

第一章 引言

人类主要使用语言进行表达与交流。语言能力通常在人类幼儿时代就已初步形成，并且在人的一生中不断发展与完善 [1, 2]。为了使计算机能够与人类进行有效交流，科研人员一直致力于研发具有类人语言能力的人工智能（Artificial Intelligence, AI）算法，使之能够掌握以自然语言形式进行沟通与交流。让机器拥有像人类一样阅读、理解、写作和交流的能力是一个长期的研究挑战 [3]。

从技术路径上来说，语言模型（Language Model, LM）是提升机器语言智能（Language Intelligence）的主要技术途径之一，全书将聚焦这一主题展开讨论。本章将主要回顾语言模型的发展历程，并且介绍大语言模型与传统语言模型的不同之处及其对于科研发展所带来的机遇与挑战。

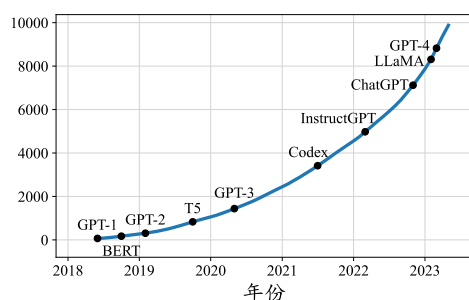
1.1 语言模型的发展历程

一般来说，语言模型旨在对于人类语言的内在规律进行建模，从而准确预测词序列中未来（或缺失）词或词元（Token）的概率。根据所采用技术方法的不同，针对语言模型的研究工作可以分为以下四个主要发展阶段：

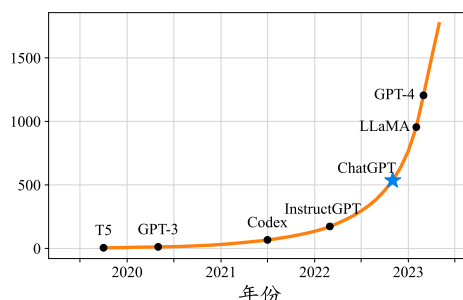
- 统计语言模型（Statistical Language Model, SLM）。在 20 世纪 90 年代兴起的统计语言模型 [4, 5] 是基于统计学习方法研发的。具体来说，统计语言模型使用马尔可夫假设（Markov Assumption）来建立语言序列的预测模型，通常是根据词序列中若干个连续的上下文单词来预测下一个词的出现概率，即根据一个固定长度的前缀来预测目标单词。具有固定上下文长度 n 的统计语言模型通常被称为 n 元（ n -gram）语言模型，如二元或三元语言模型。统计语言模型被广泛应用于信息检索（Information Retrieval, IR）和自然语言处理（Natural Language Processing, NLP）等领域的早期研究工作。对于高阶统计语言模型来说，随着阶数 n 的增加，需要估计的转移概率项数将会指数级增长，经常会受到“维数灾难”（Curse of Dimensionality）的困扰。为了缓解数据稀疏问题，需要设计专门的语言模型平滑策略，如回退估计（Back-off Estimation）和古德-图灵估计（Good-Turing Estimation）。然而平滑方法对于高阶上下文的刻画能力仍然较弱，无法精确建模复杂的高阶语义关系。

- 神经语言模型（Neural Language Model, NLM）。神经语言模型 [6, 7] 使用神经网络来建模文本序列的生成，如循环神经网络（Recurrent Neural Networks, RNN）。图

灵奖获得者 Yoshua Bengio 在一项早期工作中 [6] 引入了分布式词表示 (Distributed Word Representation) 这一概念, 并构建了基于聚合上下文特征 (即分布式词向量) 的目标词预测函数。分布式词表示使用低维稠密向量来表示词汇的语义, 这与基于词典空间的稀疏词向量表示 (One-Hot Representation) 有着本质的不同, 能够刻画更为丰富的隐含语义特征。同时, 稠密向量的非零表征对于复杂语言模型的搭建非常友好, 能够有效克服统计语言模型中的数据稀疏问题。分布式词向量又称为“词嵌入” (Word Embedding)。这种基于隐含语义特征表示的语言建模方法为自然语言处理任务提供了一种较为通用的解决途径。在这一系列工作中, word2vec [8, 9] 是一个具有代表性的词嵌入学习模型, 它构建了一个简化的浅层神经网络来学习分布式词表示, 所学习到的词嵌入可以用作后续任务的语义特征提取器, 在自然语言处理任务中得到了广泛使用, 取得了显著的性能提升。这些创新性的研究工作将语言模型用于文本表示学习 (超越了原始的词序列建模目标), 在自然语言处理领域产生了重要影响。



(a) 查询 = “Language model”



(b) 查询 = “Large language model”

图 1.1 标题中包含查询短语 “*Language Model*” (从 2018 年 6 月起) 和 “*Large Language Model*” (从 2019 年 10 月起) 的 arXiv 论文累计数量的变化趋势 (图片来源: [10])

● 预训练语言模型 (Pre-trained Language Model, PLM). 与早期的词嵌入模型相比, 预训练语言模型在训练架构与训练数据两个方面进行了改进与创新。ELMo [11] 是一个早期的代表性预训练语言模型, 提出使用大量的无标注数据训练双向 LSTM (Bidirectional LSTM, biLSTM) 网络, 预训练完成后所得到的 biLSTM 可以用来学习上下文感知的单词表示, 这与 word2vec 学习固定的词表示有着显著不同。进一步, ELMo 可以根据下游任务数据对 biLSTM 网络进行微调 (Fine-Tuning), 从而实现面向特定任务的模型优化。然而, 传统序列神经网络的长文本建模能力较弱, 并

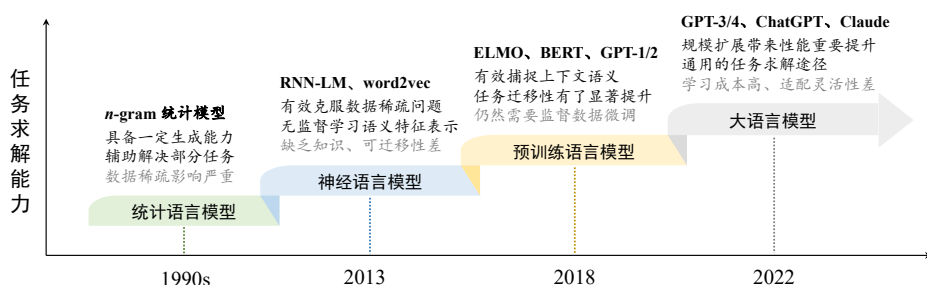


图 1.2 基于任务求解能力的四代语言模型的演化过程（图片来源：[10]）

且不容易并行训练，这些缺点限制了早期预训练模型（如 ELMo）的性能。在 2017 年，谷歌提出了基于自注意力机制（Self-Attention）的 Transformer 模型 [12]，通过自注意力机制建模长程序列关系。Transformer 的一个主要优势就是其模型设计对于硬件非常友好，可以通过 GPU 或者 TPU 进行加速训练，这为研发大语言模型提供了可并行优化的神经网络架构。基于 Transformer 架构，谷歌进一步提出了预训练语言模型 BERT [13]，采用了仅有编码器的 Transformer 架构，并通过在大规模无标注数据上使用专门设计的预训练任务来学习双向语言模型。在同期，OpenAI 也迅速采纳了 Transformer 架构，将其用于 GPT-1 [14] 的训练。与 BERT 模型不同的是，GPT-1 采用了仅有解码器的 Transformer 架构，以及基于下一个词元预测的预训练任务进行模型的训练。一般来说，编码器架构被认为更适合去解决自然语言理解任务（如完形填空等），而解码器架构更适合解决自然语言生成任务（如文本摘要等）。以 ELMo、BERT、GPT-1 为代表的预训练语言模型确立了“预训练-微调”这一任务求解范式。其中，预训练阶段旨在通过大规模无标注文本建立模型的基础能力，而微调阶段则使用有标注数据对于模型进行特定任务的适配，从而更好地解决下游的自然语言处理任务。

- 大语言模型（Large Language Model, LLM）。研究人员发现，通过规模扩展（如增加模型参数规模或数据规模）通常会带来下游任务的模型性能提升，这种现象通常被称为“扩展法则”（Scaling Law）[15]。一些研究工作尝试训练更大的预训练语言模型（例如 175B 参数的 GPT-3 和 540B 参数的 PaLM）来探索扩展语言模型所带来的性能极限。这些大规模的预训练语言模型在解决复杂任务时表现出了与小型预训练语言模型（例如 330M 参数的 BERT 和 1.5B 参数的 GPT-2）不同的行为。例如，GPT-3 可以通过“上下文学习”（In-Context Learning, ICL）的方式来利用少样本数据解决下游任务，而 GPT-2 则不具备这一能力。这种大模型具有但小模型不具有的能力通常被称为“涌现能力”（Emergent Abilities）。为了区

别这一能力上的差异，学术界将这些大型预训练语言模型命名为“大语言模型”¹ (Large Language Model, LLM) [16]。作为大语言模型的一个代表性应用，ChatGPT 将 GPT 系列大语言模型适配到对话任务中，展现出令人震撼的人机对话能力，一经上线就取得了社会的广泛关注。ChatGPT 发布后，与大语言模型相关的 arXiv 论文数量迅速增长（如图 1.1 所示），这一研究方向受到了学术界的高度关注。

通过回顾上述发展历程，可以看到语言模型并不是一个新的技术概念，而是历经了长期的发展历程。早期的语言模型主要面向自然语言的建模和生成任务，而最新的语言模型（如 GPT-4）则侧重于复杂任务的求解。从语言建模到任务求解，这是人工智能科学思维的一次重要跃升，是理解语言模型前沿进展的关键所在。图 1.2 通过任务求解能力的角度对比了四代语言模型所表现出的能力优势与局限性。首先，早期的统计语言模型主要被用于（或辅助用于）解决一些特定任务，主要以信息检索、文本分类、语音识别等传统任务为主。随后，神经语言模型专注于学习任务无关的语义表征，旨在减少人类特征工程的工作量，可以大范围扩展语言模型可应用的任务。进一步，预训练语言模型加强了语义表征的上下文感知能力，并且可以通过下游任务进行微调，能够有效提升下游任务（主要局限于自然语言处理任务）的性能。随着模型参数、训练数据、计算算力的大规模扩展，最新一代大语言模型的任务求解能力有了显著提升，能够不再依靠下游任务数据的微调进行通用任务的求解。综上所述，在语言模型的演化过程中，可以解决的任务范围得到了极大扩展，所获得的任务性能得到了显著提高，这是人工智能历史上的一次重要进步。

1.2 大语言模型的能力特点

大语言模型的出现为科研人员再次带来了实现通用人工智能（Artificial General Intelligence）的曙光。尽管通用人工智能在学术界被广泛讨论与探索，但是之前的机器学习算法的泛化性和通用性非常局限，只有大语言模型初步实现了通过统一形式来解决各种下游任务。本部分内容将简要介绍一下大语言模型的主要能力特点，特别是针对传统模型不具备的性能优势进行讨论。

- 具有较为丰富的世界知识. 与传统机器学习模型相比，大语言模型经过超大规模文本数据的预训练后能够学习到较为丰富的世界知识。实际上，最早期的专

¹值得注意的是，大语言模型不一定比小型预训练语言模型具有更强的任务效果，而且某些大语言模型中也可能不具有某种涌现能力。

家系统也是希望能够通过设计基于知识库与知识表示的推理引擎系统，进而解决特定领域的应用任务。然而，当时所采用的技术路径主要是基于逻辑、规则以及初期的机器学习算法，系统能力还比较局限，无法充分建模以及利用世界知识信息。尽管早期的预训练模型（如 BERT、GPT-1 等）也是基于相似的预训练思路，但是模型参数规模与数据规模都相对较小，无法充分学习到海量的世界知识。因此，之前的预训练语言模型需要使用微调为主要手段来解决下游任务。

- 具有较强的通用任务解决能力. 大语言模型第二个代表性的能力特点是具有较强的通用任务求解能力。大语言模型主要通过预测下一个词元的预训练任务进行学习，虽然并没有针对特定的下游任务进行优化，却能够建立远强于传统模型的通用任务求解能力。实际上，基于大规模无标注文本的下一个词元预测任务本质上可以看作一个多任务学习过程 [17]，因为针对不同词元的预测任务可能涉及到情感分类（“... 这部电影真好看”）、数值计算（“ $3+4=7$ ”）、知识推理（“中国陆地面积最大的省份是新疆”）等非常多样的训练任务。由于具有通用的任务求解能力，大语言模型深刻地影响了很多研究领域的科研范式。例如，在自然语言处理领域，很多传统任务（如摘要、翻译等）都可以采用基于大语言模型的提示学习方法进行解决，而且能够获得较好的任务效果，早期任务特定的解决方案已经被逐步替代。

- 具有较好的复杂任务推理能力. 除了具有通用性外，大语言模型在复杂任务中还展现出了较好的推理能力。例如，大语言模型能够回答知识关系复杂的推理问题 [18]，还可以解决涉及复杂数学推理过程的数学题目 [19]。在这些任务中，传统方法的性能相对较差，为了提升与其相关的特定能力，往往需要针对性地修改模型架构或者使用特定训练数据进行学习。相比之下，大语言模型在大规模文本数据预训练后，能够展现出比传统模型更强的综合推理能力。尽管有些研究工作认为大语言模型不具备真正的推理能力，而是通过“记忆”数据模式来进行任务求解，但在许多复杂应用场景中（参阅微软针对 GPT-4 的测试报告 [20]），大语言模型展现出了令人震撼的推理性能，这种现象很难完全通过数据模式的记忆与组合来进行解释。

- 较强的人类指令遵循能力. 大语言模型建立了自然语言形式的统一任务解决模式：任务输入与执行结果均通过自然语言进行表达。通过预训练与微调两个阶段的学习，大语言模型具备了较好的人类指令遵循能力，能够直接通过自然语言描述下达任务指令（又称为“提示学习”）。在早期的对话系统中，指令遵循

就是一个受到广泛关注的研究方向。然而，传统模型缺乏通用的任务理解与执行能力，仍然需要依赖人工规则或者先验信息辅助指令理解模块的设计与训练。由于具有较强的指令遵循能力，大语言模型为人机交互提供了一种自然的、通用的技术路径，这对于打造很多以人为中心的应用服务（如智能音箱、信息助手等）具有重要的意义。

- 具有较好的人类对齐能力. 机器学习模型的安全性一直以来是一个重要的研究课题。然而，传统模型的智能性、通用性相对较弱，很多科研学者对于模型安全性的关注程度通常远低于对于提升模型性能的关注。随着大语言模型的出现，由于其具有出色的模型性能，如果不进行有效的对齐与监管，将可能带来非常严重的后果。目前广泛采用的对齐方式是基于人类反馈的强化学习技术，通过强化学习使得模型进行正确行为的加强以及错误行为的规避，进而建立较好的人类对齐能力。目前很多线上部署的大语言模型应用，都能够有效阻止典型的模型功能滥用行为，一定程度上规避了常见的使用风险。

- 具有可拓展的工具使用能力. 在机器学习领域，模型的设计和实现往往都具有一定的局限性，例如会受到所采用的归纳假设以及训练数据的限制。同样地，大语言模型的能力也具有一定的局限性。例如，它仍然无法有效回答涉及到预训练数据时间范围之外的问题，并且对于数学中的数值计算问题也表现不佳。作为解决方案，由于大语言模型具有较为通用的任务求解形式，可以通过微调、上下文学习等方式掌握外部工具的使用，如搜索引擎与计算器。实际上，世界上最会使用工具的智能体就是人类，人类不断发明新的技术与工具，拓展自己的认知与能力边界。工具学习实际上就是借鉴了这一思路，通过具有特殊功能的工具来加强大语言模型的能力。然而，工具的有效使用对于模型的任务理解能力和推理能力有着较高的要求，因此传统模型以及没有经过特殊微调的大语言模型往往不能很好地使用丰富的工具库。目前最先进的大语言模型如 GPT-4 等能够支持多种工具的使用，从而极大地提升了模型的任务解决能力。

除了上述主要的能力特点外，大语言模型还能够展现出很多其他重要能力，如长程对话的语义一致性、对于新任务的快速适配、对于人类行为的准确模拟等。本书将在后续的内容中对于这些特点再进行专门介绍。

1.3 大语言模型关键技术概览

从早期的统计语言模型到大语言模型，科研人员进行了一系列的技术探索，从而实现了模型能力的显著提升。下面将概括性地介绍一下大语言模型能够取得重要进展背后的关键技术。具体的技术细节可以参考本书后续章节的详细介绍。

- **规模扩展**。规模扩展是大语言模型的一个关键成功因素。在较早期的研究中，OpenAI 从参数、数据、算力三个方面深入地研究了规模扩展对于模型性能所带来的影响，建立了定量的函数关系，称之为“扩展法则”（Scaling Law）[15, 21]（论文在 2020 年发表），并在 GPT-3 中探索了千亿级模型参数规模（175B 参数）所带来的性能优势，为后期研发 GPT 系列模型打下了重要的基础。随后，谷歌研究团队也在 2022 年推出了具有 540B 参数的 PaLM 模型，子公司 DeepMind 也在同年发表了重要研究成果—Chinchilla 扩展法则 [22]。研究人员发现这些超大规模语言模型能够展现出一些小型语言模型不具备的能力特点，如上下文学习能力、思维链能力等 [23–25]，这也成为区分上一代预训练语言模型与大语言模型的重要标志。早期的研究主要关注模型参数规模所带来的性能优势，最近的工作则是加大对于高质量数据的规模扩展。针对十亿级别（如 2B 或 7B）参数的模型使用超大规模的数据（如 2T 或 3T 词元）进行训练，仍然可能无法达到这些模型的最大数据容量。实现规模扩展的关键在于模型架构的可扩展性。Transformer 模型的可扩展性非常强，对于硬件并行优化的支持也比较友好，特别适合大语言模型的研发，很多工作也在进一步针对其进行优化与改进。

- **数据工程**。OpenAI 于 2019 年就在 GPT-2 的论文中 [17] 给出了当前大语言模型的技术路线图：通过在海量文本上进行下一个词预测的优化，使得模型能够学习到丰富的语义知识信息，进而通过文本补全的方式解决各种下游任务。这种方式最大的好处是，极大地简化了模型的设计与优化过程，使得模型训练与使用都是基于自然语言生成的模式进行的。实际上，人工智能技术的几次重要升级都体现出了这种“大道至简”的思想。例如，早期的深度学习系统通过端到端的训练方法来建立输入与输出间的映射关系，而抛弃了传统耦合多个组件的复杂系统。在这种通用的预训练范式下，模型能力本质上是来源于所见过的训练数据，因此数据工程就变得极为重要，不是简单的扩大数据规模就能够实现的。目前来说，数据工程主要包括三个方面。首先，需要对于数据进行全面的采集，拓宽高质量的数据来源；其次，需要对于收集到的数据进行精细的清洗，尽量提升用于大模型

训练的数据质量；第三，需要进行有效的数据配比与数据课程，加强模型对于数据语义信息的利用效率。这三个方面的数据工程技术直接决定了最后大语言模型的性能水平。目前来说，针对英文的开源高质量数据集比较丰富，相关的数据工程技术讨论也相对较多，但是对于其他语言的研究关注度还有待进一步加强。

- **高效预训练.** 与传统预训练语言模型相比，成功训练出一个性能较强的大语言模型极具挑战性。由于参数规模巨大，需要使用大规模分布式训练算法优化大语言模型的神经网络参数。在训练过程中，需要联合使用各种并行策略以及效率优化方法，包括 3D 并行（数据并行、流水线并行、张量并行）、ZeRO（内存冗余消除技术）等。为了有效支持分布式训练，很多研究机构发布了专用的分布式优化框架来简化并行算法的实现与部署，其中具有代表性的分布式训练软件包括 DeepSpeed [26] 和 Megatron-LM [27]，它们能够有效支持千卡甚至万卡的联合训练。在实现上，大语言模型的训练过程需要搭建一个全栈式的优化体系架构，能够支持大规模预训练数据的调度安排，建立起可迭代的模型性能改进闭环，加强效果反馈机制，从而能够快速、灵活地进行相关训练策略的调整。由于大语言模型的训练需要耗费大量的算力资源，通常需要开展基于小模型的沙盒测试实验，进而确定面向大模型的最终训练策略。为此，GPT-4 构建了一整套面向大模型的基础训练架构，可以使用较少的算力开销来可靠地预测大模型的最终性能。此外，研发过程也需要关注较为实用的优化技巧，提升训练稳定性和优化效率，如混合精度训练。

- **能力激发.** 大语言模型经过超大规模数据的预训练后，能够编码大量的文本语义知识信息。然而，这个阶段的模型能力仍然是通过通用的下一个词预测任务建立的，主要目的是为了进行预训练文本数据的恢复。为了提升模型的任务求解能力，需要设计合适的指令微调以及提示策略进行激发或诱导。在指令微调方面，可以使用自然语言表达的任务描述以及期望的任务输出对于大语言模型进行指令微调，从而增强大语言模型的通用任务求解能力，提升模型在未见任务上的泛化能力。通常来说，现有的研究认为指令微调无法向大模型注入新的知识，而是训练大模型学会利用自身所掌握的知识与信息进行任务的求解。在提示学习方面，需要设计合适的提示策略去诱导大语言模型生成正确的问题答案。为此，研究人员提出了多种高级提示策略，包括上下文学习、思维链提示等，通过构建特殊的提示模板或者表述形式来提升大语言模型对于复杂任务的求解能力。提示工程已经成为利用大语言模型能力的一个重要技术途径。进一步，大语言模型还具有较好

的规划能力，能够针对复杂任务生成逐步求解的解决方案，从而简化通过单一步骤直接求解任务的难度，进一步提升模型在复杂任务上的表现。

- **人类对齐**. 互联网上开放的无标注文本数据的内容覆盖范围较广，可能包含低质量、个人隐私、事实错误的数据信息。因此，经过海量无标注文本预训练的大语言模型可能会生成有偏见、泄露隐私甚至对人类有害的内容。在实践应用中，需要保证大语言模型能够较好地符合人类的价值观。目前，比较具有代表性的对齐标准是“3 H 对齐标准”，即 **Helpfulness**（有用性）、**Honesty**（诚实性）和 **Harmlessness**（无害性）。与传统的任务优化目标不同，这三个对齐标准一定程度上都与人类主观感知相关，很难直接建立形式化的特定优化目标。为了解决这一问题，OpenAI 提出了基于人类反馈的强化学习算法（**Reinforcement Learning from Human Feedback, RLHF**）[28]，将人类偏好引入到大模型的对齐过程中：首先训练能够区分模型输出质量好坏的奖励模型，进而使用强化学习算法来指导语言模型输出行为的调整，让大语言模型能够生成符合人类预期的输出。由于强化学习算法的优化过程较为复杂，最近学术界开始涌现出一批使用监督微调的对齐方式，从而简化 RLHF 优化过程的算法，如 **DPO** 算法等 [29]。随着人工智能算法能力的不断提升，有效监管模型行为以及使用风险变得愈发重要，特别是当模型能力达到一个较高水平之后（如超级智能或人类水平）。为此，OpenAI 还专门发布了“超级对齐”（**Super-alignment**）的研究项目，旨在研究如何监管具有强人工智能能力的算法。

- **工具使用**. 由于大语言模型的能力主要是通过大规模文本数据的语义学习所建立的，因此在非自然语言形式的任务（如数值计算）中能力较为受限。此外，语言模型的能力也受限于预训练数据所提供的信息，无法有效推断出超过数据时间范围以及覆盖内容的语义信息。为了解决上述问题，工具学习成为一种扩展大语言模型能力的关键技术 [30, 31]，通过让大语言模型学会使用各种工具的调用方式，进而利用合适的工具去实现特定的功能需求。例如，大语言模型可以利用计算器进行精确的数值计算，利用搜索引擎检索最新的时效信息。为了能够有效地使用外部工具，GPT 系列模型通过插件机制来形成系统性的工具调用方式，这些插件可以类比为大语言模型的“眼睛和耳朵”，能够有效扩展大语言模型的能力范围。在技术路径上，工具调用能力主要是通过指令微调以及提示学习两种途径实现，而未经历过特殊训练或者缺乏有效提示的大语言模型则很难有效利用候选工具。本质上来说，工具使用这一思想来源于人类行为的启发，人类能够充分利用

各种外部工具来提升某种特定技能。例如，人类发明了汽车，能够有效缩短通勤的往返时间。随着应用范围的不断拓展，创建广泛的、可供大模型使用的工具资源变得愈为重要。

尽管大语言模型技术已经取得了显著进展，但是对于它的基本原理仍然缺乏深入的探索，很多方面还存在局限性或者提升空间。首先，大模型中某些重要能力（如上下文学习能力）的涌现仍然缺乏形式化的理论解释，需要针对大语言模型基础能力的形成原因进行深入研究，从而揭示大语言模型内部的工作机理。其次，大语言模型预训练需要大规模的计算资源支持，研究各种训练策略的效果并进行可重复性的消融实验的成本非常高昂。学术界难以获得充分的算力来系统性研究大语言模型；虽然工业界或者大型研究机构不断推出性能优异的开源大模型，但是这些模型的训练过程的开源程度还不够充分，许多重要的训练细节仍缺乏公开的研究报道。特别地，现有的大语言模型非常依赖于工程方法的优化（如数据清洗等），但是这些技术的理论支撑还比较缺乏。第三，让大语言模型充分与人类价值观或偏好对齐也是一项重要的科研挑战。尽管大语言模型已经具有较好的模型能力，但是在特定场景下或者蓄意诱导下，仍然可能生成虚构、有害或具有负面影响的内容。这一问题随着模型能力的提升而变得更为难于解决。为了应对模型能力未来可能超越人类监管能力的情况，需要设计更为有效的监管方法来消除使用大语言模型的潜在风险。综上所述，大语言模型技术的研究才刚刚开始，仍然存在众多的研究挑战等待突破，需要研究人员和工程人员携手努力解决。

1.4 大语言模型对科技发展的影响

大语言模型真正令我们震撼的地方是，它与小型预训练语言模型采用了相似的网络架构以及训练方法，但通过扩展模型参数规模、数据数量以及算力资源，却带来了令人意料之外的模型性能跃升。大语言模型首次实现了单一模型可以有效解决众多复杂任务，人工智能算法从未如此强大。

大语言模型对人工智能技术的未来发展方向带来了重要影响，ChatGPT 和 GPT-4 的出现引发了人们对于实现通用人工智能（Artificial General Intelligence, AGI）可能性的重新思考。2023 年 2 月，OpenAI 发布了一篇名为“Planning for AGI and beyond”的技术文章，讨论了实现通用人工智能的短期和长期计划 [32]；来自微软的研究团队也在一篇 arXiv 论文中详细地展示了 GPT-4 强大的模型性能，并认为 GPT-4 可能被视为通用人工智能系统的早期版本 [20]。随着大语言模型技

术的迅猛发展，人工智能相关研究领域正发生着重要的技术变革，下面以四个典型的领域进行介绍：

- 自然语言处理. 在自然语言处理领域，大语言模型可以作为一种通用的语言任务解决技术，能够通过特定的提示方式解决不同类型的任务，并且能够取得较为领先的效果。进一步，很多传统任务的研究意义在衰减，甚至有些任务被宣告“结束”（如摘要任务），研究范式开始全面转向大语言模型技术，研究人员的关注重点由“解决特定任务”迁移到“如何进一步提升大语言模型的综合能力”。语言智能开始成为主导人工智能发展方向的重要路径。

- 信息检索. 在信息检索领域，传统搜索引擎受到了人工智能信息助手（即 ChatGPT）这一新型信息获取方式的冲击。在基于大语言模型的信息系统中，人们可以通过自然语言对话的形式获得复杂问题的答案。微软也推出了基于大语言模型增强的搜索引擎 New Bing，将大语言模型与传统搜索引擎进行融合。但是，目前大语言模型信息系统的精确性与实时性还有待提升，无法完全胜任现有搜索引擎的角色。鉴于大语言模型与搜索引擎各自的优势，信息检索领域主要关注两个新兴方向的研究，即检索增强的大语言模型以及大语言模型增强的搜索系统，全面围绕大语言模型技术展开。

- 计算机视觉. 在计算机视觉领域，研究人员为了更好地解决跨模态或多模态任务，正着力研发类 ChatGPT 的视觉-语言联合对话模型，GPT-4 已经能够支持图文多模态信息的输入。由于开源大语言模型的出现，可以极大地简化多模态模型的实现难度，通过将图像、视频等模态的信息与文本语义空间相融合，可以通过计算量相对较少的微调方法来研发多模态大语言模型。进一步，基于下一个词元预测的思路也可能会带来多模态领域的基础模型架构的转变，例如 OpenAI 最新推出的 Sora 模型就是基于图像块序列建模的思路进行构建的。

- 人工智能赋能的科学研究（AI4Science）. 近年来，AI4Science 受到了学术界的广泛关注，目前大语言模型技术已经广泛应用于数学、化学、物理、生物等多个领域，基于其强大的模型能力赋能科学研究。例如，著名数学家陶哲轩曾多次在社交网络表示，他在数学科研中广泛使用大语言模型，用于辅助提供解题灵感甚至用于论文的撰写。此外，大语言模型也多次被证明在新材料发现、生物制药等多个方面都能起到一定的促进作用。随着大语言模型训练数据规模与范围的扩展，在未来将会在人类科学研究中扮演更为重要的角色。

除了在特定学科领域的应用，大语言模型对于整体的科研范式也正产生着重

要影响。为了有效提升大模型的性能，研究人员需要深入了解大模型相关的工程技术，对于理论与实践的结合提出了更高的需求。例如，训练大模型具备大规模数据处理与分布式并行训练方面的实践经验。进一步，大语言模型将改变人类开发和使用人工智能算法的方式。与小型预训练语言模型不同，访问大语言模型的主要方法是通过提示接口（Prompting Interface），例如 GPT-4 API。为了更好地发挥模型性能，使用者需要了解大语言模型的工作原理，并按照大语言模型能够遵循的方式来描述需要解决的任务。

此外，大语言模型对于产业应用带来了变革性的技术影响，将会催生一个基于大语言模型的应用生态系统。例如，微软 365（Microsoft 365）正利用大语言模型（即 Copilot）来加强自动化办公软件的自动化办公工作；OpenAI 也进一步推动 Assistants API 和 GPTs 来推广大模型智能体（Agent）的研发，从而实现特定任务的求解工具。在未来，将出现更多的以大语言模型为基础技术架构的科技应用产品，简化原来繁复的功能处理流程，加快软件研发周期，极大地改善用户体验。

1.5 本书的内容组织

本书主要面向希望系统学习大语言模型技术的读者，将重点突出核心概念与算法，并且配以示例与代码（伪代码）帮助读者理解特定算法的实现逻辑。由于大语言模型技术的快速更迭，本书无法覆盖所有相关内容，旨在梳理最具代表性的基础知识内容，帮助读者更好地了解大语言模型技术的核心知识点，能够快速上手相关的科研与工程项目。为了配合本书的阅读与使用，我们创建了一个 GitHub 项目网站，该网站收集了关于大语言模型的相关资源，链接为 <https://github.com/RUCAIBox/LLMSurvey>。

本书共设置了五个主要部分，分别是背景与基础知识部分、预训练部分、微调与对齐部分、大模型使用部分以及评测与应用部分，按照如下的内容组织进行设置：

- 背景与基础知识部分. 第 2 章将首先介绍大语言模型的构建过程，随后介绍大语言模型相关的背景知识以及重要概念，包括涌现能力、扩展定律以及二者之间的联系与区别；进一步介绍 GPT 系列模型的发展历程以及各个阶段的重要技术创新，从而能够更好地了解大语言模型的技术升级历史。第 3 章将介绍目前大语言模型相关的资源信息，包括公开可用的模型检查点与 API、数据集以及代码工具库，为读者梳理与汇总相关资源。

● 预训练部分. 第 4 章将主要介绍预训练数据的准备工作, 主要包括数据的收集、清洗以及词元化方法, 随后将介绍数据课程的设计方法。第 5 章将主要介绍大语言模型的架构, 主要包括 Transformer 模型、各种组件的详细配置、长文本建模以及一些新型的模型架构。第 6 章将主要介绍预训练过程所涉及到的预训练任务、优化参数设置、可扩展的训练技术以及参数量计算与效率分析方法, 并通过相关实战代码进行讲解。

● 微调与对齐部分. 第 7 章将主要介绍指令微调所涉及的数据构建、优化策略; 进一步将介绍典型的轻量化微调技术, 减少模型训练的开销; 并且通过实践示例介绍指令微调的具体流程。第 8 章将主要介绍大模型的人类对齐技术, 将以 RLHF 为主要方法进行详细介绍, 并且进一步介绍非强化学习的对齐方法, 最后探讨 SFT 与 RLHF 之间的关系。

● 大模型使用部分. 第 9 章将主要介绍大模型的解码与部署方法, 包括解码策略、解码加速算法、低资源部署策略以及其他模型压缩方法。第 10 章将主要介绍面向大语言模型的提示学习技术, 包括基础的提示学习设计方法、上下文学习方法以及思维链方法等。第 11 章将主要介绍面向复杂任务的规划技术, 探索如何将复杂任务进行有效分解, 并通过回溯、反思等关键技术形成有效的解决方案; 进一步, 将介绍如何构建基于大语言模型的智能体以及多智能体系统。

● 评测与应用部分. 第 12 章将主要介绍面向大语言模型性能的评测方法, 针对不同的能力维度介绍相关的评测集合、评测指标以及评测方法, 并且指出大语言模型目前存在的问题。第 13 章将主要介绍大语言模型的应用情况, 具体将分别从代表性的研究领域以及应用领域两个维度展开讨论, 我们将以代表性工作为驱动, 使得读者能够了解如何将大语言模型进行领域特化以及任务特化。

最后, 第 14 章将对于全文的内容进行总结, 进一步梳理目前每个部分存在的技术挑战以及研究趋势。

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-2 7B）。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，不仅仅是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 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 A100 80G）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。

2.2 扩展法则

大语言模型获得成功的关键在于对“规模扩展”(Scaling)的充分探索与利用。在实现上，大语言模型采用了与小型预训练语言模型相似的神经网络结构(基于注意力机制的 Transformer 架构)和预训练方法(如语言建模)。但是通过扩展参数规模、数据规模和计算算力，大语言模型的能力显著超越了小型语言模型的能力。有趣的是，这种通过扩展所带来的性能提升通常显著高于通过改进架构、算法等方面所带来的改进。因此，建立定量的建模方法，即扩展法则 (Scaling Law)，来研究规模扩展所带来的模型性能提升具有重要的实践指导意义。在本部分，将首先介绍两种常见的语言模型扩展法则的定义，并且进一步对于扩展法则进行深入讨论。

2.2.1 KM 扩展法则

2020 年，Kaplan 等人 [15] (OpenAI 团队) 首次建立了神经语言模型性能与三个主要因素——模型规模 (N)、数据规模 (D) 和计算算力 (C) 之间的幂律关系

(Power-Law Relationship)。由于原始论文中没有给出具体的扩展法则命名，本部分内容中使用两位共同第一作者姓氏的首字母来进行命名。在给定算力预算 c 的条件下，可以近似得到以下三个基本指数公式来描述扩展法则：

$$\begin{aligned} L(N) &= \left(\frac{N_c}{N} \right)^{\alpha_N}, \quad \alpha_N \sim 0.076, N_c \sim 8.8 \times 10^{13} \\ L(D) &= \left(\frac{D_c}{D} \right)^{\alpha_D}, \quad \alpha_D \sim 0.095, D_c \sim 5.4 \times 10^{13} \\ L(C) &= \left(\frac{C_c}{C} \right)^{\alpha_C}, \quad \alpha_C \sim 0.050, C_c \sim 3.1 \times 10^8 \end{aligned} \quad (2.1)$$

这里， $L(\cdot)$ 表示用以 nat^1 为单位的交叉熵损失。其中， N_c 、 D_c 和 C_c 分别表示非嵌入参数数量、训练数据数量和实际的算力开销。为了便于讨论，我们在不影响表达和理解的情况下对于原始的公式符号进行了适度简化。这三个公式是通过模型在不同数据规模（22M 到 23B 词元）、模型规模（768M 到 1.5B 非嵌入参数）和算力规模下的性能表现拟合推导得到的。为了推导这些公式，需要约定一些基本假设：一个因素的分析不会受到其他两个因素的限制，如当变动模型参数规模的时候，需要保证数据资源是充足的。

由公式 2.1 可见，模型性能与这三个因素之间存在着较强的依赖关系，可以近似刻画为指数关系。上述公式为规模扩展效应提供了一种定量的普适建模方法。通过普适规则能够更好地探究问题的本质，排除其他复杂因素的影响与干扰（如 OpenAI 研究团队发现模型形状对于上述公式的影响并不大）。

为了便于理解扩展法则对于模型性能的影响，OpenAI 的研究团队又将这里的损失函数进一步分解为两部分 [21]，包括不可约损失（真实数据分布的熵）和可约损失（真实分布和模型分布之间 KL 散度的估计）：

$$L(x) = \underbrace{L_\infty}_{\text{不可约损失}} + \underbrace{\left(\frac{x_0}{x} \right)^{\alpha_x}}_{\text{可约损失}}, \quad (2.2)$$

这里 x 是一个占位符号，可以指代公式 2.1 中的 N 、 D 和 C 。其中，不可约损失由数据自身特征确定，无法通过扩展法则或者优化算法进行约减；模型性能的优化只能减小可约损失部分。

¹nat 用来表示以 e 为底信息量的自然对数。

2.2.2 Chinchilla 扩展法则

Hoffmann 等人 [22] (DeepMind 团队) 于 2022 年提出了一种可选的扩展法则, 旨在指导大语言模型充分利用给定的算力资源进行优化训练。通过针对更大范围的模型规模 (70M 到 16B 参数) 和数据规模 (5B 到 500B 词元) 进行实验, 研究人员拟合得到了另一种关于模型性能的幂律关系:

$$L(N, D) = E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}, \quad (2.3)$$

其中 $E = 1.69$, $A = 406.4$, $B = 410.7$, $\alpha = 0.34$ 和 $\beta = 0.28$ 。进一步, 利用约束条件 $C \approx 6ND$ 对于损失函数 $L(N, D)$ 进行推导, 能够获得算力资源固定情况下模型规模与数据规模的最优分配方案 (如下所示):

$$N_{\text{opt}}(C) = G \left(\frac{C}{6} \right)^a, \quad D_{\text{opt}}(C) = G^{-1} \left(\frac{C}{6} \right)^b, \quad (2.4)$$

这里, $a = \frac{\alpha}{\alpha+\beta}$, $b = \frac{\beta}{\alpha+\beta}$, G 是由 A 、 B 、 α 和 β 计算得出的扩展系数。

进一步, 研究人员 [22] 发现 KM 扩展法则和 Chinchilla 扩展法则都可以近似表示成上述算力为核心的公式 (公式 2.4):

$$N_{\text{opt}} \approx C^a, D_{\text{opt}} \approx C^b, \quad (2.5)$$

即当算力 C 给定的情况下, 最优的模型参数规模和数据规模由指数系数 a 和 b 分别确定。可以看到, a 和 b 决定了参数规模和数据规模的资源分配优先级: 当 $a > b$ 时, 应该用更多的算力去提高参数规模; 当 $b > a$ 时, 应该用更多的算力去提高数据规模。尽管 KM 扩展法则和 Chinchilla 扩展法则具有相似的公式形式, 但是在模型规模和数据规模的扩展上存在一定的差异。随着算力预算的增加, KM 扩展法则 ($a \approx 0.73$, $b \approx 0.27$ [22]) 倾向于将更大的预算分配给模型规模的增加, 而不是分配给数据规模的增加; 而 Chinchilla 扩展法则主张两种规模参数应该以等比例关系增加 ($a \approx 0.46$, $b \approx 0.54$ [22])。

Chinchilla 扩展法则这项研究的意义并不在于给出了资源在数据规模与模型规模上的具体分配方案, 而是首次形式化指出了之前的预训练工作可能忽视了训练数据的规模扩展。例如, 具有 175B 参数的 GPT-3 仅仅使用了 300B 的词元进行训练, 所使用的数据量远远没有达到模型能够编码的最大数据容量。根据 Chinchilla 扩展法则的指导, DeepMind 的研究团队进一步训练得到了具有 70B 参数的 Chinchilla 模型, 使用大概 1.4T 的词元进行训练。虽然后续有些人借鉴 Chinchilla 模型的线

性分配比例（数据规模大概是模型参数规模的五倍），但是目前这一分配系数已经基本没有参考意义了。越来越多的工作表明，现有的预训练语言模型对于数据的需求量远高于这些扩展法则中所给出的估计规模。例如，LLaMA-2 (7B) 的模型就使用了 2T 的词元进行训练，很多更小的模型也能够通过使用超大规模的预训练数据获得较大的模型性能提升。这种现象的一个重要原因是由于 Transformer 架构具有较好的数据扩展性，到目前为止，还没有实验能够有效验证特定参数规模语言模型的饱和数据规模（即随着数据规模的扩展，模型性能不再提升）。

2.2.3 关于扩展法则的讨论

在介绍完上述两个扩展法则后，我们围绕可预测的扩展以及任务层面的可预测性展开深入讨论，以加强读者对于扩展法则的理解。

● **可预测的扩展 (Predictable Scaling)**：在实践中，扩展法则可以用于指导大语言模型的训练，通过较小算力资源可靠地估计较大算力资源投入后的模型性能，这被称为可预测的扩展 [35]。这种可预测性主要体现在两个方面：使用小模型的性能去预估大模型的性能，或者使用大模型的早期训练性能去估计训练完成后的性能。可预测扩展对于大模型训练具有两个主要的指导作用。首先，对于大语言模型来说，详细进行各种训练技巧或变体的测试需要耗费巨大的算力资源。因此，一个较为理想的经验性方法是，基于小模型获得训练经验然后应用于大模型的训练，从而减少实验成本。例如，可以训练小型代理模型来确定适合大型模型的预训练数据混合的最佳比例 [36]。其次，大语言模型的训练过程较长，经常面临着训练损失波动情况，扩展法则可以用于监控大语言模型的训练状态，如在早期识别异常性能。尽管扩展法则刻画了模型性能增长（或模型损失减少）的平滑趋势，但是指数形式的变化趋势意味着可能会出现随规模扩展的收益递减情况，即后期的扩展增益开始变得缓慢甚至停滞。根据 OpenAI 团队的一项研究表明 [21]，即使接近递减收益点（即接近不可规约的模型损失，见公式 2.2），模型表征的质量仍然能够随着规模扩展而有效提升 [21]。这一发现表明，训练大型模型对于改善下游任务的性能是非常重要的。随着模型规模的不断增加，一个潜在问题是可供用来训练大语言模型的数据量实际上是有限的，公共文本数据将很快变得“枯竭”。因此，如何在数据受限的情况下建模扩展法则，仍然具有重要的实践意义。在这一情况下，数据重复或数据合成可能有助于缓解数据稀缺问题。

● **任务层面的可预测性**：现有关于扩展法则的研究大多数是基于语言建模损失

开展的，例如预测下一个词元的平均交叉熵损失 [15]，这一度量本身是平滑的，是对于模型整体能力的宏观度量。在实践中，我们则更关注大语言模型在真实任务中的性能提升。为了建立扩展法则与模型任务性能的关联，一个基础问题就是语言建模损失的减少是否真正意味着（或对应着）真实任务上模型性能的提高 [21]。整体上来说，语言建模损失较小的模型往往在下游任务中表现更好，因为语言建模的能力可以被认为是一种模型整体能力的综合度量。然而，语言建模损失的减少并不总是意味着模型在下游任务上的性能改善。对于某些特殊任务，甚至会出现“逆向扩展”（Inverse Scaling）现象，即随着语言建模损失的降低，任务性能却出人意料地变差 [37]。总体而言，探索和描述任务层面的扩展法则更加困难，因为它可能还依赖于任务相关的信息（如任务指标、任务难度等）。根据 GPT-4 的报告 [35]，通过扩展法则可以准确预测某些任务能力（例如编码能力），但是对于有些任务的性能预测是非常困难的。此外，有些重要能力（例如上下文学习能力 [23]）根据扩展法则是不可预测的，只有当模型大小超过一定规模时才会出现，如下文所讨论的涌现能力。

2.3 涌现能力

在现有文献中 [24]，大语言模型的涌现能力被非形式化定义为“在小型模型中不存在但在大模型中出现的能力”，具体是指当模型扩展到一定规模时，模型的特定任务性能突然出现显著跃升的趋势，远超过随机水平。类比而言，这种性能涌现模式与物理学中的相变现象有一定程度的相似，但是仍然缺乏相应的理论解释以及理论证实，甚至有些研究工作对于涌现能力是否存在提出质疑 [38]。整体来说，涌现能力的提出有助于使得公众认识到大语言模型所具有的能力优势，能够帮助区分大语言模型与传统预训练语言模型之间的差异。在本书中，涌现能力用来指代大语言模型所具有的典型能力，并不关注该能力是否存在于小模型中。下面，首先在第 2.3.1 节中介绍三种具有代表性的涌现能力，随后在第 2.3.2 节中讨论涌现能力的不同观点以及可能存在争议的原因。

2.3.1 代表性的涌现能力

尽管涌现能力可以定义为解决某些复杂任务的能力水平，但我们更关注可以用来解决各种任务的普适能力。下面简要介绍大语言模型的三种典型涌现能力。

• 上下文学习 (In-context Learning, ICL) . 上下文学习能力在 GPT-3 的论文中 [23] 被正式提出。具体方式为, 在提示中为语言模型提供自然语言指令和多个任务示例 (Demonstration), 无需显式的训练或梯度更新, 仅输入文本的单词序列就能为测试样本生成预期的输出。在 GPT 系列模型中, 175B 参数的 GPT-3 模型展现出强大的上下文学习能力, 而 GPT-1 和 GPT-2 模型则不具备这种能力。此外, 上下文学习能力还取决于具体的下游任务。例如, 13B 参数的 GPT-3 模型可以在算术任务 (例如 3 位数的加减法) 上展现出上下文学习能力, 但 175B 参数的 GPT-3 模型在波斯语问答任务上甚至不能表现出良好的性能 [24]。

• 指令遵循 (Instruction Following) . 指令遵循能力是指大语言模型能够按照自然语言指令来执行对应的任务 [28, 39, 40]。为了获得这一能力, 通常需要使用自然语言描述的多任务示例数据集进行微调, 称为指令微调 (Instruction Tuning) 或监督微调 (Supervised Fine-tuning)。通过指令微调, 大语言模型可以在没有使用显式示例的情况下按照任务指令完成新任务, 有效提升了模型的泛化能力。相比于上下文学习能力, 指令遵循能力整体上更容易获得, 但是最终的任务执行效果还取决于模型性能和任务难度决定。例如, FLAN-PaLM 模型 [41] 测试了 8B、62B 以及 540B 三个参数规模的模型在指令微调之后的效果, 当参数规模达到 62B 及以上的情况, 才能够在包含 23 个复杂推理任务的 BBH 评估基准上, 展现出较好的零样本推理能力。对于规模相对较小的语言模型 (如 2B), 也可以通过使用高质量指令数据微调的方式习得一定的通用指令遵循能力 (主要是简单任务, 如文本摘要等) [42]。

• 逐步推理 (Step-by-step Reasoning) . 对于小型语言模型而言, 通常很难解决涉及多个推理步骤的复杂任务 (如数学应用题), 而大语言模型则可以利用思维链 (Chain-of-Thought, CoT) 提示策略 [25] 来加强推理性能。具体来说, 大语言模型可以在提示中引入任务相关的中间推理步骤来加强复杂任务的求解, 从而获得更为可靠的答案。在思维链的原始论文中发现 [25], 对于 62B 和 540B 参数的 PaLM 模型, 思维链提示可以提高其在算术推理基准上的效果, 但是 8B 参数的模型则很难获得提升。进一步, 思维链所带来的提升在 540B 参数的 PaLM 模型上会更加明细。此外, 思维链提示对不同任务的性能提升也不完全相同, 例如 PaLM 模型在三个数据集上产生了不同的提升幅度 (GSM8K > MAWPS > SWAMP) [25]。思维链提示特别适合帮助大语言模型解决复杂数学问题, 而具有思维链能力也是大语言模型能力的重要体现。

通常来说，很难统一界定大语言模型出现这些上述能力的临界规模（即具备某种能力的最小规模），因为能力涌现会受到多种因素或者任务设置的影响。最近的研究表明，经过了高质量的预训练与指令微调后，即使较小的语言模型（如 LLaMA-2 (7B)）也能够一定程度上展现出上述提到的三种能力，并且对于参数规模的要求随着预训练数据规模的扩展以及数据质量的提升在不断下降。此外，现有的研究对于能力涌现的实验往往局限于少数几个模型规模。例如，PaLM 模型的相关公开研究只在 8B、62B 和 540B 三种模型规模上进行了测试，对于未测试过的模型规模的性能尚不清楚。

2.3.2 涌现能力与扩展法则的关系

扩展法则和涌现能力提供了两种不同观点来理解大模型相对于小模型的优势，但是刻画了较为不同的扩展效应趋势。扩展法则使用语言建模损失来衡量语言模型的整体性能，整体上展现出了较为平滑的性能提升趋势，具有较好的可预测性，但是指数形式暗示着可能存在的边际效益递减现象；而涌现能力通常使用任务性能来衡量模型性能，整体上展现出随规模扩展的骤然跃升趋势，不具有可预测性，但是一旦出现涌现能力则意味着模型性能将会产生大幅跃升。由于这两种观点反映了不同的模型性能提升趋势（持续改进 v.s. 性能跃升），可能在一些情况下会导致不一致的发现与结论。

关于涌现能力的合理性也存在广泛的争议。一种推测是，涌现能力可能部分归因于特殊任务的设置 [43, 44]：现有评测设置中通常采用不连续的评估指标（如生成代码的准确性使用测试数据通过率评估）以及较为有限的模型参数规模（如 PaLM 技术报告里只展示了 8B、62B 和 540B 三个版本的模型）。在上述这两种情况下，很容易在下游任务的评测效果上产生不连续的变化趋势，导致了所谓的模型能力的“涌现现象”。特别地，如果针对性地修改评估指标时，或者提供更为连续的模型尺寸候选使之变得更为平滑后，涌现能力曲线的突然跃升趋势有可能会消失。这种分析一定程度上解释了模型性能的陡然跃升现象，为涌现能力的存在性打上了问号。然而，在实际使用中，用户就是以一种“不连续”的方式去感知大语言模型的性能优劣。换句话说，模型输出的正确性更为重要，用户满意度的体验过程本身就是离散的。例如，用户更倾向于使用能够正确通过所有测试用例的代码，而不愿意在两个失败代码之间选择一个包含错误较少的代码。

目前还缺少对于大语言模型涌现机理的基础性解释研究工作。与这一问题较

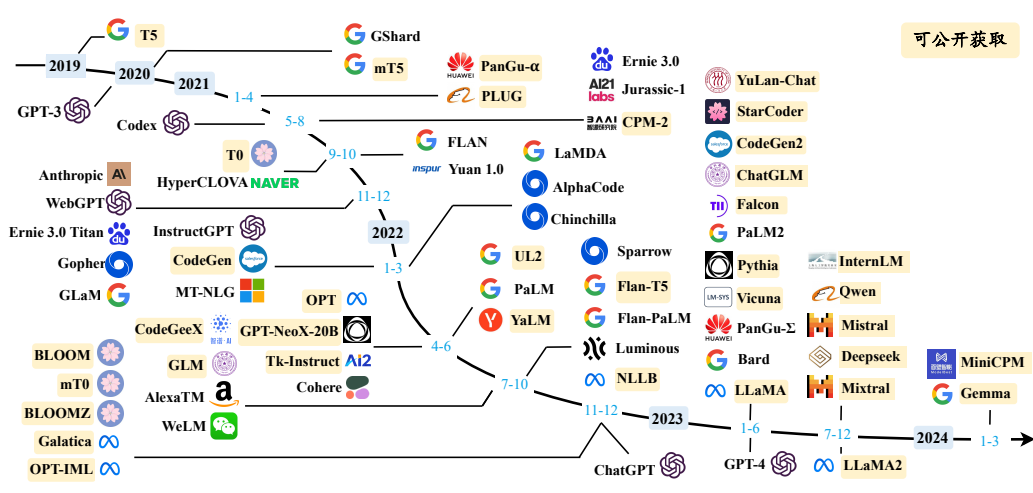


图 2.1 大语言模型发展时间线（图片来源：[10]）

为相关的研究叫做“顿悟”（Grokking），是指训练过程中的一种数据学习模式：模型性能从随机水平提升为高度泛化 [45]。在未来研究中，还需要更为深入的相关讨论，才能够有效解释大模型的涌现机理。通俗来讲，扩展法则与涌现能力之间微妙的关系可以类比人类的学习能力来解释。以语言能力为例，对于儿童来说，语言发展（尤其是婴儿）可以被看作一个多阶段的发展过程，其中也会出现“涌现现象”。在这一发展过程中，语言能力在一个阶段内部相对稳定，但是当进入另一个能力阶段时可能会出现重要的提升（例如从说简单的单词到说简单的句子）。尽管儿童实际上每天都在成长，但是语言的提升过程本质上是不平滑和不稳定的（即语言能力在时间上不以恒定速率发展）。因此，经常可以看到年轻的父母会对宝宝所展现出的语言能力进展感到惊讶。

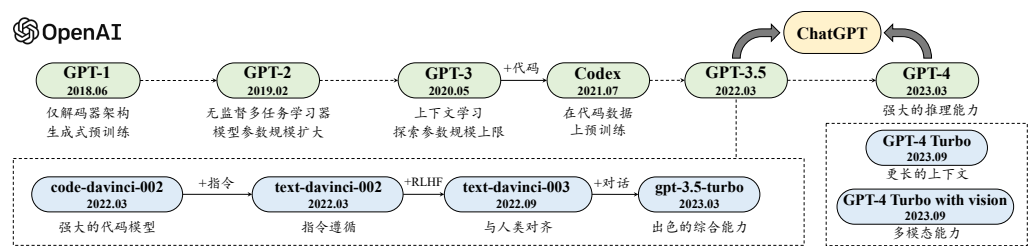


图 2.2 GPT 系列模型技术发展的历程图（图片来源：[10]）

2.4 GPT 系列模型的技术演变

2022 年 11 月底, OpenAI 推出了基于大语言模型的在线对话应用 — ChatGPT。由于具备出色的人机对话能力和任务解决能力, ChatGPT 一经发布就引发了全社会对于大语言模型的广泛关注, 众多的大语言模型应运而生, 并且数量还在不断增加 (图 2.1)。由于 GPT 系列模型具有重要的代表性, 本部分内容将针对 GPT 系列模型的发展历程进行介绍, 并且凝练出其中的重要技术创新。

GPT 系列模型的基本原理是训练模型学习恢复预训练文本数据, 将广泛的世界知识压缩到仅包含解码器 (Decoder-Only) 的 Transformer 模型中, 从而使模型能够学习获得较为全面的能力。其中, 两个关键要素是: (I) 训练能够准确预测下一个词的 Transformer (只包含解码器) 语言模型; (II) 扩展语言模型的规模以及扩展预训练数据的规模。图 2.2 展示了 GPT 系列模型的技术演进示意图, 这里主要根据 OpenAI 的论文、博客文章和官方 API 说明的信息进行绘制。该图中 实线表示在两个模型之间的进化路径上存在明确的证据 (例如, 官方声明新模型是基于基础模型开发的), 而 虚线 表示相对较弱的进化关系。截止到目前, OpenAI 对大语言模型的研发历程大致可分为四个阶段: 早期探索阶段、路线确立阶段、能力增强阶段以及能力跃升阶段, 下面进行具体介绍²。

2.4.1 早期探索

根据对于 Ilya Sutskever (OpenAI 联合创始人、前首席科学家) 的采访³, OpenAI 在成立初期就尝试使用语言模型研发人工智能系统, 但当时使用的是循环神经网络 [46], 模型能力和并行训练能力还存在较大的局限性。Transformer 刚刚问世, 就引起了 OpenAI 团队的高度关注, 并且将语言模型的研发工作切换到 Transformer 架构上, 相继推出了两个初始的 GPT 模型, 即 GPT-1 [14] 和 GPT-2 [17], 这两个早期工作奠定了后续更强大的 GPT 模型 (如 GPT-3 和 GPT-4) 的研究基础。

- *GPT-1*. 2017 年, Google 推出 Transformer 模型后, OpenAI 团队马上意识到这种神经网络架构将显著优于传统序列神经网络的性能, 有可能对于研发大型神经网络产生重要的影响。他们很快着手使用 Transformer 架构研发语言模型, 并于 2018 年发布了第一个 GPT 模型, 即 GPT-1, 模型名称 GPT 是生成式预训练

²本部分讨论的内容是作者基于调研 OpenAI 发布的论文、博客文章、采访报道和 API 说明文档所形成的个人理解。

³<https://hackernoon.com/an-interview-with-ilya-sutskever-co-founder-of-openai>

(Generative Pre-Training) 的缩写。GPT-1 基于生成式、仅有解码器的 Transformer 架构开发，奠定了 GPT 系列模型的核心架构与基于自然语言文本的预训练方式，即预测下一个词元。由于当时模型参数规模还相对较小，模型仍然缺乏通用的任务求解能力，因而采用了无监督预训练和有监督微调相结合的范式。与 GPT-1 同期发布的预训练语言模型是大名鼎鼎的 BERT 模型。BERT 与 GPT-1 虽然都采用了基于 Transformer 架构的预训练学习方式，但是它主要面向自然语言理解任务 (Natural Language Understanding, NLU)，为此只保留了 Transformer 中的编码器，其中 BERT-Large 模型在众多的自然语言理解任务上取得了非常重要的提升，成为当时备受瞩目的“明星模型”。可以说，BERT 当时引领了自然语言处理社区的研究浪潮，涌现了大量针对它改进与探索的工作。由于 GPT-1 模型规模实际上与小规模的 BERT-Base 模型相当 (100M 左右参数)，在公开评测数据集上的性能尚不能达到当时众多竞争模型中的最优效果，没有引起学术界的足够关注。

● **GPT-2.** GPT-2 沿用了 GPT-1 的类似架构，将参数规模扩大到 1.5B，并使用大规模网页数据集 WebText 进行预训练。与 GPT-1 不同，GPT-2 旨在探索通过扩大模型参数规模来提升模型性能，并且尝试去除针对特定任务所需要的微调环节。这点在 GPT-2 的论文 [17] 中得到了着重论述，它试图使用无监督预训练的语言模型来解决各种下游任务，进而不需要使用标注数据进行显式的模型微调。形式化来说，多任务学习 (Multi-task Learning) 可以通过一种较为通用的概率形式刻画，即 $P(\text{output}|\text{input}, \text{task})$ ——根据输入和任务信息来预测输出。为了建立通用的多任务学习框架，GPT 系列模型将输入、输出和任务信息都通过自然语言形式进行描述，进而后续任务的求解过程就可以看作是任务方案（或答案）的文本生成问题。OpenAI 团队在 GPT-2 的论文中还尝试解释无监督预训练在下游任务中取得良好效果的原因：“由于特定任务的有监督学习目标与无监督学习目标（语言建模）在本质上是相同的（预测下一个词元），主要区别就在于它们只是在全部训练数据的子集上进行优化，因此对于特定下游任务而言，优化无监督的全局学习目标本质上也是在优化有监督的任务学习目标” [17]。对这一说法的通俗理解是，语言模型将每个（自然语言处理）任务都视为基于世界文本子集的下一个词预测问题。因此，如果无监督语言建模经过训练后具有足够的能力复原全部世界文本，那么本质上它就能够解决各种任务。这些 GPT-2 论文中的早期讨论与 Ilya Sutskever 在接受 Jensen Huang 采访时的观点非常类似：“神经网络学到的是生成文本过程中的某种表示，这些模型的生成文本实际上是真实世界的投影……（语言模型）对

下一个单词的预测越准确，（对于世界知识）保真度就越高，在这个过程中获得的分辨率就越高……”⁴。

2.4.2 规模扩展

虽然 GPT-2 的初衷是成为一个“无监督多任务学习器”，但在很多任务上与有监督微调方法相比，模型效果整体上还是要逊色一些。在 GPT-2 基础上，GPT-3 针对（几乎相同的）模型参数规模进行了大幅扩展，在下游任务中初步展现出了一定的通用性（通过上下文学习技术适配下游任务），为后续打造更为强大的模型确立了关键的技术发展路线。

● **GPT-3.** OpenAI 在 2020 年发布了 GPT-3 模型，将模型参数扩展到了 175B 的规模。与 GPT-2 相比，GPT-3 直接将参数规模提升了 100 余倍，对于模型扩展在当时给出了一个极限尝试，其雄心、魄力可见一斑。值得一提的是，OpenAI 的两篇关于扩展法则的论文 [15, 21] 都是在 2020 年发表的，这说明在 GPT-3 开始训练时可能已经进行了比较充分的实验探索，包括小版本模型的尝试、数据收集与清洗、并行训练技巧等。在 GPT-3 的论文中，它正式提出了“上下文学习”这一概念，使得大语言模型可以通过少样本学习的方式来解决各种任务。上下文学习可以指导大语言模型学会“理解”自然语言文本形式描述的新任务，从而消除了针对新任务进行微调的需要。基于这一学习范式，大语言模型的训练与利用可以通过语言建模的形式进行统一描述：模型预训练是在给定上下文条件下预测后续文本序列，模型使用则是根据任务描述以及示例数据来推理正确的任务解决方案。GPT-3 不仅在各种自然语言处理任务中表现出了优异的效果，对于一些需要复杂推理能力或领域适配能力的特定任务也具有较好的解决能力。虽然 GPT-3 的论文没有明确提出上下文学习能力是大语言模型的涌现能力，但是指出了上下文学习对于大模型的性能增益会更加显著，而对于小模型来说则收益较小（见 GPT-3 论文 [23] 的原始图 1.2）。总体而言，GPT-3 可以被看作从预训练语言模型到大语言模型演进过程中的一个重要里程碑，它证明了将神经网络扩展到超大规模可以带来大幅的模型性能提升，并且建立了以提示学习方法为基础技术路线的任务求解范式。

⁴<https://lilearnitect.ai/ilya/>

2.4.3 能力增强

由于具有较强的模型性能，GPT-3 成为 OpenAI 开发更强大的大语言模型的研究基础。根据公开资料披露的内容来说，OpenAI 探索了两种主要途径来改进 GPT-3 模型，即代码数据训练和人类偏好对齐。

- 代码数据训练. 原始的 GPT-3 模型的复杂推理任务能力仍然较弱，如对于编程问题和数学问题的求解效果不好。为了解决这一问题，OpenAI 于 2021 年 7 月推出了 Codex [47]，这是一个在大量 GitHub 代码数据集合上微调的 GPT 模型。实验结果表明，Codex 可以解决非常困难的编程问题，还能显著提升大模型解决数学问题的能力 [48]。此外，2022 年 1 月 OpenAI 还公开了一种用于训练文本和代码嵌入的对比方法 [49]，结果表明该方法能够改善一系列相关任务的性能，包括线性探测分类、文本搜索和代码搜索等。根据 OpenAI 所发布的 API 信息所示，GPT-3.5 模型是在基于代码训练的 GPT 模型（即 `code-davinci-002`）基础上开发的，这表明在代码数据上进行训练有助于提高 GPT 模型的综合性能，尤其是代码能力。另一个可能的启发是对于可用于预训练的数据范围的扩展，可能并不局限于自然语言形式表达的文本数据。

- 人类对齐. OpenAI 关于人类对齐的公开研究工作可以追溯到 2017 年（实际时间或更早）。在一篇题为“Learning from Human Preferences”⁵的博客文章中，OpenAI 的研究团队介绍了一项使用强化学习算法从人类标注的偏好数据中学习如何改进模型性能的工作 [50]。在这篇强化学习工作发表不久，2017 年 7 月 OpenAI 研究团队又提出了一项改进的强化学习算法—PPO 算法（Proximal Policy Optimization, PPO）[51]，这也成为了 OpenAI 在后续人类对齐技术里所采用的标配强化学习算法。在 2020 年，OpenAI 研究团队将人类对齐算法应用于提升自然语言处理任务上的能力，训练了一个根据人类偏好进行优化的摘要模型 [52]。以这些前期工作为基础，2022 年 1 月，OpenAI 正式推出 InstructGPT [28] 这一具有重要影响力的学术工作，旨在改进 GPT-3 模型与人类对齐的能力，正式建立了基于人类反馈的强化学习算法，即 RLHF 算法。值得一提的是，在 OpenAI 的论文和相关文档中，很少使用“指令微调”（Instruction Tuning）一词，主要是使用“监督微调”一词（即基于人类反馈的强化学习算法的第一步 [28]）。除了提高指令遵循能力，基于人类反馈的强化学习算法有助于缓解有害内容的生成，这对于大语言模型在实际应用中的安全部署非常重要。OpenAI 在一篇技术博客文章中描述了他们对齐

⁵<https://openai.com/research/learning-from-human-preferences>

研究的技术路线 [53], 并总结了三个有前景的研究方向: 训练人工智能系统以达到 (1) 使用人类反馈、(2) 协助人类评估和 (3) 进行对齐研究。

通过这些增强技术, OpenAI 将改进后的具有更强能力的 GPT 模型命名为 GPT-3.5 模型 (参见第 3.1 节中有关 OpenAI API 的讨论)。

2.4.4 性能跃升

在历经上述近五年的重要探索, OpenAI 自 2022 年底开始发布了一系列重要的技术升级, 其中具有代表性的模型是 ChatGPT、GPT-4 以及 GPT-4V/GPT-4 Turbo, 这些模型极大提高了现有人工智能系统的能力水平, 成为了大模型发展历程中的重要里程碑。

- *ChatGPT*. 2022 年 11 月, OpenAI 发布了基于 GPT 模型的人工智能对话应用服务 ChatGPT。OpenAI 官方博客文章 [54] 概要地介绍了 ChatGPT 的研发技术, 主要是沿用了 InstructGPT (原帖中称 ChatGPT 为 “InstructGPT 的兄弟模型”) 的训练技术, 但是对于对话能力进行了针对性优化。在训练数据的收集过程中, ChatGPT 将人类生成的对话数据 (同时扮演用户和人工智能的角色) 与训练 InstructGPT 的相关数据进行结合, 并统一成对话形式用于训练 ChatGPT。ChatGPT 在与人机对话测试中展现出了众多的优秀能力: 拥有丰富的世界知识、复杂问题的求解能力、多轮对话的上下文追踪与建模能力、与人类价值观对齐的能力等。在后续的版本更迭中, ChatGPT 进一步又支持了插件机制, 通过现有工具或应用程序扩展了它的功能, 能够超越以往所有人机对话系统的能力水平。ChatGPT 一经推出就引发了社会的高度关注, 对于人工智能的未来研究产生了重要影响。

- *GPT-4*. 继 ChatGPT 后, OpenAI 于 2023 年 3 月发布了 GPT-4 [35], 它首次将 GPT 系列模型的输入由单一文本模态扩展到了图文双模态。总体来说, GPT-4 在解决复杂任务方面的能力显著强于 GPT-3.5, 在一系列面向人类的考试中都获得了非常优异的结果。GPT-4 发布后, 微软的研究团队针对其进行了大规模人类生成问题的性能测试 [20], 实验结果表明 GPT-4 具有令人震撼的模型性能, 论文作者认为 GPT-4 的到来展现出了通用人工智能的曙光。此外, 由于进行了为期六个月的迭代对齐 (在基于人类反馈的强化学习中额外增加了安全奖励信号), GPT-4 对恶意或挑衅性查询的响应更加安全。在技术报告中 [35], OpenAI 强调了安全开发 GPT-4 的重要性, 并应用了一些干预策略来缓解大语言模型可能出现的问题——幻觉、隐私泄露等。例如, 研究人员引入了 “红队攻击” (Red Teaming) 机制 [55]

来减少生成有害或有毒的内容。更重要的是，GPT-4 搭建了完备的深度学习训练基础架构，进一步引入了可预测扩展的训练机制，可以在模型训练过程中通过较少计算开销来准确预测模型的最终性能。

● *GPT-4V*、*GPT-4 Turbo* 以及多模态支持模型。基于发布的 GPT-4 初版模型 [35]，OpenAI 在 2023 年 9 月进一步发布了 GPT-4V，重点关注 GPT-4 视觉能力的安全部署。在 GPT-4V 的系统说明中 [56]，广泛讨论了与视觉输入相关的风险评估手段和缓解策略。GPT-4V 在多种应用场景中表现出了强大的视觉能力与综合任务解决能力。在 2023 年 11 月，OpenAI 在开发者大会上发布了升级版的 GPT-4 模型，称为 *GPT-4 Turbo*，引入了一系列技术升级：提升了模型的整体能力（比 GPT-4 更强大），扩展了知识来源（拓展到 2023 年 4 月），支持更长上下文窗口（达到 128K），优化了模型性能（价格更便宜），引入了若干新的功能（如函数调用、可重复输出等）。同时，*Assistants API* 功能也被推出，旨在提升人工智能应用助手的开发效率，开发人员可以利用特定的指令、外部知识和工具，在应用程序中快速创建面向特定任务目标的智能助手。此外，新版本的 GPT 模型还进一步增强了多模态能力，分别由 *GPT-4 Turbo with Vision*、*DALL·E-3*、*TTS (Text-to-speech)* 以及 *Listen to voice samples* 等支持实现。这些技术升级进一步提高了 GPT 模型的任务性能，扩展了 GPT 模型的能力范围。更重要的是，随着模型性能和支撑功能的改进，极大地加强了以 GPT 模型所形成的大模型应用生态系统。

尽管 GPT 系列模型取得了巨大的科研进展，这些最前沿的大语言模型仍然存在一定的局限性。例如，GPT 模型可能在某些特定上下文中生成带有事实错误的内容（即幻觉）或存在潜在风险的回应 [35]。更多针对大语言模型局限性的讨论将在本书的第 12 章进行详细讨论。从人工智能的发展历程来看，开发能力更强、更安全的大语言模型是一项长期的研究挑战。为了有效降低使用模型的潜在风险，OpenAI 采用了迭代部署策略 [57]，通过多阶段开发和部署的生命周期来研发模型与产品。

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）。

10.1 基础提示

因为大语言模型的微调代价较高，基于自然语言的提示方法已经成为了使用大语言模型解决下游任务的主要途径。由于提示的质量在很大程度上会影响大语言模型在特定任务中的表现，因此一系列工作深入研究了通过人工设计或自动优化的方法来生成合适的任务提示。本节将对这两种提示方法进行详细的介绍。

10.1.1 人工提示设计

针对特定任务设计合适的任务提示，这一过程被称为“提示工程”（Prompt Engineering）。在本节中，我们将首先介绍构成提示的关键要素，随后介绍人工设计提示的重要基本原则，并提供一些相关的实用建议和案例。读者可以进一步参考相关论文 [263] 和网站¹，以获得更为全面的提示设计建议。

关键要素

一般而言，针对大语言模型的提示设计需要考虑四个关键要素，即任务描述、输入数据、上下文信息和提示策略。下面将对这四个关键要素进行具体介绍。

- 任务描述. 任务描述部分展示了大语言模型应当遵循的具体指令。一般来说，用户应该使用清晰的、具体的表达来描述任务目标。进一步，某些特定任务还需

¹<https://platform.openai.com/docs/guides/gpt-best-practices>

要对于输入或输出的格式进行更详细的说明，可以使用关键词或者特殊符号来强调特殊设置以指导大语言模型更好地完成任务。

知识问答的任务描述：请使用所提供的以三个井号（###）分隔的文章回答问题。如果在文章中找不到答案，请回答“无法找到答案。”

代码补全的任务描述：你是一名程序员。给你一个代码片段，你的目标是完成这段代码，确保它能实现描述的功能。

对话推荐的任务描述：推荐 10 个符合用户偏好的商品。推荐列表可以包含对话框之前提到的商品。推荐列表的格式为：商品 ID 标题（年份）。请勿在推荐列表中提及商品标题以外的任何内容。

例 10.1 提示设计中任务描述的例子

- 输入数据. 通常情况下，用户可以直接使用自然语言描述输入数据的内容。对于特殊形式的输入数据，则需要采用合适的方法使其能够被大语言模型读取与理解。例如，对于结构化数据（如知识图谱、表格等），通常使用线性化方法将其转换为易于处理的文本序列 [264]。此外，由于结构化数据具有较好的组织形式，可以使用编程代码中的数据结构进行存储表示，将结构化数据中的属性（例如图上的节点）表示为数据结构中的变量。基于代码表示的结构化数据可以使用外部工具（如程序执行器）进行准确地读取。

线性化后的表格数据：

学号 姓名 成绩 # 01278 张三 89.0 # 03813 李四 92.5 # 06714 王五 79.0

代码格式的图数据：

```
Graph[name="Knowledge-Graph"]{
  entity_list=["James Cameron", "Ontario", ...],
  triple_list=[("James Cameron" -> "Ontario")[relation="born in"], ...],
}
```

例 10.2 提示设计中输入数据的例子

- 上下文信息. 除了任务描述和输入数据外，上下文信息对某些特定任务也非常重要。例如，搜索引擎可以为开放问答任务提供参考文档，可以通过将检索到的参考文档以上下文信息的形式引入提示作为大语言模型的输入。在引入外部信息

时，需要对于这些信息进行合适的格式化，以加强大语言模型对它们的利用。此外，上下文学习（参考第 10.2 节）中的任务示例数据也有助于提升大语言模型处理复杂任务的能力，大模型可以通过这些示例数据学习任务目标、输出格式以及输入和输出之间的映射关系。

基于任务示例回答问题：

示例问题：罗杰有 5 个网球，他又买了 2 罐网球，每罐有 3 个网球。他现在有多少个网球？

示例回答：罗杰一开始有 5 个球，2 罐每罐 3 个网球就是 6 个网球。 $5 + 6 = 11$ 。答案是 11。

用户输入：袋子里有 16 个球，其中一半是高尔夫球，并且所有高尔夫球中有一半是蓝色的球。请问有多少个蓝色高尔夫球？

模型输出：袋子里有 16 个球，其中一半是高尔夫球。因此有 $16 / 2 = 8$ 个高尔夫球。一半的高尔夫球是蓝色的。因此有 $8 / 2 = 4$ 个蓝色高尔夫球。答案是 4。

例 10.3 提示设计中上下文信息的例子（灰色文字表示模型生成结果）

- **提示策略.** 针对不同的大语言模型设计合适的提示策略对于激发模型解决特定任务的能力非常重要。在某些情况下，添加特定的前缀或后缀有助于引导大语言模型解决复杂任务。例如，使用前缀“让我们一步一步地思考”可以激发大语言模型的逐步推理能力，而使用前缀“你是这项任务（或这个领域）的专家”可以提高大语言模型在某些特定任务（或领域）中的表现。此外，对于对话式的大语言模型（例如 ChatGPT），由于其使用了大量对话数据进行训练，因此更合适的做法是将提示拆分为多个子任务提示，以多轮对话的方法逐步输入给大语言模型。

解答数学问题：

用户输入：袋子里有 16 个球，其中一半是高尔夫球，并且所有高尔夫球中有一半是蓝色的球。请问有多少个蓝色高尔夫球？

模型输出：让我们一步一步地思考。

袋子里一共有 16 个球，其中一半是高尔夫球，也就是说，一共有 8 个高尔夫球。所有高尔夫球中有一半是蓝色的，也就是说有 4 个蓝色的高尔夫球。

推荐旅游景点：

用户输入：你是一名旅行向导。我会把我的位置告诉你，你需要向我推荐附近的旅游景点。我的第一个请求是“我在伊斯坦布尔，我只想参观博物馆”。

模型输出：当然，以下是一些您可以考虑参观的博物馆：

1. 伊斯坦布尔考古博物馆：这个博物馆位于苏丹艾哈迈德广场，是土耳其最重要的考古博物馆之一。它展示了来自不同历史时期的珍贵文物，包括古代近东、古希腊和古罗马的艺术品。
2. 伊斯坦布尔现代艺术博物馆：位于博斯普鲁斯海峡畔的卡拉库埃斯梅斯地区，这个博物馆致力于展示当代土耳其和国际艺术。它拥有丰富的永久收藏和临时展览，涵盖绘画、雕塑、摄影、视频艺术和装置艺术等多个领域。
3. 圣索菲亚博物馆：这座博物馆位于伊斯坦布尔历史中心的圣索菲亚大教堂内，展示了拜占庭和奥斯曼帝国时期的艺术和文物。您可以在这里欣赏到壮丽的拜占庭建筑和壁画，以及众多的历史文物。

例 10.4 提示设计中提示策略的例子**设计原则**

基于上述四个提示的关键要素，我们进一步总结了四项提示设计的基本原则，整体上有助于设计更为有效的任务提示。

- 清晰地表达任务目标. 模棱两可的任务描述很可能导致模型产生不准确甚至错误的回答。因此，在使用大语言模型时需要给出清晰明确的指令。具体来说，一个清晰详尽的任务描述中应当包含任务的各种要素信息，如任务目标、输入/输出数据（例如，“给定一份长文档，我希望你生成一份简明摘要”）和回复限制（例如，“摘要长度不能超过 50 个单词”）。通过提供清晰的任务描述，大语言模型可以更为有效地理解给定任务并生成所需的输出结果。

建议 (“清晰地表达任务目标” 原则的一些实用建议和样例)

- Make your prompt **as detailed as possible**, e.g., “Summarize the article into a short paragraph within 50 words. The major storyline and conclusion should be included, and the unimportant details can be omitted.”
- It is helpful to let the LLM know that it is **an expert with a prefixed prompt**, e.g., “You are a sophisticated expert in the domain of compute science.”
- Tell the model **more what it should do**, but not what it should not do.
- To avoid the LLM to generate too long output, you can just use the prompt: “Question: Short Answer: ”. Besides, you can also use the following suffixes, “in a or a few words”, “in one of two sentences”.
- If you want LLMs to provide the score for a text, it is necessary to provide a **detailed description about the scoring standard** with examples as reference.
- For few-shot chain-of-thought prompting, you can also use the prompt “Let’s think step-by-step”, and the few-shot examples should be **separated by “\n”** instead of full stop.
- The prompt should be **self-contained**, and better not include pronouns (e.g., it and they) in the context.
- When using LLMs for **comparing** two or more examples, the order affects the performance a lot.
- Before the prompt, **assigning a role for the LLM** is useful to help it better fulfill the following task instruction, e.g., “I want you to act as a lawyer”.
- For multi-choice questions, it is useful to **constrain the output space** of the LLM. You can use a more detailed explanation or just imposing constraints on the logits.
- For sorting based tasks (e.g., recommendation), instead of directly outputting the complete text of each item after sorting, one can **assign indicators** (e.g., ABCD) to the unsorted items and instruct the LLMs to directly output the sorted indicators.

● 分解为简单且详细的子任务. 该原则的目标是将一个复杂任务分解为若干个相对独立但又相互关联的子任务, 每个子任务都对应原始任务的某个方面或步骤。特别地, 我们可以显式地将子任务按编号列出 (例如, “通过依次执行以下任务形成一段连贯的叙述: 1. ...; 2. ...; 3. ...”)。这种策略有助于减少复杂任务的解决难度: 通过将复杂任务分解为若干个子任务并按照一定的顺序处理这些子任务, 模型能够逐步获得最终的答案。

建议（“分解为简单且详细的子任务”原则的一些实用建议和样例）

- For complex tasks, you can clearly describe the required intermediate steps to accomplish it, e.g., “Please answer the question step by step as: Step 1 - Decompose the question into several sub-questions, . . .”
- When LLMs generate text according to some context (e.g., making recommendations according to purchase history), instructing them with the explanation about the generated result conditioned on context is helpful to improve the quality of the generated text.
- An approach similar to tree-of-thoughts but can be done in one prompt: e.g., *Imagine three different experts are answering this question. All experts will write down one step of their thinking, then share it with the group of experts. Then all experts will go on to the next step, etc. If any expert realizes they’re wrong at any point then they leave. The question is*
- As a symbol sequence can typically be divided into multiple segments (e.g., $i_1, i_2, i_3 \rightarrow i_1, i_2$ and i_2, i_3), the preceding ones can be used as in-context exemplars to guide LLMs to predict the subsequent ones, meanwhile providing historical information.
- Let the LLM check its outputs before draw the conclusion, e.g., “Check whether the above solution is correct or not.”

• 提供少样本示例. 正如第 10.2 节所介绍的上下文学习方法, 在提示中加入少量目标任务的输入输出作为任务示例 (即少样本示例), 可以提升大语言模型解决复杂任务的能力。少样本示例有助于大语言模型在无需调整参数的前提下学习输入与输出之间的语义映射关系。在实践中, 我们可以根据目标任务专门为大语言模型设计若干高质量的示例, 这能够显著提升任务表现。

建议（“提供少样本示例”原则的一些实用建议和样例）

- Well-formatted in-context exemplars are very useful, especially for producing the outputs with complex formats.
- For few-shot chain-of-thought prompting, you can also use the prompt “Let’s think step-by-step”, and the few-shot examples should be separated by “\n” instead of full stop.
- You can also retrieve similar examples in context to supply the useful task-specific knowledge for LLMs. To retrieve more relevant examples, it is useful to first obtain the answer of the question, and then concatenate it with the question for retrieval.
- The diversity of the in-context exemplars within the prompt is also useful. If it is not easy to obtain diverse questions, you can also seek to keep the diversity of the

solutions for the questions.

- When using chat-based LLMs, you can decompose in-context exemplars into multi-turn messages, to better match the human-chatbot conversation format. Similarly, you can also decompose the reasoning process of an exemplars into multi-turn conversation.
- Complex and informative in-context exemplars can help LLMs answer complex questions.
- As a symbol sequence can typically be divided into multiple segments (e.g., $i_1, i_2, i_3 \rightarrow i_1, i_2$ and i_2, i_3), the preceding ones can be used as in-context exemplars to guide LLMs to predict the subsequent ones, meanwhile providing historical information.
- Order matters for in-context exemplars and prompts components. For very long input data, the position of the question (first or last) may also affect the performance.
- If you can not obtain the in-context exemplars from existing datasets, an alternative way is to use the zero-shot generated ones from the LLM itself.

• 采用模型友好的提示格式。大语言模型采用专门构建的数据集进行预训练，因此可以从数据集中学习到大量的语言表达模式，发现并利用这些语言表达模式可以帮助我们更有效地使用大语言模型完成特定任务。对于提示中需要重点强调的部分，OpenAI 官方文档中建议用户可以使用特殊符号（例如 ###、三引号 “”””和””””、XML 标签等）进行分隔，从而让大语言模型更好地理解相关内容。此外，大多数现有的大语言模型主要在英语文本上进行训练，理解英语指令的能力更强，因此在执行任务时使用英语指令可能会获得更好的执行效果。对于非英语用户来说，通过机器翻译工具将非英语任务指令转换为英语指令再输入给大语言模型，可能会是一个更有效的策略。

建议（“采用模型友好的提示格式”原则的一些实用建议和样例）

- For the question required factual knowledge, it is useful to first retrieve relevant documents via the search engine, and then concatenate them into the prompt as reference.
- To highlight some important parts in your prompt, please use special marks, e.g., *quotation (""")* and *line break (\n)*. You can also use both of them for emphasizing, e.g., *“### Complete sqlite SQL query only and with no explanation.\n\n### Sqlite SQL tables, with their properties: \n\n{table}\n\n{foreign_key}\n\n{question}\n\nSELECT”*.
- You can also retrieve similar examples in context to supply the useful task-specific

knowledge for LLMs. To retrieve more relevant examples, it is useful to **first obtain the answer** of the question, and then concatenate it with the question for retrieval.

- If the LLM can not well solve the task, you can **seek help from external tools** by prompting the LLM to manipulate them. In this way, the tools should be encapsulated into callable APIs with detailed description about their functions, to better guide the LLM to utilize the tools.
- OpenAI models can perform a task better in English than other languages. Thus, it is useful to first **translate the input into English** and then feed it to LLMs.
- For mathematical reasoning tasks, it is more effective to design specific prompts based on the format of programming language, e.g., “Let’s use python to solve math problems. Here are three examples how to do it,\n Q: Olivia has \$23. She bought five bagels for \$3 each. How much money does she have left?\n “def solution():\n """Olivia has \$23. She bought five bagels for \$3 each. How much money does she have left?""" \n money_initial = 23\n bagels = 5\n bagel_cost = 3\n money_spent = bagels * bagel_cost\n money_left = money_initial - money_spent\n result = money_left\n return result”\n \n How about this question?\n Q:”.

10.1.2 自动提示优化

人工设计提示虽然比较直接，但是需要耗费较多的人工成本，同时要求设计人员具有丰富的提示工程经验。此外，大语言模型对于提示设计比较敏感，人工设计的提示有时很难获得最优的效果，还可能导致任务性能的下降。因此，本小节将针对离散和连续这两种主要的提示形式，详细介绍提示的自动优化方法。需要注意的是，由于大语言模型参数量巨大，并且很多工作机制已经与传统预训练模型有着较大的差异，许多提示优化方法已经不再适用于大语言模型。然而，为了内容的完整性，本书仍然将这部分内容进行了收录。

离散提示优化

离散提示通常是由一系列自然语言词元组成的完整句子表达（如“请根据提供的检索信息回答下列问题”）。然而，在离散的词元空间中进行组合搜索，不仅时间复杂度高，而且可能导致非最优的搜索结果。下面将介绍四种常见的离散提示优化方法，能够提升离散任务提示的有效性与搜索效率。

- 基于梯度的方法. 这类方法通过梯度更新技术以最大化模型的似然分数来优化离散提示的搜索过程。一种早期的代表性方法 [265] 使用梯度引导技术，首先将

提示初始化为一系列 “[MASK]” 标记，然后迭代地将提示中的词元替换为词典中的其他词元，通过词元替换产生的对数似然变化来近似估计梯度，进而为提示的每个位置贪心搜索出最佳的词元。由于该方法对提示的每个位置都进行所有候选词元的替换和梯度评估，因此需要模型进行多次前向和后向计算，导致搜索过程的效率较低。为了改进搜索效率，可以将离散词元转化为连续嵌入表示（又称为“软词元”），使用梯度直接对连续嵌入参数进行优化，最后将每个连续嵌入映射为词典中最邻近的离散词元。

- 基于强化学习的方法. 为了实现更有效的离散词元选择，另一种解决方法是将离散提示优化问题转换为强化学习问题，并使用强化学习算法进行求解。具体来说，可以将预训练语言模型作为强化学习中的策略网络并依次生成提示中的词元。在提示生成结束之后，策略网络可以获得任务特定的奖励信号，该奖励信号可通过强化学习算法用于策略网络参数的训练。在实践中，可以设计不同类型的奖励信号，比如真实标签与基于提示的预测标签是否一致、生成文本与给定条件的匹配程度。在最后的测试阶段，基于训练好的策略网络，可以采用贪心搜索策略来生成任务提示中的每个词元。

- 基于编辑的方法. 这类方法主要关注如何通过编辑现有的提示来改进模型的性能，通常是基于模型在目标任务上的表现来判断提示的好坏。它特别适用于无法直接访问模型内部状态（如梯度）的情况，例如只能通过 API 调用的模型。在这类方法中，通常需要事先定义好编辑操作，然后迭代地对提示进行修改，直至达到最大迭代轮次或者模型最佳性能。根据第 10.1.1 节的介绍，提示的关键要素包括任务描述、输入数据、上下文信息和提示策略。因此，常用的提示编辑操作有修改任务描述、添加或删除上下文任务示例、调整输入到输出的标签映射器（例如可以使用 “positive/negative” 或者 “正/负” 表示二分类）等。此外，提示编辑操作也可以根据不同的场景或者需求进行设计，以适配下游具体任务。整体流程可以概述如下：基于预定义的编辑操作，在现有提示的基础上修改得到新提示，并输入至模型得到目标任务上的表现，根据表现筛选出合适的提示。由于上述过程可能需要迭代进行，可以只选择少量测试样例来评估模型表现，以减少计算开销。

- 基于大语言模型的方法. 由于大语言模型具有通用的任务能力，因此可以将提示优化看作一个待求解的任务，进而直接使用大语言模型作为提示生成器来生成或改进提示 [266, 267]。基于大语言模型的自动提示生成框架将提示优化过程看作是一个由大语言模型指导的黑盒优化问题。该框架首先利用提示生成模型（用

于生成提示指令的大语言模型) 基于少量上下文示例生成一批候选的任务指令。随后, 使用“目标模型”(用于下游测试的大语言模型) 对这些候选指令在目标任务上的表现进行逐一评估。在评估过程中, 可以采用模型困惑度或任务准确率作为衡量指令质量的指标。上述过程可以通过基于蒙特卡洛搜索的多轮优化策略进行扩展。在每一轮迭代中, 根据模型表现对候选指令进行筛选得到高分指令, 并利用大语言模型生成与高分指令相似的新指令, 从而扩展候选指令集。迭代完成后, 选择模型表现最佳的候选指令作为最终使用的提示。然而, 上述方法没有充分考虑提示的整个历史改进轨迹, 因此可能在提示搜索过程中陷入局部最优或者产生效果震荡, 无法生成更好的提示。为了解决这一问题, 可以将所有改进的历史提示及其分数纳入提示优化过程, 以指导大语言模型逐步生成更好的新提示。

连续提示优化

与离散提示不同, 连续提示由一组连续空间中的嵌入向量组成, 可以根据下游任务的损失直接通过梯度更新进行优化。值得注意的是, 已有连续提示优化的工作主要是基于预训练语言模型开展的, 由于大语言模型参数量巨大, 连续提示受到的关注较为有限。已有的连续提示优化研究通常依赖于有监督学习方法。当数据稀缺的情况下, 还可以采用迁移学习方法来缓解目标任务标注数据不足的问题。下面将详细介绍这两种方法。

- 监督学习方法. 这类方法将连续提示向量视为可训练的模型参数, 基于下游任务数据, 通过最小化交叉熵损失来优化连续提示。根据第 7.3 节中所讨论的内容, Prefix-tuning [212] 会在语言模型的每个 Transformer 层预置一串前缀 (即一组可训练的连续向量), 而 Prompt-tuning [213] 只会在输入层加入可训练的提示向量。通过固定语言模型的大规模参数而只微调这些连续的提示向量, 可以有效节省训练所需要的参数量。然而, 这些提示优化方法通常与输入无关, 缺乏对于输入语义的充分考虑。

- 迁移学习方法. 有监督学习方法通常需要充足的训练数据来学习最优的任务提示, 很难在数据稀缺场景下获得较好的模型性能。为了解决这个问题, 基于提示的迁移学习方法首先为若干个具有代表性的源任务学习一个所有任务共享的连续提示, 然后使用该提示初始化目标任务的提示, 这可以为下游任务的提示优化提供良好的初始点。然而, 这种方法存在一定的局限性, 它在解决目标任务的所有实例时都使用了相同提示, 而即使是一个精心优化过的提示也未必适合所有的任务实例。为了解决这一问题, 可以为每个源任务独自学习任务特定的连续提示

(而不是所有源任务共享)，在解决目标任务的实例时，可以采用注意力机制等方式学习目标实例与每个源任务提示的相关性权重系数，对若干个源任务的提示向量进行加权组合，将组合后的新提示（为连续向量形式）用于帮助模型解决当前任务实例。

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 $D_k = \{f(x_1, y_1), \dots, f(x_k, y_k)\}$ 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：

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

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

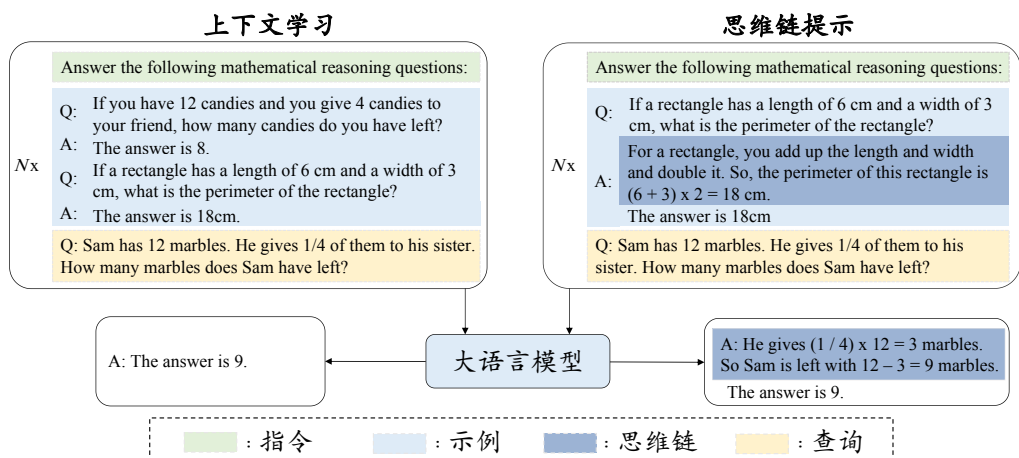


图 10.1 一个关于上下文学习和思维链提示的比较说明（图片来源：[10]）

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]。

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：

输入：罗杰有 5 个网球，他又买了 2 罐网球，每罐有 3 个网球。他现在有多少个网球？

输出：11。

示例模板：问题：{输入} 答案：{输出}

具体示例：问题：罗杰有 5 个网球，他又买了 2 罐网球，每罐有 3 个网球。他现在有多少个网球？答案：11。

(2) 增加任务描述的示例格式：

输入：罗杰有 5 个网球，他又买了 2 罐网球，每罐有 3 个网球。他现在有多少个网球？

输出：11。

示例模板：下面是一个小学数学问题。问题：{输入} 答案：{输出}

具体示例：下面是一个小学数学问题。问题：罗杰有 5 个网球，他又买了 2 罐网球，每罐有 3 个网球。他现在有多少个网球？答案：11。

(3) 增加思维链的示例格式：

输入：罗杰有 5 个网球，他又买了 2 罐网球，每罐有 3 个网球。他现在有多少个网球？

输出：让我们一步一步地思考。罗杰一开始有 5 个球，2 罐每罐 3 个网球就是 6 个网球。 $5 + 6 = 11$ 。因此答案是 11。

示例模板：下面是一个小学数学问题。问题：{输入} 答案：{输出}

具体示例：下面是一个小学数学问题。问题：罗杰有 5 个网球，他又买了 2 罐网球，每罐有 3 个网球。他现在有多少个网球？答案：让我们一步一步地思考。罗杰开始有 5 个球，2 罐每罐 3 个网球就是 6 个网球。 $5 + 6 = 11$ 。因此答案是 11。

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：

示例输入： Sentence: This house is surprisingly not constructed very well, and you probably need more money to fix it after you buy it. If you ask me, I would suggest you to consider other candidates.

示例输出： This house does not seem to be constructed well, so you may need to spend more money to fix it after you purchase it. I would suggest that you look at other properties.

示例指令： Suggest a better and more professional rephrasing of the following sentence.

示例输入：

Application Form:

Name:___ Age:___ Sex:___

示例输出： Name: John Doe. Age: 25. Sex: Male

示例指令： I am looking for a job and I need to fill out an application form. Can you please help me complete it?

示例输入： [10, 92, 2, 5, -4, 92, 5, 101]

示例输出： [-4, 2, 5, 5, 10, 92, 92, 101]

示例指令： Sort the given list ascendingly.

输入： Address: 123 Main Street, City: San Francisco

输出： 94105

指令： Given an address and city, come up with the zip code.

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，dan 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。

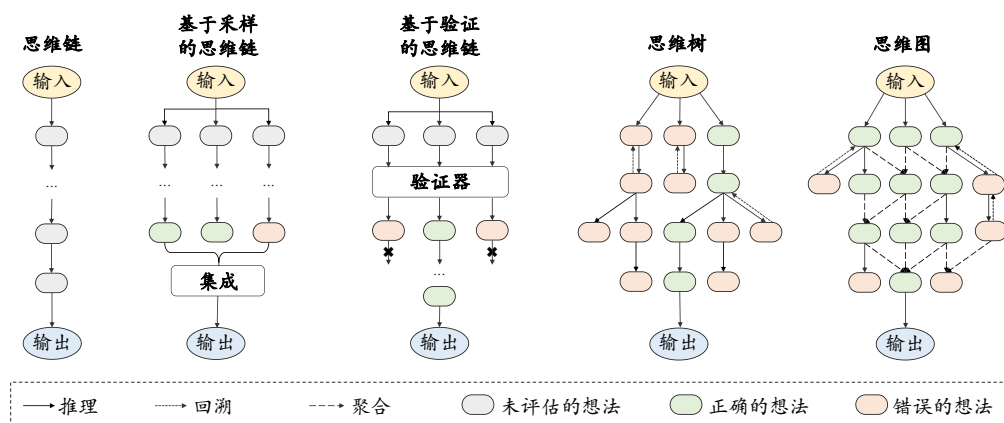


图 10.2 思维链提示技术的演化过程（图片来源：[10]）

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

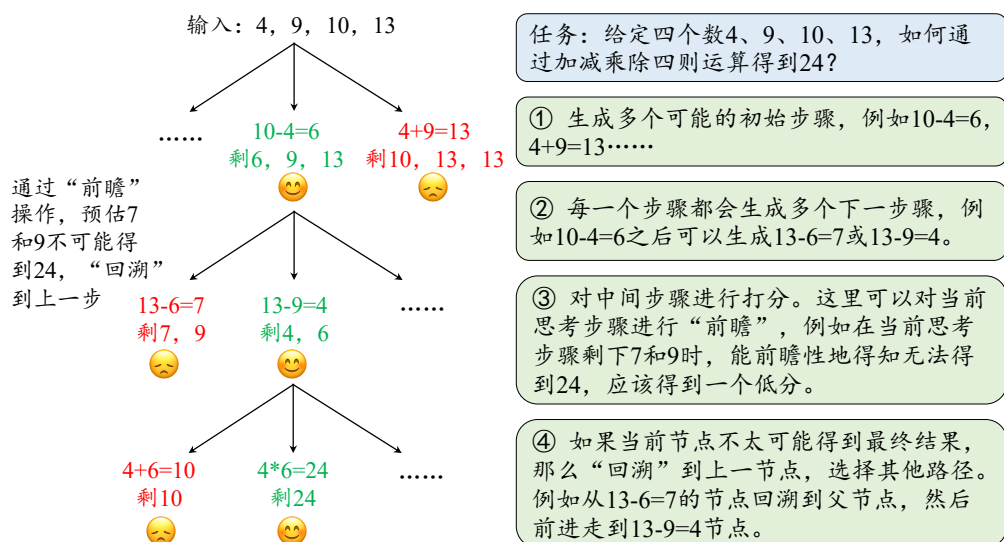


图 10.3 大语言模型使用思维树方法解决 24 点游戏

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。

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。

²给定 4 个数，通过四则运算得到 24

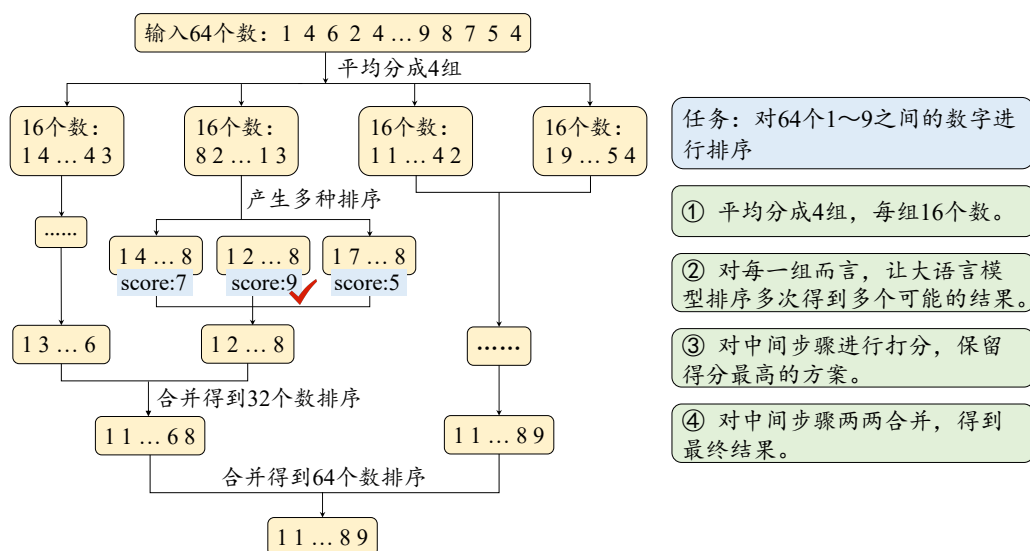


图 10.4 大模型使用思维图方法解决数组排序

● 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。