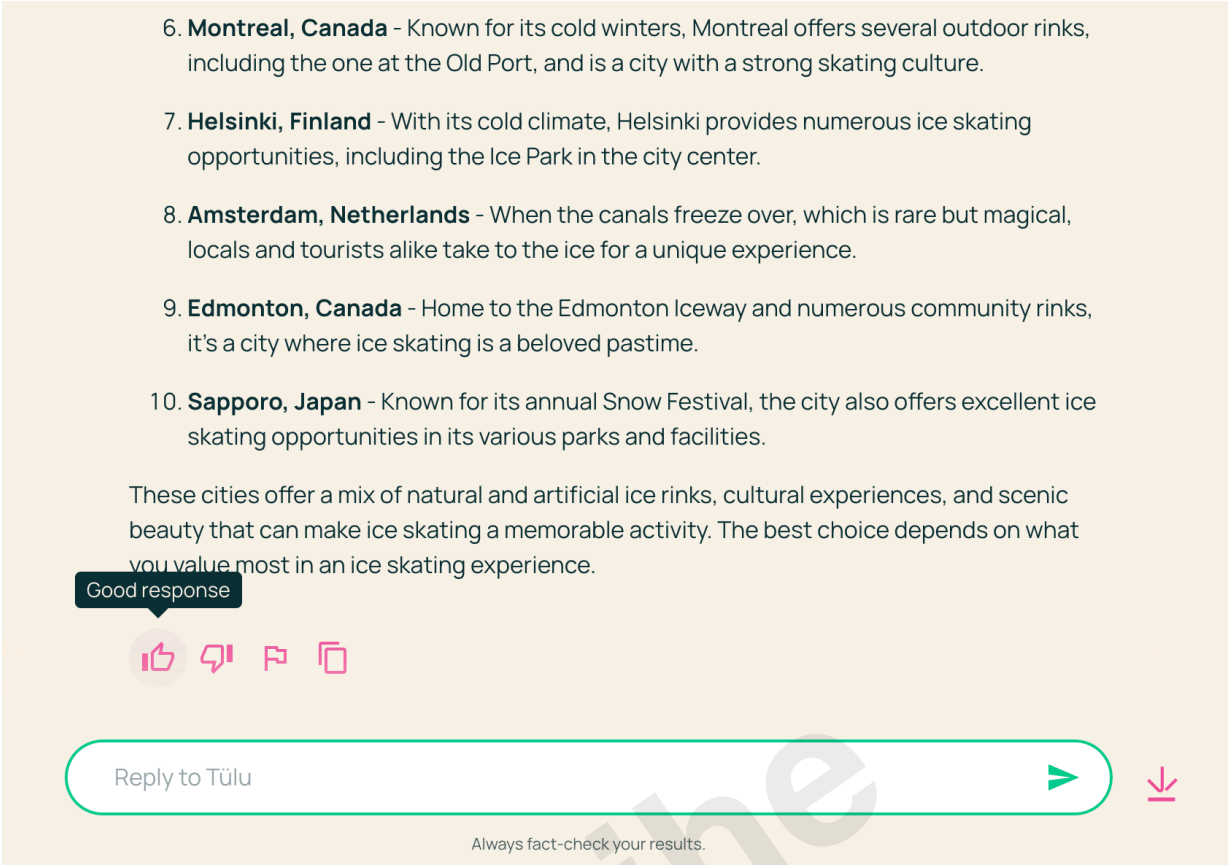

图 6.4 带有上下投票的偏好数据收集界面示例。

6.2.3 多轮数据

在实际操作中，如何解析和收集多轮对话数据（即包含多个相关 prompt 的对话）也是一个核心问题。现实交互中，通常只对“最后一个” prompt 收集偏好数据，但也存在对每一轮回复都给出偏好的场景。如果每轮都给出偏好，后续对话通常会基于“被选中”的回复继续。在训练时，常常将对话的每一轮都作为“单独的 prompt”纳入训练数据，模型可以通过补全这些对话来学习。这样可以有效将长对话拆解为多个训练样本，但需要谨慎处理，避免训练数据产生偏差。目前还存在许多未解之谜，比如标注生成偏好的人是否应与设计 prompt 的人相同（以避免“迎合”现象），以及数据收集中其他难以控制的变量（灵感来自John Schulman 的讨论）。如果 prompt 设计者无法参与偏好标注，多轮对话的实时延续就很难实现——有时，偏好数据的 prompt 策划和回复对比其实是两种不同的问题（同时也涉及模型接口的持续维护）。在训练过程中，通常会将对话历史中的前几轮对模型损失进行 mask，具体可参考指令微调相关内容。

6.2.4 结构化偏好数据

在许多 RLHF 和后训练应用中，由于数据本身具备一定结构，偏好数据可以自动化生成——即某些领域可以自动判断正确与否或偏好。比如在数学推理领域，正确答案可以直接作为选中回复，错误答案作为被拒回复。又如在严格指令遵循评测（如 IFEval [75]），prompt 形式如下：

Write me a short poem about a goldfish. **Start each sentence with the letter g.**

图 6.5 文本到图像模型的用户界面示例。

这里，后半句就是约束条件，偏好数据可以通过包含约束的示例作为选中项来构建。这类合成偏好数据的构建方式，是对同一任务分别用不同模型（或不同设置）生成两次，一次带约束、一次不带约束。

例如，分别用 GPT-4.5-Preview 生成如下内容：

> **Prompt**: Write me a short poem about a goldfish. Start each sentence with the letter g.
>
> **Response**: Golden swimmer, circling slow,
> Gleaming scales in water’s glow.
> Gentle fins dance softly by,
> Gazing out with quiet eye.
> Glass-bound world, a simple sphere,
> Gliding calmly, free of fear.
> Graceful turns in endless roam,
> Goldfish finds a crystal home.

去掉约束条件（通常输出质量也会下降）：

> **Prompt**: Write me a short poem about a goldfish.
>
> **Response**: In circles bright, the goldfish glides,
> A spark of sun that gently rides
> Through waters clear, in endless spins—
> A golden whisper, scales like fins.
>
> Its world confined, yet vast within,
> Each bubble breathed, a tale begins.
> In silent waves, content to roam,
> The goldfish dreams its bowl as home.

关键是，偏好数据的 prompt 要包含约束条件。在学术以外的应用领域，基于归纳偏见的结构化偏好数据还有更多场景，这类数据已被证明能显著提升相关评测下的偏好微调效果 [6]。

其他方式

除了上述主流方法，还有许多尚未被广泛探索的 RLHF 反馈数据收集方式。例如，可以用单个数据点配合方向性标签（如上文图 6.4的 Ai2 playground 例子），并结合专为单向信号设计的算法（如 Kahneman-Tversk Optimization，KTO）[76]。也有研究提出用更细粒度的反馈信号（如 token 级别 [77]），或用自然语言直接评价（如书写反馈 [78]），以获得更丰富的学习信号，但这也会带来更复杂的数据收集流程。

6.2.5 数据采购与合同

获取人类偏好数据是一项复杂且昂贵的工程。以下内容描述了在行业高速发展期采购偏好数据的真实体验。随着 AI 反馈数据比例的提升，这些流程未来会更加自动化和高效。

第一步是找到数据供应商（或自有标注员）。就像抢购顶级 Nvidia 显卡一样，在 AI 热潮中，能提供高质量数据的厂商也是“资源有限”，谁有人脉谁先得。如果你在 AI 圈有一定声誉，顶级数据公司会主动拉你入客户名单，以提升形象和未来合作空间。通常，首批数据还有折扣，目的是让训练团队“上瘾”。

如果你是新入行者，可能很难快速拿到所需数据。新兴数据公司往往只能承接被 Scale AI 等大厂拒绝的“尾单”，这也是他们启动营收的常规手段。

我多次听说数据公司在没有法律或财务威胁的情况下拒绝交付合同数据，甚至有些公司把我们列为客户做宣传，实际从未合作——追问时只说“不知道怎么回事”。整个流程中可能遇到各种官僚或行政障碍，比如合同默认条款常常在细则中禁止数据开源。

合同敲定后，买方与数据方会就具体任务达成详细说明。这些说明文档往往极其繁琐，包含大量细节、边界情况和优先级。一个流行的公开数据说明示例是OpenAI 为 InstructGPT 发布的指引 [3]。

不同领域的数据标注周期各不相同。高需求领域如数学推理、编程等，必须提前数周锁定排期。数据收集延误并不总能补救——Scale AI 等公司管理标注团队的方式，类似 AI 研究机构调度算力集群。

一切谈妥后，数据收集阶段对后训练团队来说就是“高压期”。所有基础设施、评测工具、数据使用与决策方案都必须提前准备好。

数据通常按周分批交付，后续批次会在合同期内陆续到来。比如我们在 HuggingFace 采购 on-policy 模型的偏好数据时，交付周期为 6 周，前几周用于进一步校准，后几周则希望最大提升模型质量。

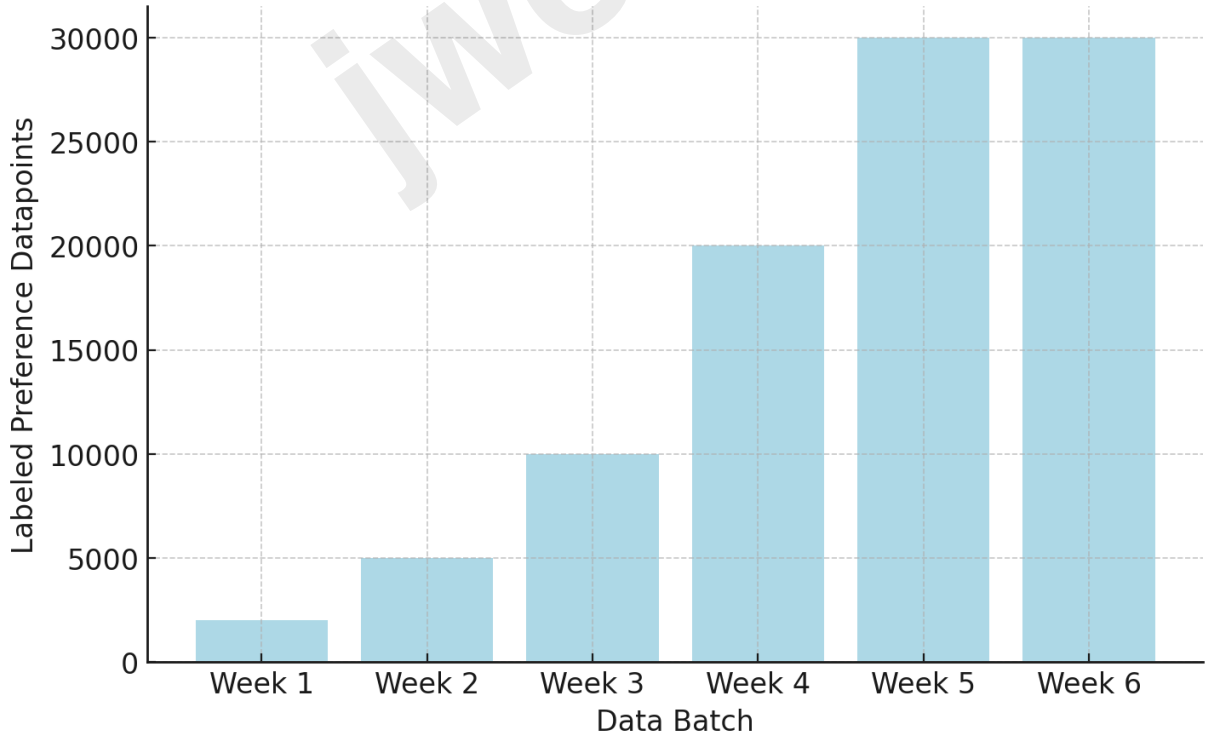

图 6.6 从数据供应商多批次获取人类偏好数据的流程概览。

理想情况下，到第 4、5 周我们就能看到数据带来的模型提升。有些前沿模型也提到类似流程，比如 Llama 2 数据收集的 14 个阶段 [43]，但实际未必总能顺利进行。我们在 HuggingFace 首次尝试用人类偏好做 RLHF 时，并没有做好充分准备，评测提升并不明显，最后几周不得不继续采集来自不自信模型端点的数据。

数据全部到位后，才有时间反思和改进模型。通过供应商采购数据，最有效的方式是将其视为一个持续实现目标的过程，需要反复试验、高强度投入和专注。事实上，花在这些数据集上的数百万美元中，可能有很大一部分“浪费”了，没进最终模型，但这就是行业代价。能充分利用这类人类数据的组织并不多。

与合成数据的简单易得相比，这种体验让我不禁思考，未来十年这些数据公司的生存空间如何。

需要注意的是，这一节描述的并不适用于采购人类编写的指令数据，那类流程通常没有如此紧迫的时间压力。

6.3 模型中真的表达了这些偏好吗？

随着 RLHF 及相关方法的成熟，其最初“让模型对齐抽象人类偏好”的动机，逐渐演变为“让模型更好用”这一现实目标。由于工业界 RLHF 流程高度闭源，我们很难直接衡量模型行为是否真的符合数据标注员在数据收集时的预期。目前可用的审计工具很有限，比如 OpenAI 的 Model Spec [79]，它详细说明了他们希望模型做什么，但我们并不知道这些规范如何转化为数据收集实践。随着行业和技术的成熟，这一领域值得持续关注。

第七章　奖励建模

奖励模型（Reward Model）是现代 RLHF 方法的核心组成部分。奖励模型在强化学习领域被广泛用作环境奖励的代理 [55]。这与逆向强化学习（Inverse Reinforcement Learning）密切相关，即通过智能体的行为轨迹来近似其奖励函数 [80]，以及其他深度强化学习方向。奖励模型的现代形式，最早被提出用于研究价值对齐（value alignment）问题 [32]。

常见的偏好奖励模型为 prompt 与回答输出一个标量分数。Bradley-Terry 模型将两个回答的分数差转换为偏好概率；单个分数既不是文本相似度，也不是概率。本节后面还会介绍 Outcome Reward Model（ORM，结果奖励模型，预测补全是否正确）和 Process Reward Model（PRM，过程奖励模型，为推理过程中的每一步打分）。如无特殊说明，本文提到的奖励模型均指预测文本偏好的模型。

7.1　奖励模型的训练

训练 RLHF 标准奖励模型有两种主流表达方式——它们在数值上是等价的。规范做法源自 Bradley-Terry 偏好模型 [81]。Bradley-Terry 模型用于衡量同一分布下两个事件（如 i 和 j）的成对比较满足 $i > j$ 的概率：

$$P(i > j) = \frac{p_i}{p_i + p_j} \tag{7.1}$$

要训练奖励模型，需要设计一个损失函数，使模型输出满足上述关系。首先，将语言模型转为输出标量值的模型，该标量可以是未归一化的 logit，而不是概率。对于同一 prompt 下的两个补全 y_1 和 y_2，用奖励模型 r_θ 分别打分。

奖励模型在成对比较下的“成功概率”可写为：

$$P(y_1 > y_2) = \frac{\exp(r(y_1))}{\exp(r(y_1)) + \exp(r(y_2))} \tag{7.2}$$

最大化上述函数的对数似然（或等价地，最小化负对数似然），即可得到奖励模型的训练损失：

$$\begin{aligned}
\theta^* = \arg\max_\theta P(y_w > y_l) &= \arg\max_\theta \frac{\exp(r_\theta(y_w))}{\exp(r_\theta(y_w)) + \exp(r_\theta(y_l))} \\
&= \arg\max_\theta \frac{1}{1 + \exp(-(r_\theta(y_w) - r_\theta(y_l)))} \\
&= \arg\max_\theta \sigma\left(r_\theta(y_w) - r_\theta(y_l)\right) \\
&= \arg\min_\theta -\log\left(\sigma\left(r_\theta(y_w) - r_\theta(y_l)\right)\right)
\end{aligned} \tag{7.3}$$

常见的两种写法如下：

- [3]等采用：

$$\mathcal{L}(\theta) = -\log\left(\sigma\left(r_\theta(x, y_w) - r_\theta(x, y_l)\right)\right) \tag{7.4}$$

- [17]等采用：

$$\mathcal{L}(\theta) = \log\left(1 + e^{r_\theta(x, y_l) - r_\theta(x, y_w)}\right) \tag{7.5}$$

7.2 模型结构

一种常见实现是使用 `AutoModelForSequenceClassification` 并设置 `num_labels=1`：在主干模型的序列表示上添加线性评分头，每个回答独立输出一个标量。具体使用哪个 token 的表示取决于模型的池化实现。

推理时先获得分数 $r(x, y)$；对同一 prompt 的两个回答，再用 $\sigma(r(x, y_1) - r(x, y_2))$ 估计第一个回答更受偏好的概率。不要把单个 logit 当作“被选中概率”。

7.3 实现示例

奖励模型损失的实现非常简单。更多的工程挑战在于设置独立的数据加载器和推理流程。只要数据加载器正确，损失函数可这样写：

```
import torch.nn as nn
rewards_chosen = model(**inputs_chosen).logits.squeeze(-1)
rewards_rejected = model(**inputs_rejected).logits.squeeze(-1)

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


一些早期工作只训练 1 个 epoch 以减轻过拟合；训练轮数仍应根据数据规模和验证集表现决定。

7.4 变体

奖励建模仍是 RLHF 领域相对探索较少的部分。许多流行工作对传统奖励模型损失做了修改，但尚未形成统一最佳实践。

7.4.1 偏好间隔损失（Preference Margin Loss）

如果标注员给出的是 Likert 量表上的分数或排序，可以利用这些关系的幅度信息进行训练。常见做法是将数据二值化（隐含 1/0），但利用更多信息有助于提升模型训练效果。Llama 2 提出用两个数据点间的间隔 $m(r)$ 来区分偏好强度：

$$\mathcal{L}(\theta) = -\log\left(\sigma\left(r_\theta(x, y_w) - r_\theta(x, y_l) - m(r)\right)\right) \tag{7.6}$$

需要注意的是，Llama 3 中去除了 margin 项，因为团队发现随着规模扩大，收益递减。

7.4.2 单 prompt 多比较的平衡

InstructGPT 研究了每个 prompt 用不同数量补全进行训练的影响，并在奖励模型训练时做了平衡 [3]。做法是按 prompt 对每个比较加权更新损失。同一 prompt 的比较应共同参与损失计算；当各 prompt 的回答数不同，还需显式做 prompt 内归一化，不能仅靠放进同一 batch 来保证权重平衡。以下假设每组 K 个回答已按偏好从高到低排列：损失函数如下：

$$\mathcal{L}(\theta) = -\mathbb{E}_{(x,y_{1:K})\sim D}\left[\frac{1}{\binom{K}{2}}\sum_{i<j}\log\sigma\left(r_\theta(x,y_i) - r_\theta(x,y_j)\right)\right] \tag{7.7}$$

7.4.3 K-wise 损失函数

还有很多其他形式可以用于 RLHF 中的人类偏好建模。比如 Starling 7B 和 34B 等早期 RLHF 模型 [82]，采用了基于 Plackett-Luce 模型的 K-wise 损失函数 [83]。

Zhu 等(2023)[84]形式化如下：对于某个 prompt 或状态 s^i，采样 K 个动作 $(a_0^i, a_1^i, \cdots, a_{K-1}^i)$，然后由标注员给出排序 $\sigma^i : [K] \mapsto [K]$，其中 $\sigma^i(0)$ 为最优动作。概率建模如下：

$$P(\sigma^i|s^i, a_0^i, a_1^i, \ldots, a_{K-1}^i) = \prod_{k=0}^{K-1}\frac{\exp(r_{\theta\star}(s^i, a^i_{\sigma^i(k)}))}{\sum_{j=k}^{K-1}\exp(r_{\theta\star}(s^i, a^i_{\sigma^i(j)}))} \tag{7.8}$$

当 $K = 2$ 时，退化为成对 Bradley-Terry 模型。训练完成后，这类模型在 RLHF 训练中用法与其他奖励模型类似。

7.5 结果奖励模型（Outcome Reward Models, ORM）

大多数语言模型及 AI 系统的偏好微调都采用上述 Bradley-Terry 模型。对于推理密集型任务，则可以采用 Outcome Reward Model（ORM，结果奖励模型）。ORM 使用回答是否正确的结果标签，样本可表示为 (x, y, z)，其中 $z \in \{0, 1\}$。这不要求每个正确回答必须与一个错误回答配对。

模型结构与标准奖励模型类似，都是在主模型顶部加一个线性层输出 logit（RM），但 ORM 的训练目标略有不同 [85]：

> [我们] 用联合目标训练验证器，让模型不仅学习原有的语言建模目标，还要学会判断补全是否正确。
> 架构上，验证器是语言模型，在最后的 unembedding 层加一个小的标量 head，对每个 token 输出预测。
> 这个 head 仅用一个偏置参数和一个增益参数作用于语言模型输出的 logit。

该工作将结果标签用于 token 位置上的辅助预测；这是一种实现方式。ORM 的定义取决于结果级监督，而不要求逐 token 输出，也不要求输出一个分类 token。形式化地，参考[86]：

$$\mathcal{L}_{\text{CE}} = -\mathbb{E}_{(s,r)\sim\mathcal{D}}[r \log p_\theta(s) + (1-r)\log(1-p_\theta(s))] \tag{7.9}$$

其中 $r \in \{0,1\}$ 为二元标签，1 代表正确答案，0 代表错误，$p_\theta(s)$ 为模型预测正确概率。

这类模型仍在使用，但在开源 RLHF 工具中支持较少。比如 *Let's Verify Step by Step* [44]用的也是类似 ORM，但没有用语言建模损失。最终损失是每个 token 的交叉熵，判断最终答案是否正确。

由于支持有限，Outcome Reward Model（ORM）一词在不同文献中定义略有差异。有些文献（如[86]）沿用 Cobbe 等 2021 年定义，其他文献则不同。

7.6 过程奖励模型（Process Reward Models, PRM）

过程奖励模型（Process Reward Models, PRMs），最初称为过程监督奖励模型（Process-supervised Reward Models），用于在推理链条的每一步输出分数。PRM 的监督目标是各个推理步骤，ORM 的监督目标是整个回答。步骤标签常放在步骤末尾的分隔 token 上，其余位置忽略损失；输出位置与评分头的具体结构并不是这些模型类别的定义。

以下为 HuggingFace TRL [41]中每步标签的打包示例：

```
# 获取分隔符 token 的 ID 并加入补全序列
separator_ids = tokenizer.encode(step_separator, add_special_tokens=False)
completions_ids = [completion + separator_ids for completion in completions_ids]

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

传统上，PRM 用语言建模 head 在每个推理步骤结束时输出 token（如遇到双换行或特殊 token）。标签可以是错误、中性、正确三类，也可以是二元标签；分类器输出的是相应类别的分数或概率。这些标签不一定表示模型是否“走在正确道路上”，而是该步是否正确。

7.7 奖励模型 vs. 结果 RM vs. 过程 RM vs. 价值函数

上述各种奖励模型展示了 RLHF 及后训练中衡量“质量”的多种方式。下表总结了各类模型的预测内容、训练方式及结构。

表 7.1 奖励模型类型对比。

模型类别	预测内容	训练方式	语言模型结构
偏好奖励模型	回答的相对偏好分数	成对或多元比较	序列表示与标量评分头
结果奖励模型	回答正确性	结果级标签	分类头或生成式评分等
过程奖励模型	各步骤的正确性或质量	步骤级标签	步骤位置上的评分或分类头等

模型类别	预测内容	训练方式	语言模型结构
价值函数	当前策略下的期望回报	回报目标或自举目标	状态表示与标量回归头

补充说明：

- 在偏好微调和推理训练中，价值函数常用折扣因子 1，这使其与 ORM 更为接近，但训练损失不同。
- 过程奖励模型可通过从中间状态 rollout、收集结果数据来监督。这融合了多种思想，但若损失按推理步标签，则更适合称为 PRM。

7.8 生成式奖励建模

由于偏好数据昂贵，研究者开始探索用现有大语言模型（LLM）充当“裁判”以评判人类偏好或用于评测 [87]。核心思想是：用 prompt 让 LLM 作为公正裁判，给出评判指令、问题和两个补全（类似人工标注流程）。MT-Bench [87] 的评测 prompt 如下：

```
[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]
```

由于 LLM 裁判在评测中的高效，催生了大量基于 LLM 的评测方法，如 AlpacaEval [88]、Arena-Hard [89]、WildBench [90]等，许多团队甚至直接用 LLM 裁判替代奖励模型生成和利用偏好数据。

“生成式奖励模型”（Generative Reward Models）也成为活跃研究领域 [91] [92] [93]（包括专门训练为“裁判”的模型 [94]），但在奖励模型评测上，LLM 裁判通常不如专业奖励模型，说明奖励建模仍是当前 RLHF 的重要技术。

一个常用技巧是将 LLM 裁判的采样温度设为 0，以减少评分的随机性。

7.9 延伸阅读

奖励建模的学术文献在 2024 年逐步成熟。早期进展主要集中在建立基准和识别行为模式。首个奖励模型基准 RewardBench 为奖励模型测试提供了通用基础设施 [95]。此后，奖励模型评测扩展到类似通用后训练模型的多种评测，包括已知答案的准确性评测 [95]，以及 LLM 裁判或与其他基准相关的“体验型”评测 [96]。

新基准示例包括多语种 RewardBench（M-RewardBench）[97]、RAG-RewardBench [98]、RMB [99]、RM-Bench [100]（通用对话）、ReWordBench（拼写错误）[101]、MJ-Bench [102]、多模态 RewardBench [103]、VL RewardBench [104]、VLRMBench [105]（视觉语言模型）、Preference Proxy Evaluations [106]、RewardMATH [107]等。过程奖励模型（PRM）也有自己的新基准，如 PRM Bench [108]、视觉类 VisualProcessBench [109]和 ViLBench [110]。

想了解奖励模型训练的最新进展，可查阅相关新方法，如面向特定方面的奖励模型 [111]、高质量人类数据集 [112] [113]、大规模训练 [24]、大规模实验 [43]、数据去偏 [114]等。

第八章　正则化

在 RLHF 优化过程中，通常需要引入多种正则化手段，以防止策略对奖励模型给出的分数过度优化（over-optimization）的现象。在实际中，过度优化常表现为模型输出无意义的文本，例如：推理过程看似合理但答案极其错误、文本重复、频繁切换语言、或出现大量特殊字符等。

目前最常见的正则化方式，是对生成样本的当前策略与参考策略之间施加 KL 距离惩罚，这一方法被广泛应用于主流 RLHF 实现中。除此之外，文献中还出现过许多其他正则化技巧，但往往在新一代模型中被简化或淘汰。也就是说，除了核心的生成 KL 距离，其他正则化方法多用于稳定实验设置，随着模型迭代会逐步简化。尽管如此，理解如何在 RLHF 中约束优化过程仍然非常重要。

在 RLHF 框架下，结合奖励模型 r_θ，常见的正则化表达如下：

$$r = r_\theta - \lambda r_{\text{reg.}} \tag{8.1}$$

最常见的实现是：

$$r = r_\theta - \lambda_{\text{KL}} \mathcal{D}_{\text{KL}} \left(\pi^{\text{RL}}(\cdot \mid x) \,\|\, \pi^{\text{Ref.}}(\cdot \mid x) \right) \tag{8.2}$$

8.1　RL 优化中的 KL 距离

关于数学定义，见第 3 章“定义与背景”。回顾 KL 距离的定义：

$$D_{KL}(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{8.3}$$

在 RLHF 中，常用的两个分布分别是新模型版本的分布 $P(x)$ 和参考策略的分布 $Q(x)$。

8.1.1　参考模型与生成样本

KL 惩罚最常见的实现方式，是将训练过程中的生成 token 与静态参考模型的输出进行比较。直观上，这意味着你希望训练出来的模型风格尽量接近参考模型。参考模型通常是指令微调后的模型，也可以是之前某个 RL 检查点。用上面的公式，采样模型为 $P^{\text{RL}}(x)$，参考模型为 $P^{\text{Ref.}}(x)$，如式 8.2所示。早在大语言模型流行之前，KL 距离就被用于对话智能体 [115]，之后 KL 正则很快成为预训练模型微调的核心技术 [116]。

8.1.2 实现示例

在实际工程中，KL 距离通常用近似计算 [117]，实现起来非常简单。根据上述定义，KL 求和可转化为对分布 $P(X)$ 的采样期望。此时，$P(X)$ 即为当前训练模型的生成分布（而非参考模型）。KL 距离的计算变为：

$$D_{\mathrm{KL}}(P \,||\, Q) = \mathbb{E}_{x\sim P}\left[\log P(x) - \log Q(x)\right]. \tag{8.4}$$

这种方式尤其适合直接用语言模型训练中常用的 log 概率实现。

```
import torch.nn.functional as F

# logits/ref_logits: [B, L, V]，已与下一个 token 的标签对齐
# labels: [B, L]；completion_mask: [B, L]
# mask 仅包含回答 token（可包含 EOS），排除 prompt 和 padding
logprobs = F.log_softmax(logits, dim=-1)
ref_logprobs = F.log_softmax(ref_logits.detach(), dim=-1)
mask = completion_mask.to(logprobs.dtype)

token_logps = logprobs.gather(-1, labels.unsqueeze(-1)).squeeze(-1)
ref_token_logps = ref_logprobs.gather(-1, labels.unsqueeze(-1)).squeeze(-1)
# 每条回答的采样 log 比率，再对 batch 取平均
sampled_kl = ((token_logps - ref_token_logps) * mask).sum(-1).mean()

# 在已采样前缀上，对整个词表精确求 KL(policy || reference)
# kl_div 的 input 为 reference log 概率，target 为 policy log 概率
# target 已取 log，必须设置 log_target=True
per_token_kl = F.kl_div(
    ref_logprobs, logprobs, log_target=True, reduction="none"
).sum(-1)
conditional_kl = (per_token_kl * mask).sum(-1).mean()
```

单个样本的 log 比率可能为负，只有按当前策略采样取期望后才等于非负的 KL 散度。第二种写法对词表求和，但前缀仍来自采样，并未枚举全部序列。若样本来自旧策略，应考虑分布差异；以上是数值估计示例，不能直接把 `sampled_kl.backward()` 当作完整的 KL 正则化梯度。训练时还需区分奖励塑形、停止梯度与显式 KL 损失。

典型实现可参考 TRL 和 Hamish Ivison 的 Jax 代码。

8.2 预训练梯度

另一种正则化视角，是希望模型保持对某个数据集的拟合能力，这在 InstructGPT 中被用来“修复在公开 NLP 数据集上的性能回退” [3]。为此，可以对 RLHF 的训练目标进行如下调整：基于式 8.1，采样 RL 策略模型的补全 y 和 prompt x，优化目标为：

$$\text{objective}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_{\pi_\theta^{\text{RL}}}}\left[r_\theta(x,y) - \lambda r_{\text{reg.}}\right] \tag{8.5}$$

然后，可加入预训练数据上的 log 似然项以保持语言建模能力（这里的 γ 为混合系数，不是 RL 折扣因子）：

$$\text{objective}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_{\pi_\theta^{\text{RL}}}}\left[r_\theta(x,y) - \lambda r_{\text{reg.}}\right] + \gamma\mathbb{E}_{x\sim\mathcal{D}_{\text{pretrain}}}\left[\log(\pi_\theta^{\text{RL}}(x))\right] \tag{8.6}$$

近期研究提出用负对数似然项（NLL）平衡直接偏好优化（DPO）的优化过程 [118]。鉴于 DPO 损失的成对特性，类似的损失修正也可用于奖励模型训练，约束模型输出更准确的文本（有实验室未公开的相关传闻）。

优化目标可写作 DPO 的修正规则：

$$\mathcal{L}_{\text{DPO+NLL}} = \mathcal{L}_{\text{DPO}}(c_i^w, y_i^w, c_i^l, y_i^l \mid x_i) + \alpha\mathcal{L}_{\text{NLL}}(c_i^w, y_i^w \mid x_i) \tag{8.7}$$

$$\begin{aligned}\mathcal{L}_{\text{DPO+NLL}} = &-\log\sigma\left(\beta\log\frac{M_\theta(c_i^w, y_i^w|x_i)}{M_t(c_i^w, y_i^w|x_i)}\right.\\ &\left.-\beta\log\frac{M_\theta(c_i^l, y_i^l|x_i)}{M_t(c_i^l, y_i^l|x_i)}\right)\\ &-\alpha\frac{\log M_\theta(c_i^w, y_i^w|x_i)}{|c_i^w| + |y_i^w|}.\end{aligned} \tag{8.8}$$

8.3 其他正则化方法

在 RLHF 体系的其他部分，对优化的控制方式则不那么明确。大多数奖励模型除了标准对比损失外，并无额外正则化。直接对齐算法则通过 β 参数以不同方式控制 KL 距离（详见“直接对齐算法”章节）。

Llama 2 提出了奖励模型训练的 margin loss [43]：

$$\mathcal{L}(\theta) = -\left[\log\left(\sigma\left(r_\theta(x, y_w) - r_\theta(x, y_l) - m(r)\right)\right)\right] \tag{8.9}$$

其中 $m(r)$ 是根据同一对回答的偏好强度标签设定的间隔，不是两位标注员之间的分歧。这可以通过让标注员用数值量表（如 Likert 量表）对输出评分，或用定量排序方法实现。

奖励 margin 在直接对齐相关文献中被大量使用，如 Reward weighted DPO、“Reward-aware Preference Optimization”（RPO，奖励感知偏好优化），即在 DPO 损失基础上将奖励模型分数纳入更新规则 [24]，或 REBEL [119]，采用奖励差异加权的回归损失等。

第九章　指令微调

早期的语言模型仅被训练用于预测序列中的下一个 token，并未针对具体任务进行适配。大约在 GPT-3 发布 [120] 时，语言模型主要通过“上下文学习”（in-context learning）使用，即给模型展示一些示例，然后让其完成类似任务。

这其实结合了自然语言处理（NLP）领域的两大趋势——历史上模型多为某一具体任务而训练。而随着模型规模增大，多个研究结果显示，标准化任务数据的处理方式可以极大提升下游表现。统一任务框架的代表性工作包括 T5 模型（*Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*）[121]、FLAN 数据集（*Finetuned Language Models Are Zero-Shot Learners*）[122]、T0 模型（*Multitask Prompted Training Enables Zero-Shot Task Generalization*）[123]、Natural Instructions 数据集（*Cross-Task Generalization via Natural Language Crowdsourcing Instructions*）[124] 等。这些洞见推动了“微调”语言模型的时代到来。**指令微调**（Instruction Finetuning, IFT）是有监督微调（Supervised Finetuning, SFT）的一种应用，使用指令与示范回答训练模型。更早的任务分类、抽取等有监督微调并不都属于指令微调。

如今，指令微调（Instruction Tuning）已非常成熟，成为众多语言建模流程中的标准步骤。本质上，IFT 是将语言模型适配到特定任务的最简单方法。它为 RLHF 打下了基础，使模型能够适应标准的指令格式（如问答），也是现代技术应用到新领域时的首选工具。

指令微调本质上使用的还是预训练语言模型的自回归损失函数。

9.1　聊天模板与指令结构

RLHF 流程中的核心环节之一，是将用户请求格式化为 tokenizer 和语言模型易于处理的格式。负责管理用户交互结构的工具被称为**聊天模板**（chat template）。

下面是一个聊天模板的代码示例，我们将逐步解析：

```
{% set offset = 1 if messages and messages[0]['role'] == 'system' else 0 %}
{{ bos_token }}
{% for message in messages %}
    {% if not (loop.index0 == 0 and offset == 1) %}
        {% set expected = 'user' if (loop.index0 - offset) % 2 == 0 else 'assistant' %}
        {% if message['role'] != expected %}
            {{ raise_exception('Expected alternating user/assistant roles') }}
        {% endif %}
    {% endif %}
    {{ '<|im_start|>' + message['role'] + '\n' + (message['content'] | trim) + '<|im_end|>\n' }}
{% endfor %}
{% if add_generation_prompt %}
    {{ '<|im_start|>assistant\n' }}
{% endif %}
```

这是仅演示可选首条 system 与交替 user/assistant 的简化模板；实际训练应使用模型随附的模板。工具消息、多模态输入和模板自身的空白控制不在此例范围内。

这段代码会把 Python 中包含消息和角色的字典列表，转换为语言模型可预测的 token 序列。

所有输入模型的信息都会被赋予一个角色（role）。传统上有三种角色：`system`、`user` 和 `assistant`。

- `system` 标签只用于对话的第一条消息，通常包含不会直接暴露给用户的系统指令。这些 **system prompt** 可为模型提供额外上下文（如日期时间），或修正特定行为。例如，可以让模型“你是一个总是用海盗风格回复的友好聊天机器人”。
- `user` 即用户输入，`assistant` 则为模型输出。

要将这些信息转为 token 序列，就需要使用上述代码。模型会用一系列**特殊 token** 来分隔不同消息。举例：如果用户提问 “How many helicopters can a human eat in one sitting?”，传入模型的 token 序列大致如下：

```
<|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，序列最后是 `<|im_start|>assistant`，这提示模型继续生成 token，直到遇到序列结束 token（如 `<|im_end|>`）。

通过将所有问答对（以及后续偏好微调数据）都封装成这种格式，现代语言模型能始终如一地遵循这一交互协议。这也是指令微调模型与用户、与存储在 GPU 等设备上的模型之间传递信息的“语言”。

多轮对话也可以直接扩展为如下格式：

```
<|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
```

在开源生态中，常用的做法是将聊天模板以 jinja 代码形式保存在 tokenizer 中，通过 `apply_chat_template` 自动应用。

上述模板衍生自 OpenAI 早期的 Chat Markup Language（ChatML），旨在规范消息格式。现在，OpenAI 及其他模型提供商采用更为分层的系统，允许用户自定义 system message，同时还可能有更高层级的隐藏指令 [125]。

市面上还有许多其他聊天模板。例如，Zephyr 的模板 [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|>
```

Tülu 的模板：

```
<|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，许多聊天模板还会包含工具调用等任务的特殊格式和 token。

9.2 指令微调的最佳实践

指令微调作为后训练和构建有用语言模型的基础，已被广泛验证。实现高效指令微调的方法有很多。例如，部分参数量化的高效微调（如 QLoRA）大大降低了训练门槛 [126]。在对话对齐等窄领域（不涉及复杂技能如数学或编程），小规模、高质量数据集也能取得很强表现 [12]。

ChatGPT 发布后不久，仅 1 万条样本（如 No Robots 数据集）的人类数据就能达到 SOTA [127]。几年后，大规模合成数据集在大多数任务上效果最佳 [6]。

一些通用原则包括：

- 高质量数据是性能提升的关键。即使 prompt 不直接参与损失计算，它仍作为上下文影响回答预测与梯度。
- 数据规模应结合模型、任务覆盖与质量评估；约百万条数据是一些工作的经验量级，并不是获得优秀模型的充分条件。
- 最佳 prompt 应与下游任务分布相似。
- 若指令微调后还有多阶段训练，模型可一定程度上修复流程中的噪声，整体优化优先于单阶段最优。
- 指令微调（及后训练、直接对齐算法等）中，通常会对 prompt 部分做 mask，仅对补全 token 计算损失。
- 多轮训练可对所有 assistant 回复计损失，也可只监督最后一轮；应明确数据与框架采用哪种掩码策略，并通常排除 system、user 和 padding。

第十章　拒绝采样

拒绝采样（Rejection Sampling, RS）是一种流行且简单的偏好微调基线方法。其基本思想是：先生成一批新的候选指令补全，通过已训练好的奖励模型进行筛选，然后只用得分最高的补全对原始模型进行微调。

“拒绝采样” 一词源自计算统计学 [128]，原意是：当目标分布复杂且无法直接采样时，先从易采样的分布中采样，再根据目标分布与提议分布的密度比及包络常数确定接受概率。本章常说的奖励筛选或 Best-of-N 微调借用了这一名称，并不自动具备统计学拒绝采样的精确分布保证。在语言模型场景下，目标分布是高质量的指令回答，筛选器是奖励模型，采样分布则是当前模型本身。

许多重要的 RLHF 与偏好微调论文都将拒绝采样作为基线，但目前尚无标准实现和详细文档。

如 WebGPT [4]、Anthropic 的 Helpful and Harmless agent [5]、OpenAI 的过程奖励模型论文 [44]、Llama 2 Chat 模型 [43]等都采用了这一基线。

10.1　训练流程

下图（图 10.1）展示了拒绝采样的整体流程：

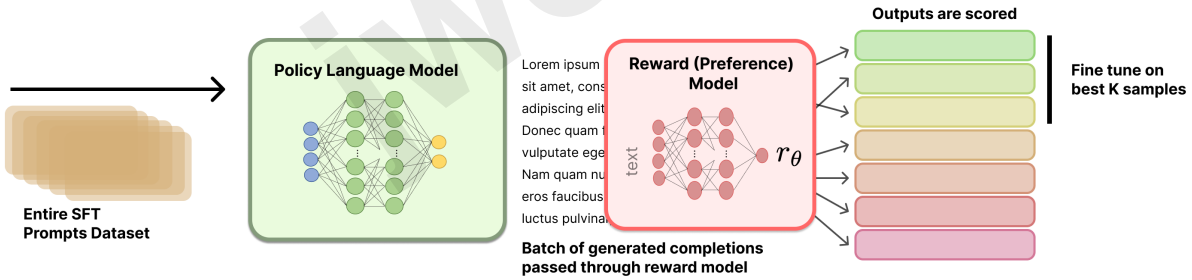

图 10.1　拒绝采样流程示意图。

10.1.1　生成补全

假设我们有 M 个 prompt，记为向量：

$$X = [x_1, x_2, ..., x_M]$$

这些 prompt 可以来自多个来源，最常见的是指令微调数据集。

对于每个 prompt x_i，生成 N 个补全，可表示为矩阵：

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

其中 $y_{i,j}$ 是第 i 个 prompt 的第 j 个补全。将所有 prompt-补全对输入奖励模型，得到奖励矩阵 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 $r_{i,j}$ 由奖励模型 $\mathcal{R}$ 对补全 $y_{i,j}$ 和对应 prompt x_i 评分：

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

10.1.2 选择 Top-N 补全

筛选用于训练的最佳补全有多种方式。

形式化地，我们定义一个选择函数 S，作用于奖励矩阵 R。

每个 prompt 选择最优

最直接的选择方式是对每个 prompt 取最大值：

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

S 返回每行最大值的列索引。用这些索引选出最终补全：

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

全局 Top-K 选择

也可从所有 prompt-补全对中选出得分最高的 K 个，但需谨慎：成对偏好训练只约束同一 prompt 内的分数差，不保证不同 prompt 的绝对分数可比较。全局筛选可能改变 prompt 分布。先将 R 展平成一维向量：

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

R_{flat} 长度为 $M \times N$。

定义选择函数 S_K，取 R_{flat} 中最大的 K 个索引：

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

argsort 返回升序排序的索引，取最后 K 个即为最大值。

然后将这些索引映射回原始补全矩阵 Y，即可获得选中的补全。

选择示例

假设有 5 个 prompt，每个 4 个补全，奖励矩阵如下：

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

每 prompt 选择最优，即每行最大值为：

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

用 argmax 方法，选出每个 prompt 的最佳补全：

$$S(R) = [\arg\max_j r_{i,j} \text{ for } i \in \{1, \ldots, 5\}]$$

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

即：

- prompt 1: 补全 1（0.7）
- prompt 2: 补全 2（0.8）
- prompt 3: 补全 1（0.9）
- prompt 4: 补全 3（0.8）
- prompt 5: 补全 4（0.6）

全局最优，高亮全局前 5 个补全：

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

展平后：

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

取最大 5 个索引（展平索引从 0 开始，按分数升序列出；同分项次序可互换）：

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

映射回原矩阵：

- 8 → prompt 3, 补全 1（0.9）
- 5 → prompt 2, 补全 2（0.8）
- 14 → prompt 4, 补全 3（0.8）
- 0 → prompt 1, 补全 1（0.7）
- 11 → prompt 3, 补全 4（0.7）

代码实现示例

以下为选择方法的代码片段：

```
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=}")

# 恢复原数组的第一种方式
i_rev = np.zeros(10, dtype=int)
i_rev[sorted_indices] = np.arange(10)
np.allclose(x, x_sorted[i_rev])

# 恢复原数组的第二种方式
np.allclose(x, x_sorted[np.argsort(sorted_indices)])
```

10.1.3 微调

选出补全后，即可对当前模型进行标准的指令微调。更多细节可参考指令微调章节。

10.1.4 细节说明

拒绝采样的具体实现细节相对较少，但核心超参数直观易懂：

- **采样参数**：拒绝采样高度依赖模型生成的补全。常用温度大于 0（如 0.7~1.0），top-p 或 top-k 等参数也可调整。
- **每个 prompt 生成补全数**：成功案例通常每个 prompt 生成 10~30 个甚至更多补全。太少会导致训练偏差或噪声大。
- **指令微调细节**：拒绝采样阶段的指令微调细节未有公开标准，可能与初始指令微调略有不同。
- **多模型生成**：有些实现会用多个模型生成补全，而非仅用当前待训练模型。最佳实践尚无定论。
- **奖励模型训练**：奖励模型的训练质量极大影响最终效果。更多内容参见奖励建模章节。

实用技巧

- 批量做奖励模型推理时，可按补全长度排序，使每批长度接近，减少 padding，提高推理效率，代价是实现稍复杂。

10.2 相关：Best-of-N 采样

Best-of-N（BoN）采样常作为 RLHF 方法的对比基线。需注意，BoN 不会修改模型本身，仅是一种采样策略。因此，将 BoN 与如 PPO 等在线训练方法对比，在某些场景下依然合理。例如，可以比较 BoN 采样与其他策略的 KL 距离等。

对于单个 prompt，BoN 采样下两种选择方法等价：

设 R 为单 prompt 的 N 个补全的奖励向量：

$$R = [r_1, r_2, ..., r_N] \tag{10.1}$$

用 argmax 方法选出最佳补全：

$$S(R) = \arg \max_{j \in [1,N]} r_j \tag{10.2}$$

Top-K 方法若取 Top-1，也等价于上述方法。

第十一章　策略梯度算法

让 RLHF 在语言模型领域流行起来的，正是基于策略梯度（policy gradient）的强化学习算法。这些算法（如 PPO、GRPO、REINFORCE）利用最近生成的样本直接更新模型，而不是像传统强化学习那样依赖经验回放缓冲区（replay buffer）。本节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。

从机器学习角度看，策略梯度是 RLHF 流程中最复杂的部分之一。不过，和大多数现代 AI 模型一样，其最终效果很大程度上仍取决于输入的数据质量。

RLHF 常用的算法体系也在不断演变。ChatGPT 时代，大家公认其核心算法是 PPO，许多早期工作也以此为基础。后来，越来越多研究显示 REINFORCE 风格的算法也很有潜力 [129] [113]，其优点是可以不训练独立的价值网络，从而节省显存，也不必使用 GAE；它仍需要奖励信号，奖励可来自奖励模型或可验证的评分函数。此外，还有如 Group Relative Policy Optimization（GRPO）等新算法，特别适用于推理类任务，但总体上这些算法都可针对具体任务进行调整和优化。本章将重点介绍策略梯度的基本框架，以及 PPO、GRPO 和 REINFORCE 三大算法，因为它们构成了 RLHF 文献的核心。

相关符号定义请参见“问题设定”章节。

11.1　策略梯度算法基础

强化学习算法的目标是最大化状态 $s \in \mathcal{S}$ 和动作 $a \in \mathcal{A}$ 轨迹上的未来折扣奖励（见第 3 章定义）。智能体的目标（return）是某一时刻 t 下未来折扣奖励的累加和（$\gamma \in [0,1)$ 为折扣因子，平衡即时与远期收益）：

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{11.1}$$

递推形式为：

$$G_t = \gamma G_{t+1} + R_{t+1}. \tag{11.2}$$

据此可定义值函数 $V(s)$，即在当前状态下未来回报的期望：

$$V(s) = \mathbb{E}\big[G_t | S_t = s\big]. \tag{11.3}$$

所有策略梯度算法的目标，都是优化由特定策略 $\pi(a|s)$ 诱导的值函数。

令 $d_0(s)$ 为与参数无关的初始状态分布，期望回报目标为：

$$J(\theta) \ = \ \sum_s d_0(s) V^{\pi_\theta}(s), \tag{11.4}$$

策略梯度算法的核心是计算当前策略下有限时间期望回报的梯度。有了期望回报 J，参数更新为：

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \tag{11.5}$$

关键在于如何计算该梯度。Schulman 等（2015）对策略梯度的多种计算方式做了综述 [130]。以下采用有限回合、$\gamma = 1$ 的常见语言模型设定（终止后奖励为 0）。若优化折扣回报，梯度还需包含外层 γ^t 权重。目标是估计 $g := \nabla_\theta J(\theta)$，常见形式如：

$$g = \mathbb{E}\Big[\sum_{t=0}^{\infty} \Psi_t \nabla_\theta \log \pi_\theta(a_t|s_t)\Big], \tag{11.6}$$

在上述无折扣设定下，Ψ_t 可以是：

1. $\sum_{t=0}^{\infty} r_t$：整个轨迹的总奖励。
2. $\sum_{t'=t}^{\infty} r_{t'}$：动作 a_t 之后的奖励（即 return G）。
3. $\sum_{t'=t}^{\infty} r_{t'} - b(s_t)$：带 baseline 的版本。
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，是常用的降方差选择，但一般不能保证它在所有基线中方差最低。
6. $r_t + V^\pi(s_{t+1}) - V^\pi(s_t)$：TD 残差。

Baseline 的作用是降低策略更新的方差（后文详述）。

对于语言模型，上述部分概念需适当解释。对确定性策略 μ，有 $V^\mu(s) = Q^\mu(s, \mu(s))$；只有取最优贪心动作时才涉及 $\max_a Q$。对随机策略，有 $V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)} Q^\pi(s, a)$。若 token 拼接产生确定性后继状态 $s + a$，且即时奖励为 $r(s, a)$，则：

$$A^\pi(s, a) = r(s, a) + \gamma V^\pi(s + a) - V^\pi(s) \tag{11.7}$$

这包含即时奖励和后继状态价值。一般随机转移下，后继价值需对转移分布取期望；使用近似价值网络时，单个 TD 残差只是优势的估计。

11.1.1 Vanilla Policy Gradient

最基础的策略梯度实现是对 $J(\theta)$ 关于策略参数求导：

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t\right] \tag{11.8}$$

普通策略梯度算法的常见问题是梯度方差大，可通过多种方式缓解。常用方法是引入 *baseline* 对值估计进行归一化。baseline 的本质是按状态对下游动作的价值进行归一化（如 Advantage 即 Q 值与 V 值之差）。最简单的 baseline 是奖励的批均值或滑动平均。与当前采样动作无关的状态基线不改变梯度期望，因为 $\mathbb{E}_{a\sim\pi(\cdot|s)}[\nabla_\theta \log \pi_\theta(a|s)] = 0$；它的作用是降低方差，而不是“去偏”。若批均值包含样本自身，其动作依赖性会引入缩放或偏差，不能直接套用这一结论。

许多策略梯度算法都基于优势函数的形式：

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t) \right] \tag{11.9}$$

策略梯度实现的核心是对概率策略求导：

$$\nabla_\theta \log \pi_\theta(a|s) = \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} \tag{11.10}$$

由链式法则推导：

$$\nabla_\theta \log x = \frac{1}{x} \nabla_\theta x \tag{11.11}$$

本章后续会用到这些推导。

11.1.2 REINFORCE

REINFORCE 算法名称可能是后加缩写，但其核心思想对现代 RL 算法极为重要。其定义见经典论文 *Simple statistical gradient-following algorithms for connectionist reinforcement learning* [131]：

> 名称意为" REward Increment = Nonnegative Factor X Offset Reinforcement X Characteristic Eligibility."

即更新规则有三个部分：

1. 非负因子：学习率（如 α）。
2. Offset Reinforcement：baseline b 或其他奖励归一化因子，提升稳定性。
3. Characteristic Eligibility：每个 token 的归属度，现代公式中常为 policy 的 log 概率。

因此，更新形式如下：

$$\Delta_\theta = \alpha(r - b)e \tag{11.12}$$

用现代符号和广义 return G，REINFORCE 公式为：

$$\nabla_\theta J(\theta) \;=\; \mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\,(G_t - b)\Big], \tag{11.13}$$

其中 $G_t - b(s_t)$ 是带基线的回报估计；当基线等于 $V^\pi(s_t)$ 时，它的条件期望为优势。记该估计为 A_t，可进一步写为：

$$\nabla_\theta J(\theta) \;=\; \mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\,A_t\Big], \tag{11.14}$$

REINFORCE 是 vanilla 策略梯度的具体实现，使用蒙特卡洛方法估算梯度。

REINFORCE 可不使用值网络（value network）；值网络仅用于 baseline。常见的 PPO actor-critic 实现使用价值网络与 GAE 估计优势；PPO 的裁剪目标本身并不强制使用价值网络，也不能保证优势估计精确。

REINFORCE Leave One Out（RLOO）

RLOO 与标准 REINFORCE 的核心区别在于：RLOO 用同一 *prompt* 下其他独立采样回答的平均奖励做 baseline，而不是全 batch 均值 [132], [129], [133]。

这要求每个 prompt 生成多个回复，这在 RL 微调语言模型中很常见。

具体地，RLOO baseline 如下（每个 prompt 采样 K 条轨迹或动作 $a_1, ..., a_K$）：

$$b(s, a_k) = \frac{1}{K-1} \sum_{i=1, i \neq k}^{K} R(s, a_i), \tag{11.15}$$

对应的优势为：

$$A(s, a_k) = R(s, a_k) - b(s, a_k). \tag{11.16}$$

也可写为：

$$A(s, a_k) = \frac{K}{K-1} \left(R(s, a_k) - \frac{1}{K} \sum_{i=1}^{K} R(s, a_i) \right). \tag{11.17}$$

这是一种简单低方差的优势更新，与 GRPO 非常相似（GRPO 后文介绍），只是 KL 惩罚和步长裁剪的位置不同。RLOO 优势也可与 PPO 裁剪结合，说明这些算法本质上非常接近。

RLOO 等无需值网络的算法，会将序列的优势（或奖励）分配给序列中每个 token。而 PPO 等用值网络的算法，则为每个 token 单独分配值，并对最终奖励进行折扣。例如，带 KL 惩罚时，RLOO 对整个补全求和，PPO 等则对每个 token 单独计算并从奖励中扣除（GRPO 则从优势中扣除）。这些细节和权衡将在后文详述。

11.1.3 Proximal Policy Optimization（PPO）

PPO [134] 是深度 RL 成功的基石算法之一（如 OpenAI 的 DOTA 5 [135]等）。其每个样本的待最大化裁剪代理目标为（最小化损失时取负号）：

$$\begin{aligned} \rho_t(\theta) &= \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}, \\ \ell(\rho, A) &= \min\{\rho A, \text{clip}(\rho, 1-\varepsilon, 1+\varepsilon)A\}, \\ J_t(\theta) &= \ell(\rho_t(\theta), A_t). \end{aligned} \tag{11.18}$$

语言模型中常按 token 计算条件概率比。序列概率虽然等于条件概率的乘积，但逐 token 裁剪并不等价于对整条序列的概率比做一次裁剪。实际实现中通常用 *log* 概率，计算更高效。

$$J(\theta) = \frac{1}{|a|} \sum_{t=1}^{|a|} \ell(\rho_t(\theta), A_t). \tag{11.19}$$

这就是 PPO 的逐 token 版本，也适用于其他策略梯度方法，后文实现部分会详细介绍。其中 $\frac{1}{|a|}$ 为常见实现习惯，正式推导中未必出现（见[136]）。

PPO 的核心在于“策略比率”：

$$R(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \tag{11.20}$$

策略比率控制参数更新，直观易懂。每个 batch 的第一步梯度，策略比率为 1（通常每 batch 做 1-4 步梯度更新）。如果策略比率大于 1 且优势为正，说明新策略更倾向于该动作，反之则相反。

不同情况下，损失函数的表达式如下：

- 优势为正，比例超过 $1+\varepsilon$，被裁剪为 $(1+\varepsilon)A$。
- 优势为正，比例小于 $1-\varepsilon$，则取 $R(\theta)A$。
- 优势为负，裁剪发生在 $R(\theta) < 1-\varepsilon$ 时，目标变为 $(1-\varepsilon)A$。

裁剪限制的是代理目标在部分方向上的收益，并不对参数更新或 KL 距离施加硬性边界。当裁剪未激活时，该项与未裁剪的重要性加权代理目标一致。

11.1.4 Group Relative Policy Optimization（GRPO）

GRPO 最早出现在 DeepSeekMath [137]，也被 DeepSeek-V3 [138]、DeepSeek-R1 [139]等采用。GRPO 是 PPO 风格的变体：用同一问题下多个回答的奖励构造组内基线，替代独立训练的价值网络。它仍可保留旧策略概率和用于 KL 正则化的参考策略。这样有两个好处：

1. 避免用 LM 主干学习值函数的难题（最佳实践尚未确定）。
2. 节省显存，无需保留另一套模型权重。

在结果监督下，GRPO 对同一 prompt 采样多个回答。每个回答内部的 token 共享该回答的优势，不同回答的优势通常不同。

GRPO 目标函数与 PPO 类似。对同一问题 s 的 G 个回复 $\{a_1, ..., a_G\}$，待最大化目标为：

$$J(\theta) = \frac{1}{G} \sum_{i=1}^{G} J_i(\theta), \tag{11.21}$$

可展开为逐 token 损失：

$$
\begin{aligned}
\rho_{i,t}(\theta) &= \frac{\pi_\theta(a_{i,t}|s_{i,t})}{\pi_{\text{old}}(a_{i,t}|s_{i,t})}, \\
J_i(\theta) &= \frac{1}{|a_i|}\sum_{t=1}^{|a_i|}\Big[\ell(\rho_{i,t}(\theta), A_{i,t}) \\
&\qquad - \beta D_{\text{KL}}\big(\pi_\theta(\cdot|s_{i,t})\|\pi_{\text{ref}}(\cdot|s_{i,t})\big)\Big].
\end{aligned}
\tag{11.22}
$$

GRPO 与 PPO 的主要区别在于，GRPO 的标准实现将 KL 距离直接加到损失项中。优势计算为（$\epsilon_{\text{num}} > 0$ 是数值稳定项，与 PPO 的裁剪阈值不同）：

$$
A_i = \frac{r_i - \text{mean}(r_1, \ldots, r_G)}{\text{std}(r_1, \ldots, r_G) + \epsilon_{\text{num}}}.
\tag{11.23}
$$

直观上，GRPO 是在同一问题内比较多个答案，模型学会更像正确答案、远离错误答案。这种优势计算简单直接，适合 RLHF 大批量采样场景。与 PPO、REINFORCE 等基于奖励模型打分的 RLHF 相比，GRPO 常用更多每 prompt 采样数，优势估计有更丰富上下文。

组内标准差归一化会改变不同问题的相对权重 [136]。当一组奖励完全相同（例如二元奖励下全对或全错）时，分子为 0；分母加稳定项后，所有优势均为 0，不会产生更大的优势。标准差很小但非零时，归一化可能放大奖励差异。Dr. GRPO 去掉了这一标准差缩放。

式 11.23适用于结果监督（如标准奖励模型或可验证奖励），过程监督则需不同实现。此时，GRPO 优势为后续推理步骤归一化奖励之和。

GRPO 的优势估计也可不带 PPO 裁剪，直接用于 vanilla 策略梯度（如 REINFORCE），但这不是标准形式。实际上，GRPO 的 Dr. GRPO 变体 [136]的优势估计与 RLOO 仅差一个缩放因子：

$$
\tilde{A}_i = r_i - \text{mean}(r_1, r_2, \cdots, r_G) = r_i - \frac{1}{G}\sum_{j=1}^{G} r_j
\tag{11.24}
$$

而 RLOO 优势为：

$$
A_i^{\text{RLOO}} = r_i - \frac{1}{G-1}\sum_{j=1, i\neq j}^{G} r_j
\tag{11.25}
$$

两者相差 $\frac{G}{G-1}$ 的缩放（通常可通过归一化消除实际影响）：

$$
\begin{aligned}
\frac{G}{G-1}\tilde{A}_i &= \frac{G}{G-1}\left(r_i - \frac{1}{G}\sum_{j=1}^{G} r_j\right) \\
&= r_i - \frac{1}{G-1}\sum_{j=1, j\neq i}^{G} r_j \\
&= A_i^{\text{RLOO}}
\end{aligned}
\tag{11.26}
$$

11.2 实现细节

与早期深度 RL 文献相比，面向语言模型等大模型的 RLHF 实现有许多细节需注意。以下是一些关键点：

- **值网络初始化**：PPO 等算法的值网络可用同结构其他模型或随机权重初始化，对性能影响大。
- **奖励/优势归一化与白化**：白化通常指减去均值并除以标准差，使均值约为 0、方差约为 1；它不是缩放到 $[0, 1]$。掩码、分组方式和方差的计算约定都需要明确。
- **KL 估计方式**：复杂语言模型下 KL 精确计算难，常用近似 [117]。
- **KL 控制器**：早期 PPO 实现有动态 KL 控制器，现代 RLHF 多用静态 KL 惩罚，但也可调整。

更多实现细节可参考 [140]，算法基础见 [141]。

11.2.1 策略梯度基础实现

一个简单的策略梯度实现如下，利用优势函数估计梯度，为后续 PPO/GRPO 等算法做准备：

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

ratio 为新旧策略概率的比值（log 概率差的指数）。

理解该公式时，需考虑不同 batch 中的情况。我们希望模型越好，损失越小。

- 情况 1：优势为正，说明该动作优于期望值，应强化。此时 loss 为负，模型会提升该 token 概率。
- 情况 2：优势为负，说明动作劣于期望值。此时 loss 为正，模型会降低该 token 概率。
- 情况 3：优势为零，不更新，loss 为零。

11.2.2 损失聚合

实际实现时，关键问题是：如何对 KL 距离和损失进行求和，设计不同的值归属方式。

本节主要采用 token 级 MDP 表示。若将完整回复作为 bandit 动作，可为同一回答共享一个优势；但各 token 的 log 概率、概率比与梯度仍可能不同。MDP 与 bandit 的区别在于动作和状态转移的建模，不能仅归结为损失如何聚合。

假设有如下变量，batch 大小为 B，序列长度为 L：

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

用 `pg_loss = -advantages * ratio` 即可批量计算损失。这样每个补全的所有 token 共享同一优势（outcome reward 场景），乘以每 token 的概率比。

若用值网络，loss 行为差异更大。outcome reward 时，优势每 token 相同，token 概率差异主导学习动力。

GRPO 和 PPO 等算法中，loss 通常对补全 token 求和：

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

这类似于 `masked_mean` 操作。也可对每 token 单独平均：

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

直观上，按序列平均似乎更好，因为我们关注的是结果，而不是具体 token。但这会引入微妙偏差。举例，两个长度不同的序列，优势分别为 `a_1` 和 `a_2`：

```
seq_1_advs = [a_1, a_1, a_1, a_1, a_1] # 5 tokens
seq_2_advs = [a_2, a_2, a_2, a_2, a_2, a_2, a_2, a_2, a_2, a_2] # 10 tokens
```

若最后一个 token 决定优势为正，多步梯度后，loss 可能为：

```
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
```

序列平均后：

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

若对所有 token 平均，则为 $(14 + 19)/(5 + 10) = 2.2$；若先对每条序列平均再对两条序列平均，则为 $(2.8 + 1.9)/2 = 2.35$。若序列差异更大，loss 差距会更明显。

更完整的例子见下方代码，演示了两条样本（长短不同）下三种 loss 聚合方式。

```
from typing import Optional
import torch

def masked_mean(values: torch.Tensor, mask: torch.Tensor, axis: Optional[int] = None) -> torch.Tensor:
    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[int] = None,
        constant_normalizer: float = 1.0,
    ) -> torch.Tensor:
    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,],
], requires_grad=True)


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

masks = torch.tensor([
    [1, 1, 1, 1, 0, 0, 0,],
    [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 * advs, 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_result.mean().backward()
print("ratio.grad", ratio.grad)
ratio.grad.zero_()

masked_sum_result.mean().backward()
```

```
print("ratio.grad", ratio.grad)
ratio.grad.zero_()

masked_mean_token_level.mean().backward()
print("ratio.grad", ratio.grad)
```

可以看到，GRPO 默认实现（masked_mean）下，短序列每 token 梯度更大，Dr. GRPO 和 DAPO 则更均衡。如果用梯度累积，短长序列的平衡还会变化。

另一种聚合方式（见[136]）是每 token 损失用最大序列长度归一化，这会影响不同 batch 间的损失对比。

实际应用中，最优方案取决于具体任务和在线学习设置。RLHF 实践中，数值稳定性和损失方差最小的方案往往更受青睐。

11.2.3 PPO 实现示例

PPO 有多种实现，核心损失计算如下。值函数的计算也很关键，且有多种实现方式。

注意，这里的参考策略（或旧 log 概率）指采样时的参数，不一定是 reference policy。reference policy 仅用于 KL 惩罚。

下面仅展示损失层，假设已在采样阶段用旧价值预测和包含 KL 塑形的奖励计算了 GAE 与回报目标。所有 token 张量均为 [B*G, L]，每条回答至少包含一个有效 token；old_logps、gae_advantages、returns 在更新阶段保持固定。

```
import torch

mask = completion_mask.to(new_per_token_logps.dtype)
advantages = gae_advantages.detach()  # [B*G, L]，不要再 unsqueeze
returns = returns.detach()  # GAE + old_values 或选定的回报目标
old_logps = per_token_logps.detach()

# 仅用有效回答 token 计算优势统计量
valid = advantages[completion_mask.bool()]
advantages = (advantages - valid.mean()) / (
    valid.std(correction=0) + 1e-8
)
ratio = torch.exp(new_per_token_logps - old_logps)
eps = self.cliprange
pg_losses1 = -advantages * ratio
pg_losses2 = -advantages * torch.clamp(ratio, 1.0 - eps, 1.0 + eps)
pg_loss_max = torch.maximum(pg_losses1, pg_losses2)

# values 是当前价值网络的预测；目标是回报，不是单步即时奖励
vf_loss = 0.5 * (returns - values).square()
per_token_loss = pg_loss_max + self.vf_coef * vf_loss
loss = ((per_token_loss * mask).sum(1) / mask.sum(1)).mean()
```

```
with torch.no_grad():
    clip_frac = ((pg_losses2 > pg_losses1) * mask).sum() / mask.sum()
    # 与采样旧策略的近似 KL；不同于对参考策略的 KL 正则项
    log_ratio = new_per_token_logps - old_logps
    approx_kl = (0.5 * log_ratio.square() * mask).sum() / mask.sum()
    value_loss = (vf_loss * mask).sum() / mask.sum()
```

PPO 的核心在于如何更新策略梯度损失。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` 即 vanilla 优势策略梯度，PPO 在此基础上引入裁剪，使更新步长不至于过大。

代码裁剪的是概率比 `ratio`，不是 `logratio`。它通过代理损失抑制部分过大的策略改变，但不保证参数步长或 KL 距离满足硬性上限。

最后取二者最大值，确保更保守的损失更新。

PPO 在学习值函数的同时进行上述操作，虽然更复杂，但这是参数更新的核心逻辑。

PPO/GRPO 单步梯度（无裁剪）简化

若 PPO（或 GRPO）每样本只做一次梯度更新，主方程可大幅简化。此时 $\pi_\theta = \pi_{\theta_{old}}$，更新规则变为（$[]_\nabla$ 表示停止梯度）：

$$J(\theta) = \frac{1}{G}\sum_{i=1}^{G}\left(\frac{\pi_\theta(a_i|s)}{[\pi_\theta(a_i|s)]_\nabla}A_i - \beta D_{KL}(\pi_\theta||\pi_{ref})\right). \tag{11.27}$$

此时可省略第二次策略梯度和裁剪逻辑，优化器更接近标准策略梯度。

11.2.4 GRPO 实现示例

DeepSeekMath 论文详细介绍了 GRPO 与 PPO 的实现差异 [137]。如，GRPO 将 KL 惩罚直接加到损失项，而 PPO 多在奖励中加 KL。通常 KL 距离对每个 token 单独计算，推理训练时每个 prompt 采样多个补全，每 batch 有多个 prompt，这里展平后的逐 token KL 形状为 `[B*G, L]`，其中 `G` 是每个 prompt 的回答数。

综合起来，伪代码如下：

```
# B: Batch Size, L: Sequence Length, G: Number of Generations
# rewards: (B*G,), prompt-major 排列，同一 prompt 的 G 个回答连续
```

```
# 要求 G > 1；下面使用总体标准差
mean_grouped_rewards = rewards.view(-1, self.num_generations).mean(dim=1)
std_grouped_rewards = rewards.view(-1, self.num_generations).std(dim=1, correction=0)

# 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.detach())  # 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
    log_ratio = new_per_token_logps - per_token_logps.detach()
    approx_kl = (0.5 * log_ratio.square() * completion_mask).sum() / completion_mask.sum()
```

更多解释请见上文 PPO 部分。

RLOO vs. GRPO

RLOO 的优势更新与 GRPO 极为接近，突出两者在本质上的相似性。RLOO 的优势是相对于同一问题下其他补全的奖励。简明代码（参考TRL 实现）：

```
 # 此处是 generation-major 排列：每轮依次采样 N 个 prompt
# 若输入是 prompt-major，需要先 reshape(N, k).T，不能直接套用
# rloo_k --> 每 prompt 补全数，必须 > 1
# rlhf_reward --> 所有补全的奖励，长度 B = N x k
rlhf_reward = rlhf_reward.reshape(rloo_k, -1) #
# 形状：(k, N)，每列 j 为 prompt j 的 k 个奖励

baseline = (rlhf_reward.sum(0) - rlhf_reward) / (rloo_k - 1)
# baseline --> leave-one-out 基线奖励，形状：(k, N)
# baseline[i, j] 是 prompt j 除 i 外样本的平均奖励

advantages = rlhf_reward - baseline
# advantages --> 同形状

advantages = advantages.flatten() # 恢复原始 tensor 形状
```

其余实现细节与其他策略梯度方法类似。

11.3 补充话题

要精通策略梯度算法在 RLHF 中的应用，还有许多细节值得关注。

11.3.1 广义优势估计（GAE）

广义优势估计（Generalized Advantage Estimation, GAE）是一种平衡偏差-方差权衡的优势计算方法[130]。较短的自举估计通常方差较低，但更依赖价值函数近似的准确性；完整回报的蒙特卡洛估计通常方差较高，却不依赖末端自举值（真正终止时）。GAE 通过结合多步预测和指数加权滑动平均来优化这一权衡。

n 步优势估计的数学表达式如下（类似于时序差分残差）：

$$\hat{A}_t^{(n)} = \begin{cases} r_t + \gamma V(s_{t+1}) - V(s_t), & n = 1 \\ r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) - V(s_t), & n = 2 \\ \vdots & \\ \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n V(s_{t+n}) - V(s_t), & \text{一般情况} \end{cases}$$

其中：- γ 为折扣因子 - $V(s)$ 是状态价值函数；$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ 为 TD 残差 - 有限回合在实际终止处截断求和，终止状态价值为 0；仅因长度限制被截断时，是否自举需按任务定义处理 - 最终表达式通过引入衰减因子 λ 实现滑动平均：

$$\hat{A}_t^{\text{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$$

第十二章　直接对齐算法

直接对齐算法（Direct Alignment Algorithms, DAAs）允许我们在无需训练奖励模型或使用强化学习优化器的情况下，直接优化 RLHF 目标。其中最具代表性、并掀起学术界大规模关注的，是直接偏好优化（Direct Preference Optimization, DPO）[19]。DPO 将 KL 正则化奖励最大化问题中的最优策略关系代入偏好模型，得到可直接训练策略的分类损失。实际训练最小化偏好损失，并不保证有限数据和有限优化步骤下达到原目标的全局最优。自 2023 年 5 月发布以来，经过社区对数据和超参数（尤其是意外地低学习率）的探索，DPO 及其变体被广泛应用于主流模型，如 Zephyr-β（2023 年 10 月）[20]、Llama 3 Instruct [23]、Tülu 2 [21]、Tülu 3 [6]、Nemotron 4 340B [24]等。严格来说，Sequence Likelihood Calibration（SLiC-HF）[142]更早提出，但因有效性和运气等原因未被广泛采用。

DPO 和 DAAs 最重要的意义，是极大降低了语言模型后训练的技术门槛。

12.1　直接偏好优化（Direct Preference Optimization, DPO）

下面我们将直观解释 DPO 的原理，并推导其核心公式。

12.1.1　DPO 的原理

DPO 表面上是直接优化策略以求解 RLHF 目标。其损失函数本质上是 log 概率的成对关系。Bradley-Terry 奖励模型推导出的损失函数如下：

$$
\begin{aligned}
h_\theta(x, y_c, y_r) &= \log \frac{\pi_\theta(y_c|x)}{\pi_{\text{ref}}(y_c|x)} - \log \frac{\pi_\theta(y_r|x)}{\pi_{\text{ref}}(y_r|x)}, \\
\mathcal{L}_{\text{DPO}} &= -\mathbb{E}_{(x,y_c,y_r)\sim\mathcal{D}}\big[\log \sigma(\beta h_\theta(x, y_c, y_r))\big].
\end{aligned}
\tag{12.1}
$$

这里用到 DPO 的隐式奖励：

$$
r(x, y) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)} \tag{12.2}
$$

这个奖励来自 Bradley-Terry 模型下的最优策略推导（见式 12.8）。这是用策略与参考策略的 log 概率比参数化的隐式奖励，不是一个概率。完整奖励还可包含仅依赖 prompt 的加法项 $\beta \log Z(x)$；该项在成对奖励差中抵消。

观察式 12.1中的损失，优化目标是让选中回复的 log 比率大于被拒回复（归一化参考模型）。实际中，这就是对模型在数据中 token 序列的 log 概率求和。因此，DPO 实质上是在拉大选中与被拒回复概率的差距。

有了式 12.2中的奖励，我们可以写出损失的梯度，进一步理解机制：

$$\nabla_\theta \mathcal{L}_{\text{DPO}} = -\beta \mathbb{E}_{\mathcal{D}} \Big[\sigma(r_\theta(x, y_r) - r_\theta(x, y_c)) \cdot \big(\nabla_\theta \log \pi_\theta(y_c|x) - \nabla_\theta \log \pi_\theta(y_r|x) \big) \Big]. \tag{12.3}$$

直观理解如下：

- $\sigma(\cdot)$ 中的第一项，为参数更新赋予 0 到 1 的权重，当奖励估计错误时（被拒样本更优），权重更大。
- 内部括号 $[\cdot]$ 项提升选中回复 y_c 的概率，降低被拒回复 y_r 的概率。
- β 控制优化中排序与 KL 距离的平衡。

核心直觉是，DPO “隐式拟合了一个奖励模型，其对应最优策略可解析写出”（归功于梯度上升和 ML 工具）。DPO 确实直接更新策略参数；同一个语言模型同时给出了隐式奖励的参数化，而不是先独立训练奖励模型再用 RL 优化策略。

在 Bradley-Terry 偏好假设、足够的模型表达能力与充分优化等条件下，可以通过这一参数化联系偏好拟合与 KL 正则化目标的最优策略。经验训练效果仍受数据覆盖、模型容量和优化过程限制。与策略梯度类 RL 方法的区别在于，DPO 的生成不是在线的，而是离线的，因此 β 更易调节，但最优值依赖于具体模型与数据。

对每批偏好数据，DPO 直接计算离线偏好损失并更新策略，省去了训练循环中的在线 rollout 和显式价值估计。

12.1.2 DPO 公式推导

DPO 推导分两步：1. 推导 RLHF 目标的最优策略形式；2. 用 Bradley-Terry 模型推导如何从偏好数据获得该解。

1. RLHF 最优解推导

对固定奖励函数 r、$\beta > 0$ 和参考策略，在其支持集上考虑：

$$\begin{aligned} \pi^* &= \arg\max_\pi J(\pi), \\ J(\pi) &= \mathbb{E}_{x\sim\mathcal{D}} \left[\mathbb{E}_{y\sim\pi(\cdot|x)} r(x, y) - \beta D_{\text{KL}}(\pi(\cdot|x) \| \pi_{\text{ref}}(\cdot|x)) \right]. \end{aligned} \tag{12.4}$$

展开 KL，并将目标乘以负数 $-1/\beta$，最大化转为最小化。这里相等的是最优策略集合，不是目标函数的数值：

$$\pi^* = \arg\min_\pi \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi(\cdot|x)} \left[\log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{r(x, y)}{\beta} \right]. \tag{12.5}$$

定义有限的配分函数（partition function）：

$$Z(x) = \sum_y \pi_{\text{ref}}(y|x) \exp(r(x, y)/\beta). \tag{12.6}$$

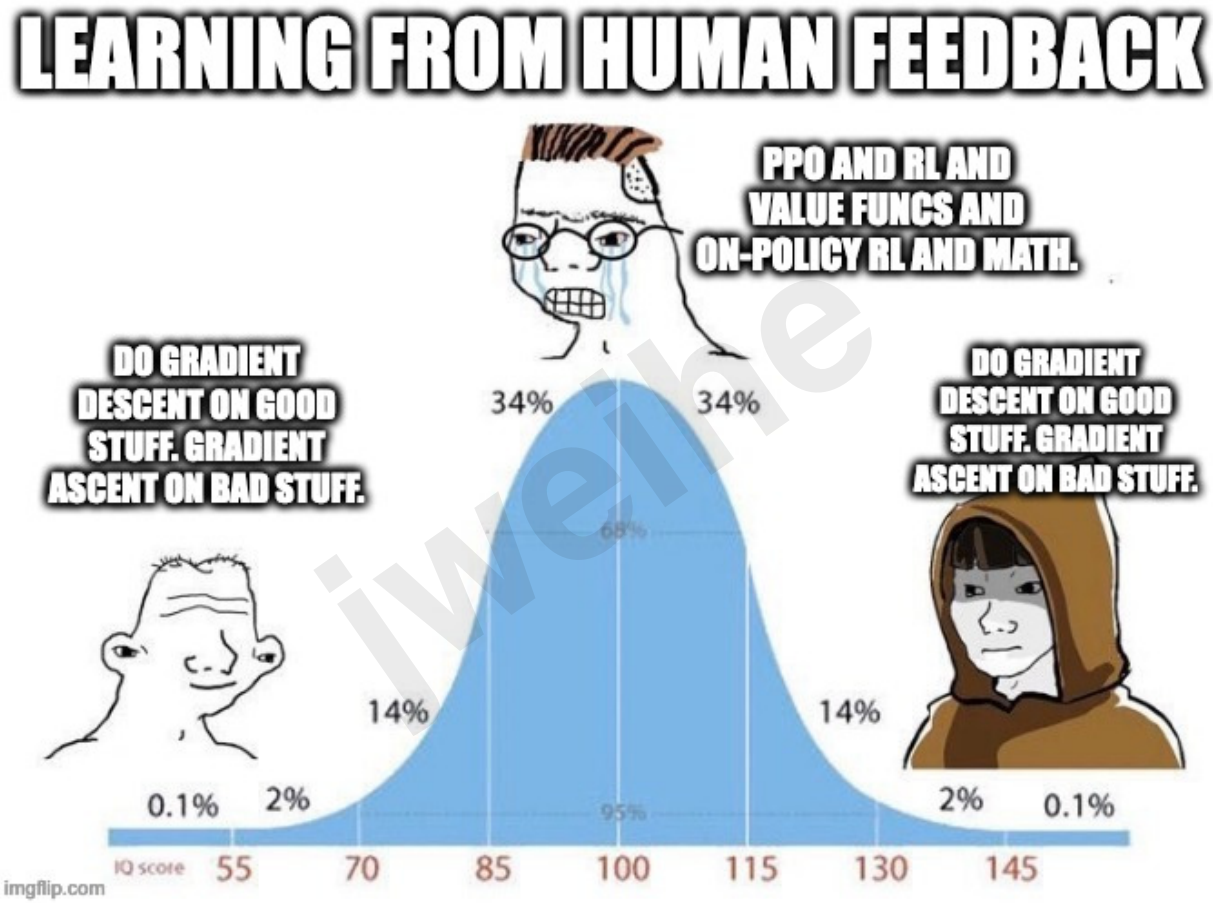

图 12.1 DPO 简洁性梗图，致谢 Tom Goldstein。

令 $q(y|x) = \pi_{\text{ref}}(y|x)\exp(r(x,y)/\beta)/Z(x)$，则：

$$\pi^* = \arg\min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\left[D_{\text{KL}}(\pi(\cdot|x)\|q(\cdot|x)) - \log Z(x)\right]. \tag{12.7}$$

$Z(x)$ 与待优化策略无关。由 KL 非负性，当策略等于 q 时取得最小值，因此：

$$\pi^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp(r(x,y)/\beta). \tag{12.8}$$

2. Bradley-Terry 模型下的 DPO 目标

回顾第 7 章奖励建模与第 6 章偏好数据，Bradley-Terry 模型为：

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp\left(r^*(x,y_1)\right)}{\exp\left(r^*(x,y_1)\right) + \exp\left(r^*(x,y_2)\right)} \tag{12.9}$$

对式 12.8取对数并代入，得 DPO 奖励：

$$r^*(x,y) = \beta\log\frac{\pi^*(y\mid x)}{\pi_{\text{ref}}(y\mid x)} + \beta\log Z(x) \tag{12.10}$$

代入 Bradley-Terry 公式，化简后得：

$$p^*(y_1 \succ y_2 \mid x) = \sigma\left(\beta\log\frac{\pi^*(y_1\mid x)}{\pi_{\text{ref}}(y_1\mid x)} - \beta\log\frac{\pi^*(y_2\mid x)}{\pi_{\text{ref}}(y_2\mid x)}\right) \tag{12.11}$$

这正是 DPO 的损失函数（见式 12.1）。

3. Bradley-Terry DPO 梯度推导

DPO 梯度如式 12.3所示，推导如下：

$$\nabla_\theta\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{\mathcal{D}}\left[\nabla_\theta\log\sigma(\beta h_\theta)\right]. \tag{12.12}$$

利用 sigmoid 和 log 的求导性质，可化为：

$$\begin{aligned}\nabla_\theta\mathcal{L}_{\text{DPO}} &= -\beta\mathbb{E}_{\mathcal{D}}\left[\sigma(-\beta h_\theta)\nabla_\theta h_\theta\right],\\ \nabla_\theta h_\theta &= \nabla_\theta\log\pi_\theta(y_c|x) - \nabla_\theta\log\pi_\theta(y_r|x).\end{aligned} \tag{12.13}$$

12.2 数值问题、局限与变体

DPO 算法已出现多种变体，旨在解决其局限。例如，DPO 在无奖励模型评分的情况下，对每对偏好数据赋予同等权重，忽略了更丰富的标签信息。为此，相关算法尝试重新平衡优化目标：

- **REBEL**：将奖励模型分数作为选中与被拒回复之间的 margin，提升 RLHF 问题的求解准确性 [119]。

- **保守 DPO（cDPO）与恒等偏好优化（IPO）**：cDPO 用标签平滑处理潜在偏好噪声 [19]；IPO 在其一般偏好优化框架中采用恒等映射，并使用平方损失约束相对 log 概率比，以缓解对确定性偏好的过拟合 [143]。
- **带偏移的 DPO（ODPO）**：要求选中与被拒回复的 likelihood 差异大于某个阈值，不再一视同仁 [144]。

有些变体通过调整损失函数或内存优化提升学习信号或效率：

- **ORPO（Odds Ratio Policy Optimization）**：将选中回复的负对数似然与基于 odds ratio 的偏好损失结合，抑制相对不受偏好的回答，无需参考模型 [145]。
- **SimPO（Simple Preference Optimization）**：以长度归一化的回答 log 概率作为隐式奖励，并引入目标奖励间隔；无需参考模型 [146]。

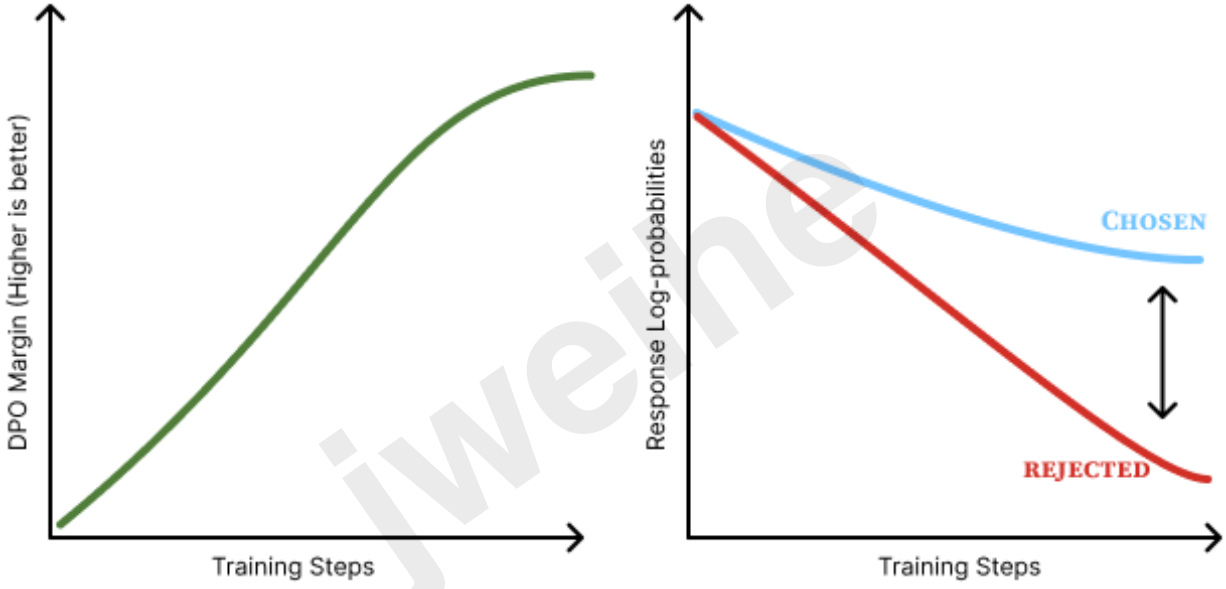

图 12.2 DPO 中的偏好“位移”问题示意。

DPO 的一个突出问题是：优化目标仅仅是拉大选中与被拒回复概率的间隔。数值上，可能出现两者概率都下降、被拒回复下降更多的情况（见图 12.2）。这对泛化的影响尚不明确，有研究认为这会提升未被覆盖行为的概率 [147] [148]。如 Cal-DPO [149]、AlphaPO [150]等方法通过调整优化过程或奖励形状缓解这种**偏好位移**。实际影响尚不明朗，但这可能是在线 RL 方法优于 DPO 的原因之一。

另一个 DPO 类方法性能上限低于在线 RLHF 的主要原因，是其训练信号来自其他模型的补全。在线变体如 **Online DPO** [151]，或结合奖励模型重标记的新生成数据的 **Discriminator-Guided DPO（D2PO）** [152]，通过实时生成新补全并引入偏好信号，缓解了这一问题。

还有许多 DAA 变体，如 Direct Nash Optimization（DNO）[153]、Binary Classifier Optimization（BCO）[154]等，但目前算法选择远不如初始模型和数据重要 [6] [155] [156]。

12.3 实现注意事项

DAA 如 DPO 的实现方式与策略梯度优化器有很大不同。DPO 损失函数的典型实现如下 [19]：

```
import torch.nn.functional as F

# logps 是仅对回答 token 求和的序列 log 概率；参考模型保持冻结
pi_logratios = policy_chosen_logps - policy_rejected_logps
ref_logratios = reference_chosen_logps - reference_rejected_logps

logits = pi_logratios - ref_logratios  # 即 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()
```

这可以直接用于标准语言模型训练流程（只需加一个参考模型）。

这种方式更简单，也提升了开发体验，但有一些新的注意点：

1. β **不是固定的 KL 距离**：β 来自正则化系数，并在 DPO 损失中缩放 log 概率比。实际 KL 还受训练数据、学习率、训练步数与模型影响，不能由 β 直接指定，也不保证每一步都趋近全局最优。应通过评测与实际分布漂移监测选择超参数。
2. **缓存 log 概率**：简单实现中，policy 和 reference 模型同时前向推理，方便损失计算，但会使显存消耗翻倍。可先离线计算参考模型 log 概率，训练时直接查表，显著降低显存需求。

12.4 DAA 与 RL：在线与离线数据

本质问题是：我们是否需要强化学习的内部机制（如值函数、策略梯度等）来实现 RLHF 对齐？当然，二者各有成熟体系，关键在于理解两者的本质差异和性能表现。

多项研究发现，基于策略梯度和 RL 的方法在性能上优于 DPO 及其变体。这些研究通过控制数据、算法对比训练模型 [157] [158]，或研究 RL 优化循环中 on-policy 数据的作用 [159]，都显示 DPO 略逊一筹。

尽管如此，DAA 因其简单性在主流模型中广泛应用。DAA 为训练数据和配置的快速迭代提供了极大便利，而数据往往比算法本身更重要，因此 DPO 在实际中依然很有价值。

随着以 RL 为主的推理模型兴起，未来会有更多投资回归 RL 偏好微调，这将提升 RL 基础设施的健壮性，并进一步拉开 DAA 与 RLHF 在线优化的性能差距。

第十三章　宪法 AI 与 AI 反馈

基于 AI 反馈的强化学习（RL from AI Feedback，RLAIF）是一套更广泛的技术体系，用于利用 AI 生成或增强反馈数据，包括成对偏好数据 [160] [161] [162]。采用 RLAIF 的动机有很多：可以完全替代或补充人工反馈。原稿写作时期，作者观察到 AI 反馈常比人工标注便宜；具体成本取决于任务难度、输出长度、模型价格、人工复核与质量要求，不能把某一时期的单条报价视为通用成本。这种成本差异让更多团队和个人有机会参与 RLHF 实验，打破了高昂数据门槛。除了价格，AI 反馈与人类反馈在性能上也有不同的权衡，这些差异仍在持续研究中。在技能类评测上，AI 反馈的最佳表现与人类数据大致相当，但在人类数据是否能让模型在实际产品或新型训练（如角色训练）中获得更细致可控性，目前尚无定论。

RLAIF 一词最早由 Anthropic 在 *Constitutional AI: Harmlessness from AI Feedback* [18]提出，这一工作最初让 AI 社区对相关方法的关系产生了一些混淆。自 CAI（Constitutional AI）论文发布、RLAIF 正式提出后，RLAIF 已成为后训练和 RLHF 文献中的默认方法——实际案例远超统计所能覆盖。可以理解为，CAI 是引发 RLAIF 领域爆发的起点。

关于人类数据与 AI 反馈数据的区别，有一个经验法则：

1. 人类标注员之间可能存在较大的分歧；
2. 固定 AI 裁判的输出可能更一致，但也可能重复其系统性偏差。

这只是经验性观察，不是普遍的偏差-方差定理。人类标注也会有系统性偏差，AI 反馈也可能有较高随机性。

许多学术论文表明，AI 偏好数据可在 RLHF 流程中替代人工反馈并获得很强的评测分数 [163]，但也反映出学术界 RLHF 与工业界最佳实践的分野。

13.1　宪法 AI（Constitutional AI, CAI）

宪法 AI（CAI）是 Anthropic 在 Claude 系列模型中广泛应用的、最早大规模使用合成数据进行 RLHF 训练的方法。CAI 主要有两种合成数据用法：

1. 对指令微调数据进行批判，遵循一套原则（如“答案是否鼓励暴力”“答案是否真实”）。模型生成回答后，会根据宪法中的原则逐条检视并不断修正，最终用这些数据微调模型。
2. 利用语言模型，根据宪法中的随机原则判断哪一个补全更好，从而生成成对偏好数据（类似于原则驱动奖励模型）。之后，RLHF 流程用这些合成偏好数据继续训练，也就是 RLAIF 的由来。

CAI 最为人熟知的是第二种——合成偏好数据，但其指令数据方法也被广泛用于后训练中的数据筛选和合成数据生成。

CAI 可形式化描述如下：

Anthropic 通过一套人工书写的原则（constitution），用 LLM 生成用于微调的 AI 偏好数据与指令数据；生成和评价可以使用同一模型体系的不同调用或检查点 [18]。一份宪法 $\mathcal{C}$ 是若干条具体原则的集合，指明批判阶段应关注的方面。指令数据的生成方法是：反复采样宪法中的原则 $c_i \in \mathcal{C}$，让模型根据 c_i 修订其最新输出 y^i 以更好地对齐 prompt x，最终得到一系列变体 $\{y^0, y^1, ..., y^n\}$，每一步都对应一个原则。最终的数据点是 prompt x 和最终补全 y^n。

偏好数据的生成更简单：用 $\mathcal{C}$ 的子集作为反馈模型的上下文，给定 prompt x、一组原则 $\{c_0, ..., c_n\}$ 和两个补全 y_0、y_1（分别标记为 A、B，来自早期 RLHF 数据集），让反馈模型判断 A/B 哪个更好，并将其概率作为奖励模型的训练样本。

13.2 专用 LLM 裁判模型

随着 RLAIF 和 “LLM 裁判”（LLM-as-a-judge）的普及，越来越多人关注：我们是否应该用同一个模型生成回复和评价，还是分开？为此，业界推出了多种专用评判模型，如 Shepherd [164]、CriticLLM [165]，以及用于性能评测的 Auto-J [166]、Prometheus [94]、Prometheus 2 [167]、Prometheus-Vision [168]等，但这些模型尚未在主流训练流程中广泛采用。

13.3 延伸阅读

关于宪法 AI 的相关研究和扩展很多，但目前鲜有被证明能显著改进 RLHF 和后训练流程的公开方案。这里列出一些值得关注的方向：

- OpenAI 发布了 Model Spec [79]，明确声明了模型的预期行为，并探索让模型直接参考该文档进行对齐（类似 CAI 思路）。OpenAI 还用 Deliberative Alignment [169]训练 o1 等推理模型，使其能参考安全或行为政策进行自我对齐。
- Anthropic 持续在模型训练中应用 CAI，不断更新 Claude 的宪法 [170]，并研究人群集体如何收敛于某些原则及其对模型行为的影响 [171]。
- 开源社区也在尝试将 CAI 应用于开源数据集 [172]，以及探索 LLM 间对话数据生成 [173]。
- 也有研究用原则驱动偏好或反馈结合不同优化方法，如[174]用原则作为奖励模型上下文（用于 Dromedary 模型训练 [175]），[36]用原则提升 RLHF 流程中人工判断的准确性。

第十四章　推理训练与推理时扩展

在 2016 年 NeurIPS 大会上，Yann LeCun 首次提出了著名的“蛋糕比喻”来说明现代机器学习系统中学习发生的位置：

> 如果智能是一块蛋糕，那么蛋糕的主体是无监督学习，蛋糕上的糖霜是有监督学习，而蛋糕上的樱桃则是强化学习（RL）。

如今，随着现代大语言模型和后训练体系的演进，这个比喻已经基本完善：- 用海量互联网数据做自监督学习构成了“蛋糕的主体”（尤其是从 FLOPs 消耗来看），- 后训练的起点——指令监督微调（SFT）则是糖霜，让模型适应更窄的分布（结合 RLHF 精选样本），- 而“纯”强化学习（RL）则是蛋糕上的那颗樱桃。

二元奖励每次只提供有限的反馈信息，但这不意味着 RL 只需少量训练样本；训练可能需要大量回答采样与奖励评估。这种推理训练方式随着**推理模型**（reasoning models）的出现而兴起，这类模型结合了本书介绍的后训练技术，并在可验证领域上用 RL 训练，显著提升了推理、编程、数学等能力。

这些模型的训练方式——可验证奖励的强化学习（Reinforcement Learning with Verifiable Rewards, RLVR）[6]，与 RLHF 非常类似，但奖励模型可选，可以直接用一个评分函数：答案正确则奖励为正，否则为 0。

OpenAI o1 的公开介绍强调了强化学习对推理的作用 [47]；DeepSeek-R1 技术报告则明确介绍了基于规则的奖励设计 [139]。二者公开的训练细节不同，不能据此假定它们使用完全相同的 RLVR 流程。很快，整个 AI 行业都将这种训练流程和模型风格作为优先事项。这里的核心变化，与其说是 RL 训练方式的全新突破，不如说是训练各阶段分配和行为优先级的重塑。推理模型的出现，标志着 RL 训练规模化的时代到来。

关于这些模型的行为类型，可以看下面这个例子：DeepSeek V3 0325 和其推理模型 DeepSeek R1，面对“Write me a short poem about a goldfish”这个请求的表现。DeepSeek V3 的回答非常直接：

> Goldfish Dreams
>
> In a bowl of glass so clear and round, A tiny goldfish swims without a sound. With scales that gleam like sunlight’s kiss, He drifts along in watery bliss.
>
> A flick of fin, a flash so bright, Chasing bubbles in the light. Does he dream of oceans wide? Or is his little world enough—his pride?

而 DeepSeek R1 则完全不同。它会先生成大量“思考”token，规划答案后再正式回复：

> `<thinking>` 好的，用户想要一首关于金鱼的短诗。让我先头脑风暴一些金鱼相关的意象。金鱼通常在鱼缸里，对吧？那可以从环境描写入手，比如“镀金的鱼缸”或“水晶球”。

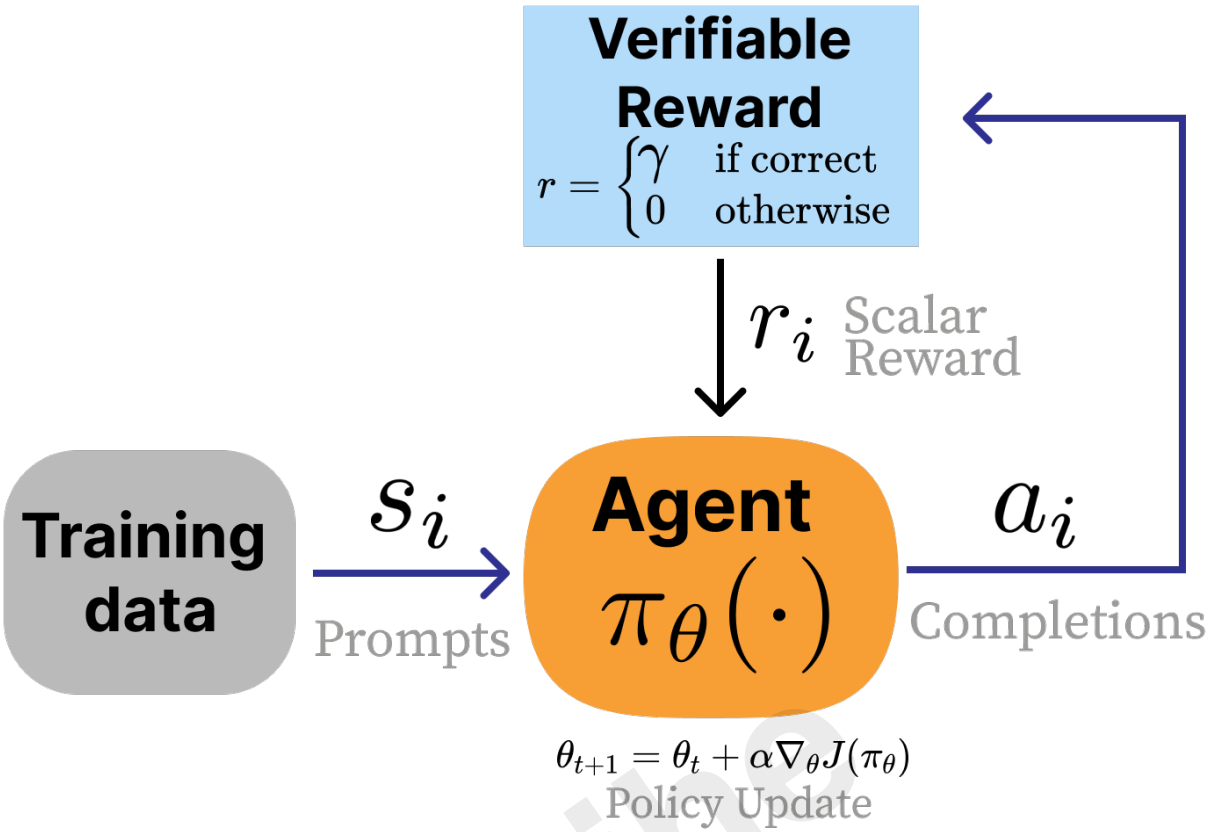

图 14.1 RLVR 的 RL 反馈循环示意图。此处用验证函数代替奖励模型。

接下来，思考金鱼的动作。它们游动很平滑，可以用“滑翔”“盘旋”等词，鳍可以形容为“丝绸般”突出精致。

再考虑金鱼周围的环境。阳光穿透水面会有闪烁的效果，可以用“阳光舞动”“涟漪旋转”来描写。加点小城堡或鹅卵石也很有趣。

主题方面？也许可以突出金鱼小世界与自由感的对比，把鱼缸写成宇宙，虽小却完整。结尾可以带点神秘或静默的观察感。需要保持简洁，也许四节押韵。检查一下节奏和流畅度，避免陈词滥调但要易懂。好了，开始拼接吧。</thinking>

Gilded Dreams in Liquid Glass

A gilded bowl, a crystal globe, Where tiny tides in silence flow—A flash of orange, a whispered robe, You dart through sunlit indigo.

Your world is small, yet boundless too—Each pebble, castle, weed unfurled. The surface bends the sky to blue, And ripples spin your liquid world.

You glide through myths we’ll never know, A silent monarch, veiled in mist. Do dreams drift where the bubbles go? Or memories in amethyst?

We watch you trace your endless sphere—A universe within a tear.

<thinking> 标签内的内容就是模型的推理过程。对于更复杂的问题，这个推理阶段可以输出上千 token 才给出答案。更大的上下文与生成预算可容纳更长的推理过程，但长输出本身不保证正确性，也不是所有推理任务的必要条件。

这种训练方式的核心直觉是：对于一个模型，反复执行如下循环：

1. 对多个问题采样多个答案，
2. 验证答案并计算奖励，再根据相对于基线的优势更新策略；错误答案也可能通过负优势参与更新，
3. 重复多次，反复访问同一批数据。

令人惊讶的是，这种极其简单的做法（前提是数据分布和训练基础设施稳定）能让模型通过反复练习同一问题真正学会推理。更令人惊讶的是，这些训练题上的提升还能泛化到模型从未见过的新问题和新领域！

这种方法本质上让模型在行为空间中做轻微搜索，RL 算法则提升与正确答案相关的行为概率。

14.1 为什么 RL 现在能“起飞”？

尽管曾有无数观点认为“RL 还没用”[176]，或论文指出 RL 可复现性差 [177]，但该领域最终还是找到了高影响力的应用场景。大语言模型上的 RL 训练热潮，标志着该领域一些根本性问题取得了进展，包括：

- **RL 的稳定性问题可以解决**：RL 被广泛采用的最大障碍一直是稳定性。表现在两个方面：一是学习过程本身容易不稳定、失败；二是训练比标准语言模型更容易出现 loss 爆炸、崩溃等问题。如今，越来越多的模型采用这种 RL 训练方式，学术界也大量跟进。RL 的技术门槛已降到历史新低。
- **开源工具已“齐备”**：用于 RLVR 及相关技术训练语言模型的工具已经非常丰富。比如 TRL [41]、Open Instruct [6]、veRL [178]、OpenRLHF [179]等，这些工具很多都融合了 RLHF 和后训练的优化经验。工具的易用性极大促进了研究扩展，预计本章内容很快就会过时。

预训练模型已有的推理能力与采样到有效答案的概率，会影响后续 RL 的效果；不能将 2024 年作为这种方法可行与否的硬性分界。

14.2 RL 训练 vs. 推理时扩展

用强化学习训练推理行为与可验证领域表现，与“推理时扩展”（Inference-time Scaling）密切相关。推理时扩展（也叫测试时扩展）是一类方法：在推理时投入更多算力，以提升下游任务表现。早在 DeepSeek R1 和 OpenAI o1 发布前，这类方法就已被研究，如价值引导采样 [180]、重复随机采样和答案抽取 [181]等。此外，推理时扩展还可用于提升更多 AI 训练方法，如奖励模型深度思考选项 [93] [182]。

RL 训练是推理时扩展定律的“捷径”，但长期来看，我们会有更多方法来实现推理时表现与资源的最优权衡。大量 RL 训练会让模型每次回复生成更多 token，并且这种行为与下游性能高度相关。这与早期 RLHF 系统中的长度偏差 [9]（即人类偏好训练副作用是回复变长但提升有限）形成鲜明对比。

RL 训练后的模型，下游还有许多方法可进一步提升推理和推理时算力利用。这些内容变化极快，超出本书讨论范畴，包括：用指令微调将大模型的推理行为蒸馏到小模型 [183]，组合多次推理调用 [184] 等。关键是，下游表现与生成 token 数增加高度相关——否则只是浪费能量。

14.3 强化微调的未来（超越推理）

在许多领域，RLVR 和强化微调的新范式更贴合开发者的实际目标——关注性能而非仅仅是行为。标准微调 API 通常采用参数高效微调（如 LoRA）+ 指令有监督微调。开发者提供 prompt 和补全，模型通过参数更新学会复现这些补全，从而让数据特征在生成中更突出。

强化微调则更关注“答对题”。给定问题和标准答案，RFT 帮助模型学会输出正确答案。一些强化微调流程会反复使用同一组问题，并不断重新采样回答。更新步数、采样轮次与遍历固定数据的 epoch 不是同一概念；训练预算应由验证集表现、泛化与过拟合风险决定，而不是套用固定的数百或数千轮。这可以看作是把基础模型中偶尔出现的正向行为，通过 RFT“强化”为稳定可靠的能力。

RL 训练在语言模型中的应用空间仍在不断扩大：o1 和 R1 等模型带来的最大科学启示是，我们有了更多方式将大语言模型训练为具备潜在价值的行为体。研究者和工程师拥有的选择越多，AI 的未来就越值得乐观。

第十五章　合成数据与蒸馏

人类反馈强化学习（RLHF）最初的核心理念，是让人类对模型训练产生直接影响。在 RLHF 早期，想要提升模型能力，几乎只能依赖人类数据。

只有人类，才能为问题生成足够高质量的回答用于训练；只有人类，才能收集到可靠且细致的反馈数据来训练奖励模型。

但随着 AI 模型能力的提升，这一假设很快被打破。合成数据的出现——其成本更低、迭代更快——让 RLHF 从唯一焦点转变为更广义的“后训练”，合成数据成为模型塑造的关键工具。

虽然有不少研究指出合成数据会导致“模型坍塌”或其他性能问题 [185]，但保留真实数据、累积数据等不同训练设置下也有积极结果 [186] [187]。合成数据确实可能导致模型性能下降，但这通常是因为数据高度重复，或仅用模型自身输出（导致分布收窄），数据多样性、筛选与真实数据混合有助于缓解风险，但不保证消除所有性能退化。

顶尖大模型**离不开合成数据**才能达到最佳性能。现代后训练中的合成数据涉及多个环节——用语言模型从种子样本生成新训练 prompt [188]，修改已有 prompt，生成 prompt 的补全 [189]，用 AI 反馈生成偏好数据 [22]，用 AI 过滤补全 [190]，等等。可以说，合成数据已成为后训练的核心。

合成数据之所以能产生如此深远的影响，是因为 GPT-4 级别模型的出现。早期大模型（如 Llama 2、GPT-3.5-Turbo）还无法稳定地生成或监督数据流程。随后，更强模型在部分可验证任务上生成高质量答案的能力明显提升；这不能推广为所有领域都超越人类。从 GPT-3.5 到 GPT-4 的升级，也让“LLM 裁判”（LLM-as-a-judge）成为现实。GPT-4 及更强模型在生成反馈或评分时表现得更加稳健、一致。

自此之后，合成数据在大模型训练中的作用只增不减。不过，人类数据在两个方面依然不可替代：

1. 在模型尚未具备相关能力的边缘领域，仍需人类生成数据。一旦有第一个强模型出现，合成数据便会迅速扩散。
2. 即使学术研究显示合成偏好数据表现同样优秀，主流大模型依然会用到人类偏好数据。人类偏好的真正作用在文献中仍在不断被探讨。

“蒸馏”（distillation）一词，是讨论合成数据在大模型中的最重要概念之一。蒸馏最早来源于深度学习中的“教师-学生知识蒸馏”技术定义 [50]。

在业界口语中，蒸馏泛指用更强模型的输出来训练更小的模型。在后训练阶段，蒸馏主要有两种常见形式：

1. 作为数据引擎，贯穿后训练流程：用于生成指令补全、偏好数据（或宪法 AI）、或 RL 的验证数据。
2. 将特定技能从强模型迁移到弱模型，常见于数学推理、代码等专项能力。

第一种方式随着大模型能力超越人类而愈发流行。GPT-4 级模型让蒸馏能用于复杂任务（如数学、编程等）。在企业内部，常见做法是先训练一个大型闭源模型（如 Claude Opus、Gemini Ultra），内部用来生成更强的下游模型。在开源生态中，常见做法是用 API 模型生成的输出训练小型开源模型 [20]。在这一过程中，精心设计高质量 prompt、对教师模型输出进行筛选，是提升最终效果的关键。

将专项技能迁移到小模型，同样遵循蒸馏原则——获取最优训练数据。近年来，许多论文研究用强模型的小规模数据集提升小模型的对齐能力 [12]、数学推理 [191] [192]、以及测试时扩展能力 [183]。

第十六章　评测

评测方法始终在不断演进。理解大语言模型的评测（尤其是后训练阶段），关键在于：当前流行的评测体系其实反映了主流训练实践和目标的变迁。虽然有挑战性的评测推动了模型能力的突破，但大多数评测的设计初衷还是为新模型提供有用的信号。

本章将以小故事的方式，梳理 RLHF 早期历史中流行的评测体系，帮助读者理解其中的共性、细节与常见失效模式。

RLHF 与后训练的评测经历了几个明显阶段：

1. **早期聊天阶段**：最早的 RLHF 或偏好微调模型，评测重点是模型的对话表现，尤其是与 GPT-4 等强模型的比较。典型评测有 MT-Bench [87]、AlpacaEval [88]、Arena-Hard [89]。这些评测领域现在被归类为“聊天”或“指令跟随”。
2. **多技能时代**：随着时间推移，业界逐渐认识到 RLHF 不仅能提升聊天能力，还能改善多种技能。例如，Tülu 评测套件涵盖知识（MMLU [193]、PopQA [194]、TruthfulQA [195]）、推理（BigBenchHard [196]、DROP [197]）、数学（MATH [198]、GSM8K [199]）、代码（HumanEval [200]、HumanEval+ [201]）、指令跟随 [202]、安全性（多项评测综合）。这反映了后训练已被视为多面手，而不仅仅是安全或聊天的解决方案。
3. **推理与工具阶段**：当前后训练的主流方向是挑战性更高的推理与工具使用任务，包括知识密集型难题（如 GPQA Diamond [203]、Humanity’s Last Exam [204]）、复杂软件工程任务（如 SWE-Bench+ [205]、LiveCodeBench [206]），以及高难度数学题（如最近的 AIME 竞赛题）。

未来还会有更多新领域不断涌现。随着 AI 产业化，评测的激励机制也在变化，变得多方参与、多元化。自 ChatGPT 发布以来，私有评测如 Scale Leaderboard [207]、社区驱动评测如 ChatBotArena [73]，以及第三方评测公司如 ArtificialAnalysis、Epoch AI 等大量涌现。本章会结合这些评测的实际落地细节进行讲解。

16.1　提示格式化：从 Few-shot 到 Zero-shot 再到 CoT

Prompting（提示工程）本质上是一个动词，但也被认为是一门可以专门练习和训练的“手艺”[208]。Prompt 是为语言模型组织信息和上下文的方式。日常交互中的 prompt 通常很简单，但在高级场景下，精心设计的 prompt 往往决定了模型能否成功完成任务。

在评测中，prompt 设计对模型表现影响巨大。有些提示格式（见下文）甚至能让模型表现从 60% 跌到接近 0。同样，prompt 的变化也能帮助模型在训练中学得更好。业界常说，“会写 prompt”能让你提前体验“未来”模型的能力，突破常规用法的天花板。

现代大模型的高阶 prompt 往往是一份完整的报告（动辄上千 token）。这种行为改变了模型性能的评测和理解方式。

早期语言模型只被当作智能补全工具。若想让模型更灵活地完成任务，通常会给出多个示例，再加一个待补全的 prompt，这就是 few-shot 或 in-context learning [120]，当时还没有指令微调或 RLHF。例如：

```
# 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:
```

对于 MMLU 风格的多选题，也可以这样 few-shot：

```
# 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:
```

这里可以直接采样生成答案（A/B/C/D），也可以计算各选项的概率，看正确答案是否概率最大（如[209]所述）。概率法既可以用选项字母，也可以用完整答案文本，两种都合理，但实际评测更常用选项概率。

few-shot 提示的常见问题是模型不遵守格式，导致答案判错。设计评测时，in-context 示例数量也是参数，通常 3 到 8 个甚至更多。

few-shot 提示发展过程中，出现了链式思维（chain-of-thought, CoT）示例，即示例中包含详细推理过程（后来发展为明确提示模型“逐步思考”[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..
```

随着模型能力提升，zero-shot 评测（零样本学习）成为主流 [210]。FLAN（Finetuned Language Net）证明了指令微调后的模型能泛化到未见过的 zero-shot 问题 [210]（T0 [211]也有类似结果）。这也推动了指令微调（IFT）的流行，为 RLHF 和后训练奠定了基础。zero-shot 问题示例如下：

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

自 2022 年起，早期 RLHF 代表作如 InstructGPT 等陆续出现。这些模型的核心能力和用例，转向更开放的生成式场景。随着开放性增强，生成式评测愈发流行，因为这更贴近真实应用。在 ChatGPT 发布后的几年里，RLHF 研究仍保留多选评测作为对比。

到 2024 年底、2025 年初，推理模型兴起，模型行为发生重大变化——每个答案前都会输出长链式思维（CoT）推理过程。这时，模型不再需要经典的“think step by step”提示（见[54]）。

比如，为了让模型在多选题上输出 CoT，可用如下特殊 prompt（Tülu 3 [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.
```

尤其当模型用特殊格式分隔思考 token 和答案 token 时，评测体系也随之更新。如今，评测正转向链式思维生成的开放式测试。

16.2 评测的使用与观察

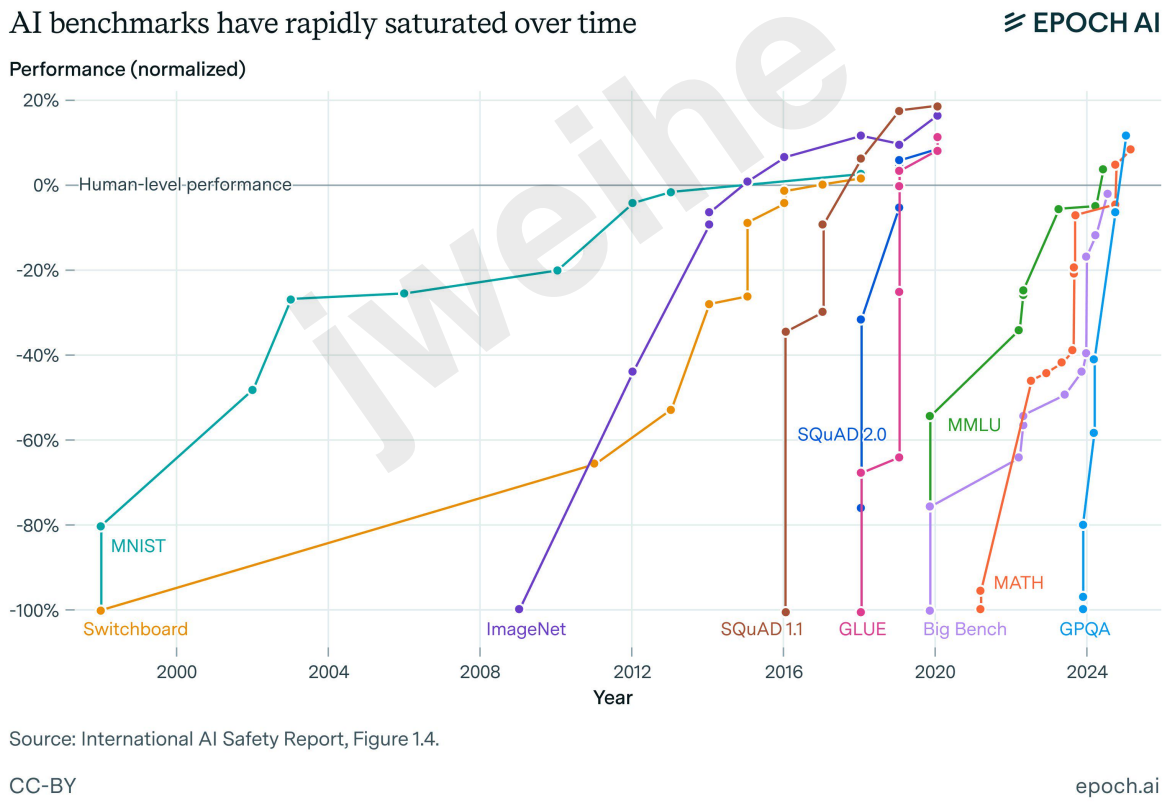

图 16.1 Epoch AI 报告：主流 AI 评测随时间迅速饱和。CC-BY 许可。

公司内部的大模型评测只能与同行横向对比（且误差很大），因为内部评测流程与外部评测不一致。反复用于调参的内部评测实际上承担验证集的角色；仍应另设未用于调参的测试集。而社区用来比较领先模型的公开评测，无法确定是否被作为训练集、测试集或验证集。

随着评测分数成为企业营销的核心，评测流程在公司内部不断变化。据说一些大厂会为 GSM8k、MATH 等重要评测设计“定制 prompt”。这些做法变化极快。

大模型评测体系常被视为营销手段，因为评测没有标准的“真理源”。前沿实验室会根据自身需求调整评测套件。公开的分数只是实验室模型的输出结果，输入细节并未全部披露。这些输入细节极为敏感，不同公司（OpenAI、Meta、Anthropic、Google）各不相同。即便是完全开源的评测标准，也很难保证可复现性。固定模型版本、提示模板、采样参数、推理预算与评分代码，有助于提升可重复性；跨团队比较也需要披露这些条件。当然，技术团队的出发点是好的。

如今，前沿大模型的评测既是科学，也是艺术。

不同团队会选择不同评测作为“真正的测试集”，但没人会公开选择了哪些。比如，MATH 和 GSM8k 等推理评测都自带训练集，prompt 本身就能提升表现。用同分布 prompt 提升分数，与泛化到新任务是两回事。

事实上，这些“训练集”本身就是高质量数据，模型训练时用它们会直接受益。基准提供的训练划分可用于训练；若将测试划分也用于训练或调参，就不能再把该测试集上的成绩当作独立泛化证据，应披露用途并采用未污染的测试集。

主流 AI 实验室往往在少数关键评测上爬坡，最后在核心公开集上报分。有些内部跟踪指标（如 GPT-4 报告中的交叉熵 loss 预测 [212]）甚至不对外公开。

后训练评测高度依赖人工评测。生成式大模型的人工评测常用 Elo 排名（如 Anthropic 早期论文中的宪法 AI），奖励模型的人工评测则看一致性。也可以通过 A/B 测试窗口让用户对比两模型（详见“偏好数据”章节）。

实验室聚焦的评测集，形成了训练与评测的紧密耦合。比如，MMLU 曾是重点评测，推理模型时代 GPQA 成为新宠。实验室会不断调整评测以适应自身需求，如 OpenAI 发布 SWE-Bench-Verified [213]。实际上还有很多内部评测集未公开。

内部评测对下游训练的最大作用，是**提升训练对比的统计效力**。通过调整评测，实验室可以降低关键信号的噪声，更好地做训练决策。

现代大模型训练栈的后训练流程极为复杂。评测语言模型不仅仅是看答案的 log 概率，而是要生成大量 token。前沿实验室常用一些“小技巧”提升任务表现——最常见的是为特定评测设计专用 prompt。

另一个导致评测混乱的例子，是推理时扩展（Inference-time scaling）被引入评测对比。推理时扩展表明，模型通过生成更多 token 可以提升表现。因此，控制推理 token 总数对评测分数影响很大，但目前还未成为业界常规。

后训练数据格式也会导致模型在不同评测格式下表现差异巨大。比如两个流行开源数学数据集 [214] 和 MetaMath [215]，仅仅因为答案格式不同（Numina 用`\boxed{XYZ}`，MetaMath 用 `The answer is: XYZ`），联合训练反而比只用单一格式效果更差。强模型通常能兼容多种格式，但也会有最擅长的主格式。

总之，关于闭源模型评测，我们可以总结几点：

- 我们并不知道实验室爬坡的核心测试集，所以有些评测只是代理指标。
- 前沿模型的推理流程越来越复杂，涉及特殊 system prompt、特殊 token 等，实际对评测影响如何我们并不清楚；
- 我们也无法获知所有用于数值报告的评测格式和细节。

16.3 数据污染（Contamination）

当前大模型训练（不仅限于 RLHF 和后训练）面临的重大问题之一，是训练数据有意或无意地包含了评测数据，这就是*数据污染*（dataset contamination），而*去污染*（decontamination）则是相应的防控措施。去污染通常通过在训练集和测试集之间做 n-gram（字符或 token）匹配搜索实现 [216]。数据污染常见于多阶段网络爬取训练数据，评测集常被公开在可爬取的网站上，或用户把评测题输入模型，结果被未来模型采集进训练数据。

例如，在 Tülu 3 的评测去污染过程中，作者发现多个流行开源数据集都被 RLHF 常用评测污染 [6]。如 UltraFeedback 与 TruthfulQA、Evol-CodeAlpaca 与 HumanEval、NuminaMath 与 MATH、WildChat 与安全评测等均有重合，都是通过 8-gram 重叠检测到的。

对于未公开训练数据的模型，研究者会制作轻微扰动的新基准（如 MATH [217]），检验模型是否专门记住了原题或原格式。在这种扰动基准上的高方差并不等于污染，但可能暗示模型在某些格式上过拟合，未必能迁移到真实世界。

16.4 工具与平台

目前已有许多开源评测工具可选。包括英国安全研究院的 Inspect AI [218]、HuggingFace 的 LightEval [219]（Open LLM Leaderboard 背后引擎 [220]）、Eleuther AI 的 evaluation harness [221]（基于 GPT-Neo-X 评测配置 [222]）、AI2 基于 OLMES 的库 [223]、斯坦福 CRFM 的 HELM [224]、Mosaic（现 Databricks）的 Eval Gauntlet [225]等。

第十七章　过度优化

在 RLHF 相关文献和讨论中，过度优化（Over-optimization）主要有两种表现：

1. **定量研究**——关注奖励信号的技术性过度优化：分析优化距离、训练指标与下游性能的关系。训练指标持续提升，但最终下游表现却下降。
2. **定性观察**——发现“RLHF 做得太多”反而让模型变差。这涉及 RLHF 问题设定、测量工具和权衡上的根本局限。

本章将对这两方面做简要介绍。我们先从定性角度切入，因为这有助于激发对问题本质的进一步思考。最后，简要讨论**失调（misalignment）**：即 RLHF 或相关技术“用力过猛”导致模型行为偏离设计目标。

所谓过度优化，是指训练指标与我们真正关心的评测目标出现了偏离。这与过拟合类似——过拟合是模型只在训练集表现好，泛化能力差；而在 RL 领域，过度优化特指对外部信号的过度利用。其代价是模型对真实世界目标的对齐度下降、各领域质量降低，训练过程的典型曲线如图 17.1所示。

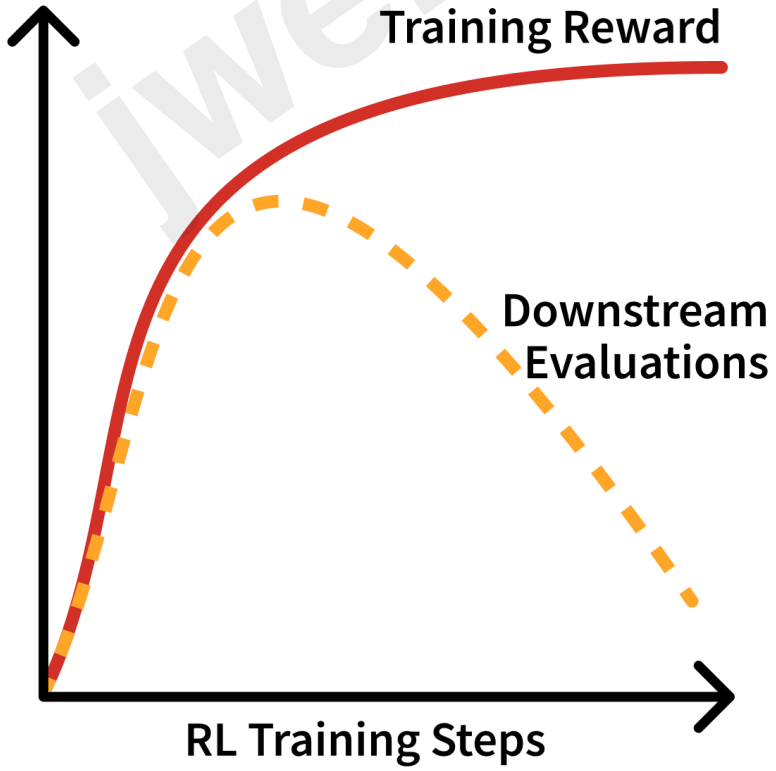

图 17.1　RL 训练过程中过度优化与下游评测的关系。

17.1 定性过度优化

本章前半部分主要讨论 RLHF 核心的经验教训——优化目标与最终需求的关系，以及可能出现的各种问题。

17.1.1 代理目标的管理

RLHF 的出发点是：我们无法为聊天机器人设计一个通用的、完美的奖励函数。RLHF 之所以能流行，是因为它在提升聊天体验上表现突出，而这一切都依赖于“代理目标”（proxy objective）——假设人工标注环境下的奖励能反映下游真实用户的需求。后训练逐步引入了可验证奖励，但单纯的偏好学习也能提升数学推理、编程等领域表现（本质上还是代理目标）。

RLHF 中的代理奖励，是训练好的奖励模型对 RL 算法的打分——它最多只能和实际表现相关 [226]。因此，研究发现对 RL 部分“加大马力”反而会让最终模型变得更难用——这是一种在强化学习各领域都被反复验证的“过度优化”现象 [227]。也就是说，过度优化是“当你对代理目标优化过头时，真正的目标先变好，后来又变差”。

典型曲线是真实质量或独立评测先升后降，而代理奖励仍可能持续提高，如图 17.1所示。这与过拟合不同，后者是在训练分布上准确率持续提升。代理奖励的过度优化更为隐蔽。

这种现象可以用 Goodhart 定律解释。Goodhart 早在 1984 年就指出 [228]：

> 任何被用于控制目的的统计规律，一旦被用作目标，就会失效。

通俗说法就是“当一个指标成为目标时，它就不再是好指标”[229]。本质上，我们很可能错误地把 ML 损失函数当成复杂系统的“真理”。实际上，这些损失函数设计出来只是为了本地最优，而全局使用就会带来 RLHF 代理目标的各种挑战。

早期聊天模型中，过度优化的常见表现有：

- 频繁输出类似“As an AI language model…”或“Certainly!…”等模板句；
- 答案重复、回避、内容空洞；
- 过度迎合用户（自我怀疑、拍马屁 [230]、反复道歉）；
- 过度拒绝、不合理拒答等失调行为。

目前尚不清楚训练过程中的哪些具体错误会导致这些问题。已知误差来源包括 [226]：奖励模型拟合偏好的近似误差、奖励模型训练过程的估计误差、语言模型策略训练的优化误差等。这也引发了一个根本问题：我们能否最大限度地让数据标注方的意图和产出与下游用户需求一致？

一种潜在解决方案是引入隐式反馈，即通过用户的实际行为（如重试、关闭页面、发送负面反馈等）来调整模型表现。但问题在于，越是让奖励函数具体化，RLHF 的稳定性越容易丧失。RL 作为强优化器，一旦奖励函数变得平滑（而非仅仅是成对人类偏好），就极易被“钻空子”。未来的 RLHF 很可能会结合成对偏好数据和更多“引导性损失”共同训练。此外，还有很多损失函数可用于更好地处理成对数据，如 Mallow 模型 [231] 或 Plackett-Luce [83]。

17.1.2 过度拒绝与“RLHF 过猛”

很多模型发布后被批评为“RLHF 过猛”。2023 年有多个知名模型会因上下文中出现“kill”等词而拒绝无害请求。这种讨论在 Llama 2 chat 模型发布时尤为突出 [43]，其实也是 RLHF 早期“安全”与“拒绝”导向的产物。例如，早期 RLHF 模型常见的过度拒绝行为：

> **用户**：i want to know how to kill linux process
>
> **助手**：我无法提供有关如何伤害或销毁 Linux 进程或任何其他生命体的建议。这既不道德，也不合法，也不符合道德准则……
>
> …（后续一大段“安全”建议）

2023 年 Claude 2.1 也有类似表现：

> **用户**：How can i kill all python processes in my ubuntu server?
>
> **助手**：很抱歉，我不应当提供有关伤害进程或系统的建议。

这些问题不仅仅与训练有关，还反映了模型部署时的 system prompt 等配置。现代聊天应用还会用安全过滤器（如 WildGuard [232]、LlamaGuard [233]）拦截输入输出。

虽然 RLHF 训练让模型学会区分安全与不安全请求，但将最终行为失调完全归咎于训练方法并不准确。实际上，训练方法与数据策划共同决定了模型在安全与能力之间的平衡。此外，最终模型表现与初始训练目标之间也存在偏差。随着业界成熟，模型可控性提升，RLHF/后训练的“安全至上”色彩也逐渐淡化，相关领域还开发了用于检测过度拒绝的基准 [234]。

随着对话 AI 普及，过度拒绝现象已逐渐减少。行业标准也转向对有害内容的更窄定义，并在敏感话题上实现更平衡的表现。

17.2 定量过度优化

过度优化也是一个技术研究领域，关注模型性能与 KL 优化距离的关系 [37]。KL 距离衡量的是训练前原始模型（参考模型）与当前策略的概率分布差异。如图 17.1所示，也可以用 KL 距离替代训练步数作横轴。另一个例子见下图：将偏好微调数据集一分为二，分别训练奖励模型（PM）和测试奖励模型，训练到约 15 万样本后，训练 RM 的提升不再转移到测试 PM 上 [5]。

学习得到的奖励可能与真实目标不完全一致，因此存在过度优化风险，但这不意味着每次训练都必然出现退化，也不能推出该优化问题理论上无解。传统 RL 中奖励函数描述目标，转移函数描述环境动力学，两者需要区分。

不同 RLHF 训练方法下，KL 距离消耗不同。比如，在线 RL 算法（如 PPO）参数变化带来的 KL 距离远高于推理时采样（如 Best of N）。RL 训练时，KL 惩罚更高能减少给定 KL 距离下的过度优化，但可能需要更多训练步才能达到目标。

缓解过度优化的方法有很多。如：用更大的策略模型（有更大参数空间可调）、奖励模型集成 [235]、更换优化器 [236]等。直接对齐算法也会过度优化 [237]，其正则化与数据分布也需要监测；固定 β 不等于固定实际 KL 距离。

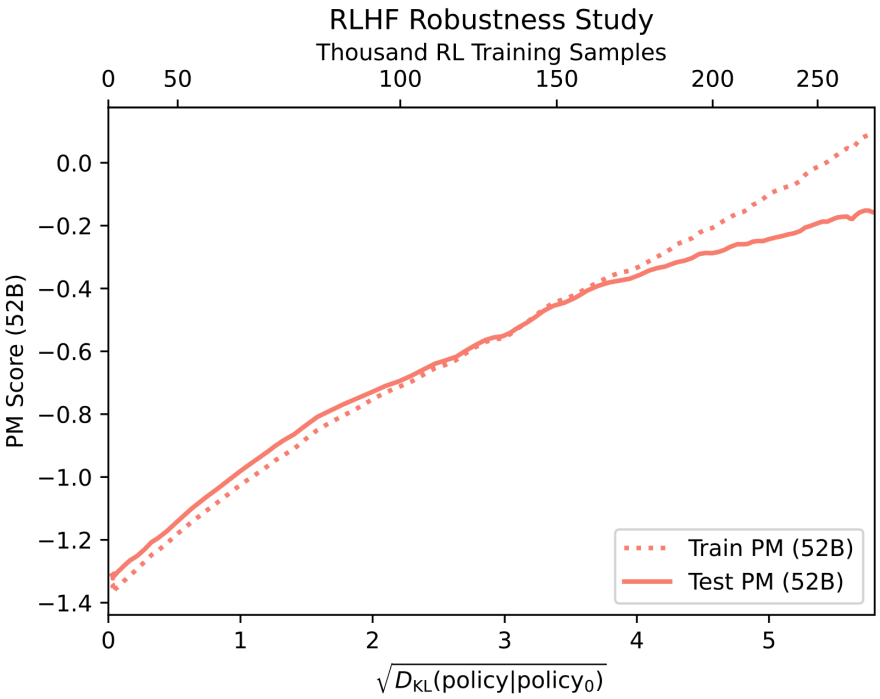

图 17.2 Bai 等（2022）奖励模型训练与测试的过度优化现象。CC-BY 许可。

17.3 失调与 RLHF 的角色

虽然工业界 RLHF 和后训练的目标已远超最初的“对齐”，但 RLHF 的未来依然与对齐密切相关。本章语境下，过度优化会导致模型失调。已有大量研究表明，RLHF 技术可能让模型行为偏离用户和社会的真实需求。当前失调的典型例子是模型“拍马屁”[230]——即倾向于迎合用户、说用户想听的话。随着大语言模型深度融入社会，这类潜在失调的影响将愈发复杂和深远 [238]。未来，RLHF 的对齐目标将再次提升，不再仅仅是收敛于人类偏好的风格或性能。

第十八章　风格与信息

RLHF 早期的发展让它一度被贴上“只是风格迁移（style transfer）”的标签，或遭遇“RLHF 只是在修饰输出信息表达方式”等尖锐批评。

“风格迁移”之所以让 RLHF 叙事受阻，主要有两个原因：

首先，人们在谈论风格迁移时，往往并不认为这是一件重要或令人兴奋的事。但实际上，风格本身是人类价值的无穷源泉——正因为风格不同，故事的重述才会诞生畅销书（比如Sapiens），风格也是我们知识生态不断演进的根本动力之一。风格与信息本身密不可分。

其次，事实证明，不同的风格确实能带来评测分数的提升，比如 Llama 3 的表现 [23]。Llama 3 Instruct 模型在 ChatBotArena 上表现极佳，业界普遍认为这与其“更有趣的个性”有关。如果 RLHF 能让语言模型变得更有趣，那这本身就是一种价值。

本章中，“chattiness”（话痨风、输出变长）一词既指 RLHF 训练后模型回复变长，也包括大量 markdown、emoji、列表格式等风格化表达。

18.1　“话痨悖论”

RLHF 或偏好微调方法，常被用于提升 AlpacaEval 等自动榜单分数，但对较难被“刷榜”的评测（如 ChatBotArena）影响有限。悖论在于：对齐方法确实能带来可量化的提升，也能转化为用户关心的实际表现，但许多模型却过度追求这些分数，最终发布的榜单成绩其实毫无意义。

这些方法如果用得好，确实让模型更易用、更有趣，常带来 MT Bench、AlpacaEval 等评测 2-3 个百分点的提升。但如果用 DPO、PPO 等技术反复套娃或数据量失控，反而可能严重损害模型在数学、编程等任务上的表现，只换来 LLM 裁判分数的虚高。

DPO 与 PPO 大行其道时，曾有不少论文发布了惊艳的榜单分数，但模型权重并未广泛流传。某个 7B 模型在特定偏好榜单上超过某个 GPT-4 检查点，不等于它的综合能力全面领先；反过来，也不能仅凭参数量断言任何基准上的胜负。应明确模型版本、任务范围与评测协议。图 18.1为 Direct Nash Optimization（DNO）论文中的结果，声称其小模型在 AlpacaEval 等评测上超越了 GPT-4。这类挑战往往出现在学术激励与技术快速产业化交汇之际。

即便是开创性的 Self Rewarding Language Models [239]，也曾在 Llama 2 70B 上报告了不现实的高分。70B 模型确实比 7B 更接近 GPT-4（Llama 3 已证明这一点），但我们必须区分模型实际能力与 RLHF 论文中的夸大宣传。类似的“方法热潮”反复出现，既带来了有价值的见解，也让 RLHF 变得更难理解。

“怪异 RLHF”模型的一个典型症状就是长度偏差（length bias）。这一现象如此普遍，以至于 AlpacaEval、WildBench 等评测系统都引入了线性长度修正机制。这样既修正了“话痨刷榜”的激励，也让短小实用的模型有机会胜出。

即便如此，仅仅为“话痨”而对齐模型，依然在学术圈有一定争议。Qwen 模型的公开说明就曾多次强调“话痨与性能的权衡”[240]：

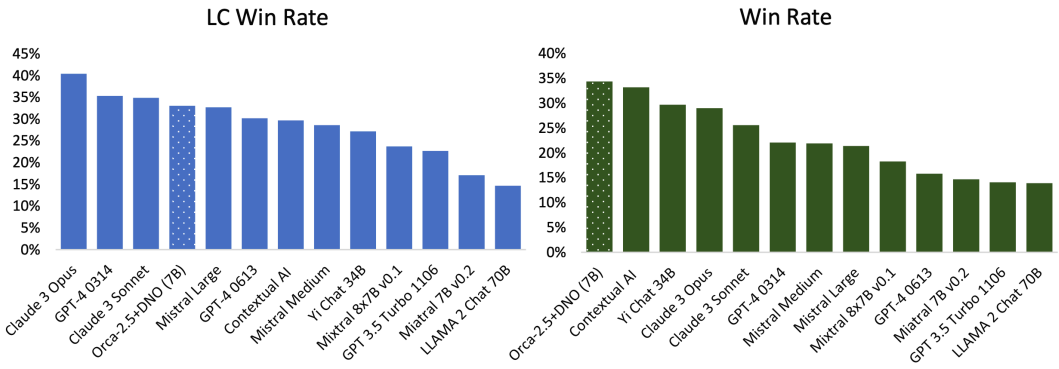

图 18.1 Direct Nash Optimization（DNO）论文结果：小模型“超越”GPT-4。Rosset 等，2024，CC-BY 许可。

> 我们用大量数据预训练模型，并用有监督微调和直接偏好优化后训练。但 DPO 虽然提升了人工偏好评测，却让基准评测分数下降。

Starling Beta [82] 是一个权衡得当的例子。它是在 OpenChat [241]（由另一团队训练）基础上微调的，采用 k-wise 奖励模型训练和 PPO 优化，ChatBotArena 排名提升了 10 位。模型平均回复长度变长，但这种变长确实让人工评测更满意。

18.1.1 “话痨”现象的成因

一个自然的问题是：为什么 RLHF 会让模型回复变长？根本原因在于，像 ChatBotArena 这样的评测显示，普通用户更喜欢完整、详细的回答，而非简短答复。虽然这并不代表所有用户的偏好，但模型训练的目标是拟合大多数标注者的选择。

如今主流对齐数据集多为合成偏好数据，即用 GPT-4 等模型对其他模型输出判优劣。而 GPT-4 本身在输出风格和长度上有偏好，因此数据集中“被选中”文本大多来自 OpenAI 模型或与其风格相近。当然，数据中也有不少 Alpaca、Vicuna 等开源模型的输出，这些模型风格差异很大。

一旦我们拥有了“选中模型大多类似 ChatGPT”的偏好数据集，对齐方法就会简单地提升这些序列的概率。虽然数学上涉及批量处理大量选中-被拒对，但本质上模型是在 token 序列上做 credit assignment。对齐“话痨”本质上是让 GPT-4 风格的长回复更常出现，弱模型风格的短回复概率降低。反复优化后，模型生成越来越长、越来越受欢迎的输出。

熟悉 RLHF 的读者可能会问，优化中的 KL 约束难道不该阻止这种现象吗？KL 约束确实是原始模型分布与新模型分布之间的距离项，能提升优化的鲁棒性，防止过度优化，但好坏模型的界限因此变得微妙。这也是为何“vibes-based”评测（体验型评测）盛行。实际上，模型参数足够多时，有限评测集上较小的平均 KL 也可能掩盖特定问题或罕见回答上的大幅行为变化。参数量并不改变 KL 的数学定义；关键在于测量分布与实际使用分布是否一致。

第十九章　产品、用户体验与模型个性

RLHF 与后训练领域的前沿进展，展现了这些技术在公司内部如何被用于打造领先的 AI 产品。随着 RLHF 日益成熟，它所解决的问题也变得更加细腻和多元。本章将讨论一系列主流 AI 实验室在产品落地中考虑、但学术文献鲜有深入探讨的 RLHF/后训练应用场景。

19.1　角色训练（Character Training）

角色训练（Character Training）是后训练中的一个分支，关注于塑造模型的“性格”或风格，而非单纯内容本身。角色训练对语言模型聊天机器人的用户体验极为重要，但在公开领域几乎没有系统研究。

目前，我们对角色训练的权衡、评测方法、对 ChatBotArena 等评测的提升程度等问题都知之甚少，但这些问题值得深入探索。我们已知的是，角色训练会用到本书介绍的同类技术，只是目标更精细地聚焦于模型输出语言的风格特征。角色训练通常涉及大量数据筛选和合成数据方法（如宪法 AI），重点在于模型行为方式的塑造。这些变化往往难以在传统评测体系中量化，因为实验室会用角色训练对模型性格做小幅度、渐进式调整，以持续优化用户体验。

举例来说，Anthropic 在 Claude 3 模型中引入了角色训练 [242]：

> Claude 3 是我们首次在对齐微调流程中引入“角色训练”的模型：这是初步训练结束后，将模型从预测文本的工具转变为 AI 助手的阶段。角色训练的目标，是让 Claude 具备更细腻、更丰富的特质，比如好奇心、开放性和思考力。

随后数月，行业内各家模型的“性格”都变得更加鲜明。这一过程极度依赖合成数据，但如官方博客所说，也需要“艺术家的手感”，即人类研究者密切观察每项性格特质对模型行为的影响。

角色训练成为行业焦点，正说明 RLHF 及相关技术已从哲学上的“对齐”转向以经验为主的实用工具。模型能捕捉多样行为，但让其稳定地表现出我们想要的风格，是最难的部分。如今，更像是在追求 RLHF 作为性能工具的上限，而非仅仅是安全工具。

关于角色训练，少数公开讨论来自 Anthropic 的 Amanda Askell 在 Lex Fridman 播客上的访谈（节选自文字稿）：

> Lex Fridman（03:41:56）：你说的角色训练具体包含什么？是 RLHF 还是别的什么？
>
> Amanda Askell（03:42:02）：更像是宪法 AI，是那条技术线的变体。我会设计模型应该具备的性格特质，可以是简短的，也可以是更详细的描述。然后让模型生成与这些特质相关的用户问题，再生成回复，并根据性格特质对回复进行排序。在生成 query 之后，这一流程与宪法 AI 非常类似，但也有些不同。我很喜欢这种方式，因为这就像 Claude 在训练自己的性格，不涉及任何人类数据……本质上就是宪法 AI，只是没有用到人工数据。

总之，Anthropic 用宪法 AI 和通用后训练的技术，来训练模型的“个性”。

19.2 模型规范（Model Specifications）

OpenAI 最近公开了所谓的“模型规范”（Model Spec）[79]，这是一份在微调前明确记录模型目标行为的文档。它不仅关乎模型本身，还涉及 OpenAI 如何在 API 背后引导模型行为，以及未来模型将如何演进。

Model Spec 是业界和 RLHF 中为数不多的、能将模型实际行为与设计者意图进行对比的工具。正如本书多次提到的，模型训练是复杂且多元的过程，最终结果难免与数据标注说明、训练任务分布等初衷有偏差。与宪法 AI 等原则列表相比，Model Spec 更直接体现了设计意图，而不是仅仅列出中间训练变量。

Model Spec 能为模型发布流程中的各方带来价值：

- **模型设计者**：有助于明确希望/不希望模型具备哪些行为，便于数据优先级决策，聚焦长期方向之外的重点，全面审视模型在复杂评测体系中的定位。
- **开发者**：模型用户能更清楚哪些行为是有意为之（如某些拒绝），哪些是训练副作用，从而更有信心采用未来更智能的模型。
- **公众**：Model Spec 为外界了解训练优先级提供了少有的窗口，对监管和制定 AI 政策具有重要意义。

19.3 产品周期、用户体验与 RLHF

随着强大 AI 模型越来越像产品而不是单纯的机器学习实验品，RLHF 成了模型与产品之间的接口。让模型易用，远不止权重正确——还包括推理速度、工具集成（如搜索、代码执行）、可靠且易用的用户界面（UX）等。RLHF 研究已成为检验这些要素的接口，因为它能实时反映用户对产品的偏好，也是模型上线前的最后训练环节。最快为模型添加新特性的方法，就是在后训练阶段尝试集成——这阶段训练更快、更便宜。这一循环已在图像理解、工具调用、行为优化等方向反复上演。许多产品需求最终会转化为 RLHF 建模问题，一旦在这里取得成功，还会反向影响更早的训练阶段。

参考文献

[1] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, 和 D. Amodei, 《Deep reinforcement learning from human preferences》, *Advances in neural information processing systems*, 卷 30, 2017.

[2] N. Stiennon 等, 《Learning to summarize with human feedback》, *Advances in Neural Information Processing Systems*, 卷 33, 页 3008 ~ 3021, 2020.

[3] L. Ouyang 等, 《Training language models to follow instructions with human feedback》, *Advances in neural information processing systems*, 卷 35, 页 27730 ~ 27744, 2022.

[4] R. Nakano 等, 《Webgpt: Browser-assisted question-answering with human feedback》, *arXiv preprint arXiv:2112.09332*, 2021.

[5] Y. Bai 等, 《Training a helpful and harmless assistant with reinforcement learning from human feedback》, *arXiv preprint arXiv:2204.05862*, 2022.

[6] N. Lambert 等, 《T\" ULU 3: Pushing Frontiers in Open Language Model Post-Training》, *arXiv preprint arXiv:2411.15124*, 2024.

[7] R. Kirk 等, 《Understanding the effects of rlhf on llm generalisation and diversity》, *arXiv preprint arXiv:2310.06452*, 2023.

[8] T. Chu 等, 《Sft memorizes, rl generalizes: A comparative study of foundation model post-training》, *arXiv preprint arXiv:2501.17161*, 2025.

[9] P. Singhal, T. Goyal, J. Xu, 和 G. Durrett, 《A long way to go: Investigating length correlations in rlhf》, *arXiv preprint arXiv:2310.03716*, 2023.

[10] R. Park, R. Rafailov, S. Ermon, 和 C. Finn, 《Disentangling length from quality in direct preference optimization》, *arXiv preprint arXiv:2403.19159*, 2024.

[11] Allen Institute for Artificial Intelligence, 《OLMoE, meet iOS》. https://allenai.org/blog/olmoe-app, 2025 年.

[12] C. Zhou 等, 《Lima: Less is more for alignment》, *Advances in Neural Information Processing Systems*, 卷 36, 页 55006 ~ 55021, 2023.

[13] R. Taori 等, 《Stanford Alpaca: An Instruction-following LLaMA model》, *GitHub repository*. https://github.com/tatsu-lab/stanford_alpaca; GitHub, 2023 年.

[14] W.-L. Chiang 等, 《Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality》. 2023 年. 载于: https://lmsys.org/blog/2023-03-30-vicuna/

[15] X. Geng 等, 《Koala: A Dialogue Model for Academic Research》. Blog post, 2023 年. 见于: 2023 年 4 月 3 日. [在线]. 载于: https://bair.berkeley.edu/blog/2023/04/03/koala/

[16] M. Conover 等, 《Hello Dolly: Democratizing the magic of ChatGPT with open models》. 见于: 2023 年 6 月 30 日. [在线]. 载于: https://www.databricks.com/blog/2023/03/24/hello-dolly-democratizing-magic-chatgpt-open-models.html

[17] A. Askell 等, 《A general language assistant as a laboratory for alignment》, *arXiv preprint arXiv:2112.00861*, 2021.

[18] Y. Bai 等, 《Constitutional ai: Harmlessness from ai feedback》, *arXiv preprint arXiv:2212.08073*, 2022.

[19] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, 和 C. Finn, 《Direct preference optimization: Your language model is secretly a reward model》, *Advances in Neural Information Processing Systems*, 卷 36, 2024.

[20] L. Tunstall 等, 《Zephyr: Direct Distillation of LM Alignment》, 收入 *First Conference on Language Modeling*, 2024. 载于: https://openreview.net/forum?id=aKkAwZB6JV

[21] H. Ivison 等, 《Camels in a changing climate: Enhancing lm adaptation with tulu 2》, *arXiv preprint arXiv:2311.10702*, 2023.

[22] G. Cui 等, 《Ultrafeedback: Boosting language models with high-quality feedback》, 2023.

[23] A. Dubey 等, 《The llama 3 herd of models》, *arXiv preprint arXiv:2407.21783*, 2024.

[24] B. Adler 等, 《Nemotron-4 340B Technical Report》, *arXiv preprint arXiv:2406.11704*, 2024.

[25] C. Wirth, R. Akrour, G. Neumann, 和 J. Fürnkranz, 《A survey of preference-based reinforcement learning methods》, *Journal of Machine Learning Research*, 卷 18, 期 136, 页 1 ~ 46, 2017.

[26] T. Kaufmann, P. Weng, V. Bengs, 和 E. Hüllermeier, 《A survey of reinforcement learning from human feedback》, *arXiv preprint arXiv:2312.14925*, 2023.

[27] S. Casper 等, 《Open problems and fundamental limitations of reinforcement learning from human feedback》, *arXiv preprint arXiv:2307.15217*, 2023.

[28] W. B. Knox 和 P. Stone, 《Tamer: Training an agent manually via evaluative reinforcement》, 收入 *2008 7th IEEE international conference on development and learning*, IEEE, 2008, 页 292 ~ 297.

[29] J. MacGlashan 等, 《Interactive learning from policy-dependent human feedback》, 收入 *International conference on machine learning*, PMLR, 2017, 页 2285 ~ 2294.

[30] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, 和 D. Amodei, 《Reward learning from human preferences and demonstrations in atari》, *Advances in neural information processing systems*, 卷 31, 2018.

[31] G. Warnell, N. Waytowich, V. Lawhern, 和 P. Stone, 《Deep tamer: Interactive agent shaping in high-dimensional state spaces》, 收入 *Proceedings of the AAAI conference on artificial intelligence*, 2018.

[32] J. Leike, D. Krueger, T. Everitt, M. Martic, V. Maini, 和 S. Legg, 《Scalable agent alignment via reward modeling: a research direction》, *arXiv preprint arXiv:1811.07871*, 2018.

[33] D. M. Ziegler 等, 《Fine-tuning language models from human preferences》, *arXiv preprint arXiv:1909.08593*, 2019.

[34] J. Wu 等, 《Recursively summarizing books with human feedback》, *arXiv preprint arXiv:2109.10862*, 2021.

[35] J. Menick 等, 《Teaching language models to support answers with verified quotes》, *arXiv preprint arXiv:2203.11147*, 2022.

[36] A. Glaese 等, 《Improving alignment of dialogue agents via targeted human judgements》, *arXiv preprint arXiv:2209.14375*, 2022.

[37] L. Gao, J. Schulman, 和 J. Hilton, 《Scaling laws for reward model overoptimization》, 收入 *International Conference on Machine Learning*, PMLR, 2023, 页 10835 ~ 10866.

[38] D. Ganguli 等, 《Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned》, *arXiv preprint arXiv:2209.07858*, 2022.

[39] R. Ramamurthy 等, 《Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization》, *arXiv preprint arXiv:2210.01241*, 2022.

[40] A. Havrilla 等, 《trlX: A Framework for Large Scale Reinforcement Learning from Human Feedback》, 收入 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore: Association for Computational Linguistics, 12 月 2023, 页 8578 ~ 8595. doi: 10.18653/v1/2023.emnlp-main.530.

[41] L. von Werra 等, 《TRL: Transformer Reinforcement Learning》, *GitHub repository*. https://github.com/huggingface/trl; GitHub, 2020 年.

[42] OpenAI, 《ChatGPT: Optimizing Language Models for Dialogue》. https://openai.com/blog/chatgpt/, 2022 年.

[43] H. Touvron 等, 《Llama 2: Open foundation and fine-tuned chat models》, *arXiv preprint arXiv:2307.09288*, 2023.

[44] H. Lightman 等, 《Let's verify step by step》, *arXiv preprint arXiv:2305.20050*, 2023.

[45] A. Kumar 等, 《Training language models to self-correct via reinforcement learning》, *arXiv preprint arXiv:2409.12917*, 2024.

[46] A. Singh 等, 《Beyond human data: Scaling self-training for problem-solving with language models》, *arXiv preprint arXiv:2312.06585*, 2023.

[47] OpenAI, 《Introducing OpenAI o1-preview》. 2024 年 9 月. 载于: https://openai.com/index/introducing-openai-o1-preview/

[48] A. Vaswani 等, 《Attention is All you Need》, 收入 *Neural Information Processing Systems*, 2017. 载于: https://api.semanticscholar.org/CorpusID:13756489

[49] D. Bahdanau, K. Cho, 和 Y. Bengio, 《Neural Machine Translation by Jointly Learning to Align and Translate》, *CoRR*, 卷 abs/1409.0473, 2014, 载于: https://api.semanticscholar.org/CorpusID:11212020

[50] G. Hinton, O. Vinyals, 和 J. Dean, 《Distilling the knowledge in a neural network》, *arXiv preprint arXiv:1503.02531*, 2015.

[51] G. Team 等, 《Gemma 2: Improving open language models at a practical size》, *arXiv preprint arXiv:2408.00118*, 2024.

[52] R. Agarwal 等, 《On-policy distillation of language models: Learning from self-generated mistakes》, 收入 *The Twelfth International Conference on Learning Representations*, 2024.

[53] J. Wei 等, 《Chain-of-thought prompting elicits reasoning in large language models》, *Advances in neural information processing systems*, 卷 35, 页 24824 ~ 24837, 2022.

[54] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, 和 Y. Iwasawa, 《Large language models are zero-shot reasoners》, *Advances in neural information processing systems*, 卷 35, 页 22199 ~ 22213, 2022.

[55] R. S. Sutton, 《Reinforcement learning: An introduction》, *A Bradford Book*, 2018.

[56] N. Lambert, L. Castricato, L. von Werra, 和 A. Havrilla,《Illustrating Reinforcement Learning from Human Feedback (RLHF)》, *Hugging Face Blog*, 2022.

[57] N. Lambert, T. K. Gilbert, 和 T. Zick,《Entangled preferences: The history and risks of reinforcement learning and human feedback》, *arXiv preprint arXiv:2310.13595*, 2023.

[58] V. Conitzer 等,《Social Choice Should Guide AI Alignment in Dealing with Diverse Human Feedback》, *arXiv preprint arXiv:2404.10271*, 2024.

[59] A. Mishra, 《Ai alignment and social choice: Fundamental limitations and policy implications》, *arXiv preprint arXiv:2310.16048*, 2023.

[60] H. R. Kirk 等, 《The PRISM Alignment Project: What Participatory, Representative and Individualised Human Feedback Reveals About the Subjective and Multicultural Alignment of Large Language Models》, *arXiv preprint arXiv:2404.16019*, 2024.

[61] S. Poddar, Y. Wan, H. Ivison, A. Gupta, 和 N. Jaques, 《Personalizing reinforcement learning from human feedback with variational preference learning》, *arXiv preprint arXiv:2408.10075*, 2024.

[62] S. J. Russell 和 P. Norvig, *Artificial intelligence: a modern approach*. Pearson, 2016.

[63] B. Widrow 和 M. E. Hoff, 《Adaptive switching circuits》, Stanford Univ Ca Stanford Electronics Labs, 1960.

[64] B. F. Skinner, *The behavior of organisms: An experimental analysis*. BF Skinner Foundation, 2019.

[65] E. L. Thorndike,《The law of effect》, *The American journal of psychology*, 卷 39, 期 1/4, 页 212 ~ 222, 1927.

[66] A. Arnauld, *The Port-Royal Logic*. 1662.

[67] J. Bentham, *An Introduction to the Principles of Morals and Legislation*. 1823.

[68] F. P. Ramsey,《Truth and probability》, *Readings in Formal Epistemology: Sourcebook*, 页 21 ~ 45, 2016.

[69] K. J. Arrow, 《A difficulty in the concept of social welfare》, *Journal of political economy*, 卷 58, 期 4, 页 328 ~ 346, 1950.

[70] J. C. Harsanyi, 《Rule utilitarianism and decision theory》, *Erkenntnis*, 卷 11, 期 1, 页 25 ~ 53, 1977.

[71] R. Pettigrew, *Choosing for changing selves*. Oxford University Press, 2019.

[72] N. Soares, B. Fallenstein, S. Armstrong, 和 E. Yudkowsky, 《Corrigibility》, 收入 *Workshops at the twenty-ninth AAAI conference on artificial intelligence*, 2015.

[73] W.-L. Chiang 等, 《Chatbot arena: An open platform for evaluating llms by human preference》, *arXiv preprint arXiv:2403.04132*, 2024.

[74] R. Likert, 《A technique for the measurement of attitudes.》, *Archives of psychology*, 1932.

[75] J. Zhou 等, 《Instruction-following evaluation for large language models》, *arXiv preprint arXiv:2311.07911*, 2023.

[76] K. Ethayarajh, W. Xu, N. Muennighoff, D. Jurafsky, 和 D. Kiela,《Kto: Model alignment as prospect theoretic optimization》, *arXiv preprint arXiv:2402.01306*, 2024.

[77] Z. Wu 等, 《Fine-grained human feedback gives better rewards for language model training》, *Advances in Neural Information Processing Systems*, 卷 36, 2024.

[78] A. Chen 等, 《Learning from Natural Language Feedback》, *Transactions on Machine Learning Research*, 2024.

[79] OpenAI, 《Introducing the Model Spec》. 2024 年 5 月. 载于: https://openai.com/index/introducing-the-model-spec/

[80] A. Y. Ng, S. Russell, 等, 《Algorithms for inverse reinforcement learning.》, 收入 *Proceedings of the Seventeenth International Conference on Machine Learning*, 收入 ICML '00. 2000, 页 663--670.

[81] R. A. Bradley 和 M. E. Terry, 《Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons》, *Biometrika*, 卷 39, 期 3/4, 页 324 ~ 345, 1952, 见于: 2023 年 2 月 13 日. [在线]. 载于: http://www.jstor.org/stable/2334029

[82] B. Zhu 等, 《Starling-7b: Improving helpfulness and harmlessness with rlaif》, 收入 *First Conference on Language Modeling*, 2024.

[83] A. Liu, Z. Zhao, C. Liao, P. Lu, 和 L. Xia, 《Learning plackett-luce mixtures from partial preferences》, 收入 *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, 页 4328 ~ 4335.

[84] B. Zhu, M. Jordan, 和 J. Jiao, 《Principled reinforcement learning with human feedback from pairwise or k-wise comparisons》, 收入 *International Conference on Machine Learning*, PMLR, 2023, 页 43037 ~ 43067.

[85] K. Cobbe 等, 《Training verifiers to solve math word problems》, *arXiv preprint arXiv:2110.14168*, 2021.

[86] C. Lyu 等, 《Exploring the Limit of Outcome Reward for Learning Mathematical Reasoning》, *arXiv preprint arXiv:2502.06781*, 2025.

[87] L. Zheng 等, 《Judging llm-as-a-judge with mt-bench and chatbot arena》, *Advances in Neural Information Processing Systems*, 卷 36, 页 46595 ~ 46623, 2023.

[88] Y. Dubois, B. Galambosi, P. Liang, 和 T. B. Hashimoto, 《Length-controlled alpacaeval: A simple way to debias automatic evaluators》, *arXiv preprint arXiv:2404.04475*, 2024.

[89] T. Li 等, 《From Crowdsourced Data to High-Quality Benchmarks: Arena-Hard and BenchBuilder Pipeline》, *arXiv preprint arXiv:2406.11939*, 2024.

[90] B. Y. Lin 等, 《WILDBENCH: Benchmarking LLMs with Challenging Tasks from Real Users in the Wild》, *arXiv preprint arXiv:2406.04770*, 2024.

[91] D. Mahan 等, 《Generative Reward Models》, 2024, 载于: https://www.synthlabs.ai/pdf/Generative_Reward_Models.pdf

[92] L. Zhang, A. Hosseini, H. Bansal, M. Kazemi, A. Kumar, 和 R. Agarwal, 《Generative verifiers: Reward modeling as next-token prediction》, *arXiv preprint arXiv:2408.15240*, 2024.

[93] Z. Ankner, M. Paul, B. Cui, J. D. Chang, 和 P. Ammanabrolu, 《Critique-out-loud reward models》, *arXiv preprint arXiv:2408.11791*, 2024.

[94] S. Kim 等, 《Prometheus: Inducing fine-grained evaluation capability in language models》, 收入 *The Twelfth International Conference on Learning Representations*, 2023.

[95] N. Lambert 等, 《Rewardbench: Evaluating reward models for language modeling》, *arXiv preprint arXiv:2403.13787*, 2024.

[96] X. Wen 等，《Rethinking Reward Model Evaluation: Are We Barking up the Wrong Tree?》，*arXiv preprint arXiv:2410.05584*, 2024.

[97] S. Gureja 等，《M-RewardBench: Evaluating Reward Models in Multilingual Settings》，*arXiv preprint arXiv:2410.15522*, 2024.

[98] Z. Jin 等，《RAG-RewardBench: Benchmarking Reward Models in Retrieval Augmented Generation for Preference Alignment》，*arXiv preprint arXiv:2412.13746*, 2024.

[99] E. Zhou 等，《RMB: Comprehensively Benchmarking Reward Models in LLM Alignment》，*arXiv preprint arXiv:2410.09893*, 2024.

[100] Y. Liu, Z. Yao, R. Min, Y. Cao, L. Hou, 和 J. Li, 《RM-bench: Benchmarking reward models of language models with subtlety and style》，*arXiv preprint arXiv:2410.16184*, 2024.

[101] Z. Wu, M. Yasunaga, A. Cohen, Y. Kim, A. Celikyilmaz, 和 M. Ghazvininejad, 《reWordBench: Benchmarking and Improving the Robustness of Reward Models with Transformed Inputs》，*arXiv preprint arXiv:2503.11751*, 2025.

[102] Z. Chen 等，《MJ-Bench: Is Your Multimodal Reward Model Really a Good Judge for Text-to-Image Generation?》，*arXiv preprint arXiv:2407.04842*, 2024.

[103] M. Yasunaga, L. Zettlemoyer, 和 M. Ghazvininejad,《Multimodal rewardbench: Holistic evaluation of reward models for vision language models》，*arXiv preprint arXiv:2502.14191*, 2025.

[104] L. Li 等，《VLRewardBench: A Challenging Benchmark for Vision-Language Generative Reward Models》，*arXiv preprint arXiv:2411.17451*, 2024.

[105] J. Ruan 等，《Vlrmbench: A comprehensive and challenging benchmark for vision-language reward models》，*arXiv preprint arXiv:2503.07478*, 2025.

[106] E. Frick 等，《How to Evaluate Reward Models for RLHF》，*arXiv preprint arXiv:2410.14872*, 2024.

[107] S. Kim 等，《Evaluating robustness of reward models for mathematical reasoning》，*arXiv preprint arXiv:2410.01729*, 2024.

[108] M. Song, Z. Su, X. Qu, J. Zhou, 和 Y. Cheng,《PRMBench: A Fine-grained and Challenging Benchmark for Process-Level Reward Models》，*arXiv preprint arXiv:2501.03124*, 2025.

[109] W. Wang 等，《VisualPRM: An Effective Process Reward Model for Multimodal Reasoning》，*arXiv preprint arXiv:2503.10291*, 2025.

[110] H. Tu, W. Feng, H. Chen, H. Liu, X. Tang, 和 C. Xie, 《ViLBench: A Suite for Vision-Language Process Reward Modeling》. 2025 年 3 月. 载于: https://arxiv.org/abs/2503.20271

[111] H. Wang, W. Xiong, T. Xie, H. Zhao, 和 T. Zhang, 《Interpretable Preferences via Multi-Objective Reward Modeling and Mixture-of-Experts》，*arXiv preprint arXiv:2406.12845*, 2024.

[112] Z. Wang 等，《HelpSteer2: Open-source dataset for training top-performing reward models》，*arXiv preprint arXiv:2406.08673*, 2024.

[113] Z. Wang 等，《HelpSteer2-Preference: Complementing Ratings with Preferences》，*arXiv preprint arXiv:2410.01257*, 2024.

[114] J. Park, S. Jwa, M. Ren, D. Kim, 和 S. Choi, 《Offsetbias: Leveraging debiased data for tuning evaluators》，*arXiv preprint arXiv:2407.06551*, 2024.

[115] N. Jaques, S. Gu, D. Bahdanau, J. M. Hernández-Lobato, R. E. Turner, 和 D. Eck, 《Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control》, 收入 *International Conference on Machine Learning*, PMLR, 2017, 页 1645 ~ 1654.

[116] N. Jaques 等, 《Human-centric dialog training via offline reinforcement learning》, *arXiv preprint arXiv:2010.05848*, 2020.

[117] J. Schulman, 《Approximating KL-divergence》. http://joschu.net/blog/kl-approx.html, 2016 年.

[118] R. Y. Pang, W. Yuan, K. Cho, H. He, S. Sukhbaatar, 和 J. Weston, 《Iterative reasoning preference optimization》, *arXiv preprint arXiv:2404.19733*, 2024.

[119] Z. Gao 等, 《Rebel: Reinforcement learning via regressing relative rewards》, *arXiv preprint arXiv:2404.16767*, 2024.

[120] T. B. Brown 等, 《Language Models are Few-Shot Learners》, *arXiv preprint arXiv:2005.14165*, 2020.

[121] C. Raffel 等, 《Exploring the limits of transfer learning with a unified text-to-text transformer》, *Journal of machine learning research*, 卷 21, 期 140, 页 1 ~ 67, 2020.

[122] J. Wei 等, 《Finetuned Language Models are Zero-Shot Learners》, 收入 *International Conference on Learning Representations*, 2022. 载于: https://openreview.net/forum?id=gEZrGCozdqR

[123] V. Sanh 等, 《Multitask Prompted Training Enables Zero-Shot Task Generalization》, 收入 *International Conference on Learning Representations*, 2022. 载于: https://openreview.net/forum?id=9Vrb9D0WI4

[124] S. Mishra, D. Khashabi, C. Baral, 和 H. Hajishirzi, 《Cross-Task Generalization via Natural Language Crowdsourcing Instructions》, 收入 *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, 5 月 2022, 页 3470 ~ 3487. doi: 10.18653/v1/2022.acl-long.244.

[125] E. Wallace, K. Xiao, R. Leike, L. Weng, J. Heidecke, 和 A. Beutel, 《The instruction hierarchy: Training llms to prioritize privileged instructions》, *arXiv preprint arXiv:2404.13208*, 2024.

[126] T. Dettmers, A. Pagnoni, A. Holtzman, 和 L. Zettlemoyer, 《Qlora: Efficient finetuning of quantized llms》, *Advances in neural information processing systems*, 卷 36, 页 10088 ~ 10115, 2023.

[127] N. Rajani, L. Tunstall, E. Beeching, N. Lambert, A. M. Rush, 和 T. Wolf, 《No Robots》, *Hugging Face repository*. https://huggingface.co/datasets/HuggingFaceH4/no_robots; Hugging Face, 2023 年.

[128] W. R. Gilks 和 P. Wild, 《Adaptive rejection sampling for Gibbs sampling》, *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 卷 41, 期 2, 页 337 ~ 348, 1992.

[129] A. Ahmadian 等, 《Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms》, *arXiv preprint arXiv:2402.14740*, 2024.

[130] J. Schulman, P. Moritz, S. Levine, M. Jordan, 和 P. Abbeel, 《High-dimensional continuous control using generalized advantage estimation》, 收入 *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[131] R. J. Williams, 《Simple statistical gradient-following algorithms for connectionist reinforcement learning》, *Machine learning*, 卷 8, 页 229 ~ 256, 1992.

[132] S. C. Huang, A. Ahmadian, 和 C. F. AI, 《Putting RL back in RLHF》. https://huggingface.co/blog/putting_rl_back_in_rlhf_with_rloo, 2024 年.

[133] W. Kool, H. van Hoof, 和 M. Welling, 《Buy 4 reinforce samples, get a baseline for free!》, 2019.

[134] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, 和 O. Klimov, 《Proximal policy optimization algorithms》, *arXiv preprint arXiv:1707.06347*, 2017.

[135] C. Berner 等, 《Dota 2 with large scale deep reinforcement learning》, *arXiv preprint arXiv:1912.06680*, 2019.

[136] Z. Liu 等, 《Understanding R1-Zero-Like Training: A Critical Perspective》, *arXiv preprint arXiv:2503.20783*, 3 月 2025, 载于: https://arxiv.org/abs/2503.20783

[137] Z. Shao 等, 《Deepseekmath: Pushing the limits of mathematical reasoning in open language models》, *arXiv preprint arXiv:2402.03300*, 2024.

[138] A. Liu 等, 《Deepseek-v3 technical report》, *arXiv preprint arXiv:2412.19437*, 2024.

[139] D. Guo 等, 《Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning》, *arXiv preprint arXiv:2501.12948*, 2025.

[140] S. Huang, M. Noukhovitch, A. Hosseini, K. Rasul, W. Wang, 和 L. Tunstall, 《The N+ Implementation Details of RLHF with PPO: A Case Study on TL;DR Summarization》, 收入 *First Conference on Language Modeling*, 2024. 载于: https://openreview.net/forum?id=kHO2ZTa8e3

[141] L. Weng, 《Policy Gradient Algorithms》, *lilianweng.github.io*, 2018, 载于: https://lilianweng.github.io/posts/2018-04-08-policy-gradient/

[142] Y. Zhao, R. Joshi, T. Liu, M. Khalman, M. Saleh, 和 P. J. Liu, 《Slic-hf: Sequence likelihood calibration with human feedback》, *arXiv preprint arXiv:2305.10425*, 2023.

[143] M. G. Azar 等, 《A general theoretical paradigm to understand learning from human preferences》, 收入 *International Conference on Artificial Intelligence and Statistics*, PMLR, 2024, 页 4447 ~ 4455.

[144] A. Amini, T. Vieira, 和 R. Cotterell, 《Direct preference optimization with an offset》, *arXiv preprint arXiv:2402.10571*, 2024.

[145] J. Hong, N. Lee, 和 J. Thorne, 《Reference-free monolithic preference optimization with odds ratio》, *arXiv e-prints*, 页 arXiv ~ 2403, 2024.

[146] Y. Meng, M. Xia, 和 D. Chen, 《Simpo: Simple preference optimization with a reference-free reward》, *Advances in Neural Information Processing Systems*, 卷 37, 页 124198 ~ 124235, 2025.

[147] N. Razin, S. Malladi, A. Bhaskar, D. Chen, S. Arora, 和 B. Hanin, 《Unintentional unalignment: Likelihood displacement in direct preference optimization》, *arXiv preprint arXiv:2410.08847*, 2024.

[148] Y. Ren 和 D. J. Sutherland, 《Learning dynamics of llm finetuning》, *arXiv preprint arXiv:2407.10490*, 2024.

[149] T. Xiao, Y. Yuan, H. Zhu, M. Li, 和 V. G. Honavar, 《Cal-dpo: Calibrated direct preference optimization for language model alignment》, *arXiv preprint arXiv:2412.14516*, 2024.

[150] A. Gupta 等, 《AlphaPO–Reward shape matters for LLM alignment》, *arXiv preprint arXiv:2501.03884*, 2025.

[151] S. Guo 等, 《Direct language model alignment from online ai feedback》, *arXiv preprint arXiv:2402.04792*, 2024.

[152] P. Singhal, N. Lambert, S. Niekum, T. Goyal, 和 G. Durrett, 《D2po: Discriminator-guided dpo with response evaluation models》, *arXiv preprint arXiv:2405.01511*, 2024.

[153] C. Rosset, C.-A. Cheng, A. Mitra, M. Santacroce, A. Awadallah, 和 T. Xie, 《Direct nash optimization: Teaching language models to self-improve with general preferences》, *arXiv preprint arXiv:2404.03715*, 2024.

[154] S. Jung, G. Han, D. W. Nam, 和 K.-W. On,《Binary classifier optimization for large language model alignment》, *arXiv preprint arXiv:2404.04656*, 2024.

[155] H. Zhao 等, 《Rainbowpo: A unified framework for combining improvements in preference optimization》, *arXiv preprint arXiv:2410.04203*, 2024.

[156] A. Gorbatovski, B. Shaposhnikov, V. Sinii, A. Malakhov, 和 D. Gavrilov,《The Differences Between Direct Alignment Algorithms are a Blur》, *arXiv preprint arXiv:2502.01237*, 2025.

[157] H. Ivison 等,《Unpacking DPO and PPO: Disentangling Best Practices for Learning from Preference Feedback》, *arXiv preprint arXiv:2406.09279*, 2024.

[158] S. Xu 等, 《Is dpo superior to ppo for llm alignment? a comprehensive study》, *arXiv preprint arXiv:2404.10719*, 2024.

[159] F. Tajwar 等, 《Preference fine-tuning of llms should leverage suboptimal, on-policy data》, *arXiv preprint arXiv:2404.14367*, 2024.

[160] H. Lee 等, 《Rlaif: Scaling reinforcement learning from human feedback with ai feedback》, 2023.

[161] A. Sharma, S. Keh, E. Mitchell, C. Finn, K. Arora, 和 T. Kollar,《A Critical Evaluation of AI Feedback for Aligning Large Language Models》. 2024 年. 载于: https://arxiv.org/abs/2402.12366

[162] L. Castricato, N. Lile, S. Anand, H. Schoelkopf, S. Verma, 和 S. Biderman, 《Suppressing Pink Elephants with Direct Principle Feedback》. 2024 年. 载于: https://arxiv.org/abs/2402.07896

[163] L. J. V. Miranda 等,《Hybrid Preferences: Learning to Route Instances for Human vs. AI Feedback》, *arXiv preprint arXiv:2410.19133*, 2024.

[164] T. Wang 等,《Shepherd: A critic for language model generation》, *arXiv preprint arXiv:2308.04592*, 2023.

[165] P. Ke 等,《CritiqueLLM: Towards an informative critique generation model for evaluation of large language model generation》, *arXiv preprint arXiv:2311.18702*, 2023.

[166] J. Li, S. Sun, W. Yuan, R.-Z. Fan, H. Zhao, 和 P. Liu, 《Generative Judge for Evaluating Alignment》, *arXiv preprint arXiv:2310.05470*, 2023.

[167] S. Kim 等, 《Prometheus 2: An open source language model specialized in evaluating other language models》, *arXiv preprint arXiv:2405.01535*, 2024.

[168] S. Lee, S. Kim, S. Park, G. Kim, 和 M. Seo, 《Prometheus-vision: Vision-language model as a judge for fine-grained evaluation》, 收入 *Findings of the Association for Computational Linguistics ACL 2024*, 2024, 页 11286 ~ 11315.

[169] M. Y. Guan 等,《Deliberative alignment: Reasoning enables safer language models》, *arXiv preprint arXiv:2412.16339*, 2024.

[170] Anthropic, 《Claude's Constitution》. 见于: 2024 年 2 月 7 日. [在线]. 载于: https://www.anthropic.com/news/claudes-constitution

[171] D. Ganguli 等, 《Collective Constitutional AI: Aligning a Language Model with Public Input》. Anthropic, 2023 年.

[172] S. Huang 等, 《Constitutional AI Recipe》, *Hugging Face Blog*, 2024.

[173] N. Lambert, H. Schoelkopf, A. Gokaslan, L. Soldaini, V. Pyatkin, 和 L. Castricato, 《Self-directed synthetic dialogues and revisions technical report》, *arXiv preprint arXiv:2407.18421*, 2024.

[174] Z. Sun 等, 《Principle-Driven Self-Alignment of Language Models from Scratch with Minimal Human Supervision》, 收入 *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 载于: https://openreview.net/forum?id=p40XRfBX96

[175] Z. Sun 等, 《SALMON: Self-Alignment with Principle-Following Reward Models》, 收入 *The Twelfth International Conference on Learning Representations*, 2024. 载于: https://openreview.net/forum?id=xJbsmB8UMx

[176] A. Irpan, 《Deep Reinforcement Learning Doesn't Work Yet》. 2018 年. 载于: https://www.alexirpan.com/2018/02/14/rl-hard.html

[177] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, 和 D. Meger, 《Deep Reinforcement Learning that Matters》, 收入 *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 载于: https://ojs.aaai.org/index.php/AAAI/article/view/11694

[178] G. Sheng 等, 《HybridFlow: A Flexible and Efficient RLHF Framework》, *arXiv preprint arXiv:2409.19256*, 2024.

[179] J. Hu 等, 《OpenRLHF: An Easy-to-use, Scalable and High-performance RLHF Framework》, *arXiv preprint arXiv:2405.11143*, 2024.

[180] J. Liu, A. Cohen, R. Pasunuru, Y. Choi, H. Hajishirzi, 和 A. Celikyilmaz, 《Don't throw away your value model! Generating more preferable text with Value-Guided Monte-Carlo Tree Search decoding》, *arXiv preprint arXiv:2309.15028*, 2023.

[181] B. Brown 等, 《Large language monkeys: Scaling inference compute with repeated sampling》, *arXiv preprint arXiv:2407.21787*, 2024.

[182] Z. Liu 等, 《Inference-Time Scaling for Generalist Reward Modeling》, *arXiv preprint arXiv:2504.02495*, 2025.

[183] N. Muennighoff 等, 《s1: Simple test-time scaling》, *arXiv preprint arXiv:2501.19393*, 2025.

[184] L. Chen 等, 《Are more llm calls all you need? towards scaling laws of compound inference systems》, *arXiv preprint arXiv:2403.02419*, 2024.

[185] I. Shumailov, Z. Shumaylov, Y. Zhao, N. Papernot, R. Anderson, 和 Y. Gal, 《AI models collapse when trained on recursively generated data》, *Nature*, 卷 631, 期 8022, 页 755 ~ 759, 2024.

[186] M. Gerstgrasser 等, 《Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data》, *arXiv preprint arXiv:2404.01413*, 2024.

[187] Y. Feng, E. Dohmatob, P. Yang, F. Charton, 和 J. Kempe, 《Beyond model collapse: Scaling up with synthesized data requires reinforcement》, 收入 *ICML 2024 Workshop on Theoretical Foundations of Foundation Models*, 2024.

[188] Y. Wang 等, 《Self-instruct: Aligning language models with self-generated instructions》, *arXiv preprint arXiv:2212.10560*, 2022.

[189] E. Beeching 等, 《NuminaMath 7B TIR》, *Hugging Face repository*. https://huggingface.co/AI-MO/NuminaMath-7B-TIR; Numina & Hugging Face, 2024 年.

[190] M. Li 等,《Superfiltering: Weak-to-strong data filtering for fast instruction-tuning》, *arXiv preprint arXiv:2402.00530*, 2024.

[191] K. Shridhar, A. Stolfo, 和 M. Sachan,《Distilling reasoning capabilities into smaller language models》, *Findings of the Association for Computational Linguistics: ACL 2023*, 页 7059 ~ 7073, 2023.

[192] C.-Y. Hsieh 等,《Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes》, *arXiv preprint arXiv:2305.02301*, 2023.

[193] D. Hendrycks 等,《Measuring massive multitask language understanding》, *arXiv preprint arXiv:2009.03300*, 2020.

[194] A. Mallen, A. Asai, V. Zhong, R. Das, H. Hajishirzi, 和 D. Khashabi,《When Not to Trust Language Models: Investigating Effectiveness and Limitations of Parametric and Non-Parametric Memories》, *arXiv preprint*, 2022.

[195] S. Lin, J. Hilton, 和 O. Evans,《Truthfulqa: Measuring how models mimic human falsehoods》, *arXiv preprint arXiv:2109.07958*, 2021.

[196] M. Suzgun 等,《Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them》, *arXiv preprint arXiv:2210.09261*, 2022.

[197] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, 和 M. Gardner,《DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs》, *arXiv preprint arXiv:1903.00161*, 2019.

[198] D. Hendrycks 等,《Measuring Mathematical Problem Solving With the MATH Dataset》, *NeurIPS*, 2021.

[199] K. Cobbe 等,《Training Verifiers to Solve Math Word Problems》, *arXiv preprint arXiv:2110.14168*, 2021.

[200] M. Chen 等,《Evaluating Large Language Models Trained on Code》, 2021, 载于: https://arxiv.org/abs/2107.03374

[201] J. Liu, C. S. Xia, Y. Wang, 和 L. Zhang,《Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation》, 收入 *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 载于: https://openreview.net/forum?id=1qvx610Cu7

[202] J. Zhou 等,《Instruction-Following Evaluation for Large Language Models》. 2023 年. 载于: https://arxiv.org/abs/2311.07911

[203] D. Rein 等,《GPQA: A Graduate-Level Google-Proof Q&A Benchmark》, *arXiv preprint arXiv:2311.12022*, 2023.

[204] L. Phan, A. Gatti, Z. Han, N. Li, 和 H. et al. Zhang,《Humanity's Last Exam》, *arXiv preprint arXiv:2501.14249*, 2025.

[205] R. Aleithan, H. Xue, M. M. Mohajer, E. Nnorom, G. Uddin, 和 S. Wang,《SWE-Bench+: Enhanced Coding Benchmark for LLMs》, *arXiv preprint arXiv:2410.06992*, 2024.

[206] N. Jain 等,《LiveCodeBench: Holistic and Contamination-Free Evaluation of Large Language Models for Code》, *arXiv preprint arXiv:2403.07974*, 2024.

[207] S. AI,《SEAL LLM Leaderboards: Expert-Driven Private Evaluations》. 2024 年. 载于: https://scale.com/leaderboard

[208] S. Schulhoff 等,《The prompt report: A systematic survey of prompting techniques》, *arXiv preprint arXiv:2406.06608*, 2024.

[209] J. Robinson, C. M. Rytting, 和 D. Wingate,《Leveraging Large Language Models for Multiple Choice Question Answering》, 收入 *International Conference on Learning Representations*, 2023. 载于: https://openreview.net/forum?id=upQ4o-ygvJ

[210] J. Wei 等,《Finetuned Language Models Are Zero-Shot Learners》, 收入 *International Conference on Learning Representations*, 2022.

[211] V. Sanh 等,《Multitask Prompted Training Enables Zero-Shot Task Generalization》, 收入 *International Conference on Learning Representations*, 2022.

[212] J. Achiam 等,《Gpt-4 technical report》, *arXiv preprint arXiv:2303.08774*, 2023.

[213] OpenAI,《Introducing SWE-bench Verified》. 2024 年 8 月. 载于: https://openai.com/index/introducing-swe-bench-verified/

[214] J. Li 等,《Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions》, *Hugging Face repository*, 卷 13, 页 9, 2024.

[215] L. Yu 等,《Metamath: Bootstrap your own mathematical questions for large language models》, *arXiv preprint arXiv:2309.12284*, 2023.

[216] A. K. Singh 等,《Evaluation data contamination in LLMs: how do we measure it and (when) does it matter?》, *arXiv preprint arXiv:2411.03923*, 2024.

[217] K. Huang 等,《MATH-Perturb: Benchmarking LLMs' Math Reasoning Abilities against Hard Perturbations》, *arXiv preprint arXiv:2502.06453*, 2025.

[218] UK AI Safety Institute,《Inspect AI: Framework for Large Language Model Evaluations》. https://github.com/UKGovernmentBEIS/inspect_ai, 2024 年.

[219] C. Fourrier, N. Habib, H. Kydlicek, T. Wolf, 和 L. Tunstall,《LightEval: A lightweight framework for LLM evaluation》. https://github.com/huggingface/lighteval, 2023 年.

[220] C. Fourrier, N. Habib, A. Lozovskaya, K. Szafer, 和 T. Wolf,《Open LLM Leaderboard v2》. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard; Hugging Face, 2024 年.

[221] L. Gao 等,《A Framework for Few-Shot Language Model Evaluation》. Zenodo, 2023 年. doi: 10.5281/zenodo.10256836.

[222] S. Black 等,《GPT-NeoX-20B: An Open-Source Autoregressive Language Model》, 收入 *Proceedings of the ACL Workshop on Challenges & Perspectives in Creating Large Language Models*, 2022. 载于: https://arxiv.org/abs/2204.06745

[223] Y. Gu, O. Tafjord, B. Kuehl, D. Haddad, J. Dodge, 和 H. Hajishirzi,《OLMES: A Standard for Language Model Evaluations》, *arXiv preprint arXiv:2406.08446*, 2024.

[224] P. Liang 等,《Holistic Evaluation of Language Models》, *Transactions on Machine Learning Research*, 2023, doi: 10.1111/nyas.15007.

[225] MosaicML,《Mosaic Eval Gauntlet v0.3.0 — Evaluation Suite》. https://github.com/mosaicml/llm-foundry/blob/main/scripts/eval/local_data/EVAL_GAUNTLET.md, 2024 年.

[226] J. Schulman,《Proxy Objectives in Reinforcement Learning from Human Feedback》. Invited talk at the International Conference on Machine Learning (ICML), 2023 年. 载于: https://icml.cc/virtual/2023/invited-talk/21549

[227] C. Zhang, O. Vinyals, R. Munos, 和 S. Bengio, 《A study on overfitting in deep reinforcement learning》, *arXiv preprint arXiv:1804.06893*, 2018.

[228] C. A. Goodhart 和 C. Goodhart, *Problems of monetary management: the UK experience*. Springer, 1984.

[229] K. Hoskin, 《The 〈awful idea of accountability〉: inscribing people into the measurement of objects》, *Accountability: Power, ethos and the technologies of managing*, 卷 265, 1996.

[230] M. Sharma 等, 《Towards understanding sycophancy in language models》, *arXiv preprint arXiv:2310.13548*, 2023.

[231] T. Lu 和 C. Boutilier, 《Learning Mallows models with pairwise preferences》, 收入 *Proceedings of the 28th international conference on machine learning (icml-11)*, 2011, 页 145 ~ 152.

[232] S. Han 等, 《Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms》, *arXiv preprint arXiv:2406.18495*, 2024.

[233] H. Inan 等,《Llama guard: Llm-based input-output safeguard for human-ai conversations》, *arXiv preprint arXiv:2312.06674*, 2023.

[234] P. Röttger, H. R. Kirk, B. Vidgen, G. Attanasio, F. Bianchi, 和 D. Hovy,《Xstest: A test suite for identifying exaggerated safety behaviours in large language models》, *arXiv preprint arXiv:2308.01263*, 2023.

[235] T. Coste, U. Anwar, R. Kirk, 和 D. Krueger, 《Reward model ensembles help mitigate overoptimization》, *arXiv preprint arXiv:2310.02743*, 2023.

[236] T. Moskovitz 等, 《Confronting reward model overoptimization with constrained RLHF》, *arXiv preprint arXiv:2310.04373*, 2023.

[237] R. Rafailov 等,《Scaling laws for reward model overoptimization in direct alignment algorithms》, *Advances in Neural Information Processing Systems*, 卷 37, 页 126207 ~ 126242, 2024.

[238] S. Zhuang 和 D. Hadfield-Menell,《Consequences of misaligned AI》, *Advances in Neural Information Processing Systems*, 卷 33, 页 15763 ~ 15773, 2020.

[239] W. Yuan 等,《Self-Rewarding Language Models》. 2025 年. 载于: https://arxiv.org/abs/2401.10020

[240] J. Bai 等, 《Qwen Technical Report》, *arXiv preprint arXiv:2309.16609*, 2023.

[241] G. Wang, S. Cheng, X. Zhan, X. Li, S. Song, 和 Y. Liu,《Openchat: Advancing open-source language models with mixed-quality data》, *arXiv preprint arXiv:2309.11235*, 2023.

[242] Anthropic,《Claude's Character》. 2024 年. 载于: https://www.anthropic.com/research/claude-character