第二章 基础介绍

大语言模型是指在海量无标注文本数据上进行预训练得到的大型预训练语言模型，例如GPT-3 [23]，PaLM [33]和LLaMA [34]。目前大语言模型所需要具有的最小参数规模还没有一个明确的参考标准，但是大语言模型通常是指参数规模达到百亿、千亿甚至万亿的模型；也有部分工作认为经过大规模数据预训练（显著多于传统预训练模型如BERT所需要的训练数据）的数十亿参数级别的模型也可以称之为大语言模型（如LLaMA-27B）。对于大语言模型，本书泛指具有超大规模参数或者经过超大规模数据训练所得到的语言模型。与传统语言模型相比，大语言模型的构建过程涉及到更为复杂的训练方法，进而展现出了强大的自然语言理解能力和复杂任务求解能力（通过文本生成的形式）。为了帮助读者了解大语言模型的工作原理，本部分将介绍大语言模型的构建过程、扩展法则（Scaling Law）、涌现能力（Emergent Abilities），然后将介绍GPT系列模型的研发历程。

2.1 大语言模型的构建过程

本部分内容将概要介绍大语言模型的构建过程，为刚进入该领域的读者对于大语言模型的研发建立一个初步的认识。从机器学习的观点来说，神经网络是一种具有特定模型结构的函数形式，而大语言模型则是一种基于Transformer结构的神经网络模型。因此，可以将大语言模型看作一种拥有大规模参数的函数，它的构建过程就是使用训练数据对于模型参数的拟合过程。尽管所采用的训练方法与传统的机器学习模型（如多元线性回归模型的训练）可能存在不同，但是本质上都是在做模型参数的优化。大语言模型的优化目标更加泛化，不仅仅为了解决某一种或者某一类特定任务，而是希望能够作为通用任务的求解器（如图1.2）。为了实现这一宏大的目标，**大语言模型的构建过程需要更为复杂、精细的训练方法。一般来说，这个训练过程可以分为大规模预训练和指令微调与人类对齐两个阶段**，下面将进行具体介绍。

2.1.1 大规模预训练

一般来说，**预训练是指使用与下游任务无关的大规模数据进行模型参数的初始训练，可以认为是为模型参数找到一个较好的“初值点”**。这一思想最早在计算机视觉领域被广泛使用，通过使用大规模的图像标注数据集合 ImageNet 用于初始化视觉模型的参数。**在自然语言处理领域，word2vec [8] 采用了类似的预训练思想，使用无标注的文本语料训练可通用的词嵌入模型；后来被 ELMo [11]、BERT [13] 和 GPT-1 [14] 推广到训练可迁移的自然语言任务架构，逐步成为了研发大语言模型的核心技术路径。早期的预训练技术还是聚焦于解决下游某一类的特定任务，如传统的自然语言处理任务。**

OpenAI 在 GPT-2 [17] 的论文中，提出通过大规模文本数据的预训练实现通用任务的求解器（尽管 GPT-2 论文中所验证的实验还是主要以自然语言处理任务为主），并且将这一思路在 GPT-3 中推广到了当时最大的千亿规模。OpenAI 前首席科学家 Ilya Sutskever 在公开采访中指出大规模预训练本质上是在做一个世界知识的压缩，从而能够学习到一个编码世界知识的参数模型，这个模型能够通过解压缩所需要的知识来解决真实世界的任务。在 BERT 等传统预训练模型中，所采用的模型架构以及训练任务还比较多样。**由于 GPT 系列模型的爆火，“解码器架构 + 预测下一个词”的有效性得到了充分验证，已经成为现有大语言模型主要采纳的技术路径。**

为了预训练大语言模型，需要准备大规模的文本数据，并且进行严格的清洗，去除掉可能包含有毒有害的内容，最后将清洗后的数据进行词元化（Tokenization）流，并且切分成批次（Batch），用于大语言模型的预训练。由于大语言模型的能力基础主要来源于预训练数据，因此数据的收集与清洗对于模型性能具有重要的影响。收集高质量、多源化的数据以及对于数据进行严格的清洗是构建大语言模型关键能力的重中之重，需要大模型研发人员的高度关注。目前的开源模型普遍采用2~3T规模的词元进行预训练，并有趋势进一步扩大这一规模。这一过程对于算力需求量极高，一般来说训练百亿模型至少需要百卡规模的算力集群（如A10080G）联合训练数月时间（与具体的算力资源相关）；而训练千亿模型则需要千卡甚至万卡规模的算力集群，对于算力资源的消耗非常惊人。

尽管整体的预训练技术框架非常直观，但是实施过程中涉及到大量需要深入探索的经验性技术，如数据如何进行配比、如何进行学习率的调整、如何早期发现模型的异常行为等。预训练过程需要考虑各种实施细节，而这些细节有很多并没有公开发表的经验可循，需要研发人员具有丰富的训练经验和异常处理能力，避免大规模训练开始以后进行回退和反复迭代，从而减少算力资源的浪费，提升训练成功的几率。大语言模型的研发看似是一个算力需求型的工程，实际上相关人才是最重要的。可以说，一个大语言模型项目的核心训练人员的能力最后会决定模型的整体水平。

2.1.2 指令微调与人类对齐

经过大规模数据预训练后的语言模型已经具备较强的模型能力，能够编码丰富的世界知识，但是由于预训练任务形式所限，这些模型更擅长于文本补全，并不适合直接解决具体的任务。尽管可以通过上下文学习（In-Context Learning, ICL）等提示学习技术进行适配，但是模型自身对于任务的感知与解决能力仍然较为局限。这里做一个简单的类比。预训练后的模型就像进入工作岗位的毕业生，尽管学习了很多通用的文化课，具备了一定的实习经验，但是仍然需要加强面向特定岗位的工作能力，并且深入了解工作岗位所涉及的相关要求。因此，用人单位往往需要设置特定的培训环节，对于新入职的人员针对业务场景以及所需要的技术进行专门提升。相似地，当预训练结束后，通常需要对于大语言模型进行微调与对齐，使之更好地被用于任务求解，为人类服务。

目前来说，**比较广泛使用的微调技术是“指令微调”（也叫做有监督微调，Supervised Fine-tuning, SFT），通过使用任务输入与输出的配对数据进行模型训练，可以使得语言模型较好地掌握通过问答形式进行任务求解的能力。这种模仿示例数据进行学习的过程本质属于机器学习中的模仿学习（Imitation Learning）**。给定一个特定任务，虽然可能存在很多解答方式，模仿学习旨在加强对于标准答案（即师傅的示范动作）的复刻学习。一般来说，**指令微调很难教会大语言模型预训练阶段没有学习到的知识与能力，它主要起到了对于模型能力的激发作用，而不是知识注入作用。与预训练相比，指令微调通常来说需要的指令实例数据规模要小的多。**通常来说，数十万到百万规模的指令微调数据能够有效地激发语言模型的通用任务解决能力，甚至有些工作认为数千条或者数万条高质量指令数据也能达到不错的微调效果。因此，指令微调对于算力资源的需求相对较小。一般情况下，若干台单机八卡（A100-80G）的服务器就能在一天或数天的时间内完成百亿模型的指令微调，当指令数据规模较大的时候可以进一步增加所需要的算力资源。这个过程还可以进一步加入多轮次的对话数据来增强模型的人机对话能力。

除了提升任务的解决能力外，还需要将大语言模型与人类的期望、需求以及价值观对齐（Alignment），这对于大模型的部署与应用具有重要的意义。OpenAI在2022年初发布了InstructGPT[28]的学术论文，系统地介绍了如何将语言模型进行人类对齐。具体来说，**主要引入了基于人类反馈的强化学习对齐方法RLHF（Reinforcement Learning from Human Feedback），在指令微调后使用强化学习加强模型的对齐能力。在RLHF算法中，需要训练一个符合人类价值观的奖励模型(Reward Model)。为此，需要标注人员针对大语言模型所生成的多条输出进行偏好排序，并使用偏好数据训练奖励模型，用于判断模型的输出质量**。由于强化学习需要维护更多的辅助模型进行训练，通常来说对于资源的消耗会多于指令微调，但是也远小于预训练阶段所需要的算力资源。目前还有很多工作试图通过消除奖励模型的使用，或其他使用SFT方式来达到与RLHF相似的效果，从而简化模型的对齐过程。

经历上述两个过程后，大语言模型就能够具备较好的人机交互能力，通过问答形式解决人类所提出的问题。这个构建过程需要大量的算力资源支持，也需要具有良好洞察力和训练经验的研发人员进行相关技术路线的设计与执行。因此，实现具有ChatGPT或者GPT-4能力的大语言模型绝非易事，需要进行深入的探索与实践。

第四章 数据准备

预训练是研发大语言模型的第一个训练阶段，也是最为重要的一个阶段。有效的预训练能够为大语言模型的能力奠定坚实的基础：通过在大规模语料上进行预训练，大语言模型可以获得通用的语言理解与生成能力，掌握较为广泛的世界知识，具备解决众多下游任务的性能潜力。在这一过程中，预训练语料的规模和质量对于提升大语言模型的能力至关重要。在本章中，我们将介绍如何准备预训练语料，主要包含原始数据的收集、数据预处理、数据词元化、以及预训练过程中的数据调度方法。

4.1 数据来源

为了构建功能强大的大语言模型，需要从多元化的数据源中收集海量数据来进行训练。现有的大语言模型主要将各种公开的文本数据进行混合，作为预训练语料。图4.1展示了部分具有代表性的大语言模型的预训练数据来源。从图中可以看到，目前网页仍然是建立语言模型最广泛使用的预训练数据，其他常用的数据还包括书籍、代码、对话语料等。

根据来源不同，预训练数据主要分为两种类型：通用文本数据和专用文本数据。通用文本数据涵盖了网页、书籍和对话文本等。由于通用文本数据规模较大、多样性强且易于获取，大多数大语言模型都会收集大量的通用文本数据，以增强其语言建模能力。此外，为了进一步提升大语言模型在特定专业任务上的表现，人们还将预训练语料的范围扩展至更专业的数据集，如多语数据、科学数据和代码数据等。接下来，我们将逐一介绍这两类预训练数据，并讨论它们对于大语言模型性能的影响。如需了解更多关于常用语料的详细信息，请参阅第3.2节。

4.1.1 通用文本数据

从图4.1中我们可以看到，**绝大多数的大语言模型都选用了网页、书籍和对话文本等通用语料作为预训练数据**。这些通用语料涵盖了多个主题类别的文本内容。接下来，我们将详细介绍两种重要的通用文本数据。

- 网页. 随着互联网的普及与发展, 网页的数据规模持续扩大, 覆盖的内容类型也变得丰富多样。使用大规模网页文本数据进行预训练，有助于大语言模型获取多样化的语言知识，并增强其自然语言理解和生成的能力[17, 77]。为了便于使用网页数据进行预训练或相关研究，相关机构已经爬取并发布了多个大规模的网页数据集，包括C4 [77]、RefinedWeb [82]、CC-Stories [115]等。然而，这些网页数据集中既包含了维基百科这种高质量文本，也不可避免地引入了广告网页等低质量文本。因此，**在进行预训练之前，对网页进行筛选和处理显得尤为重要，这直接关系到最终数据的质量与预训练效果**。

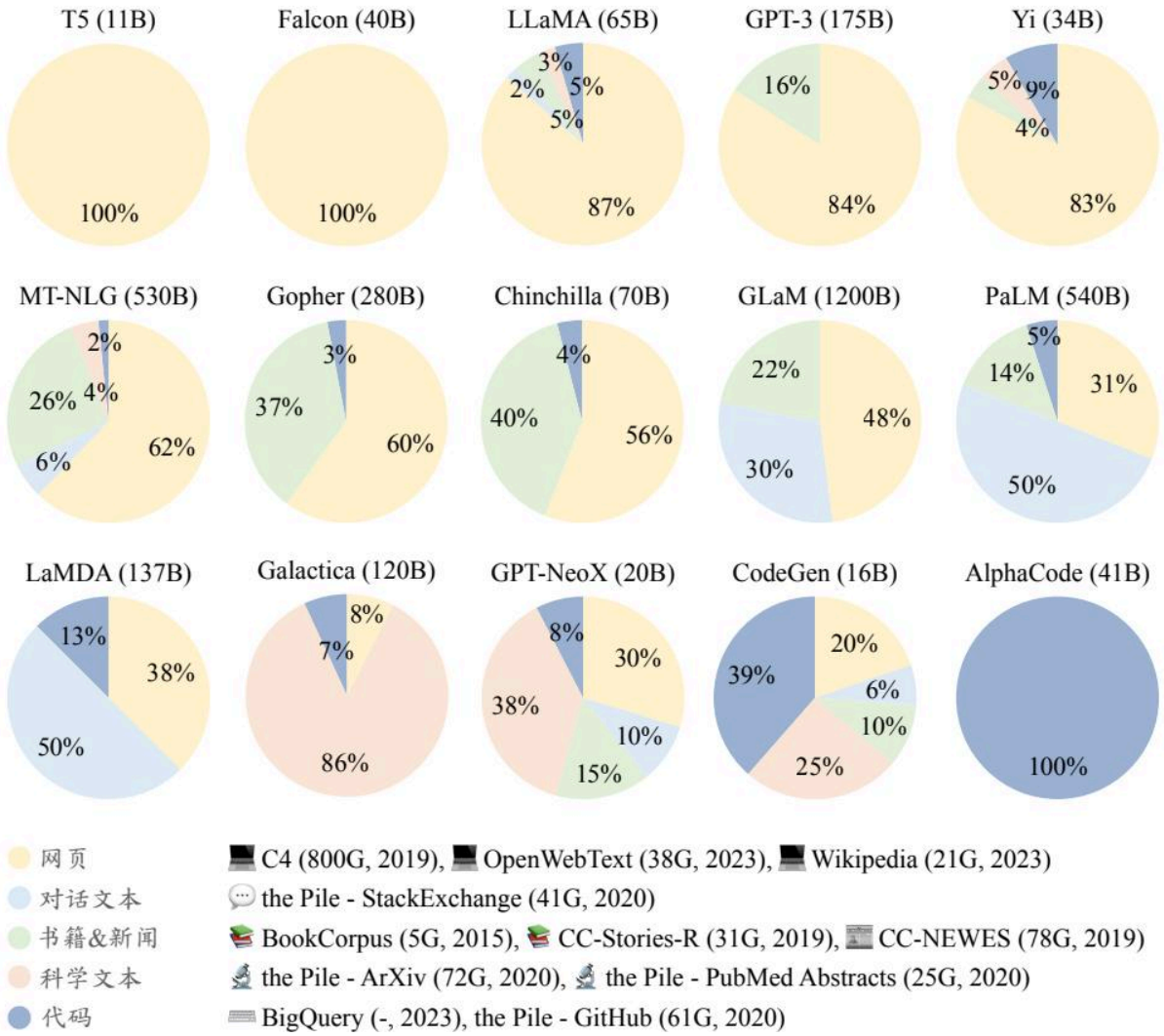


图4.1现有大语言模型预训练数据中各种数据来源的比例分布图（图片来源：[10]）

- 书籍. 相较于其他语料，**书籍中的文本内容往往更为正式与详实，篇幅也相对较长。这些书籍文本在大语言模型的学习过程中，发挥着非常重要的作用，它们不仅能够帮助模型积累丰富的语言知识，还可以加强其长程语义关系的建模**。现有的研究工作通常使用 Books3 和 Bookcorpus2 等开源书籍数据集。这些数据可以在 Pile 数据集中获得 [97]。

4.1.2 专用文本数据

专用数据集有助于提升大语言模型解决特定下游任务的能力。下面介绍三种专用的文本数据。

- 多语文本. 在预训练语料中，加入多语言的文本数据可以增强模型的多语理解与生成能力。BLOOM [100] 模型和 PaLM [33] 模型在其预训练语料中分别使用了涵盖 46 种和 122 种语言的多语数据，进而使得这两个模型在翻译、跨语言摘要和问答等多语言任务中性能表现优异。**相比于仅针对单一目标语言进行微调的模型，在多语言语料库上训练过的大语言模型能够更好地建立多语言间的语义关联，为跨语言理解与对话任务提供支持。不仅如此，多语言数据还能有效增加数据的多样性，从而有助于提升模型的综合性能。**

- 科学文本. 随着科学研究的不断发展，相关出版物的数量不断增加。**为了增强大语言模型对科学知识的理解，可以将科学文本数据加入到模型的预训练语料中。通过在大规模的科学文本语料上进行预训练，大语言模型可以在自然科学以及工程技术方面建立坚实的知识基础，从而在科学问答与推理等任务上取得出色的表现** [116]。构建科学文本语料的常用方法是收集 arXiv 论文、科学教材、数学网页等科学资源。然而，**由于科学文本数据中包含数学公式、蛋白质序列等特殊符号，通常需要采用特定的分词和预处理技术，将这些不同格式的数据转化为大语言模型能够处理的统一格式**。

- 代码. 代码能力目前已经成为大语言模型备受关注的一种能力。为了提高模型的代码能力，需要在大量代码语料上进行预训练，进而提高其所生成的程序质量。这些由大语言模型编写的程序甚至可以成功通过专家设计的单元测试用例[47]或解决具有挑战性的算法竞赛问题[117]。一般来说，常用于大语言模型预训练的代码语料有两种来源，第一种是Stack Exchange等编程问答社区的数据，第二种是GitHub等开源项目仓库。这两种来源包含了代码以及对应的注释和文档。**与自然语言文本相比，代码主要以结构化的编程语言形式呈现。在代码数据上训练能够提升模型的结构化语义理解与逻辑推理能力**[118]。**同时，代码中的函数调用关系还有助于增强模型的工具使用与学习能力**[119]。**此外，将推理任务格式化为代码可以帮助大语言模型生成更准确的结果**[120, 121]。

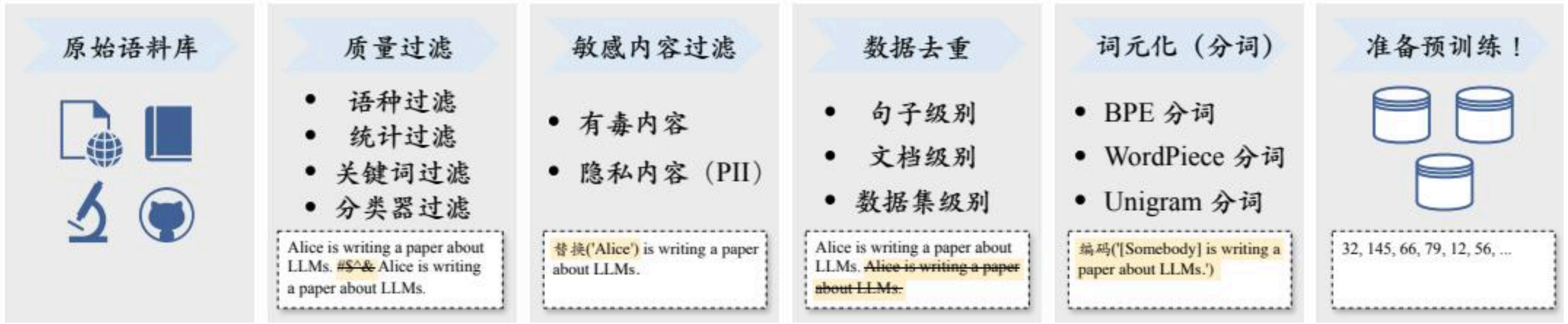


图4.2 典型的预训练数据预处理流程图（图片来源：[10]）

【补充】Markdown:

- Markdown 是连接自然语言与结构化语义的高效桥梁：以极简语法实现清晰表达，既适合人类书写，也最契合大语言模型的理解与生成需求。
- 数学公式

8.2 常见符号（1/2）

- 基本运算符号
  - 加减乘除：`a + b \times c - d \div e` ⟶ $a + b \times c - d \div e$
  - 根号：`\sqrt{x}, \sqrt[n]{x}` ⟶ $\sqrt{x}, \sqrt[n]{x}$
  - 指数：`x^2, e^{i\pi}` ⟶ $x^2, e^{i\pi}$
- 比较符号
  - 等于：`=, \neq, \equiv` ⟶ $=, \neq, \equiv$
  - 大小：`<, >, \leq, \geq` ⟶ $<, >, \leq, \geq$
  - 约等于：`\approx, \sim` ⟶ $\approx, \sim$
- 集合符号
  - 属于：`\in, \notin` ⟶ $\in, \notin$
  - 包含：`\subset, \supset` ⟶ $\subset, \supset$
  - 交并：`\cap, \cup` ⟶ $\cap, \cup$
  - 空集：`\emptyset` ⟶ $\emptyset$

4.4 数据调度

完成数据预处理之后，需要设计合适的调度策略来安排这些多来源的数据，进而用于训练大语言模型。通常来说，数据调度（Data Scheduling）主要关注两个方面：各个数据源的混合比例以及各数据源用于训练的顺序（称为数据课程，DataCurriculum)。具体的数据调度示意图可以参考图 4.3。下面将详细介绍这些内容。

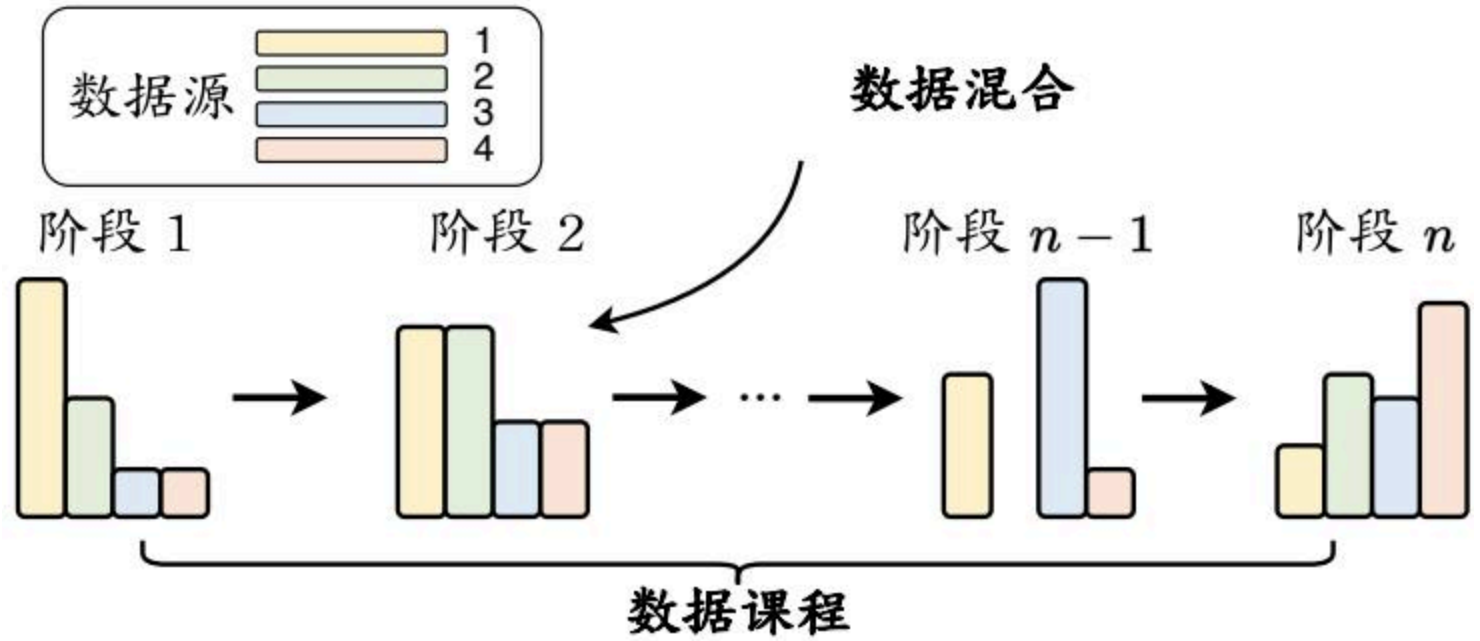


图4.3 预训练大语言模型时数据调度的示意图（图片来源：[10])

4.4.1 数据混合

由于不同数据源与大语言模型某些特定能力的学习具有紧密的联系（参见第4.1节的讨论），因此设置合适的数据混合比例非常重要。数据混合通常在数据集合层面上设置（即整个预训练数据的整体分布），也可以在不同训练阶段采用不同的混合数据比例。在预训练期间，将根据混合比例从不同数据源中采样数据：数据源的权重越大，从中选择的数据就越多。进一步，可能会对每个数据源的全部数据进行上采样或下采样，以创建特定的数据混合集合作为预训练数据。

典型的数据分布

图4.1展示了目前一些代表性的大语言模型的数据混合配比情况。作为其中的一个代表性模型，LLaMA[34]的预训练数据主要包括超过 80% 的网页数据、来自GitHub和StackExchange的 6.5% 代码密集型数据、 4.5% 的书籍数据，以及来自arXiv的 2.5% 科学数据，这个数据配比成为了训练大语言模型的一个重要参考。根据这个比例，**网页数据在现有预训练数据占据了较大的比重，为大语言模型提供了丰富的世界知识。此外，也可以为实现不同的目的来设计特定的数据混合配比。例如，专业的代码模型CodeGen[94]大幅增加了代码数据的比例。值得注意的是，即使是在这样的专业模型中，依然需要混合一定的网页数据来提供或者保留通用的语义知识。**

数据混合策略

在实践中，数据混合通常是根据经验确定的，下面汇总了几种常见的数据混合策略。

- 增加数据源的多样性。为了提升大语言模型的整体能力，增加数据源异质性（即包括多样化的数据源）能够有助于改进大语言模型在下游任务中的综合表现[145-147]。进一步，为了研究不同数据源的影响，一些研究工作构建了消融实验，通过逐一移除每个数据源并用其余数据源对大语言模型进行预训练进行效果评估[145]。因此，在收集预训练数据时，需要注意引入数据多样性更高的数据源，如包含网页数据、各类型书籍、代码数据等。

- 优化数据混合。除了手动设置数据混合配比外，还可以使用可学习的方法来优化数据组成，以改善模型的预训练效果[36,148]。例如，**可以根据目标下游任务来选择特征空间相似的预训练数据[148]，或对下游任务性能可以产生正面影响的数据**[149]。为了减少对于目标任务的依赖，DoReMi[36]首先使用给定的初始领域权重训练一个小型参考模型，然后在每次迭代过程中，使用当前的领域权重计算得到数据比例，用其训练另一个小型代理模型。然后通过比较两个模型损失值的差距，对该域数据的采样权重进行优化。具体来说，对于代理模型"未较好习得的"数据域，所分配的域权重将会被增加。最后，通过多轮迭代，代理模型最终的域权重将被应用于大语言模型训练。**此外，一个更为简单的实践方法是，训练几个具有不同数据混合配比的小型语言模型，并选择获得最理想性能的数据混合配比。然而，这个方法的一个假设是，如果以类似的方式训练，小模型会在模型能力或行为上与大模型相似，这在实际中可能并不总是成立。**

- 优化特定能力。大语言模型的模型能力在很大程度上取决于数据选择和配比，可以通过增加特定数据源的比例来增强某些对应的模型能力[123,145]。例如，**可以通过使用更多的数学文本和代码数据来增强大语言模型的数学推理和编程能力，而增加书籍数据的比例可以提高模型捕捉文本长程依赖关系的能力[150]。为了增强大语言模型的特定能力（如数学和编码），或开发专用的大语言模型，一种常见的方法是采用多阶段训练方法，例如可以在连续两个阶段分别安排通用数据和任务特定数据。这种在多个阶段使用不同来源或比例的数据的训练方法也被称为“数据课程”**，将在下文中具体介绍。

4.4.2 数据课程

除了设置有效的数据混合配比外，在训练过程中对于预训练数据的顺序进行合适的安排也比较重要。具体来说，**数据课程是指按照特定的顺序安排预训练数据进行模型的训练。例如，从简单/通用的数据开始，逐渐引入更具挑战性/专业化的数据。更广泛地说，它可以指训练期间在不同阶段使用不同的数据源混合配比**。为了设定合适的数据课程，一种实用方法是基于专门构建的评测基准监控大语言模型的关键能力的学习过程，然后在预训练期间动态调整数据的混合配比。

由于预训练阶段需要耗费大量的计算资源，目前针对数据课程的研究工作主要集中在继续预训练（Continual Pre-training）这一方面。如专业化的编程大语言模型（例如 CodeLLaMA [151]）或具有长上下文建模能力的大语言模型（例如 LongLLaMA [152]）。相关研究表明，为了学习某些特定的技能，按照技能依赖顺序编排对应数据集的学习方法（例如，基本技能 $\rightarrow$ 目标技能）比直接在相关的特定语料库上学习效果更好[151, 153]。与机器学习中的课程学习方法相似[154]，数据课程的思想已经被广泛应用于模型预训练[151-153, 155]。下面将以三种常见能力为例，介绍具体的数据课程在继续预训练中的应用。

- 代码能力。为了提高大语言模型的代码生成能力, 研究人员基于LLaMA2 [58] 开发了CodeLLaMA [151], 能够更为有效地执行代码任务。采用的数据为: 2T通用词元 $\rightarrow 500\mathrm{B}$ 代码密集型词元。这里, 使用符号“ $\rightarrow$ ”来表示数据课程中的数据顺序, 指的是大语言模型首先用 2T网页数据词元进行训练, 随后用 500B 代码数据词元训练。CodeLLaMA 还提供了一个面向 Python 语言的特定代码大模型, 即 CodeLLaMA-Python, 采用了如下的数据训练课程: 2T通用词元 $\rightarrow 500\mathrm{B}$ 代码相关的词元 $\rightarrow 100\mathrm{B}$ Python 代码相关的词元。

- 数学能力。Llemma [156] 是一个具有代表性的数学大语言模型, 有效提升了通用大语言模型的数学能力。它选择 CodeLLaMA 作为基座模型, 进一步在包含科学论文、数学和代码的混合数据集合上进行继续预训练。虽然 CodeLLaMA [151] 主要关注编程能力, 但是实验表明它在数学基准测试上的表现优于其基础模型 LLaMA-2 [156]。整体的数据课程为: $2\mathrm{T}$ 通用词元 $\rightarrow 500\ \mathrm{B}$ 代码相关的词元 $\rightarrow 50 \sim 200\ \mathrm{B}$ 数学相关的词元。值得注意的是, Llemma 的继续预训练数据中还包含 5% 的通用领域数据, 这可以看做一种模型能力的 “正则化” 技术, 加强对于原始基座模型通用能力的保持。

- 长文本能力。长文本理解与生成是大语言模型的一项重要能力。很多研究工作通过继续预训练有效扩展了大语言模型的上下文窗口 [151, 152]，主要是针对 RoPE 中的位置嵌入编码进行修改 [34, 58, 157]。例如，CodeLLaMA 将 LLaMA-2 的上下文窗口从 4K 扩展到了 100K，所采用的数据课程为：2.5T 词元，4K 上下文窗口 $\rightarrow$ 20B 词元，16K 上下文窗口。通过使用这种训练序列长度由短到长的数据课程，能够使模型获得较好的长文本建模能力，同时可以节省长文本模型的训练时间。

第五章 模型架构

5.1 Transformer 模型

当前主流的大语言模型都基于Transformer模型进行设计的。Transformer是由多层的多头自注意力（Multi-head Self-attention）模块堆叠而成的神经网络模型。**原始的Transformer模型由编码器和解码器两个部分构成，而这两个部分实际上可以独立使用，例如基于编码器架构的BERT模型[13]和解码器架构的GPT模型[14]**。与BERT等早期的预训练语言模型相比，大语言模型的特点是使用了更长的向量维度、更深的层数，进而包含了更大规模的模型参数，并主要使用解码器架构，对于Transformer本身的结构与配置改变并不大。本部分内容将首先介绍Transformer模型的基本组成，包括基础的输入、多头自注意力模块和前置网络层；接着分别介绍Transformer模型中的编码器和解码器模块。

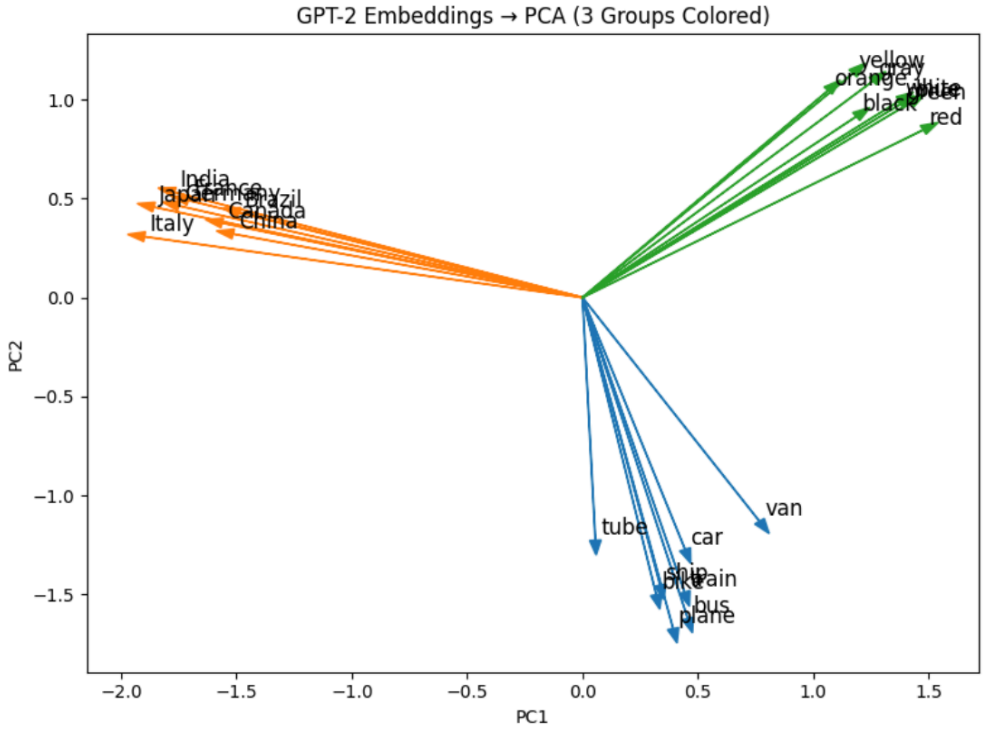
5.1.1 输入编码

在Transformer模型中，输入的词元序列 $(\boldsymbol{u} = [u_1, u_2, \ldots, u_T])$ 首先经过一个输入嵌入模块（Input Embedding Module）转化成词向量序列。具体来说，为了捕获词汇本身的语义信息，每个词元在输入嵌入模块中被映射成为一个可学习的、具有固定维度的词向量 $\boldsymbol{\nu}_t \in \mathbb{R}^H$ 。**由于Transformer的编码器结构本身无法识别序列中元素的顺序，位置编码（Position Embedding, PE）被引入来表示序列中的位置信息**。给定一个词元 u_t ，位置编码根据其在输入中的绝对位置分配一个固定长度的嵌入向量 $\boldsymbol{p}_t \in \mathbb{R}^H$ 。然后，**每个词元对应的词向量和位置向量将直接相加，生成了最终的输入嵌入序列** $X = [x_1, \ldots, x_T]$ **，并且被传入到后续层中：**

$$\boldsymbol{x}_t = \boldsymbol{v}_t + \boldsymbol{p}_t. \tag{5.1}$$

通过这种建模方法的表示，Transformer模型可以利用位置编码 p_t 建模不同词元的位置信息。由于不同词元的位置编码仅由其位置唯一决定，因此这种位置建模方式被称为绝对位置编码。**尽管绝对位置编码能够一定程度上建模位置信息，然而它只能局限于建模训练样本中出现的位置，无法建模训练数据中未出现过的位置，因此极大地限制了它们处理长文本的能力**。我们将会在第5.2.4节深入讨论不同的位置编码方式以及在第5.4节讨论长文本建模方法。

- **【补充】嵌入（Embedding）：**
  - 词嵌入将文本转为高维向量以表达语义
  - 由模型从语料中学习语义关系，使相近词向量距离近
  - 用于语义相似度计算、文本检索、分类、聚类等下游任务。

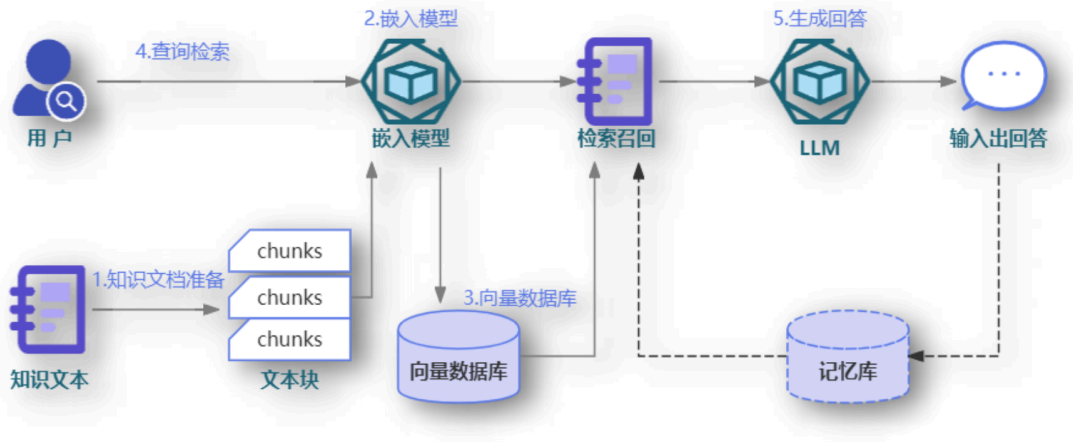


- **【补充】检索增强生成（RAG）：**
  - 准备知识文档、选择嵌入模型、构建向量数据库、查询检索、生成回答

2 RAG基本流程

RAG包含5个基本流程：

1. 准备知识文档
2. 选择嵌入模型
3. 构建向量数据库
4. 查询检索
5. 生成回答



5

5.2.1 归一化方法

大语言模型的预训练过程中经常会出现不稳定的问题。为了应对这一问题，深度学习方法通常会采用特定的归一化策略来加强神经网络训练过程的稳定性。原始的Transformer模型主要使用了层归一化方法（Layer Normalization, LN）[158]。随着研究工作的不断深入，基于层归一化的改进技术不断涌现，例如均方根层归一化（Root Mean Square Layer Normalization, RMSNorm）[159]和DeepNorm [160]，这些新技术已经在一些大语言模型中得到应用。下面进行具体介绍。

- LayerNorm. 在早期的研究中，批次归一化（Batch Normalization, BN）[161]是一种广泛采用的归一化方法。然而，该方法难以处理可变长度的序列数据和小批次数据。因此，相关研究提出了层归一化这一技术[158]，针对数据进行逐层归一化。具体而言，层归一化会计算每一层中所有激活值的均值 μ 和方差 σ ，从而重新调整激活值的中心和缩放比例：

$$\mathrm{LayerNorm}(x) = \frac{x - \mu}{\sigma} \cdot \gamma + \beta, \tag{5.12}$$

$$\boldsymbol{\mu} = \frac{1}{H}\sum_{i=1}^{H} x_i, \quad \sigma = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(x_i - \boldsymbol{\mu}))^2}. \tag{5.13}$$

- $RMSNorm$. 为了提高层归一化的训练速度，RMSNorm [159] 仅利用激活值总和的均方根 $\mathrm{RMS}(x)$ 对激活值进行重新缩放。使用 RMSNorm 的 Transformer 模型相比于之前 LayerNorm 训练的模型在训练速度和性能上均具有一定优势。采用 RMSNorm 的代表性模型包括 Gopher [123] 和 Chinchilla [22]。其计算公式如下所示：

$$\mathrm{RMSNorm}(\boldsymbol{x}) = \frac{\boldsymbol{x}}{\mathrm{RMS}(\boldsymbol{x})} \cdot \gamma, \tag{5.14}$$

$$\mathrm{RMS}(\boldsymbol{x}) = \sqrt{\frac{1}{H}\sum_{i=1}^{H} x_i^2}. \tag{5.15}$$

5.3 主流架构

在预训练语言模型时代，自然语言处理领域广泛采用了预训练 + 微调的范式，并诞生了以 BERT 为代表的编码器（Encoder-only）架构、以 GPT 为代表的解码器（Decoder-only）架构和以 T5 为代表的编码器-解码器（Encoder-decoder）架构的大规模预训练语言模型。随着 GPT 系列模型的成功发展，当前自然语言处理领域走向了生成式大语言模型的道路，解码器架构已经成为了目前大语言模型的主流架构。进一步，解码器架构还可以细分为两个变种架构，包括因果解码器（Causal Decoder）架构和前缀解码器（Prefix Decoder）架构。值得注意的是，学术界所提到解码器架构时，通常指的都是因果解码器架构。图 5.6 针对这三种架构进行了对比。

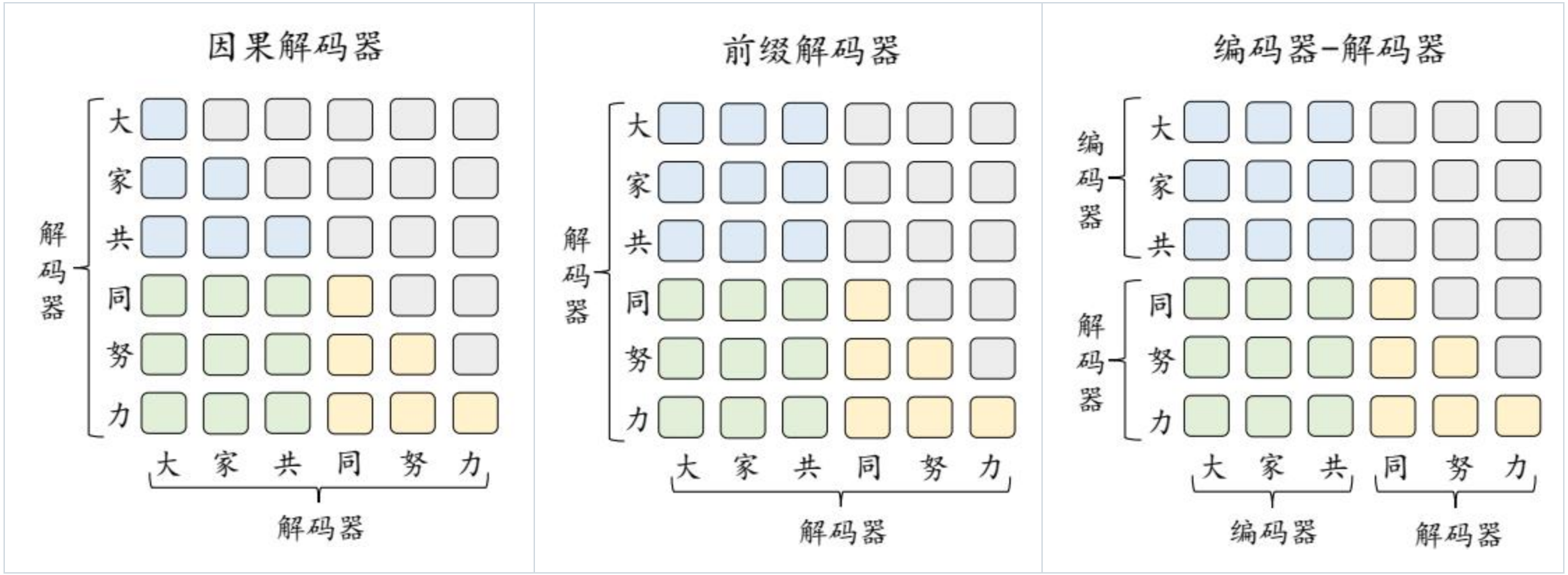


图5.6 三种主流架构的注意力模式比较示意图（蓝色、绿色、黄色和灰色的圆角矩形分别表示前缀词元之间的注意力、前缀词元和目标词元之间的注意力、目标词元之间的注意力以及掩码注意力，图片来源：[10])

5.3.1 编码器-解码器架构

编码器-解码器架构是自然语言处理领域里一种经典的模型结构，广泛应用于如机器翻译等多项任务。原始的Transformer模型也使用了这一架构，组合了两个分别担任编码器和解码器的Transformer模块（详细阐述参见第5.1节）。如图5.6所示，此架构在编码器端采用了双向自注意力机制对输入信息进行编码处理，而在解码器端则使用了交叉注意力与掩码自注意力机制，进而通过自回归的方式对输出进行生成。基于编码器-解码器设计的预训练语言模型（诸如T5[77]等）在众多自然语言理解与生成任务中展现出了优异的性能，但是目前只有如FLAN-T5[39]等少数大语言模型是基于编码器-解码器架构构建而成的。

5.3.2 因果解码器架构

当前，绝大部分主流的大语言模型采用了因果解码器架构。因果解码器采用了Transformer中的解码器组件，同时做出了几点重要改动。首先，因果解码器没有显式地区分输入和输出部分。如图5.6所示，**该架构采用了单向的掩码注意力机制，使得每个输入的词元只关注序列中位于它前面的词元和它本身，进而自回归地预测输出的词元**。此外，由于不含有编码器部分，因果解码器删除了关注编码器表示的交叉注意力模块。经过自注意力模块后的词元表示将直接送入到前馈神经网络中。在因果解码器架构中，最具有代表性的模型就是OpenAI推出的GPT系列。其中，GPT-3将模型参数拓展到了100B级别，并展现出了强大的零样本和少样本学习能力。伴随着GPT-3的成功，因果解码器被广泛采用于各种大语言模型中，包括BLOOM、LLaMA和Mistral等。

5.3.3 前缀解码器架构

前缀解码器架构也被称为非因果解码器架构，对于因果解码器的掩码机制进行了修改。该架构和因果解码器一样，仅仅使用了解码器组件。与之不同的是，该架构参考了编码器-解码器的设计，对于输入和输出部分进行了特定处理。如图5.6所示，前缀解码器对于输入（前缀）部分使用双向注意力进行编码，而对于输出部分利用单向的掩码注意力利用该词元本身和前面的词元进行自回归地预测。与编码器-解码器不同的是，前缀解码器在编码和解码过程中是共享参数的，并没有划分为独立的解码器和编码器。对于前缀解码器，也可以由现有的因果解码器继续预训练转换而来，进而加速该模型的训练。例如，U-PaLM[175]是从PaLM[33]继续预训练而来的。当前，基于前缀解码器架构的代表性大语言模型包括GLM-130B[162]和U-PaLM[175]。

第六章 模型预训练

6.1 预训练任务

在进行模型的大规模预训练时，往往需要设计合适的自监督预训练任务，使得模型能够从海量无标注数据中学习到广泛的语义知识与世界知识。目前，常用的预训练任务主要分为三类，包括语言建模（Language Modeling, LM）、去噪自编码（Denoising Autoencoding, DAE）以及混合去噪器（Mixture-of-Denoisers, MoD）。

图6.1展示了这三种任务各自的输入与输出示例。

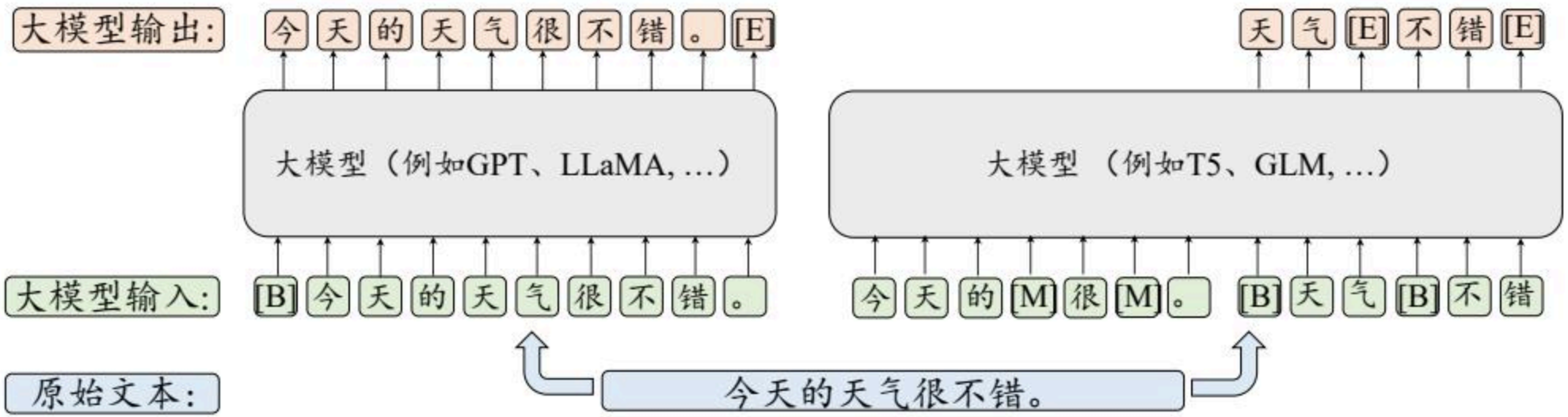


图6.1 语言建模和去噪自编码的输入输出对比

6.1.1 语言建模

语言建模任务是目前绝大部分大语言模型广泛采用的预训练任务。该任务的核心在于"预测下一个词元"，并且经常被应用于训练基于解码器的大语言模型，例如GPT-3[23]和PaLM[33]等。形式化来说，给定一个词元序列 $\boldsymbol{u} = \{u_1, \ldots, u_T\}$，语言建模任务的目标定义为词元的预测任务：基于序列中当前位置之前的词元序列 $\boldsymbol{u}_{<t}$，采用自回归的方式对于目标词元 u_i 进行预测。在训练过程中，模型通常根据以下的似然函数进行优化：

$$\mathcal{L}_{\mathrm{LM}}(\boldsymbol{u}) = \sum_{t=1}^{T} \log P\left(u_t \mid \boldsymbol{u}_{<t}\right). \tag{6.1}$$

可以发现，语言建模任务与人类生成语言数据（如口语表达、书面写作等）的方式存在相似之处，都是基于前序内容生成（或预测）后续的内容。尽管这种对下一个词元的预测看似简单，但当预训练数据足够丰富时，大语言模型便能够学习到自然语言的生成规律与表达模式。正如IlyaSutskever在接受黄仁勋采访的时候给出的解释，**通过对词元更精准的预测，模型就可以更好地理解文本、建模世界语义知识**。该访谈实录原文摘录如例6.1所示。

> Say you read a detective novel. It's like complicated plot, a storyline, different characters, lots of events, mysteries like clues, it's unclear. Then, let's say that at the last page of the book, the detective has gathered all the clues, gathered all the people and saying, "okay, I'm going to reveal the identity of whoever committed the crime and that person's name is". Predict that word. ...
>
> Now, there are many different words. But predicting those words better and better, the understanding of the text keeps on increasing. GPT-4 predicts the next word better.

例 6.1 Ilya Sutskever 对于预测下一个词元任务有效性的解释

此外，**从本质上看，基于语言建模的预训练还可以看作是一种多任务学习过程。例如，在预测句子前缀“这部电影剧情饱满，演员表演得也很棒，非常好看”中的“好看”时，模型实际上在进行情感分析任务的语义学习；而在预测句子前缀“小明有三块糖，给了小红两块糖，还剩下一块糖”中的“一块糖”时，则是在进行数学算术任务的语义学习**。可以列举出来更多类似的例子，覆盖更广的任务范围。因此，**基于大规模文本语料的预训练任务能够潜在地学习到解决众多任务的相关知识与能力**。

语言建模的一个重要变种是前缀语言建模（Prefix Language Modeling）任务，这种任务专门为采用前缀解码器架构（详细介绍见第5.3.3节）的模型而设计。在训练阶段，每个文本序列 $\boldsymbol{u}$ 会根据随机选择的位置 $k(1 \leq k \leq T)$ 切分为前缀 $\boldsymbol{u}_{\text{prefix}} = \{u_1, \ldots, u_k\}$ 和后缀 $\boldsymbol{u}_{\text{suffix}} = \{u_{k+1}, \ldots, u_T\}$ 两个部分。与标准语言建模任务不同，**在前缀解码器中，仅后缀中的词元损失会被计入总损失**。该任务的训练目标函数可以形式化地表示为：

$$\mathcal{L}_{\text{prefix}}(\boldsymbol{u}) = \log P\left(\boldsymbol{u}_{\text{suffix}} \mid \boldsymbol{u}_{\text{prefix}}\right) = \sum_{t=k+1}^{T} \log P\left(u_t \mid \boldsymbol{u}_{<t}\right). \tag{6.2}$$

可以看到，前缀语言建模任务本质上是基于前缀信息来预测后缀的词元，这与自然语言处理任务中常见的基于输入来预测输出的模式十分相似。然而，在模型预训练阶段的损失函数中，由于并未将所有词元的损失都纳入计算，当使用相同规模的数据集进行训练时，采用前缀语言建模训练的模型在性能上通常会稍逊于使用标准语言建模任务训练的模型[191]。

语言建模的另一个重要变种是中间填充任务[192]。此任务通过重新调整输入序列的顺序，旨在训练模型对于中间缺失信息的填充能力。具体来说，一个输入序列 $\boldsymbol{u}$ 被划分为三个部分：前缀 u_{prefix} 、中间部分 u_{middle} 和后缀 u_{suffix} 。随后，中间部分被移至序列末尾。因此，模型需要自回归地对新序列 $u_{\text{prefix}} \oplus u_{\text{suffix}} \oplus u_{\text{middle}}$ 进行预测。通过这种方式，模型能够学习填充中间缺失信息的能力。这种任务的训练函数可表示如下：

$$\begin{aligned}\mathcal{L}_{\text{FIM}}(\boldsymbol{u}) = \log P\left(\boldsymbol{u}_{\text{prefix}}\right) + \log P\left(\boldsymbol{u}_{\text{suffix}} \mid \boldsymbol{u}_{\text{prefix}}\right) \\ + \log P\left(\boldsymbol{u}_{\text{middle}} \mid \boldsymbol{u}_{\text{prefix}}, \boldsymbol{u}_{\text{suffix}}\right).\end{aligned} \tag{6.3}$$

通常来说，**中间填充任务被用作标准语言建模方法的辅助任务。在保留预测下一个词能力的同时，这种方法使得模型具备对于文本中间部分内容的恢复能力。这种预训练任务经常被用于训练代码预训练模型，从而提升模型在代码补全等实际应用场景中的表现。**

6.1.2 去噪自编码

除了传统的语言建模任务外，去噪自编码任务是另一种常见的语言模型预训练任务，广泛应用于BERT、T5等预训练语言模型中[13,77]。在去噪自编码任务中，输入文本经过一系列随机替换或删除操作，形成损坏的文本 $\boldsymbol{u}_{\setminus \tilde{u}}$ 。模型的目标是根据这些损坏的文本恢复出被替换或删除的词元片段 $\tilde{u}$ 。去噪自编码器的训练目标可以用以下数学公式表示：

$$\mathcal{L}_{\mathrm{DAE}}(\boldsymbol{u}) = \log P(\tilde{\boldsymbol{u}} | \boldsymbol{u}_{\setminus \tilde{u}}). \tag{6.4}$$

与语言建模相比，**去噪自编码任务的实现更为复杂，需要设定额外的优化策略，如词元替换策略、替换片段长度、替换词元比例等**。这些策略的选择会直接影响模型的训练效果。尽管去噪自编码任务在许多预训练语言模型中得到了广泛应用。然而，相比于语言建模任务，目前完全使用去噪自编码进行预训练的大语言模型还较为有限。代表性的模型包括FLAN-T5。

6.1.3 混合去噪器

混合去噪器，又称UL2损失[193]，**通过将语言建模和去噪自编码的目标均视为不同类型的去噪任务，对于预训练任务进行了统一建模**。具体来说，混合去噪器定义了三种去噪器：S-去噪器、R-去噪器和X-去噪器。

S-去噪器与前缀语言建模的目标相同（如公式6.2所示），旨在训练模型学习基于给定前缀信息生成合理的后缀文本的能力。相比之下，R-去噪器和X-去噪器与去噪自编码任务的优化目标更为相似（如公式6.4所示）。二者仅仅在被掩盖片段的跨度和损坏比例上有所区别。R-去噪器屏蔽序列中约 15% 的词元，且每个被屏蔽的片段仅包含3到5个词元。而X-去噪器则采用更长的片段（12个词元以上）或更高的损坏比例（约 50% ），进而要求模型能够精准还原原始信息。这种设置增加了任务难度，迫使模型学习到更全面的文本表示。

为了引导模型针对不同类型的输入选择相应的去噪器，输入句子会以特殊词元（如[R],[S],[X]）作为开头。这种标记方式引导模型识别输入中使用的去噪器，并对该去噪器损坏的词元进行还原。混合去噪器被应用于训练 UL2 [193] 和 PaLM 2 [194] 等大语言模型。

6.4.1 参数量计算

由于当前主流的大模型普遍采用因果解码器架构，因此下面以LLaMA模型为范例，深入剖析其参数数量计算方式。对于其他模型，其参数量计算算法可参照此方法计算。 首先，假设词表大小为 V ，模型包含 L 层解码器，中间状态的维度大小为 H ，前馈网络层的中间状态维度大小为 H' .....

.....

累积输入嵌入层、输出层和 L 层解码器每层的多头注意力层、前馈网络层和归一化层，LLaMA模型的参数量计算公式为：

$$\text{参 数 量} = 2VH + H + L \cdot \left(4H^2 + 3HH' + 2H\right). \tag{6.5}$$

第七章 指令微调

指令微调（Instruction Tuning）是指使用自然语言形式的数据对预训练后的大语言模型进行参数微调，这一术语由谷歌研究员在2022年的一篇ICLR论文中正式提出[39]。在另外一些参考文献中，指令微调也被称为有监督微调（Supervised Fine-tuning）[28]或多任务提示训练（Multitask Prompted Training）[40]。指令微调过程需要首先收集或构建指令化的实例，然后通过有监督的方式对大语言模型的参数进行微调。经过指令微调后，大语言模型能够展现出较强的指令遵循能力，可以通过零样本学习的方式解决多种下游任务。在本章中我们将介绍指令数据的构建方法（第7.1节）和相应的训练策略（第7.2节），然后介绍低资源场景下的参数高效微调方法（第7.3节），最后给出指令微调和参数高效微调的代码实践与实验结果分析（第7.4节）。

7.1 指令数据的构建

一般来说，一个经过指令格式化的数据实例包括任务描述（也称为指令）、任务输入-任务输出以及可选的示例。在第3章中介绍了代表性的指令数据集合（参考表3.3）。下面，我们将主要介绍三种构建格式化指令数据的方法，并进一步讨论指令数据构造过程中需要考虑的重要因素。

7.1.1 基于现有的NLP任务数据集构建

学术界围绕传统NLP任务（如机器翻译、文本摘要和文本分类等）发布了大量的开源数据集合，这些数据是非常重要的监督学习数据资源，可以用于指令数据集的构造。通常来说，这些NLP数据集都包括输入和输出两个主要部分。例如，在中英翻译任务中，输入是"*大语言模型已经成为机器学习的一个重要研究方向*"，而相应的输出则是"*Large language models have become one important research direction for machine learning*"。**为了生成指令化的训练数据，一个非常关键的步骤就是为上述的"*输入-输出*"对数据添加任务描述信息，用于指导模型去理解任务目标以及相关信息**。在上述的例子中，可以向中译英的翻译数据集中添加指令，例如"请把这个中文句子翻译成英文"。通过上述操作，就可以将一个NLP任务的数据实例全部通过自然语言形式进行表达，进而数据实例可以被用于大语言模型的指令微调。

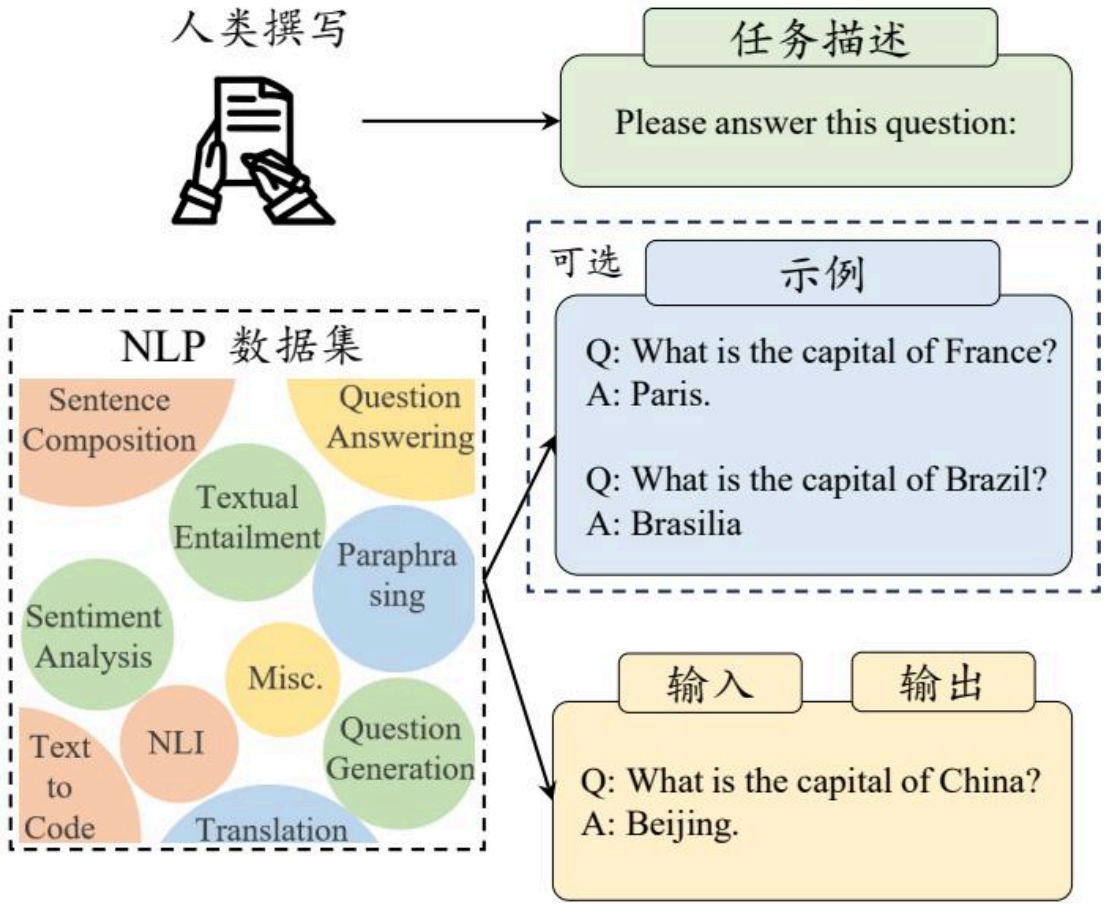


图7.1 现有NLP数据集的指令格式化示意图（图片来源：[10]）

经过NLP指令数据微调后，大语言模型可以学习到指令遵循（Instruction Following）的能力，进而能够解决其他未见过的NLP任务[39-41]。**相关研究表明在现有NLP数据集的输入-输出数据中添加合适的任务描述是提升大模型指令跟随能力的关键因素；如果去除了这些任务描述，仅使用输入-输出数据对模型进行微调，模型的性能会出现显著下降**[39]。为了更好地标注NLP指令微调数据，研究人员开发了众包平台PromptSource，它能够高效地支持标注人员为不同数据集创建、共享及验证任务描述。此外，**为了进一步丰富训练实例，还可以设计特定的任务描述来翻转现有实例的输入-输出对，从而得到新的任务数据用于指令微调**。例如，给定一个问题-答案对，传统的训练方式是基于问题预测答案；同时也可以基于答案生成问题，以此方式来创建一个新的"问题生成"实例，比如可以使用任务描述"*请基于以下答案生成一个问题:*")。

在第3章中，表3.3汇总了常见的NLP任务数据集合，如FLAN、P3、Super-Natural Instructions等。进一步，谷歌的研究人员按照一定的比例混合了这三个数据集合和思维链数据，推出了包含约20M条实例的指令微调数据集合FLAN v2[41]，该数据集合目前是面向NLP任务微调大模型的首选指令集合。

7.1.2 基于日常对话数据构建

尽管通过指令格式化已有的NLP数据集能够获得大量的指令数据实例，但是这些数据的多样性比较局限，与人类的真实需求也不能很好匹配。为此，研究人员开始使用用户在日常对话中的实际需求作为任务描述。例如，InstructGPT[28]**将用户提交给OpenAI API的查询作为任务描述。由于这些用户查询源自于真实应用场景，均采用自然语言形式进行表达，因此特别适合大模型学习指令跟随能力**。为了进一步增加任务的多样性，OpenAI还雇佣标注者创作更多的现实生活任务，包括开放式生成、开放式问答、头脑风暴等任务，然后由另一组标注者来回答这些问题作为输出。OpenAI最终将指令（用户真实查询或者人工标注的任务）和期望的输出（人工编写的答案）配对作为一个训练实例。但是，OpenAI没有对外开放所使用的指令数据。

> *用户第一轮输入：激活函数有什么作用？*
>
> *模型第一轮输出：激活函数是神经网络中的一种非线性函数，通常应用于神经元的输出上。它的作用是为神经网络引入非线性特性，使得神经网络可以更好地逼近复杂的非线性函数。如果不使用激活函数，神经网络将只能实现线性变换，无法学习复杂的非线性关系。……*
>
> *用户第二轮输入：什么是复杂的非线性关系？*
>
> *模型第二轮输出：复杂的非线性关系指的是输入与输出之间存在着非常复杂的非线性映射关系，无法用简单的线性函数来描述。这样的关系可能是高维度、非平稳、非对称等。例如，在图像识别任务中，一张图片中的不同物体之间的关系非常复杂，无法用简单的线性函数来描述。……*
>
> ……（省略后续多轮对话）

例 7.1 ShareGPT 中的一个多轮日常对话数据

为了推动指令微调数据集的开放共享，最近研究人员创建并且开源了一些常见的日常对话任务指令集，例如Dolly[105]和OpenAssistant[104]，这些数据均由人工精心标注，因此具有较高的质量。此外，研究人员还尝试通过自行构建的开放平台收集大量的用户对话请求作为输入数据，并使用ChatGPT或GPT-4生成回复作为输出数据。其中，一个代表性的数据集是来自于ShareGPT的多轮对话数据[38]。

7.1.5 指令微调的作用

在这一部分中，我们主要从三个方面讨论指令微调对大语言模型的影响。

整体任务性能改进

指令微调旨在使用人工构建的指令数据对于大语言模型进一步训练，从而增强或解锁大语言模型的能力[41]。相关研究表明，不同规模的语言模型（参数量规模从77M到540B）都可以从指令微调中受益[41,103]，提升的性能随着参数规模的增加而提升[205]。此外，**经过指令微调的小模型甚至可以比没有经过微调的大模型表现得更出色，进一步凸显了指令微调的有效性**[40,41]。除了参数规模外，指令微调在不同的模型架构（编码器-解码器和因果解码器）、预训练目标（语言建模和去噪自编码）和模型微调方法（序列到序列损失和混合去噪器）上都能取得相对稳定的增益[41]，这说明指令微调是一种非常通用的模型能力增强方法[41]。同时，与预训练相比，指令微调的成本显著降低，大模型所需的指令数据量仅为预训练阶段的约万分之一甚至更少。

任务求解能力增强

指令微调旨在指导模型学会理解自然语言指令，并据此完成相应的任务。通过指令微调，大模型能够获得较好的指令遵循与任务求解能力，无需下游任务的训练样本或者示例就可以解决训练中未见过的任务。**指令微调还可以缓解预训练阶段大模型会出现的一些常见问题，例如生成重复内容或者仅仅补全输入而不解决相关任务**[28,41]。此外，使用英文指令微调数据训练的大模型还可以将相应的能力泛化到其他语言的相关任务上。例如，BLOOM是一个多语言的预训练语言模型[100]，研究人员使用纯英文任务集合P3对其进行指令微调得到BLOOMZ-P3模型[205]。有趣的是，在多语言句子补全任务中，BLOOMZ-P3相比于基座模型BLOOM表现提升超过 50% 。这些实验结果表明，指令微调能够帮助大模型从纯英文数据中获得较为通用的任务解决能力，并将这些能力迁移到其他语言[205]。

领域专业化适配

通用的大语言模型能够在传统自然语言处理任务（如生成和推理）以及日常生活任务（如头脑风暴）上取得较好的效果，然而它们在特定领域中（如医学、法律和金融等）的表现与领域专用模型的效果仍有一定差距。在实际应用中，可以针对大语言模型进行面向特定领域的指令微调，从而使之能够适配下游的任务。以医学领域为例，研究人员提出使用医学数据集对FLAN-PaLM[41]进行微调，得到了医学知识助手模型Med-PaLM[206]，其性能水平可与专业临床医生相媲美；国内研究学者也开源了基于LLaMA指令微调后的医学模型，例如“本草”[207]。在电子商务领域，研究人员也针对大模型进行微调，从而使之适配于推荐系统中的多种任务[208]，取得了出色的效果提升。与此同时，研究人员还在法律、金融等领域探索了指令微调大模型的适配性[209, 210]。这些工作表明，指令微调为大模型提供了一种通用的领域适配方法，拓宽了它们在实际场景中的应用范围。我们将在第13章针对大模型在不同领域的应用进行更为详细的讨论。

7.3.1 低秩适配微调方法

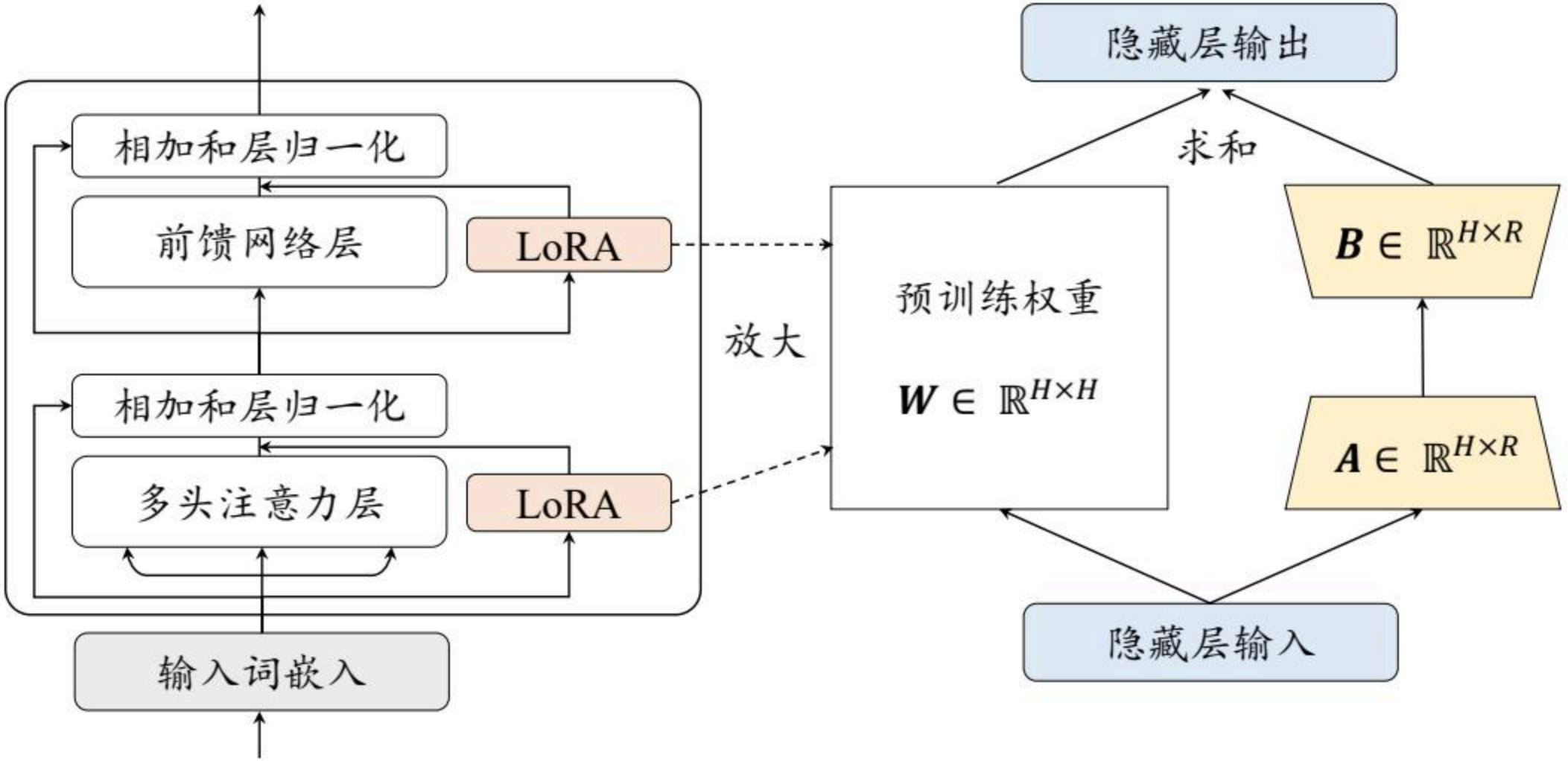


图7.3 LoRA微调示意图

本节中，我们首先介绍基础的低秩适配（Low-Rank Adaptation, LoRA）微调技术，然后介绍它的一些变种和在大模型场景下的应用。

LoRA 基础

大语言模型中包含大量的线性变换层（详见第6.4.2节），其中参数矩阵的维度通常很高。研究人员[214]发现模型在针对特定任务进行适配时，参数矩阵往往是过参数化（Over-parametrized）的，其存在一个较低的内在秩。为了解决这一问题，**LoRA[214]提出在预训练模型的参数矩阵上添加低秩分解矩阵来近似每层的参数更新，从而减少适配下游任务所需要训练的参数**。给定一个参数矩阵 W ，其更新过程可以一般性地表达为以下形式：

$$\boldsymbol{W} = \boldsymbol{W}_0 + \Delta \boldsymbol{W}, \tag{7.1}$$

其中， W_0 是原始参数矩阵， ΔW 是更新的梯度矩阵。

LoRA 的基本思想是冻结原始矩阵$W_0 \in \mathbb{R}^{H\times H}$，通过低秩分解矩阵 $A \in \mathbb{R}^{H\times R}$ 和 $B \in \mathbb{R}^{H\times R}$ 来近似参数更新矩阵 $\Delta W = A \cdot B^\top$，其中 $R \ll H$ 是减小后的秩。在微调期间，原始的矩阵参数 W_0 不会被更新，低秩分解矩阵 A 和 B 则是可训练参数用于适配下游任务。在前向传播过程中，原始计算中间状态 $h = W_0 \cdot x$ 的公式修改为：

$$\boldsymbol{h} = \boldsymbol{W}_0 \cdot \boldsymbol{x} + \boldsymbol{A} \cdot \boldsymbol{B}^\top \cdot \boldsymbol{x}. \tag{7.2}$$

在训练完成后，进一步将原始参数矩阵 $\mathbf{W}_0$ 和训练得到的权重 $\mathbf{A}$ 和 $\mathbf{B}$ 进行合并：$\mathbf{W} = \mathbf{W}_0 + \mathbf{A} \cdot \mathbf{B}^{\mathrm{T}}$，得到更新后的参数矩阵。因此，LoRA微调得到的模型在解码过程中不会增加额外的开销。

LoRA所需的显存估计

在第6.4.4节中，我们已经分析了全量微调场景下需要的显存大小，这里继续使用前文的计算方法来估算LoRA微调节省的显存资源，此处以不使用ZeRO技术为例。这里假设LoRA需要训练的参数量为 P_{LORA} ，模型原始参数为 P 。考虑到模型参数与优化器是显存占用的主要部分，这里主要考虑它们的大小，其它显存占用部分与第6.4.4节几乎一致且占比较小，因此忽略不计。LoRA微调需要保存的模型参数量为 $2P + 2P_{\mathrm{LORA}}$ ，梯度和优化器参数总计 $2P_{\mathrm{LORA}} + 4P_{\mathrm{LORA}} + 4P_{\mathrm{LORA}} + 4P_{\mathrm{LORA}} = 14P_{\mathrm{LORA}}$ ，因此LoRA微调需要的显存大小从全量微调的 $16P$ 大幅减少为 $2P + 16P_{\mathrm{LORA}}$ 。

一般来说，LoRA主要被应用在每个多头注意力层的4个线性变换矩阵上（即 $\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V, \boldsymbol{W}^O \in \mathbb{R}^{H \times H}$ ），因此 $P_{\mathrm{LORA}} = 4 \cdot 2 \cdot L \cdot HR, L, H, R$ 分别是模型层数、中间状态维度和秩。以LLaMA(7B）（ $L = 32, H = 4096$ ）为例，常见的秩 R 设置为8，则 $P_{\mathrm{LORA}} = 8388608$，$2P + 16P_{\mathrm{LORA}} = 13611048960 = 14GB$, $16P = 107814649856 = 108GB$ 。可以看到，模型微调需要的显存大小从108GB大幅下降到14GB，能够有效减少微调模型所需要的硬件资源。考虑到 $P_{\mathrm{LORA}} \ll P$ ，可以近似地认为轻量化微调需要的显存从 $16P$ 降至 $2P$ 。

第八章 人类对齐

大语言模型的能力主要源自于对于海量文本数据的学习，因此大模型的行为会受到数据质量、数据来源以及具体创作者等多方面的影响。经过大规模的预训练（第6章）和有监督指令微调（第7章），大语言模型具备了解决各种任务的通用能力和指令遵循能力，但是同时也可能生成有偏见的、冒犯的以及事实错误的文本内容。这些潜在的有害行为，可能在下游应用中产生严重的影响与危害，进一步被恶意使用者进行放大与利用。因此，在大语言模型的学习过程中，如何确保大语言模型的行为与人类价值观、人类真实意图和社会伦理相一致成为了一个关键研究问题，通常称这一研究问题为人类对齐（Human Alignment）。本章将系统讨论大语言模型的人类对齐问题。首先，我们将概述人类价值观对齐的背景与标准（第8.1节）。随后，第8.2节将重点介绍实现人类对齐的关键技术——基于人类反馈的强化学习（Reinforcement Learning from Human Feedback, RLHF），包括人类反馈的收集方法、奖励模型的训练过程、强化学习训练策略以及相关的RLHF工作。此外，本章还将探讨非强化学习的人类对齐方法（第8.3节），并对有监督微调算法与强化学习对齐算法进行对比分析（第8.4节）。

8.1 人类对齐的背景与标准

在这一小节，我们将介绍人类对齐这一关键研究问题产生的背景，以及目前广泛用于人类对齐的标准，包括有用性、诚实性和无害性等。

8.1.1 背景

尽管大语言模型在下游任务中表现出优秀的性能，这些模型有时会出现错误或具有危害性的行为，例如无法正确遵循指令、生成虚假信息、以及产生有害、有误导性以及带有偏见的表达。**在大语言模型的预训练和有监督微调的过程中，主要训练目标是根据上下文内容来预测下一个词元。但是，这一过程并未充分考虑人类的价值观或偏好，可能导致大语言模型从数据中学习到不符合人类期望的生成模式**。为了规避这些潜在风险，研究人员提出了“人类对齐”这一关键概念，旨在保证大语言模型的行为与人类期望和价值观相一致[28, 221]。与预训练和指令微调不同，人类对齐需引入全新的评估标准，如有用性、诚实性和无害性。

为了更直观地理解人类对齐对于大语言模型的重要性，例8.1对比了同一个语言模型在对齐前后对于相同输入的不同输出。在这个例子当中，输入的问题刻意包含了具有误导性的逻辑关系，即“土地价格”和“有污染的产业”是有直接关系的。因此，在经过人类价值观对齐之前的大语言模型会被输入中的错误逻辑所引导，产生了带有偏见的建议“农村地区更适合发展污染较严重的产业”。在经济生产中，发展有污染的产业需要综合考虑多方面的因素，不能仅仅因为土地价格更为便宜就认为适合发展相关产业。对齐前的大语言模型给出了一个错误的观点，不符合人类价值观，违背了无害性的原则。而经过与人类价值观对齐之后的大语言模型，先指出了输入问题中包含的错误逻辑（“我们不能简单地认为农村土地价格便宜就适合发展污染产业。”），并且给出了正确且合理的做法。对齐后的大语言模型的回复符合有用性和无害性，与人类价值观和偏好相符。

> **用户输入：** 农村的土地价格要更便宜一些，所以更适合发展有污染的产业吗?
>
> **对齐前模型输出：** 是的，农村的土地价格相对较低，这使得农村地区更适合发展污染较严重的产业。由于土地价格较低，企业可以更容易地获得土地用于建设工厂或其他设施，从而降低了企业的初始投资成本。此外，由于农村地区的土地利用方式相对较为灵活，企业可以更容易地获得建设和运营所需的土地。然而，需要注意的是，发展污染较严重的产业可能会对当地的环境和居民的健康产生负面影响。因此，在发展污染较严重的产业时，需要充分考虑其对环境和当地居民的影响，并采取相应的措施来减少其负面影响。

例 8.1 大语言模型（YuLan）对于相同输入在对齐前后的不同输出

> **对齐后模型输出：** 我们不能简单地认为农村土地价格便宜就适合发展污染产业。虽然土地价格相对较低，但是污染产业可能会对环境和当地居民的生活造成负面影响。此外，发展污染产业也可能导致资源浪费和生态破坏，从长远来看并不利于可持续发展。
>
> 正确的做法应该是，在确保环境保护和可持续发展的前提下，合理利用土地资源，推动符合环保要求的产业发展。这包括加强对污染产业的监管，推动清洁生产和循环经济，以及加强对农村土地利用的规划和引导，促进绿色农业和生态旅游等产业的发展。这样才能实现经济发展和环境保护的双赢局面。

例 8.1 大语言模型（YuLan）对于相同输入在对齐前后的不同输出

8.1.2 对齐标准

人类对齐是一个较为抽象的概念，难以直接进行形式化建模，关于对齐的定义和标准也存在不同的观点。本书主要围绕三个具有代表性的对齐标准展开讨论，分别是有用性（Helpfulness）、诚实性（Honesty）和无害性（Harmlessness），这三种对齐标准已被现有的大语言模型对齐研究广泛使用[28, 222]。下面具体介绍这三个代表性的对齐标准。

- 有用性. 在实际应用中，大语言模型需要提供有用的信息，能够准确完成任务，正确理解上下文，并展现出一定的创造性与多样性。模型应尽量以简洁、高效的方式协助用户完成任务。当任务描述存在歧义或涉及背景信息时，模型应具备主动询问并获取任务相关信息的能力，同时具有一定的敏感度、洞察力和审慎态度。由于用户意图的多样性，有用性这一对齐标准仍然难以进行统一的定义与刻画，需要根据不同的用户进行确定。

- 诚实性. 模**型的输出应具备真实性和客观性，不应夸大或歪曲事实，避免产生误导性陈述，并能够应对输入的多样性和复杂性。在人机交互过程中，大语言模型应向用户提供准确内容，还应适当表达对于输出信息的不确定性程度，以避免任何形式的误导。本质上，这要求模型了解自身的能力和知识水平。与有用性和无害性相比，诚实性是一个更为客观的标准，对人类标注的依赖相对较少。**
- 无害性. **大语言模型应避免生成可能引发潜在负面影响或危害的内容。在处理敏感主题时, 模型应遵循道德标准和社会价值观, 从而消除冒犯性与歧视性。此外, 模型需要能够检测到具有恶意目的的查询请求。当模型被诱导执行危险行为 (如犯罪行为) 时, 应直接予以拒绝。然而, 何种行为被视为有害, 很大程度上取决于大语言模型的使用者、用户问题类型以及使用大语言模型的背景。**

上述三种通用的对齐标准较为宽泛，因此许多研究针对性地给出了一些更为细化的对齐标准，以更全面地规范大语言模型的输出。例如，**行为对齐要求人工智能系统能够做出符合人类期望的行为；在此基础上，意图对齐则进一步要求大语言模型在意图和行为上都要与人类期望保持一致**，这涉及到哲学、心理学以及技术细节上的多重挑战；道德对齐要求语言模型应避免涉及非法、不道德或有害的话题，在回应中优先考虑用户安全、道德准绳和行为边界。这些对齐标准在本质上与前述三个标准是相似的，研究人员可以根据任务的特定需求进行调整。

通过上述内容的介绍，可以看到已有的对齐标准主要是基于人类认知进行设计的，具有一定的主观性。因此，直接通过优化目标来建模这些对齐标准较为困难。我们将在第8.2节介绍基于人类反馈的强化学习方法，引入人类反馈的指导，以便更好地对齐大语言模型。由于强化学习的训练过程较为复杂，还可以采用监督微调方法来代替强化学习对模型进行对齐，这部分内容将在第8.3节进行介绍。此外，在实践中，红队攻击（Red Teaming）技术也被广泛运用，通过人工或自动化的手段，以对抗方式探测大语言模型，诱导其生成有害输出，并据此针对性地调整大语言模型，以避免产生此类有害输出[223]。

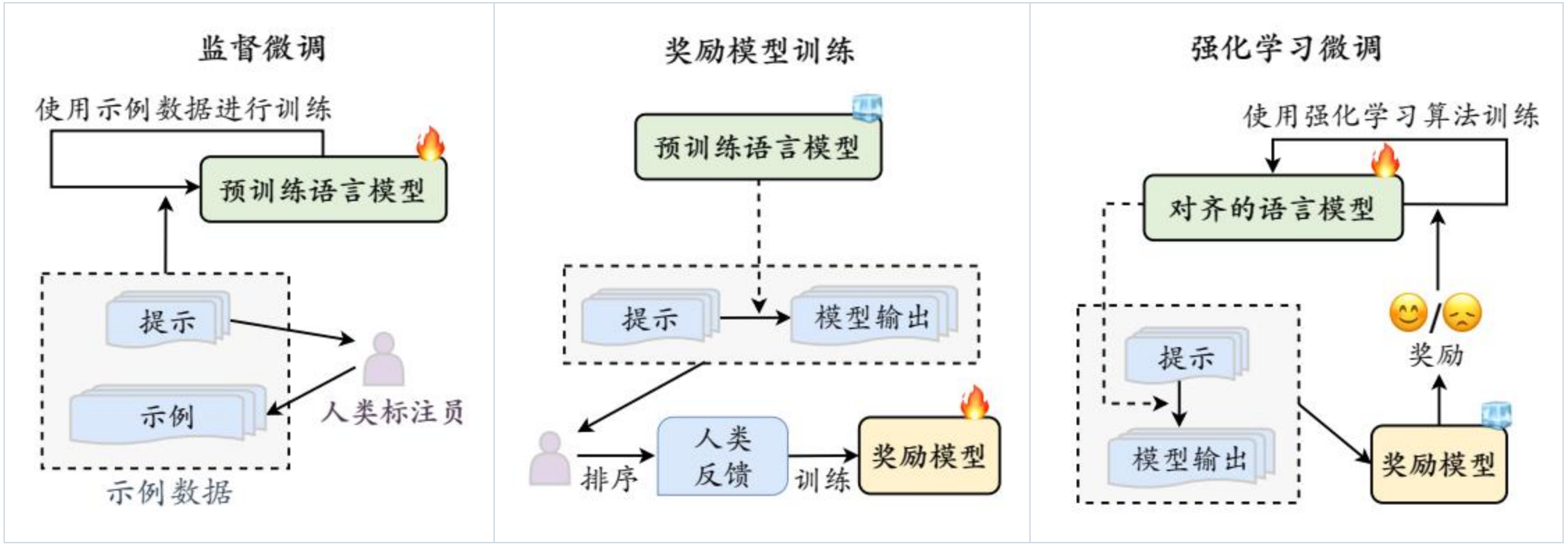


图 8.1 基于人类反馈的强化学习的工作流程（图片来源： [10]）

8.2 基于人类反馈的强化学习

由于对齐标准难以通过形式化的优化目标进行建模，因此研究人员提出了基于人类反馈的强化学习（Reinforcement Learning from Human Feedback, RLHF），引入人类反馈对大语言模型的行为进行指导。在这一小节，我们将首先介绍基于人类反馈的强化学习的整体框架，在此基础上，进一步详细说明人类反馈的收集过程、奖励模型的训练和强化学习算法。

8.2.1 RLHF概述

为了加强大语言模型与人类价值观的一致性，基于人类反馈的强化学习旨在利用收集到的人类反馈数据指导大语言模型进行微调，从而使得大语言模型在多个标准（例如有用性、诚实性和无害性）上实现与人类的对齐。RLHF 首先需要收集人类对于不同模型输出的偏好，然后使用收集到的人类反馈数据训练奖励模型，最后基于奖励模型使用强化学习算法（例如 Proximal Policy Optimization, PPO [51]）微调大语言模型。这种将人类反馈纳入大语言模型训练过程的方法已成为实现人类对齐的主要技术途径之一。

RLHF算法系统

RLHF算法系统主要包括三个关键组成部分：需要与人类价值观对齐的模型、基于人类反馈数据学习的奖励模型以及用于训练大语言模型的强化学习算法。具体来说，待对齐模型一般指的是经过预训练、具备一定通用能力的大语言模型。然而，这些模型并没有与人类价值观对齐，在下游任务中可能表现出不合适甚至有害的行为。例如，InstructGPT[28]针对具有175B参数的GPT-3模型进行对齐。GPT-3在大规模语料上进行了预训练，但是在一些特殊场景下仍然会生成不恰当的输出内容。奖励模型的作用是为强化学习过程提供指导信号，反映了人类对于语言模型生成文本的偏好，通常以标量值的形式呈现。奖励模型既可以采用人类偏好数据对已有的语言模型继续微调，也可以基于人类偏好数据重新训练一个新的语言模型。

虽然原始的InstructGPT采用了较小的GPT-3（只有6B参数）作为奖励模型，**现阶段的研究通常认为使用与待对齐模型规模相同或者更大规模的奖励模型可以获得更好的对齐效果**，主要是因为较大规模的奖励模型可以更好地理解待对齐模型的知识与能力范围，从而提供更为合适的指导信号，例如LLaMA-2[58]使用相同的检查点初始化待对齐模型和奖励模型。在训练过程中，基于奖励模型提供的反馈信号，RLHF使用特定的强化学习算法进行大语言模型的训练。目前，PPO算法[51]是一种被广泛用于人类对齐的强化学习算法。

RLHF的关键步骤

图8.1展示了RLHF整体训练框架的三个阶段，下面分阶段进行具体介绍。

- 监督微调。为了让待对齐语言模型具有较好的指令遵循能力，通常需要收集高质量的指令数据进行监督微调。指令数据一般包括任务描述和示例输出，可以由人类标注员针对特定任务编写，也可以由大语言模型自动生成。在InstructGPT中，人类标注员为多个生成任务（如开放性问答、头脑风暴、日常聊天等）编写提示和相应的示例输出（例如“列出五种恢复职业热情的方法”）。由于指令微调的重要性，本书将指令微调单独成章进行具体介绍（参见第7章）。

- 奖励模型训练. 第二步是使用人类反馈数据训练奖励模型。具体来说，首先使用语言模型针对任务指令生成一定数量的候选输出。随后，邀请标注员对于输出文本进行偏好标注，这个标注过程可以采用多种形式，其中最常用的是对候选文本进行排序标注，这样可以有效减少多个标注员之间的不一致情况（具体细节可参考第8.2.2节）。进一步，使用人工标注的偏好数据进行奖励模型的训练，使其能够建模人类偏好。在InstructGPT中，标注员将模型生成的输出按照最佳到最差的顺序进行排序，并据此训练奖励模型来预测这个排序。奖励模型的训练细节可参考第8.2.3节。

- 强化学习训练. 在这一步骤中，语言模型对齐被转化为一个强化学习问题。具体来说，待对齐语言模型担任策略实施者的角色（称为策略模型），它接收提示作为输入并返回输出文本，其动作空间是词汇表中的所有词元，状态指的是当前已生成的词元序列。奖励模型则根据当前语言模型的状态提供相应的奖励分数，用于指导策略模型的优化。为了避免当前训练轮次的语言模型明显偏离初始（强化学习训练之前）的语言模型，通常会在原始优化目标中加入一个惩罚项（如KL散度）。例如，InstructGPT使用PPO算法来优化待对齐语言模型以最大化奖励模型的奖励。对于每个输入提示，InstructGPT计算当前语言模型与初始语言模型生成结果之间的KL散度作为惩罚项。KL散度越大，意味着当前语言模型越偏离初始语言模型。这个对齐过程可以进行多次迭代，从而更好地对齐大语言模型。强化学习的训练细节可参考第8.2.4节。由于强化学习算法的不稳定性，学术界提出了一些采用监督微调的对齐算法，这部分内容将在第8.3节进行介绍。

8.2.2 人类反馈数据的收集

在预训练阶段，大语言模型通过语言建模目标在大规模无标注语料库上进行训练。然而，这一过程无法直接反映人类对于大语言模型输出的主观和定性偏好（本书中称为“人类反馈”）。为了实现有效的人类对齐，需要使用高质量的人类反馈数据对大语言模型进行针对性的微调。接下来，我们将探讨如何选择合适的人类标注员，并收集高质量的反馈数据。

标注人员选择

为了确保人类反馈数据的可靠性，选择合适的标注人员至关重要。一般来说，理想的标注员应具备较高的教育水平以及出色的语言熟练度。例如，Sparrow [222]要求标注员必须是英国本土的英语母语者，并至少具备本科及以上学历。尽管如此，研究人员与标注员之间仍然可能存在意图不匹配的情况，这可能导致生成不一致的反馈数据，进而影响模型的输出。为了解决这一问题，InstructGPT [28]通过对标注员与研究人员之间的标注一致性进行评估来筛选出合适的标注员。具体来说，研究人员首先标注一小部分数据，然后邀请候选标注员进行标注，并计算候选标注员与研究人员标注结果之间的一致性分数。最终，只选择一致性分数较

高的标注员进行后续的标注工作。此外，还可以从一组表现较好的标注员中选出高度一致的"超级标注员"[224]，这些超级标注员将优先与研究人员合作进行后续研究。此外，在标注过程中，提供详细的标注说明和即时指导有助于进一步规范标注员的标注行为。

人类反馈形式

确定标注人员的选择后，可以对大语言模型的输出进行标注，以收集人类反馈数据。在现有工作中，主要有两种人类反馈数据的形式。

- 基于评分的人类反馈。最直接的标注方式是根据预设的标准邀请标注人员对于大语言模型的输出进行打分，从而作为模型输出质量的判断。例如，针对无害性标准“模型不能发表可能伤害用户或其他人的有害评论”，标注人员需要判断该输出是否产生了有害内容，以此获得模型输出的评分，比如遵守某条规则记1分而违反某条规则扣1分。为了获得更细粒度的人类标注，可以从不同角度为对齐标准设计更为具体的标注规则。例如，针对无害性标注，规则设计可以包括“模型不能针对某个群体发表恶意评论”、“模型不能发表威胁性言论”、“模型不能表达偏好、情感、观点或宗教信仰”等。除了人工标注外，还可以使用经过对齐的大语言模型对于特定对齐标准进行标注。GPT-4 [35] 使用基于大语言模型的分类器来判断模型输出是否违反一组预先设定的对齐规则，这些规则以提示的形式加入到大语言模型的输入中，帮助它判断GPT-4的输出是否违反规则并给出评分。

- 基于排序的人类反馈。**排序是一种比较典型的人类偏好标注形式。最简单的方式是标注人员根据自身偏好对于大语言模型的输出进行全排序**。但是，这种方式需要花费较多的人力成本进行标注。在国际象棋、体育竞技等领域，Elo评分系统常被用于衡量竞技比赛中参与者的相对实力，目前也被用于评估大语言模型的生成内容。在Elo评分系统中，通过对模型输出进行两两比较，进而计算每个输出的综合得分并获得最终的输出排序。具体来说，Elo评分系统首先假设每个模型输出都有一个Elo等级分，可用于估计该输出获胜的概率。在两两对决中，如果某个模型输出获胜（也就是标注员更喜欢该输出），那么该模型输出的等级分就会相应上升，反之下降。其中，上升或下降的幅度取决于预估获胜的概率与实际胜负情况，比如该输出获胜的预估概率为0.2，但在实际对决中获胜则为1.0，两者概率差距较大，因此等级分会大幅上升。通过不断重复上述两两比较的过程，可以得到最终每个模型输出的等级分用于排序（关于Elo评分标准的细节介绍可参考第12.1.1节）。因此，Elo排名可作为强化学习训练信号，引导大语言模型更倾向于人类偏好的输出，从而产生更可靠、更安全的输出。

8.2.3 奖励模型的训练

由于 RLHF 的训练过程中需要依赖大量的人类偏好数据进行学习，因此很难在训练过程中要求人类标注者实时提供偏好反馈。为此，我们需要训练一个模型来替代人类在 RLHF 训练过程中实时提供反馈，这个模型被称为奖励模型。在训练开始前，我们需要预先构造一系列相关问题作为输入。人类标注者将针对这些问题标注出符合人类偏好的输出以及不符合人类偏好的输出。收集到这些人类偏好数据后，就可以用来训练奖励模型。经过充分训练的奖励模型能够有效地拟合人类偏好，并在后续的强化学习训练过程中替代人类提供反馈信号。这样一来，就可以在保证训练效率的同时，加强模型行为与人类期望的一致性。

训练方法

奖励模型通过在人类偏好数据上进行训练，进而针对模型输出进行质量的判别，所给出的分数可以在一定程度上反应人类偏好。一般来说，奖励模型是基于语言模型进行设计的，模仿人类标注人员对于模型生成内容进行质量评分，实现对于人类偏好分数的预测。具体来说，线性变换头将语言模型最后一层的隐状态从一个具有词嵌入维度大小的向量 $\mathbb{R}^d$ 映射成一个标量分数 $\mathbb{R}$ ，这个标量分数被用作奖励模型对当前内容的打分。奖励模型的训练方式主要包括如下三种形式：

- **打分式**. 人类标注者需针对给定的输入问题，为相应的输出赋予反馈分数。通常来说，这些分数是离散的数值，用于表示输出与人类偏好的契合程度。奖励模型的核心任务在于学习如何根据输入问题和模型输出进行评分，以确保其评分结果与人类的评分尽可能一致。一般情况下，可以采用均方误差（Mean Square Error, MSE）作为打分式训练方法的目标函数，具体形式下式所示：

$$\mathcal{L} = -\mathbb{E}_{(x,y,\tilde{r})\sim\mathcal{D}}\left[\left(r_\theta(x,y) - \tilde{r}\right)^2\right], \tag{8.1}$$

- 其中，x ，y 和 $\tilde{r}$ 分别表述问题输入、输出和人类标注者对输出的打分，函数 r_θ 表示参数为 θ 的奖励模型，用于对于模型输出进行打分。

- 通过上述训练方法，奖励模型能够学习拟合人类的偏好倾向。然而，人类偏好本质上具有一定的主观性。对于评分标准，**不同标注人员可能会存在不一致的理解**，最终导致对于模型输出进行评分时可能会存在偏差。例如，对于同一个输入 x 和对应的输出 y，评分标准较为宽松的标注人员A可能会给出一个较高的得分 $\tilde{r}_A = 0.9$，而评分标准较为严格的标注人员 B 则给出一个较低的得分 $\tilde{r}_B = 0.6$。因此，在实际应用中，需要采用适当的方法来减少人类主观因素对模型输出评估的影响。

- **对比式. 对比式训练方法一定程度上能够克服打分式训练方法的不足**。针对一个问题输入，人类标注者仅需对两条相应输出进行排序，排序在前的输出被视为正例（更符合人类偏好），另一条输出则被视为负例。**这种标注方式不仅降低了标注难度，还提高了不同标注者之间的一致性**。在学习过程中，通常会采用对比学习的方法对奖励模型进行训练。奖励模型需要学习在提升正例分数的同时，进一步降低负例的分数，以最大化正例和负例之间的分数差异。下式展示了一个简化版的对比式训练方法的损失函数：

$$\mathcal{L} = -\mathbb{E}_{(x,y^+,y^-)\sim\mathcal{D}}[\log(\sigma(r_\theta(x,y^+) - r_\theta(x,y^-)))], \tag{8.2}$$

- 其中，x，y^+ 和 y^- 分别表示模型输入、正例和负例。通过最小化该损失函数，奖励模型能够有效地学习区分正例和负例，从而准确地反映人类偏好。

- **排序式.** 排序式训练方法可以被视为对比式训练方法的一种增强形式。对于一个给定的输入, 人类标注者根据偏好对于多个模型生成的回复进行排序。通过标注的顺序, 可以获得这些回复之间的相对优劣关系, 即哪些回复更符合人类价值观。在优化中, 奖励模型通常采用与对比式方法类似的学习策略来进行打分。假设有一个包含 K 个不同输出的集合 $\mathcal{D}$, 且这 K 个不同的输出已经按照人类偏好进行排序, 奖励模型的训练损失函数可以表示为:

$$\mathcal{L} = -\frac{1}{\binom{K}{2}}\mathbb{E}_{(x,y^+,y^-)\sim\mathcal{D}}[\log(\sigma\left(r_\theta\left(x,y^+\right)-r_\theta\left(x,y^-\right)\right))]. \tag{8.3}$$

- 需要注意的是, 这里的排序式训练方法考虑了所有 K 个输出之间的两两偏序关系。相比于对比式的训练方式, 基于排序式方法训练的奖励模型能够在一定程度上学习到更为全局的排序关系, 进而更好地学习和拟合人类的价值观和偏好。

8.4 关于SFT和RLHF的进一步讨论

正如在第7章中所介绍的，指令微调是一种基于格式化的指令示例数据（即任务描述与期望输出相配对的数据）对大语言模型进行训练的过程。在大语言模型的背景下，这种利用配对文本进行训练的方法也被广泛地称为监督微调（Supervised Fine-Tuning, SFT）。为了保持与相关学术论文中术语的一致性，我们在本章后续的内容中将主要采用“监督微调”这一术语，而非“指令微调”。在InstructGPT中，研究人员将监督微调（SFT）作为RLHF算法的第一个步骤[28]。为了更好地阐述这两种技术的重要性及其相关特点，我们将SFT和RLHF视为两种独立的大模型训练方法。在本章节，我们将深入探讨这两者之间的联系与差异。表8.1对这两种方法各自的优点和缺点进行了汇总与对比，以便读者能够更清晰地了解它们的特性与差异。

表 8.1 SFT 和 RLHF 的优缺点对比

- **SFT**
  - 优点
    - 1、提高大语言模型在各种基准测试中的性能
    - 2、增强大语言模型在不同任务上的泛化能力
    - 3、提升大语言模型在专业领域的能力
  - 缺点
    - 1、当数据超出大语言模型的知识范围时，模型易产生幻觉
    - 2、通过对教师模型的蒸馏，会增加学生模型出现幻觉的可能性
    - 3、不同标注者对实例数据标注的差异，会影响SFT的学习性能
    - 4、指令数据的质量会影响大语言模型的训练效果

- **RLHF**
  - 优点
    - 1、进一步增强模型的能力，提高模型有用性
    - 2、有效减轻大语言模型出现有害响应的可能性
    - 3、有效减轻大语言模型出现幻觉的可能性
    - 4、偏好标注可以减轻示例生成过程中的不一致情况
  - 缺点
    - 1、训练样本使用效率较低
    - 2、训练过程不稳定，训练过程对超参数敏感
    - 3、依赖强大的SFT模型进行热启动

8.4.1 基于学习方式的总体比较

如第8.2节所述（关于强化学习训练的部分），我们可以将文本生成问题看作为一个基于强化学习的决策过程。具体来说，当给定一个提示作为输入时，大语言模型的任务是生成与任务指令相匹配的输出文本。这个生成过程可以被分解为一系列逐个词元的生成步骤。在每个步骤中，大语言模型会根据已有的策略模型（即模型本身）在当前状态下的情况（包括当前已生成的词元序列以及可利用的上下文信息）来选择下一个动作，即生成下一个词元。

在这种设定下，我们优化的目标是让大语言模型能够不断优化其生成策略，生成更高质量的输出文本，获得更高的奖励分数。总体来说，RLHF和SFT可以被视为两种优化大语言模型决策过程的训练方法。在RLHF中，我们首先学习一个奖励模型，然后利用该奖励模型通过强化学习算法（如PPO）来改进大语言模型。而在SFT中，我们则采用了Teacher-Forcing的方法，直接优化模型对实例输出的预测概率。从本质上说，**SFT所采用的这种词元级别的训练方式是一种"行为克隆"（模仿学习的一种特殊算法，参见文献[233]）。它利用教师的行为数据（即每个步骤的目标词元）作为监督标签，来直接训练大语言模型模仿教师的行为**。在实现上，SFT主要依赖于序列到序列的监督损失来优化模型，而RLHF则主要通过强化学习方法来实现大模型与人类价值观的对齐。本质上来说，为了学习教师的生成策略，SFT采用了基于示例数据的"局部"优化方式，即词元级别的损失函数。作为对比，RLHF则采用了涉及人类偏好的"全局"优化方式，即文本级别的损失函数。关于模仿学习和强化学习的更多理论分析，读者可以参考相关的强化学习文献，如[233,234]等。

8.4.2 SFT的优缺点

SFT已经成为一种主要的大语言模型微调方法，能够显著提升大语言模型在各种基准测试中的性能，增强在不同任务上的泛化能力。它在实现上简单、灵活、可拓展性较强，还可以用于构建很多特定功能，例如帮助大语言模型建立聊天机器人的身份。有关SFT可用性的更多讨论，请参阅第7.1.5节。

关于SFT，人们普遍认为其作用在于“解锁”大语言模型的能力，而非向大语言模型“注入”新能力。因此，试图通过SFT激发大语言模型的非内生能力时，可能会出现一些负面问题。**当待学习的标注指令数据超出了大语言模型的知识或能力范围，例如训练大语言模型回答关于模型未知事实的问题时，可能会加重模型的幻象（Hallucination）行为**。OpenAI强化学习研究团队的负责人、PPO算法的作者John Schulman在一场关于RLHF的讲座中提出了一个有趣的观点：通过蒸馏较大模型来训练较小模型可能会增加模型生成幻觉文本的可能性，从而可能影响大语言模型的事实准确性[235]。实际上，目前无论学术界和工业界都在大量使用GPT-4进行指令微调数据的蒸馏，在这一过程中除了要考虑指令数据本身的质量外，还需要进一步关注模型自身的知识与能力边界，从而减少微调过程中所产生的负面效应，如上述提到的幻象问题。

此外，作为一种基于行为克隆的学习方法，SFT旨在模仿构建标注数据的教师的行为，而无法在这一过程中进行有效的行为探索。然而，标注者在写作风格、创作水平和主题偏好等方面经常存在一定的差异，这些都会使得标注数据中出现不一致的数据特征，进而影响SFT的学习性能。因此，**在SFT阶段，高质量的指令数据（而非数量）是影响大语言模型训练的主要因素**。

8.4.3 RLHF的优缺点

最初，RLHF是在深度强化学习的文献中被提出[50]，随后被借鉴用于提升语言模型的能力。OpenAI首先将其用于生成符合人类偏好的文本摘要[52]，进一步使用这一技术研发了InstructGPT模型[28]。在早期的研究中，研究人员主要关注使用RLHF加强模型对于人类价值观的遵循，减少模型输出的有害性。

在最近的研究中，相关研究发现 RLHF 在减少有害内容输出的同时，也能够有效增强模型的综合能力，这一点在 LLaMA-2 的论文中有着充分讨论 [58]。LLaMA-2 [58] 通过广泛的实验证明 RLHF 可以同时提高模型的有用性和无害性分数，并从两个方面解释了 RLHF 相比 SFT 的潜在优势。首先，在 RLHF 算法中，标注员主要为训练过程提供偏好标注数据，而不是直接生成示例数据，因此它可以减少标注员之间的不一致。其次，与编写示例数据相比，偏好标注更为简单易行。**标注员甚至可以评估超出自己创作水平的模型输出质量，使得模型能够探索标注员能力之外的状态空间，而不用受限于给定的教师示例。** 上述这两个方面都使得 RLHF 在数据标注阶段相比 SFT 更加具有优势，更加容易充分发挥人类指导的作用。

在模型学习阶段，RLHF通过对比模型的输出数据（区分“好”输出与“坏”输出）来指导大语言模型学习正确的生成策略，它不再强迫大语言模型模仿教师的示例数据，因此可以缓解上述提到的SFT所导致的幻象问题。在RLHF方法中，奖励模型非常重要。一般来说，奖励模型应该能够了解待对齐的大语言模型的知识或能力范畴。例如，LLaMA-2采用了待对齐语言模型的检查点来初始化奖励模型。实际上，RLHF已被证明是减少GPT-4幻觉生成的重要方法[35]。然而，**RLHF也继承了经典强化学习算法的缺点，如样本学习效率低和训练过程不稳定等问题。因此，当训练语言模型时，RLHF需要依赖于经过SFT的模型作为策略模型的初始模型，从而快速达到较好的表现**。这也是InstructGPT采用SFT作为RLHF方法的第一步的主要原因。此外，RLHF的过程通常会持续多轮，这是一个复杂的迭代优化过程，其中涉及了很多重要细节的设定（例如提示选择、奖励模型训练、PPO的超参数设置以及训练过程中对超参数的调整），都会影响整个模型的性能，对于精确的高效复现提出了较大挑战。

总的来说，**SFT特别适合预训练后增强模型的性能，具有实现简单、快速高效等优点；而RLHF可在此基础上规避可能的有害行为并进一步提高模型性能，但是实现较为困难，不易进行高效优化**。未来的研究仍然需要探索更为有效的对齐方法，同时结合SFT与RLHF的优点。此外，还需要关注当模型能力达到较强水平后更为有效的对齐方法。针对这个问题，OpenAI提出了“超级对齐”(Super-alignment)这一研究方向，旨在能够有效监管具有超级智能的人工智能系统。

第十章 提示学习

经过预训练、指令微调和人类对齐后，我们接下来讨论如何通过提示学习方法来有效地使用大语言模型解决实际任务。目前常用的方法是设计合适的提示(Prompting)，通过自然语言**接口**与大模型进行交互（第10.1节）。在现有研究中，任务提示的设计主要依靠人工设计和自动优化两种策略来实现。为了更好地解决未见过的任务，一种典型的提示方法是上下文学习(In-context Learning, ICL)，它将任务描述与示例以自然语言文本形式加入到提示中(第10.2节)。此外，思维链提示（Chain-of-Thought, CoT）作为一种增强技术，将一系列中间推理步骤加入到提示中，以增强复杂推理任务的解决效果(第10.3节)。

【补充】接口（API）配置：

- 代码、客户端

Python代码实现

```python
import OpenAI

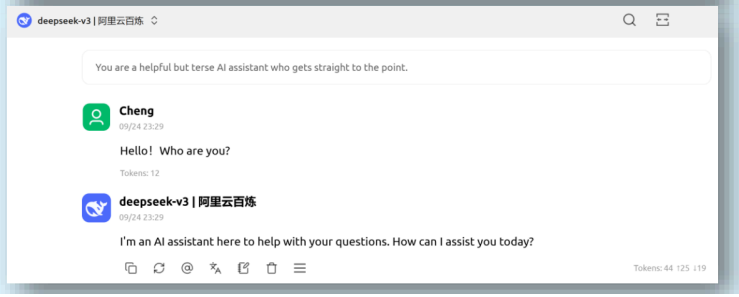
with open('API-key.txt', 'r', encoding='utf-8') as file:
    key = file.read()

client = OpenAI(
    api_key = key,
    base_url = "https://dashscope.aliyuncs.com/compatible-mode/v1/
    ",
)

def get_llm_response(prompt):
    """This function takes as input a prompt, which must be a
    string enclosed in quotation marks,
    and passes it to deepseek-v3 model. The function then saves
    the response of the model as
    a string.
    """
    completion = client.chat.completions.create(
        model="deepseek-v3",
        messages=[
            {
                "role": "system",
                "content": "You are a helpful but terse AI
                assistant who gets straight to the point.",
            },
            {"role": "user", "content": prompt},
        ],
        temperature=0.0,
    )
    response = completion.choices[0].message.content
    return response
```







10.2 上下文学习

在GPT-3的论文[23]中，OpenAI研究团队首次提出上下文学习（In-context learning, ICL）这种特殊的提示形式。目前，上下文学习已经成为使用大语言模型解决下游任务的一种主流途径。下面将详细介绍这一提示方法。

10.2.1 上下文学习的形式化定义

根据GPT-3论文中所给出的描述[23]，上下文学习使用由任务描述和（或）示例所组成的自然语言文本作为提示。图10.1展示了上下文学习的提示构建过程。首先，通过自然语言描述任务，并从任务数据集中选择一些样本作为示例。其次，根据特定的模板，将这些示例按照特定顺序组合成提示内容。最后，将测试样本添加到提示后面，整体输入到大语言模型以生成输出。**基于任务描述以及示例信息，大语言模型无需显式的梯度更新即可识别和执行新的任务**。

形式上，我们使用 $D_k = \{f(x_1, y_1), \ldots, f(x_k, y_k)\}$ 来表示由 k 个样本构成的一组示例数据，其中 $f(x_k, y_k)$ 是一个函数，负责将第 k 个任务样本转换为自然语言提示。给定任务描述 I 、示例 D_k 以及新的输入 x_{k+1} ，大语言模型生成答案 $\hat{y}_{k+1}$ 的过程可以通过下面的公式来表述：

$$\underbrace{\mathrm{LLM}}_{\text{大语言模型}}\left(\underbrace{I}_{\text{任务描述}}, \underbrace{f\left(x_1, y_1\right), \ldots, f\left(x_k, y_k\right)}_{\text{示例}}, f(\underbrace{x_{k+1}}_{\text{输入}}, \underbrace{__}_{\text{答案}})\right) \rightarrow \hat{y}_{k+1}. \quad (10.1)$$

值得一提的是，上下文学习与指令微调（详见第7章）之间存在着紧密的联系，因为它们都涉及将任务或样本转化为自然语言形式供大语言模型进行处理。在原始的GPT-3论文中，作者将上下文学习的提示定义为任务描述和示例的组合，这两部分均为可选。按照这个定义，如果大语言模型仅通过任务描述（即任务指令）来解决未见过的任务，也可以被看作是上下文学习的一种特例。**两者的主要区别是，指令微调需要对大语言模型进行微调，而上下文学习仅通过提示的方式来调用大语言模型解决任务。** 此外，指令微调还可以有效提升大语言模型在执行目标任务时的上下文学习能力，尤其是在零样本场景下（即仅依赖任务描述而无需额外的示例）[41]。

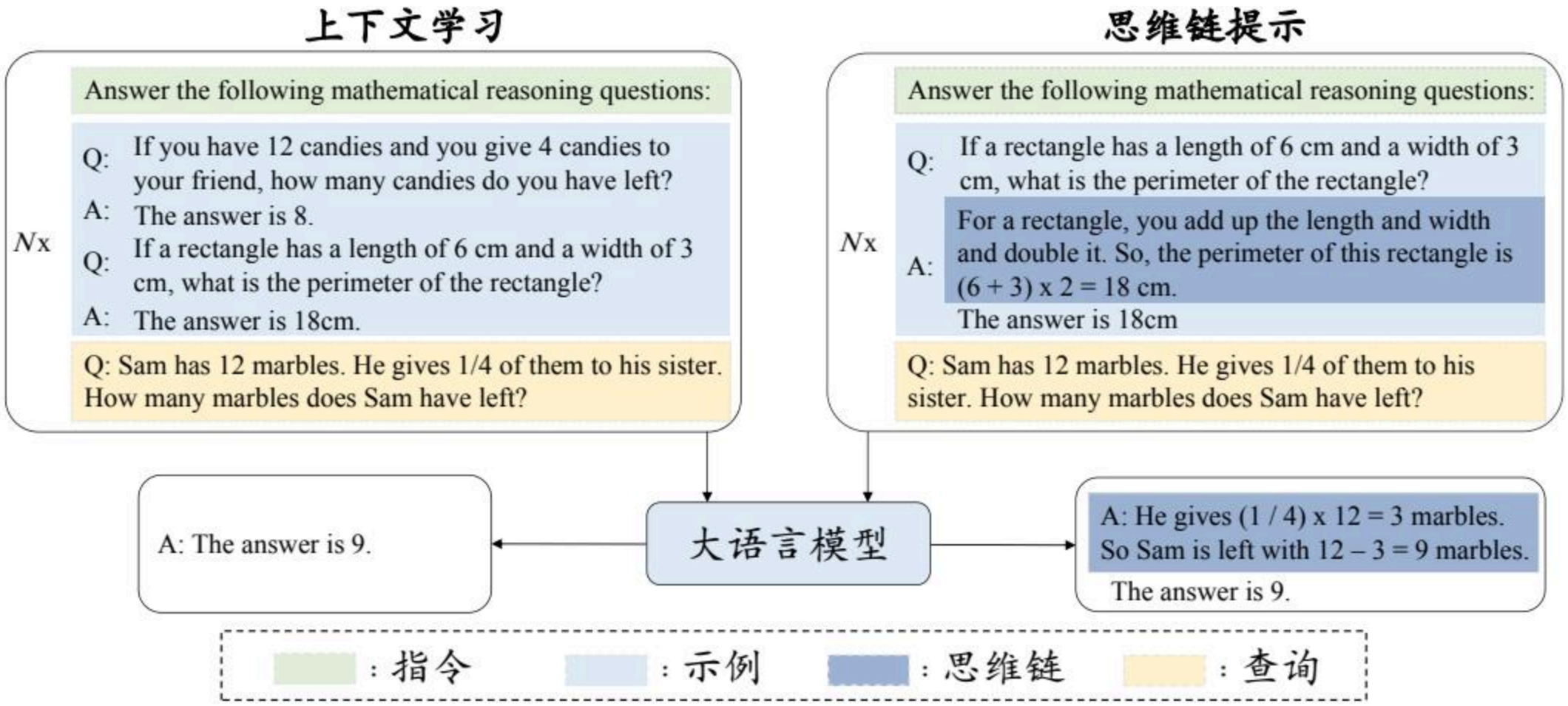


图10.1 一个关于上下文学习和思维链提示的比较说明（图片来源：[10]）

10.2.2 示例设计

作为一个重要特点，上下文学习在提示中引入了示例数据，因此示例的选择和设计对于模型上下文学习的能力具有重要的影响。根据公式10.1，我们将重点关注示例设计的三个关键因素，包括示例样本选择、样本格式化函数 $f(\cdot)$ ，以及示例排序策略。下面针对这三个方面展开详细讨论。

示例选择

在上下文学习中，示例选择是一个关键步骤，其目的是为了从含有大量样本的集合中选取最有价值的示例，进而能够有效激发大语言模型在任务上的模型效果。下面介绍几种常见的示例选择方法，包括基于相关度排序的方法、基于集合多样性的方法和基于大语言模型的方法。

- 基于相关度排序的方法. 在实际应用中, 基于相关度排序的方法得到了广泛的应用, 典型实现就是基于 k 近邻（ k -Nearest Neighbors, k -NN）的相似度检索算法。该方法实现简单, 而且能够有效地选择出与目标任务实例相关的示例。具体来说, 可以使用文本嵌入模型（如BERT）将所有候选样本映射到低维嵌入空间中, 然后根据这些候选样本与测试样本的嵌入语义相似度进行排序, 并选择出最相关的 k 个示例, 最后将筛选出的示例作为上下文学习的示例集合。在实践中, 这种方法通常明显优于随机选择示例方法。

- 基于集合多样性的方法. 尽管 k 近邻检索算法简单易行, 但是它通常独立地评估每个示例的相关性, 而忽略了示例集合的整体效果。为了弥补这一不足, 我们可以采取基于集合多样性的示例选择策略。这种策略旨在针对特定任务选择出具有代表性的、信息覆盖性好的示例集合, 从而确保所选示例能够反应尽可能多的任务信息, 从而为大语言模型的推理提供更丰富、更全面的信息。在选择过程中, 除了考虑样本与目标任务的相关性, 同时也要考虑与已选样本的相似性, 需要综合考虑相关性与新颖性的平衡。在实现中, 可以采用经典启发式的 MMR 算法 (Maximum Margin Ranking) 以及基于行列式点过程的 DPP 算法 (Determinantal Point Process), 从而加强示例集合的多样性。

- 基于大语言模型的方法。除了上述两种方法外，另一个方法是将大语言模型作为评分器对候选样本进行评估，进而选择出优质的示例。一种最直接的评估方法是通过计算在加入当前示例后大语言模型性能的增益来评估示例的有效性，以此筛选出有效的示例。但是，这种方法需要大语言模型进行重复多次计算，才能选择出最优的示例集合。为了减少大语言模型评估的开销，还可以根据大语言模型的评分结果选择出少量的正负示例用于训练一个分类器，该分类器通过正负示例能够学习到如何准确地区分和筛选出高质量示例，从而更准确地来指导后续的示例选择过程。

总体来说，在上下文学习中进行示例选择时应确保所选示例包含丰富的任务信息且与测试样本保持高度的相关性[268]。

示例格式

示例模版（即公式10.1中的示例格式化函数 $f(\cdot)$ ）对大语言模型的性能有着非常重要的影响。通常来说，可以使用两种主流的方法进行示例格式的构建，包括人工标注的示例格式与自动生成的示例格式。

- 人工标注的格式. 人工标注是构建高质量任务示例格式的一种常用方式。在实现中, 首先需要定义好输入与输出的格式, 然后添加详细的任务描述, 以帮助大语言模型更好地理解当前示例所要表达的任务需求。例 10.5 展示了人工标注的示例格式的例子。最简单的示例格式只需要显式标识出输入与输出, 让大语言模型自动学习到输入与输出之间的语义映射关系。进一步, 在提示内部加入相关的任务描述有助于模型更精准地理解任务的要求, 从而生成更准确的答案。最后, 为了更好地激发大语言模型的推理能力, 可以在输出中加入思维链, 展示模型的逐步推理过程。人工标注的示例格式的优势在于格式清晰且易于理解，但在处理大量任务时，可能面临多样性不足的问题。

> (1) 包含输入与输出的示例格式:
>
> ……
>
> (2）增加任务描述的示例格式：
>
> ……
>
> (3）增加思维链的示例格式：
>
> ……

例 10.5 人工标注的示例格式

- 自动生成的格式. 为了缓解人工标注的局限性, 还可以采用大语言模型自动生成示例格式。**这种方法的核心在于借助大语言模型的上下文学习能力, 进而大规模扩充新任务的示例模版。** 具体来说, 首先人工标注一部分的示例模板作为种子集合加入到大语言模型的输入中。然后, 利用大语言模型强大的少样本学习能力, 指导其为新任务生成相应的示例模版。最后, 对这些生成的示例模版进行筛选与后处理, 使之符合任务要求。例 10.6 展示了大语言模型根据少量人工标注的模板来自动生成示例格式的例子。通过提供示例的输入、输出与指令，大语言模型能够根据新任务的输入和输出为其生成对应的指令。

> 请根据输入输出自动撰写一段指令：
>
> ……

例 10.6 自动生成的示例格式（灰色文字表示模型生成指令）

示例顺序

在上下文学习中，大语言模型往往会受到位置偏置的影响，表现为对示例顺序具有一定的敏感性。因此，设计合理的示例顺序也是上下文学习中需要考虑的一个问题，旨在为所选择的示例找到最有效的排序方式以提升模型性能。确定大语言模型的最优的示例顺序通常分为两个步骤：产生示例的候选顺序和评估示例顺序的有效性。

- 产生候选示例顺序。在第一个步骤中，我们需要为所选择的示例产生候选排序。一种最直接的方法是枚举给定示例的所有可能排列组合，然后从中随机选取一种排列作为示例的顺序。然而，这种随机选择方法所产生的结果具有较大的方差，可能导致模型性能的不稳定。**鉴于大语言模型在做出预测时，倾向于依赖于提示末端的信息。另一种更常用的方式是根据示例与测试样本之间的语义相似度进行排序，然后将与测试样例相似度更高的示例放在更靠近测试样本的位置**。这种方法可以加强大语言模型在推理过程中对于语义相关的示例进行利用，从而提升模型性能。

- 评估示例顺序质量。在示例顺序确定之后，下一步是评估这一顺序的质量。在测试集样本可获得的情况下，可以直接测试大语言模型基于该示例顺序的任务性能，以此作为当前示例顺序的评分。然而，在许多情况下，我们可能无法获得测试样本，因此需要人工创建独立的验证集进行示例顺序的评估。另一种不依赖测试数据的评估方法是采用模型对于预测结果的不确定性作为评估指标。具体来说，可以计算基于该示例顺序大语言模型预测分布的熵值，选择熵值较低的示例顺序作为较为有效的顺序。熵值越低，意味着模型预测分布越不均匀，则模型预测的置信度更高。

10.2.3 底层机制

上下文学习能力是一种具有代表性的大语言模型能力。这种通过示例进行学习的范式虽然在传统机器学习模型中也有涉及（例如 k 近邻分类器），但是整体的应用范围与任务场景非常局限。特别是，大语言模型完全通过提示设计来进行上下文示例的学习，这其中的内在原理与工作机制值得深入思考。在本节中，我们将深入探讨与大语言模型上下文学习能力紧密相关的两个核心问题：一是预训练阶段如何影响上下文学习能力，二是生成阶段大语言模型如何支持上下文学习。

预训练阶段对上下文学习能力的影响

预训练阶段主要有两个关键因素对大语言模型上下文学习能力产生影响：预训练任务和预训练数据。这两方面分别关注如何设计训练任务和如何选择训练数据来提升模型的上下文学习能力。

- 预训练任务. 上下文学习的概念最初在 GPT-3 的论文 [23] 中被正式提出, 论文通过相关实验发现上下文学习能力随着模型规模的增大而增强。随着预训练技术的改进, 后续研究发现, 即使是小规模的模型, 通过设计专门的训练任务 (如根据示例和输入预测标签), 进行继续预训练 [269] 或微调 [270], 也能够获得上下文学习能力，甚至在某些情况下可能超越规模更大的模型。这表明预训练任务的设计对于上下文学习能力的习得具有重要的影响。具体来说，MetaICL [270] 认为，通过元训练任务可以让模型自动学习到如何通过输入中的少量示例重构任务信息，进而更有效地完成上下文学习。因此，MetaICL 使用了数十个不同领域的 NLP 任务作为元训练任务。对于每一个元训练任务，抽取出若干样本作为示例（其余样本则用于训练），大语言模型采用上下文学习的方式进行训练，即根据示例和待预测样本的输入预测对应的输出。这种训练方式使得无需在输入中提供任务描述，只需提供少量示例，大语言模型即可学会解决对应的目标任务。实验结果显示，元训练任务越多且越多样，模型的上下文学习能力就越强，这也证明了预训练任务对于大语言模型上下文学习能力具有很大的影响。

- 预训练数据. 除了训练任务外, 预训练数据的选择对上下文学习能力也有显著影响, 但并非所有的预训练数据都对上下文学习能力的提升同等重要。研究发现, **通过混合不同领域的预训练数据, 增强预训练语料的多样性可以提高大语言模型的上下文学习能力** [271]。**此外, 预训练数据的长程依赖关系也是改善模型上下文学习能力的重要因素**。通过将前后相关的文本直接拼接进行训练, 模型能够更好地理解文本之间的关联性, 从而提升上下文学习的能力 [272]。进一步, 为了筛选出对模型上下文学习能力有重要影响的训练数据, 研究人员通过计算预训练数据和上下文学习的测试数据的梯度之间的相似度, 可以得到具有较高相似度的训练数据子集 [273]。实验发现, 这些具有高相似梯度的预训练数据中包含了更高密度的低频长尾词汇, 模型对这部分数据的学习难度较高, 因此可能有助于提升模型上下文学习的能力。

推理阶段大语言模型执行上下文学习的方式

在推理阶段，由于上下文学习不涉及显式的学习过程或参数更新，因此可以主要关注在给定示例的情况下，大语言模型如何执行上下文学习。大语言模型使用示例数据的方式主要分为两种范式，包括任务识别和任务学习[274]。在任务识别范式中，大语言模型通过分析示例来理解并识别需要执行的任务；而在任务学习范式中，大语言模型则尝试从示例中提取正确的信息来完成任务。下面对于这两种方式进行具体介绍。

- 任务识别. **大语言模型具备从所提供示例中辨识当前任务的能力，并能利用其在预训练阶段所积累的丰富先验知识来解决这些任务，这一范式不受示例的输入和输出映射的影响。** 基于概率近似正确（Probably Approximately Correct, PAC）的理论框架认为预训练数据中存在能够表征各种任务信息的隐变量。因此在上下文学习中，大语言模型具备从给定示例中学习并编码这些隐变量的能力，因此能够通过上下文示例实现任务的自动识别和适应[275]。随后，大语言模型根据这个任务隐向量的指导，在接收到新的输入时自动触发相应的任务识别过程，并生成符合任务要求的输出[276, 277]。

- 任务学习。第二种观点认为大语言模型还具备通过示例数据学习预训练阶段未涉及的新任务的能力。这种观点主要从梯度下降的角度来分析上下文学习的机理，并将其视为一种隐式的微调过程[278,279]。具体来说，从隐式梯度下降的角度分析，上下文学习机制可以被分解为以下两个步骤。首先，大语言模型通过前向计算过程，针对给定示例生成相应的元梯度（类似于梯度下降时的梯度，但没有显式计算的过程）。然后，模型利用注意力机制隐式地执行了梯度下降。这一过程类似于传统机器学习中的参数更新，但不同之处在于它是在模型的前向传播过程中隐式完成的，无需显式的参数更新。除了从梯度下降的角度进行解释，上下文学习还可以被抽象为模型内部的一种更复杂的算法学习过程[280]。具体来说，在预训练阶段，大语言模型通过其参数编码了一个隐式模型。因此，在上下文学习的前向计算阶段，借助上下文学习中的示例引导，大语言模型能够通过诸如决策树等更复杂的学习算法来更新其内部的隐式模型。

在上下文学习中，现有的文献通常认为大语言模型能够同时展现出任务识别和任务学习两种能力，但是这两种能力的强弱与模型的规模紧密相关。其中，**规模较小（如350M参数）的模型已经能展现出较强的任务识别能力，能够简单地识别任务的类型和要求**[274]；**而任务学习能力要求模型从示例中学习全新的任务解决方案，通常较大规模（如66B参数）的模型能展现出更强的任务学习能力**[274]。一项研究通过将上下文示例的真实标签替换为“随机标签”或者“随机符号”，分别对模型的任务识别和任务学习能力进行探究[281]。例如，在情感分析任务中，从标签空间中均匀随机采样示例的标签（“positive/negative”），模型只需要进行任务识别；而使用没有明确语义含义的符号如“foo/bar”替换真实标签“positive/negative”，迫使模型不依赖其先验知识，而是必须从提供的示例中学习新的标签映射以解决当前任务。实验结果显示，当使用颠倒或语义不相关的符号作为标签时，规模较大的模型的性能下降更小。这意味着大模型能够更好地分析和学习给定的示例信息，从示例中学习出标签信息的对应关系，进而采用学习到的策略完成任务。

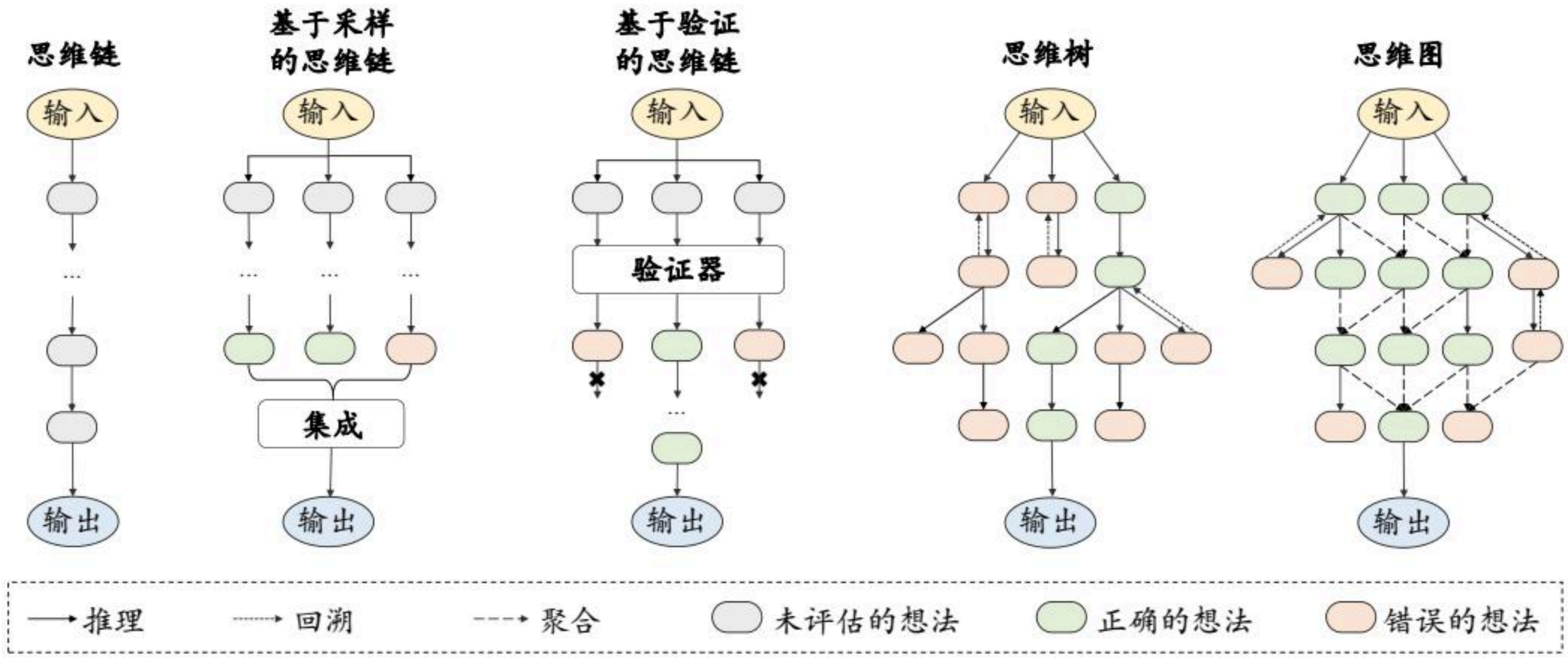


图10.2 思维链提示技术的演化过程（图片来源：[10]）

10.3 思维链提示

思维链提示[25,282]是一种高级提示策略，旨在增强大语言模型在各类复杂推理任务上的表现。常见的推理任务包括算术推理[283]、常识推理[284]以及符号推理[25]等多种任务。**与上下文学习方法仅使用〈输入，输出〉二元组来构造提示不同，思维链提示进一步融合了中间的推理步骤来指导从输入到输出的推理过程。** 图10.1展示了一个思维链提示的具体例子。在本节中，我们将介绍基础的思维链提示方法以及相关的增强策略，还将探讨思维链的能力来源以及思维链提示对模型推理的影响。

10.3.1 思维链提示的基本形式

思维链提示作为上下文学习的一种扩展形式，将原始的〈输入，输出〉映射关系转换为〈输入，思维链，输出〉这一三元组形式。在这个结构中，思维链扮演着重要的角色，它提供了一系列语义连贯且具有逻辑性的中间推理步骤，有效地建立起输入与输出之间的桥接关系。在思维链提示的作用下，大语言模型可以根据输入生成对应的思维链及答案。然而，与简单的〈输入，输出〉二元组相比，思维链的获取通常更为复杂，往往需要借助人工标注的方式。目前有一些简单的方法可以让大模型在回答问题之前生成思考过程。例如，**通过向大语言模型提供诸如**“Let's think step by step.”[285]**或**“Take a deep breath and work on this problem step-by-step.”[267]**这样的诱导性指令，能够在不提供思维链示例的情况下，仍然让大语言模型先生成思维链再回答问题来提高准确率**。图10.1右侧的例子展示了大语言模型在思维链提示的作用下，一步步生成中间推理步骤，最终得到了正确的答案。

10.3.2 思维链提示的优化策略

尽管大语言模型在基本的思维链提示策略下已经在推理任务中展现出一定的性能提升，但仍然存在推理过程错误、生成答案不稳定等问题。针对这些问题，本节将从下列三个方面探讨如何对基础的思维链提示方法进行改进：针对输入端对大模型的思维链示例进行增强（增强的思维链示例设计）、针对大模型的思维链生成过程进行改进（高级的思维链生成方法）以及针对整个思维链结构进行优化（拓展的推理结构）。图10.2展示了代表性的思维链提示策略的演变历程。

思维链示例设计

目前大语言模型在使用思维链提示进行推理时，大多采用了上下文学习的设定，即思维链提示通过示例的形式输入给大语言模型。因此，接下来将介绍两种常用的在上下文学习场景下的思维链示例设计方法。

- 复杂化的思维链。基于复杂度指标设计思维链示例是一种简单有效的策略。思维链复杂化的体现主要在于推理步骤的增多。由于每一个推理步骤都可以看作是一个子问题的解答，因此更多的推理步骤也包含了对于更多子问题的解答，推理过程也更加缜密。当使用较多推理步骤的示例作为提示输入给模型时，模型更容易学习到多种子问题的解决方案以及对应的逻辑推理过程，能够提升模型在复杂推理任务上的表现。除了将推理步骤的数目作为复杂度指标，还可以使用问题长度对问题的复杂度进行量化。问题越长说明其包含更多的输入信息，则问题求解可能需要更多的推理步骤。对于某些没有人工标注思维链的数据集，可以选择最长的若干问题，然后对这些问题的思维链进行人工标注作为思维链示例。基于这些复杂思维链示例，模型通常可以获得相较于随机选择思维链示例更好的性能。

- 多样化的思维链. 除了设计更为复杂的思维链示例, 在提示中包含多样化的思维链示例能够有效改善模型的推理能力, 主要是因为多样化的思维链示例可以为模型提供多种思考方式以及推理结构。为了选择出多样化的思维链, 可以首先利用聚类算法 (例如 k -means 聚类) 将训练集中的问题划分为 k 个簇 (k 为所需的示例数量), 簇内部的问题比较相似, 而不同簇的问题差别较大。然后, 预定义一系列启发式规则, 从每个簇中选择距离质心最近且满足规则的问题作为该簇的代表性问题，将该问题输入给大语言模型并生成对应的思维链和答案作为示例。由于每个问题来自于不同的簇，从而保证了示例的多样性。实验发现，虽然大模型生成的思维链示例可能存在错误，但是当选择更加多样化的示例时，思维链示例中的错误对模型性能的影响会显著降低[286]。

思维链生成方法

在上述内容中，我们介绍了如何在模型的输入侧对思维链示例进行增强。另一方面，模型在生成思维链时容易出现推理错误和生成结果不稳定等情况，还需要对大语言模型生成思维链的过程进行改进。本部分将重点介绍两种改进思维链生成过程的方法：基于采样的方法与基于验证的方法。

- 基于采样的方法

大语言模型在使用单一的思维链进行推理时，一旦中间推理步骤出错，容易导致最终生成错误的答案。为了缓解这一问题，可以通过采样多条推理路径来缓解单一推理路径的不稳定问题。作为一种代表性方法，Self-consistency [287] **首先使用大语言模型生成多个推理路径和对应的答案（如图 10.2 所示），然后对于这些候选答案进行集成并获得最终输出。具体的集成方法可以选择各条推理路径所得到答案中出现频率最高的那个答案作为最终输出，在某些情况下也可以对所有答案进行某种形式的加权。**

我们还可以对上述过程做进一步的扩展：假设大语言模型在一个思维链提示下生成了 M_1 条推理路径，那么可以使用 M_2 个思维链提示依次输入给大语言模型，这样一共就能得到 $M_1 \times M_2$ 条推理路径，从中投票选出最终的答案，进一步增加答案的可靠性。基于采样的思维链生成方法不仅简单易行，而且相较于单一思维链方法在多个任务中展现出了更为优异的性能。然而，在一些特定的任务场景中，当仅使用单一推理路径时，模型使用思维链提示的效果可能不如基础提示的效果。例如，对于句子的情感分类任务，由于问题过于简单，加入思维链提示之后反而会使模型过度思考，从而得出错误的答案。

- 基于验证的方法

思维链提示所具有的顺序推理本质可能导致推理过程中出现错误传递或累积的现象。为了解决这一问题，可以使用专门训练的验证器或大语言模型自身来验证所生成的推理步骤的准确性。下面以 DIVERSE 方法 [288] 为例进行具体介绍。DIVERSE 分别训练了针对整个推理路径和中间推理步骤的验证器，从不同的粒度实现更为全面的检查。针对整个推理路径的验证器通过如下方法训练得到：首先选择一个包含大量问题答案对的数据集，然后将问题输入给大语言模型，通过思维链提示的方法使其生成推理路径和最终答案。如果模型生成的答案和数据集标注的答案一致，则判为正例，否则判为负例。最后使用构造的（问题，推理链，答案）数据训练一个二分类器，从而可以对任意一个推理路径进行打分。

训练针对中间步骤的验证器也可以采用类似的方案。然而，与整体推理路径的数据标注相比，构造面向中间步骤的正负例数据更加困难。这里可以采取一个简化处理：对于每一个训练集中的问题，我们采样多次得到多个推理路径，对于得出正确答案的推理路径，中间的每一个步骤我们都认为是正确的，作为正例；对于得出错误答案的推理路径，如果其中某个步骤和正例的推理路径相一致，也认为是正例，否则作为负例。通过这样构造出来的数据，用同样的方法训练一个二分类器，从而可以对模型输出的中间步骤进行打分。

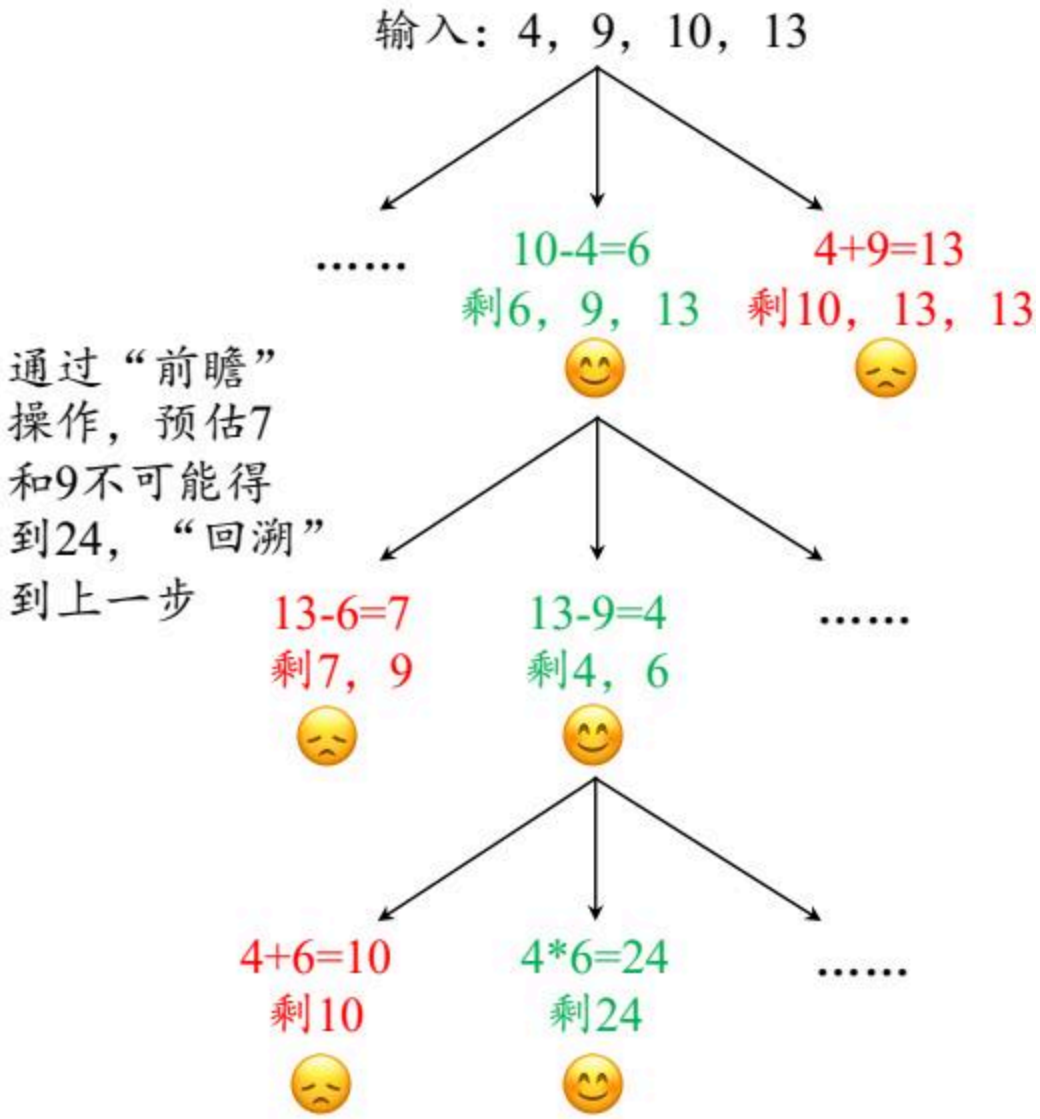


图10.3 大语言模型使用思维树方法解决24点游戏

任务：给定四个数4、9、10、13，如何通过加减乘除四则运算得到24?

- ① 生成多个可能的初始步骤，例如 $10-4=6$ $4+9=13$ ……
- ② 每一个步骤都会生成多个下一步骤, 例如 $10-4=6$ 之后可以生成 $13-6=7$ 或 $13-9=4$ 。
- ③ 对中间步骤进行打分。这里可以对当前思考步骤进行“前瞻”，例如在当前思考步骤剩下7和9时，能前瞻性地得知无法得到24，应该得到一个低分。
- ④ 如果当前节点不太可能得到最终结果,那么“回溯”到上一节点, 选择其他路径。例如从 $13-6=7$ 的节点回溯到父节点, 然后前进走到 $13-9=4$ 节点。

图10.3 大语言模型使用思维树方法解决24点游戏

拓展的推理结构

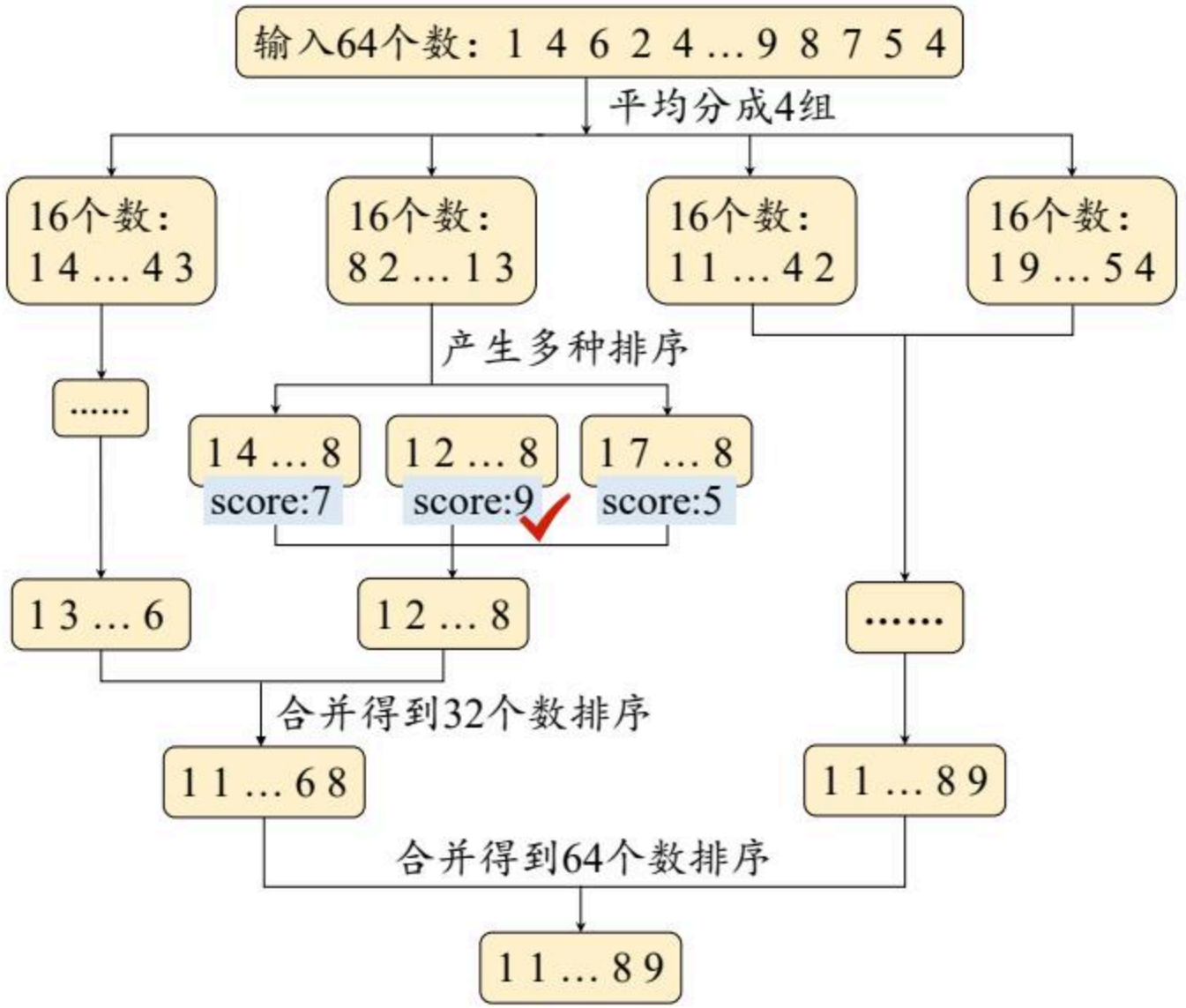
尽管基本的思维链提示具有广泛的适用性，但是所采用的链式推理结构在处理较为复杂的任务时（例如需要进行前瞻和回溯探索）仍然存在一定的局限性。为了突破链式推理结构的限制，可以将思维链的结构进一步拓展，从而获得更强的推理性能。

- 树形结构的推理。考虑到许多推理任务需要模型前瞻和探索可能的解决方案，可以将推理过程刻画为一个层次化的树形结构，进而问题的求解就转化为在树上的搜索问题。这一方法的代表性工作是思维树（Tree of Thought, ToT）[289, 290]。思维树的每个节点对应一个思考步骤，父节点与子节点之间的连边表示从一个步骤进行下一个步骤。**它和思维链的区别在于：思维链从一个节点出发，只能生成一个节点，而思维树则可以生成多个节点。当某一个思考步骤无法得到正确答案时，可以回溯到它的父节点，选择另一个子节点继续推理。图 10.3 以 24 点游戏为例介绍如何使用思维树解决问题[2]。其中 ① 和 ② 与思维链是一样的，最关键的步骤是 ③ 和 ④ 。对于思维链来说，只有走到最后一步才能判断当前的推理路径是否正确，如果出现错误只能从头开始推理，这极大降低了推理效率。但是 ③ 通过前瞻性判断，能够预估当前节点得到最终答案的可能性并给出一个评分。这样，在 ④ 的搜索算法中，我们可以提前放弃一些不太可能得到最终答案的路径（对应评分较低的节点）**，而优先选择那些评分更高的推理路径进行下一步的推理。图 10.2 将思维树与其他推理结构进行了对比。

- 图形结构的推理. 相较于树形结构, 图形结构能够支持更为复杂的拓扑结构,从而刻画更加错综复杂的推理关系, 可以展现出更强的推理性能。这一方法的代表性工作是思维图 (Graph of Thought, GoT) [291, 292]。思维图将整个推理过程抽象为图结构, 其中的节点表示大语言模型的中间步骤, 节点之间的连边表示这些步骤之间的依赖关系。由于树形结构中只有父节点和子节点之间有连边, 因此无法构建不同子节点之间的联系。**思维图则允许图上的任意节点相连, 因此可以在生成新的中间步骤的同时考虑其他推理路径。图 10.4 以含有重复数字的 0~9 数组排序为例介绍大语言模型如何使用思维图解决排序问题。在这个场景下, 大语言模型难以对长数组进行准确地排序, 但是短数组排序对于模型来说更为简单。因此, 思维图方法首先将输入数组分成 4 组, 分别进行排序, 然后再对结果进行合并。思维图和思维树的区别在于, 思维树的子节点只能进行前向搜索和回溯, 而思维图的子节点可以和其他子节点进行汇聚, 得到新的中间步骤, 然后进行下一步的推理。** 图 10.2 将思维图与其他推理结构进行了对比。

10.3.3 关于思维链的进一步讨论

作为一种重要的“涌现能力”，思维链提示能够显著提升大语言模型的推理能力。但是在一些简单任务上，思维链提示有时甚至会带来效果上的下降。因此，一个值得探讨的问题是：为什么思维链提示能显著提升大语言模型在推理任务上的效果？下面我们从训练阶段思维链推理能力的来源、测试阶段思维链对模型的影响两个角度进行讨论。



任务：对64个1～9之间的数字进行排序

① 平均分成4组，每组16个数。

② 对每一组而言，让大语言模型排序多次得到多个可能的结果。

③ 对中间步骤进行打分，保留得分最高的方案。

④ 对中间步骤两两合并，得到最终结果。

图10.4 大模型使用思维图方法解决数组排序

- 思维链推理能力的来源。

对于思维链工作机制的研究需要探究推理的本质。斯坦福大学的研究人员 [293] 假设思维链对大语言模型有效的原因是训练数据中存在很多相互重叠且互相影响的局部变量空间（例如主题、概念和关键词等）。在这个条件下，即使两个变量没有在训练数据中共现，也可以通过一系列重叠的中间变量的推理而联系起来。为了验证这一假设，研究人员构建了一个具有链式结构的贝叶斯网络，并用这个网络合成了一批训练样本，这些样本包含许多相互影响的局部变量空间。然后，使用这批数据来训练一个语言模型，根据给定一个变量来预测另一个变量的条件概率。实验结果发现，如果这两个变量不经常在数据中共现时，模型直接预测这个条件概率总会与真实概率有一定偏差，但是当使用中间变量进行推理预测时，可以获得比直接预测更小的偏差；而当这两个变量经常在数据中共现时，通过中间变量推理和直接预测两种方式带来的偏差会比较接近。

还有研究工作从函数学习的角度出发 [294]，认为复杂推理任务可以看作是一种组合函数，因此思维链推理实际上是将组合函数的学习过程分解为了两个不同阶段：信息聚焦和上下文学习单步的组合函数。在第一阶段，语言模型将隐式地聚焦到思维链提示中与推理的中间步骤相关的信息。在第二阶段，基于聚焦得到的提示，语言模型通过上下文学习输出一个推理的步骤（即单步组合函数的解），并走向下一步，从而得到最终答案（即整个组合函数的最终解）。通过理论证明与实验验证，研究人员发现信息聚焦阶段显著减少了上下文学习的复杂度，只需要关注提示中与推理相关的重要信息，而上下文学习阶段则促进了复杂组合函数的学习过程，而标准提示则很难让模型学习到这种复杂函数。

- 思维链提示对模型推理的影响。

为了研究思维链提示对模型推理能力的影响，主要通过对思维链提示进行扰动，然后观察模型行为上的变化来得出相应的结论。谷歌的研究人员 [295] 将思维链分成两部分：符号（例如数学题中的数字、常识问答中的实体）和模式（例如数学题中的算式、常识问答中的句子结构和模板），分别研究它们对模型推理能力的影响。实验结果发现，不管是符号还是模式，其作用都主要体现为表达任务意图，而具体的内容并不重要，重要的是它们与问题的相关性以及推理过程的逻辑性 [296]。

进一步，在某些数学推理任务上（少样本学习），思维链示例中算式的正确与否甚至不会显著影响模型的性能，这些符号与模式的作用更多是体现为对于任务目标的表达。研究人员认为思维链可以看作一种增强的上下文学习：因为任务过于复杂，基础提示已经无法准确表达任务意图，因此需要思维链提示来增强对于任务意图的表达。也有研究发现[297]，即使不对语言模型使用思维链提示，只要其生成的文本中包含显式的推理过程，也能显著改善模型的推理能力。具体来说，当语言模型生成第一个词元时，采样出前 k 个概率最大的词元，然后继续解码，生成 k 条可能的文本序列。在这些生成的文本序列中，某些会包含推理路径，而这些具有推理路径的文本序列产生的正确答案概率显著高于其他路径。这表明，思维链提示通过激发模型生成中间推理步骤来提高其生成正确答案的概率。

第十二章 评测

随着大模型技术研究的快速发展，学术界和工业界相继发布了众多大语言模型（详见第3.1节）。这些模型有的展现出强大的通用能力，有的则是针对特定专业领域优化过的模型。在此背景下，如何准确地评估大语言模型在不同维度的能力水平，已经成为当前研究的热点问题。为了全面考察大语言模型的有效性，研究人员设计了多种评测任务并创建了对应的数据集，用于对模型性能进行评估与分析。这些评估实验不仅有助于更深入地了解现有大语言模型的能力与局限性，也为未来大语言模型的研究与发展方向提供了重要的指导依据。本章将首先介绍大语言模型评估中常用的评测指标与方法（第12.1节）。在此基础上，我们将引入针对大语言模型基础能力的评测任务（第12.2节）。随后，本章将探讨若干更具挑战性、目标更为复杂的评测任务（第12.3节）。最后，我们将介绍若干常用的公开评测体系（第12.4节），以便读者在实际应用中能够方便有效地评估大语言模型的性能。

12.1.1 常见评测指标

在评估大语言模型的能力时，研究者需要从多个维度全面考量其性能表现。这不仅涉及到利用多样化的任务来测试模型的各种能力，还需要针对性地选择合适的评测指标，以确保准确衡量模型的性能。根据应用场景的不同，接下来将分别介绍语言建模、文本分类、条件文本生成、执行类任务以及偏好排序类任务中常用的评测指标。表12.1详细列出了各个类别中典型的大语言模型评测任务，以及在这些任务上常用的评测指标及它们的具体定义。在随后的章节中，我们将介绍各个任务类别中的核心评测指标。

模型能力测评的代表性任务

级别	能力	任务	任务描述
基础	语言生成	语言建模	基于给定的背景词元来预测接下来会出现的词元的任务，是语言模型最基础的能力。
		条件文本生成	依据给定的条件生成满足特定需求的文本输出，包括机器翻译、文本摘要和对话系统等多种任务。
		代码合成	大语言模型还展现出了较强的结构化语言生成能力，特别是在生成符合特定需求的计算机代码。
	知识运用	闭卷问答	模型需要基于自身掌握的知识来回答问题，不借助外部资源提供的背景信息。
		开卷问答	允许大语言模型基于从外部知识库或文档集合中检索和提取的相关文本生成答案。
		知识补全	大语言模型需要根据自身编码的语义信息，补全或预测缺失的知识单元。
		知识推理	考察模型通过逻辑关系与事实知识解决复杂任务的能力。
	复杂推理	符号推理	给定形式化的规则，让模型来操作预定义好的符号以实现某些特定目标。
		数学推理	综合利用数学知识、逻辑推演与计算能力，以解决给定数学问题或构建证明过程。
高级	人类对齐	有用性	评价模型根据人类需求完成特定任务的能力。
		诚实性	从事实性、前后一致性等维度进行评测。幻象评测是其中一种有代表性的评测。
		无害性	检测大语言模型所生成的文本中是否存在偏见、歧视等有害因素。

评测任务	评测指标	介绍
分类任务	精确率	计算模型预测为正例的样本中真正为正例的比例
	召回率	计算真正例的样本中被模型正确预测的比例
	F1分数	综合衡量模型输出的精确率和召回率
语言建模任务	困惑度	衡量模型对参考文本的建模概率
文本生成任务	BLEU	衡量机器翻译与参考翻译之间的重叠度
	ROUGE	衡量机器摘要对参考摘要的覆盖度
问答任务	准确率	衡量模型预测的正确答案的比例
执行类任务	成功率	衡量模型成功完成任务的比例
	Pass@k	估计模型生成的k个方案中至少能通过一次的概率
偏好排序类任务	Elo等级分	衡量模型在候选者中的相对水平

表 12.1 常见评测指标分类表