思维链 (Chain-of-Thought) 推理机制和语言代理模型综述精读

Tongji University

摘要

大预言模型极大地促进了语言制智能领域的发展，我们能够在一系列的语言计算领域看到大语言模型的强大实力。通过理论层面对大模型的逻辑推理能力进行分析，其强大的推理能力得到进一步印证，尤其在一些复杂推理任务上，大模型显现出了惊人的感知和语言能力。大模型利用思维链 (Chain-of-Thought, CoT) 推理技术，成功将推理步骤划分为不同小步骤，极大地提高了模型的推理能力，同时也为推理过程提供可解释性、可控性和极大的灵活性。利用思维链的特性，语言代理模型 (Language Agents) 能够进一步增强其在不同环境下的表现，从而进一步为代理模型的发展提供理论支持。本文将简要介绍索思维链技术基础，并进一步介绍在思维链技术下，语言模型的发展情况与研究方向。

关键词: 思维链，大语言模型，语言代理模型，Chain-of-Thought, Language Agent

1. 引言

语言智能是指通过自然语言表述的概念进行理解和推理的能力。随着语言模型规模的扩大，大型语言模型（LLMs）在追求人类水平的语言智能方面取得了显著进展。推理是语言智能领域的一个重要研究课题，其可以被描述为一个多步骤的过程，在这一过程中，语言模型可以从离散的证据片段中得出推论，最终形成更抽象的概念。这些概念有助于进行高层次的预测。最近的研究表明，与直接生成答案的方式相比，通过引入提示 (prompt) 来逐步进行问题推理，大模型在推理任务上的表现有明显提升。通过引入 prompt 使得大语言模型逐步解决问题的方法称作思维链提示 (CoT) 方法。

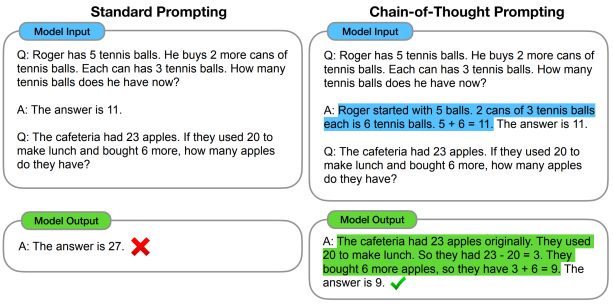

图 1. CoT 通过 prompt 将推理过程划分为不同步骤

CoT 提示技术的优化从模式上推进了大语言模型推理领域的发展，由此引发了 CoT 框架内显著的范式转变。这些转变包括三个关键方面：

- 提示模式 (prompting pattern)：从手动设计情境中的学习演示到自动构建提示
- 推理格式 (reasoning format)：从非结构化自然语言格式到结构化格式
- 应用场景 (application scenario)：从单一语言环境到多语言环境，从一种语言模式到拥抱多模态方法，以及从复杂的推理任务到通用的推理任务到通用任务

CoT 推理是 LLMs 的一种代表性新兴能力，它提供了一种相对成熟且可解释性较强的策略，可将错综复杂的问题分解为更小、更易处理的子问题，并通过循序渐进的方法系统地找到解决方案。利用在

[1]Original paper: https://arxiv.org/abs/2311.11797

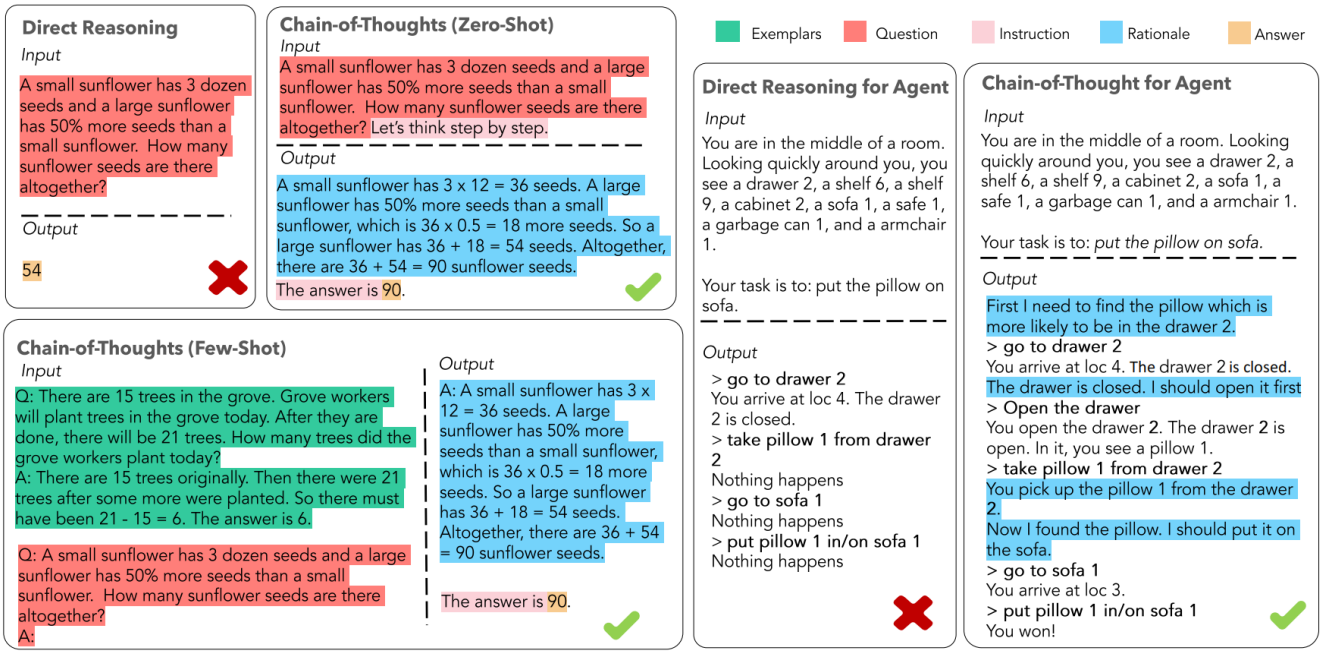

图 2. CoT 指令分类

预训练过程中生成的推理能力，CoT 技术成功使得大模型识别出推理过程中重要的原子命题、逻辑关系，并成功将命题推理连续化，使得各个命题之间能够有因果关系地产生推理链。大语言模型从而构建了推理过程步骤，完成推理任务、在解决原子命题的过程中，语言模型同时可以通过知识的检索和验证等手段进一步加强推理过程。

由此在 CoT 模型训练和推理时，语言模型已经被扩展为具有感知、记忆和推理能力的综合框架。从概念上来看，其已经具有 Agent 的雏形，即能够对环境进行感知交互，并且具有记忆和推理能力。LLM 驱动的语言代理模型能够在真实世界或模拟环境中熟练地遵从语言指令并执行动作。按照其与环境的交互程度，语言代理模型可以分为两种类型：(1) 自主代理模型 (autonomous agents) (2) 交互代理模型 (communicative agents)

本文将简要介绍 CoT 模型的内在机制，分析思维链机制的作用方式，包括定义特征和技术优点等。随后将简单介绍大语言模型引入思维链技术后 prompt 技术的应用场景。

2. Chain-of-Thought **机制介绍**

2.1. CoT 模型定义

思维链的概念是指为解决问题或得出答案而产生的一系列中间推理步骤或得出答案的一系列中间推理步骤, 其形式可以描述如下映射关系：

$$< \text{input} \rightarrow \text{reasoning chain (rationale)} \rightarrow \text{output} >$$

这种方法往往比传统的直接推理更有效，比如在传统模型中，标准的分类、多项选择和问题解答问题通常会利用直接推理的方式，即 $<$ input $\rightarrow$ output $>$ 的映射形式。而在思维链模型中，该过程被划分为不同不同思维链上的节点，通过步进推理实现原来的全部推理任务。这些思维链的节点可以被划分为不同种类，如图 2 所示。我们假设推理数据集分布为 D ，用 $s = (x, y) \sim \mathcal{D}$ 表示在 D 上的采样，其中 x 和 y 分别为问题和答案，问题与答案的形式为字符串形式。类似的，用 $|x|$ 表示序列 x 的长度，用 p_θ 表示以 θ 为参数的预训练模型，CoT 模型可定义为：

指令 (instruction). 指令通常是简短的句子，用于提示 LLM 按所需格式生成答案。其能够引导 LLM 在推理过程中逐步思考，最终得出推理结论。

演绎推理 (rationale). 我们将 CoT 中间的推理步骤称为演绎推理 (rationale)，其包括解决方案、推理步骤和其他先验知识等。将演绎推理的过程定义为 r，若 r 是由 LLMs 生成的，则 r 可以定义为 $r \sim p_\theta(x, p)$；若 r 是人工推导的，则我们无需 p 参与推理过程，可以直接使用 $r = f(x)$ 获取 r，其中 $f(\cdot)$ 表示人工推导操作。

示例 (Exemplars). 在 few-shot 场景下的 prompt 方法中，示例是在输入-输出模式下的期望输出。每一个示例都包含问题、演绎过程和对应答案。在测试预测之前，示例可作为输入输出关系的情境演示。在将示例输入模型前，通常情况下要将示例拼接到一起。具体而言，假设示例的大小为 n，示例 E 可以表述为:

$$E = [(x_1, r_1, y_1) \circ \cdots (x_{n-1}, r_{n-1}, y_{n-1} \circ (x_n, r_n, y_n))]$$

其中 $\circ$ 表示连接操作，$(x_i, y_i) \sim \mathcal{D}$，且 $r_i = f(x_i)$ 。

Zero-shot **思维链**. Zero-Shot-CoT 不要求用户提供范例。它通常依靠指令来帮助 LLM 进行逐步推理，从而进一步答案。研究人员首先使用指令 p_1（如" 让我们一步一步地思考"）诱导 LLM 生成理由 r，然后使用指令 $p2$（如" 让我们一步一步地思考"）诱导 LLM 根据问题和理由得出最终答案。从形式上看，输出 y 的计算过程如下:

$$r \sim \prod_{i=1}^{|r|} p_\theta(r_i | x, p_1, r_{<i}), \quad y \sim \prod_{i=1}^{|y|} p_\theta(y_i | x, p_1, r, p_2, y_{<i})$$

Few-shot **思维链**. Few-Shot-CoT 包括提供一系列带有相关理由的范例。这些示例与问题串联起来，促使 LLM 生成理由和答案。在这种情况下，输出 y 以端到端模式获得，可表述为:

$$y \sim \prod_{i=1}^{|y|} p_\theta(y_i | E, x, y_{<i})$$

2.2. CoT 模型优势

更好的推理表现. 通过将复杂的多步骤问题分解为中间阶段，CoT 最大限度地降低了忽略关键细节的风险。细节。它还能确保将额外的计算资源有效地分配给需要更多推理步骤的问题。大量研究已经确证了 CoT 在广泛领域中的有效性。其中包括算术推理、常识推理和符号推理。

更好的可解释性. CoT 将复杂的推理任务分解成一连串相互关联的决策步骤，由此人类更容易地理解 LLM 所做的决定或结论背后的基本逻辑和推理。它揭示了大语言模型得出特定答案的具体步骤，这为模型调试和找出推理过程中可能偏差的部分提供了宝贵的见解。但是当前完全描述模型的计算过程仍然是一个有待解决的难题。

更好的可控性. 通过促使 LLM 输出一连串相互关联的想法，用户可以对 LLM 的认知过程施加更大的影响。许多研究都致力于识别和纠正推理路径可能偏离轨道或需要额外信息的特定思维单元。这种可控性的提高使得回答问题更加审慎和准确。

更好的灵活性. 只需在 Zero-Shot-CoT 的输入问题末尾添加指令，或在 Few-Shot-CoT 的输入问题中加入 CoT 示例，就能轻松地在足够大的现成 LLM 中使用 CoT 推理。CoT 的灵活性超越了推理任务的范畴，使其适用于许多领域，包括经典的自然语言处理 (NLP)、科学应用和基于代理的系统等。

3. Chain-of-Thought **机理分析**

3.1. CoT 适用场景

从工程的角度来看，CoT 推理在以下三个条件下有帮助: i 使用 LLM; ii 任务具有挑战性，需要多步骤推理; iii 直接提示的性能不会随着模型规模的扩大而显著提高。有进一步证据表明，在去噪函数上预先训练的 200 亿个参数的 LLM 也能实现有效的 CoT 推理。否则 CoT 在使用较小的 LLM 时往往会陷入困境。由于缺乏 LLM 的支持性知识和

推理能力较差，可能会导致模型出现幻觉。CoT 推理在简单步骤的任务中也不太有效，如匹配、序列标记和单项选择题等。

从理论角度看，当训练数据（可能被视为 LLM 中的参数知识）由局部变量集群组成，而这些变量集群具有很强的局部性时，CoT 推理会很有帮助。这一发现意味着，LLM 必须具备与任务相关的知识，才能支持 CoT 推理，我们称此类知识为原子知识。CoT 推理通常是通过情境学习（ICL）激发的，例如 Zero-shot CoT 和 few-shot CoT。研究表明，CoT 推理在不同范例的提示下效果不同，与查询相关的理由和正确的推理步骤排序是 CoT 提示有效的关键。除了提示之外，在训练语料中引入推理材料和必要的知识也能显著提高 CoT 推理能力。研究发现，用代码数据进行预训练有利于有效的 CoT 推理。在相同的 LLMs 中，CoT 推理能力可以通过预训练的方式得到提高，同时在 LLMs 中，CoT 推理能力可以得到在预训练阶段必要的诱导。研究表明我们可以提高同一 LLM 中的 CoT 推理能力，或者在更小的模型中诱导 CoT 推理能力。

3.2. CoT 适用原因

最近的研究采用了实证和理论两种方法，以努力理解协同工作取得成效的根本原因。实证方面，Wei[1] 等人认为，CoT 推理的成功是一个多层面的现象，可能涉及各种新兴能力。可能涉及到各种新出现的能力。这些能力包括语义理解、符号映射、主题连贯性、运算能力和忠实性。有趣的是，Zhang 等 [2] 发现，示例理由中的错误并不会导致显著的成绩下降。理由并不会导致成绩明显下降。Wang 等人 [3] 报告了类似的观察结果，即 LLMs 可以生成连贯的推理步骤，并取得 80-90% 以上的成绩，尽管在示例中会提示无效的推理步骤。推理步骤。这些发现意味着，经过前训练后，LLMs 已经具备了与生俱来的推理能力。CoT 提示指定了一种输出格式，可规范化模型生成，使其逐步生成。模型生成的输出格式，使其在按顺序并与查询相关的情况下逐步生成（Wang 等 [4]）。换句话说，CoT 技术有助于迫使模型进行推理，而不是教它如何完成推理（Zhang 等 [5]）。

从理论上讲，贝叶斯推理是一种从理论角度研究 CoT 为何有效的流行方法在 LLM 的背景下，该证明可以解释为，LLM 中的参数知识包括与目标问题相关的知识片段，以及与目标问题相关的知识片段之间具有很强的相互联系。为了验证这一结论，Bi 等人对代码数据进行了实证研究，发现数据的局部结构属性对改进 CoT 数据的局部结构属性对提高 CoT 推理能力至关重要。Prystawski & Goodman 和 Bi[6] 等人的研究结果表明，CoT 可以帮助识别用于推理的知识原子片段，并在知识原子片段之间架起一座关系桥梁。同样，wang 使用知识图谱进行分析，发现将已知事实组织成” 链”（即 CoT）可显著影响推理的有效性。能极大地影响推理的有效性。LLMs 能够从已知事实中准确推导出以前未见过的事实，从而回答给定的问题。LLMs 就能从已知事实中准确推导出先前未见的事实，从而回答给定的查询，而无需明确编码推理规则。

4. Chain-of-Thought 范式转变

截至 2023 年 10 月，CoT 在七个最具代表性的推理任务上的表现总结如图 4 所示。这七个推理任务涵盖了不同的类别，包括：(i) 算术推理：GSM8K (Cobbe 等 [7])，AQuA (Ling 等 [8]) 和 SVAMP (Patel 等 [9])；（ii）常识推理：CSQA（Talmor 等 [10]）和 Strategy QA（Geva 等 [11]）；（iii）符号推理：Last Letter Concatenation）和 Coin Flip

在复杂推理任务的基准性能方面，CoT 对语言模型在所有七个任务中的推理能力产生了显著影响，推进迅速。值得注意的是，除了常识推理外，相对简单的 CoT 格式 Manual-CoT 在算术和符号推理中与直接提示相比，显著提高了整体准确性。关于常识推理任务，Manual-CoT 和 Zero-Shot-CoT 都观察到了 CoT 在性能上的不足。然而，当 CoT 与更大规模的 PaLM（540B）模型集成时，它在常识推理方面持续增强。值得注意的是，Zero-Shot-CoT 还指出，通过 CoT 生成的理由通常在逻辑上正确，或者仅包含人类可理解的错误。这表明，即使任务指标没有明确衡量，CoT 也鼓励改进常识推理。虽然 Manual-CoT 未能在常识推理中取得性能提升，但

图 3. CoT 在不同 task 上的表现

推理技术的优化，如自一致性答案聚合和自动示范构建（Shum 等 [12]），揭示了 CoT 在总体上取得显著结果的潜力。最新的 CoT 推理技术在整个推理过程的各个层面都得到了加强，例如多模态感知、自动提示、推理验证和基于一致性的采样。

4.1. Prompt 范式

提示模式主要可分为两个组成部分：指令生成和范例生成。指令生成主要侧重于找到最佳的指令来提示语言模型，使其能够进行逐步推理，而不是直接回答问题。这种方法主要旨在最大化语言模型的 zero-shot 能力。范例生成主要侧重于为 Few-Shot-CoT 找到最佳的输入-输出演示范例对。这些范例用于与测试输入一起提示语言模型，使模型能够预测相应的输出。

4.1.1 指令生成

指令生成可以根据它们各自的生成过程分为两种明显的方法：手动指令生成和自动指令生成。

早期的尝试主要涉及手动构建指令提示。最早和最传统的指令生成方法是由 Kojima 等人提出的 Zero-Shot-CoT。Zero-Shot-CoT 表明，通过在每个答案之前添加一个简单的提示，例如“让我们逐步思考”，大型语言模型（LLMs）可以进行零射击推理。Zero-Shot-CoT 在各种推理任务上的零射击 LLM 性能表现优于其他 zero-shot 方法，而无需手工制作的 Few-Shot 例子，标志着 Zero-Shot-CoT 的新时代的开始。

Wang 等人 [12] 进一步提出了 Plan-and-Solve（PS）提示，以解决 Zero-Shot-CoT 推理中的缺失步骤错误。它包括制定一个计划，将任务分解为较小的子任务，并根据计划执行这些子任务。PS 提示包含两个阶段。在第一阶段，作者使用提出的提示模板“让我们首先了解问题并制定解决问题的计划。然后，让我们按计划逐步解决问题”来提示 LLM，以生成推理过程和答案。第二阶段使用答案提示（“因此，答案（阿拉伯数字）是”）提取答案。

然而，手动设计指令可能并不总是产生期望的结果，用户通常需要尝试各种提示以实现所需的行为。为应对这一挑战，Zhou 等人 [13]）提出了 Automatic Prompt Engineer（APE），这是一种专为 LLMs 自动生成和选择指令的方法。APE 将指令生成视为一种自然语言程序合成，并通过搜索由 LLM 提出的指令候选池来优化这个过程。其主要目标是最大化选

择的得分函数。具体而言，APE 通过指导 LLM 使用手动制作的模板生成一组候选指令来启动该过程。最后它利用 LLM 根据输入-输出示例推断出得分最高的最可能指令。利用 LLM 的能力，APE 简化了提示工程过程，减轻了大量人为干预，并生成了高质量的指令。

Yang 等人提出了一种简单而高效的方法，即通过 PROmpting 进行优化（OPRO），它充分利用了语言模型（LLM）的优化能力。OPRO 在优化中采用了一种开创性的方法，充分发挥了 LLMs 的潜力。OPRO 通过呈现优化问题和优化轨迹的自然语言描述来启动优化过程。这个轨迹包括先前的解决方案以及它们关联的优化分数。随后，制定和评估更新的解决方案，以它们的性能和质量。在经过彻底检查后，用于随后优化步骤的提示包含了这些解决方案。随着迭代过程的展开，解决方案经历渐进性的细化，最终提高了它们的质量。

最初，OPRO 被应用于解决两个经典的优化问题：线性回归问题和旅行推销员问题。研究证明，由 OPRO 优化的提示在性能上超过了人工设计的提示，特别是在任务如 GSM8K 和 Big-Bench Hard 上。OPRO 展示了它在解决常见优化挑战和通过用自然语言呈现 LLMs 的优化任务来生成和优化解决方案的效率。

4.1.2 示例生成

类似于指令生成，示例生成也可以根据构建范例的方法分为两类：手动范例生成和自动范例生成。

Few-Shot-CoT 推理，由 Wei 等人正式探讨，代表了一种离散的提示学习方法，利用多个输入-输出对来提示 LLM 生成推理并得到最终答案。为了提供更清晰的区分，我们将称其为 Manual-CoT。Manual-CoT 遵循传统的手动范例生成方法。与传统的上下文学习不同，在上下文学习中，LLMs 会被提示一组输入-输出演示对以及一个测试输入，使模型能够预测输出。Manual-CoT 包括在目标输出之外，通过手动设计额外的逻辑推理过程来提示模型的输出。研究者通过优化范例的选择，进一步发展了 Manual-CoT，并引入了 Active-Prompt，该方法使用用人工设计的推理注释的特定任务示例提示。Active-Prompt 处于手动范例生成和自动范例生成之间。该方法使用基于不确定性的主动学习度量从任务特定的查询池中选择最不确定的问题。Active-Prompt 首先让 LLM 多次回答问题，然后基于不确定性度量（如不一致性、熵、方差等）选择最不确定的问题，手动注释推理，并将问题和推理用作推理的示例。

为了减少手工制作任务特定演示以逐个生成推理链的任务负担，Zhang 等 [13] 人提出了 Auto-CoT，该方法保持了抽样问题的多样性并生成推理链以自动构建演示。具体而言，Auto-CoT 包括两个主要阶段：（i）问题聚类：将给定的问题数据集分成几个簇；（ii）演示抽样：从每个簇中选择一个代表性问题，并使用 Zero-Shot-CoT 生成其推理链。

Shum 等人 [12] 提出了一种称为 Automate-CoT（Automatic Prompt Augmentation and Selection with Chain-of-Thought）的策略，自动化了 CoT 提示的推理链的增强和选择过程。该过程包括三个步骤：增强语言模型以生成多个伪链，根据与地面真相答案的一致性修剪伪链，并使用减小方差的策略梯度策略选择最有帮助的推理链。

4.1.3 推理范式

推理格式的增强主要涵盖三个方面：CoT 公式化、推理聚合和 CoT 验证。CoT 公式化侧重于将顺序的 CoT 转化为各种认知结构，如树状、图状或表格状格式，从而融入结构思维线索。推理聚合主要涉及通过对从 LLM 中采样的结果进行聚合来提高 LLM CoT 推理准确性。CoT 验证主要强调引入验证方法来验证和修正 CoT 推理过程。

4.2. 应用场景

4.2.1 从单一语言到多语言环境

Shi 等人 [13] 将 CoT 扩展到跨足多语境领域，并引入了 Multilingual Grade School Math（MGSM）基准，该基准评估大型语言模型在多语境环境中的推理能力。这个基准包含了 250 个小学数学问题，

已被翻译成十种语言。此外，该作者提出了一个称为“Multi-lingual CoT”的概念，涉及使用多语境示例提示 LLMs，并结合英语中间推理步骤。这种方法已被证明能够产生具有竞争力甚至更好的结果。Multi-lingual CoT 表明，将英语的 chain-of-thought 提示作为基线可能是进行多语境推理研究的一种有价值的策略。

4.2.2 从文字模态到多模态

CoT 中的多模态性可以根据引入多模态元素的位置分为两类：输入多模态和输出多模态，如图 6 所示。Zhang 等人 [14] 首次探索了输入多模态的 CoT，使 CoT 能够超越文本信息，提出了多模态 CoT（MM-CoT）。MM-CoT 不是提示 LLMs，而是专注于微调。MM-CoT 将语言（文本）和视觉（图像）模态纳入两阶段框架：理由生成和答案推理。MM-CoT 通过使用门控融合机制微调较小的 LLMs 并整合语言和视觉模态。该方法的结果表明，整合视觉信息可以增强 LLM 生成推理路径的能力，缓解虚构挑战，从而提高性能。基于 Zhang 等人，Yao 等人首次将图结构引入输入多模态 CoT，并提出了一个两阶段流水线，即思考输入图（GoT-Input）。与 GoT-Rationale（Besta 等人，2023）将思考生成过程建模为图结构不同，GoT-Input 则侧重于模型化从 CoT 理由中派生出的思考图，以增强模型的推理能力。在第一阶段，模型在给定输入问题和利用开放 IE 系统从输入中提取子动词-宾语三元组的思考图的情况下生成理由。在第二阶段，模型在给定问题和生成的理由作为输入以及基于输入文本的新思考图的情况下生成答案。GoT 分别使用不同的编码器对文本、图形和图像（可选）进行增强，并通过使用 GNN 增强演绎推理能力。然后，GoT 使用门控融合方法融合特征以生成最终答案。通过模拟 LLMs 内部人类思维的非序列化特性，GoT 证明增强了 LLMs 的演绎推理能力。

在 Huang 等人 [15] 的研究中，引入了一种能够处理各种模态的多模态语言模型 KOSMOS-1，该模型使用了多模态的 chain-of-thought 提示方法。在初始阶段，当呈现一张图像时，作者使用提示“详细介绍这张图片：”生成图像的详细描述作为理由。紧接着研究者为模型提供了理由和一个任务特定的提示，以生成最终结果。

与上述输入多模态性的 CoT 相反，Visual Chain of Thoughts（VCoT）（Rose 等人，2023）将多模态性引入输出空间。VCoT 通过为视觉元素生成标题并识别多点凝视，以在生成多模态填充时保持输入序列一致性。随后，它采用递归方法生成多模态填充，包括图像和图像标题。通过新颖驱动的递归填充和一致性驱动的视觉增强的组合，实现了这一目标。这些策略用于增强多步推理的可解释性，弥补逻辑差距，最终提高下游任务性能。

5. CoT 技术与语言代理模型的结合

通过上述先进技术的改进，CoT 推理对人工智能（AI）社区产生了更广泛的影响，尤其推动了现实生活中自主代理的发展。构建具有在不同环境中学习和行动能力的智能自主代理是人工智能（AI）的长期目标（Searle, 1969; Wooldridge & Jennings, 1995; Maes, 1995; Hendler, 1999; Wang et al., 2023b; Xi et al., 2023; Zhou et al., 2023d）。鉴于前述技术的迅速发展，CoT 推理方法已被用于感知、记忆和推理，从而使语言代理能够在日益复杂的环境中进行交互。这些能力为开发通过人-代理和代理-代理协作解决复杂任务的自主代理奠定了基础。

与 RL 代理模型的区别. 追求开发智能代理的目标一直是人工智能研究的长期目标。在早期阶段，对代理的研究主要集中在强化学习技术上。强化学习代理通过与环境的迭代交互进行决策训练，通过奖励或惩罚的形式获得反馈—正确的移动会得到奖励，而错误的则会受到惩罚。这个迭代过程旨在最小化错误，最大化准确的决策。强化学习代理具有一个关键特征：通过与环境的持续交互能够自我演化。然而，强化学习代理面临一些局限。它们严重依赖专家数据和为特定任务量身定制的精心设计的奖励函数。因此，它们的有效性通常局限于个别任务，阻碍了它们对新任务或领域的泛化能力。此外，强化学习代理的内部运作通常缺乏透明性和可解释性）。

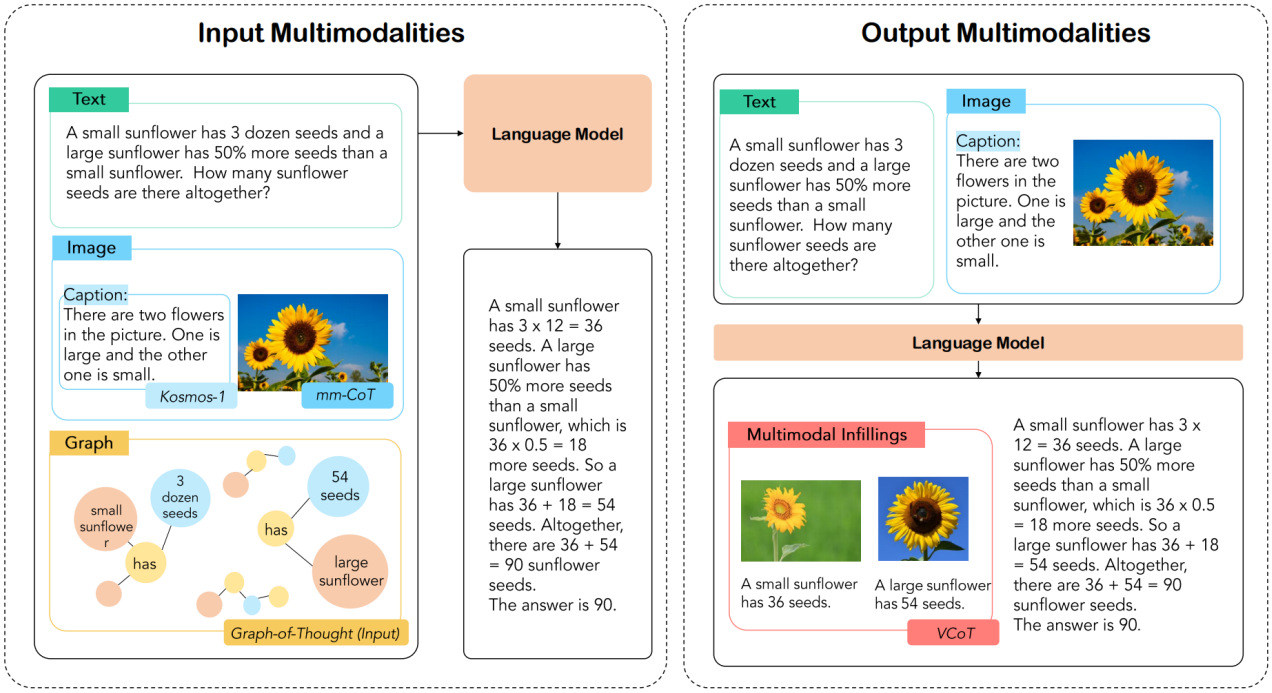

图 4. CoT 与多模态模型的结合

相比之下，语言代理通过利用嵌入在 LLMs 中的常识先验，与强化学习代理有所区别。这些先验减少了对人工注释和试错学习的依赖，使其能够轻松适应新任务或环境，并在 CoT 的帮助下提供更好的可解释性)。然而，语言代理在响应环境变化时演化其参数面临挑战，主要因为它们主要是通过提示或调整 LLMs 的高成本来适应环境。尽管最近关于语言代理的研究，例如 Retroformer（Yao 等人），已经纳入了类似强化学习的策略来增强语言代理的能力，但焦点仍然主要限于语言推理任务。有望看到如何弥合强化学习代理和语言代理之间的差距，以促进未来在复杂环境中能够以强大性能和高可解释性工作的体系结构。

5.1. 语言代理模型主干结构

语言代理可以建立在单模态 LLM 或多模态 LLM 之上。完成任务通常需要多个交互步骤。整个过程称为一个 episode，由一系列 turns 组成。为了完成任务，代理需要在每个 episode 的每个 turn 中提前规划、做决策和执行动作。规划、决策和动作执行的过程可能反映了 LLMs 的推理能力，因为 LLMs 在真实世界或虚拟环境中暴露于在 LLMs 预训练期间不存在的情境。在这样的环境中，LLM 必须感知世界的知识并采取行动，而 CoT 在这些情况下有助于弥合环境感知与 LLMs 内在能力之间的差距。

5.2. CoT 感知模型

逐步提示代理以链式感知方式解释感知步骤已被证明能够提高行动成功率。这种方法增强了对环境或上下文的理解。值得注意的是，Rawles 等人发现使用 CoT 模板“答案：让我们一步一步思考。我看到 < 屏幕截图 >，我需要...”显著提高了行动预测的准确性。如图 1 所示的例子，感知的 CoT 提示可以是“让我们一步一步思考。我在 Google 应用中看到了无关的搜索结果”。此外，Zhang 等人 (2023d) 和 Huang 等人 (2023b) 利用外部工具获取图像标题作为辅助输入，帮助提高多模态环境的感知。这些标题被放置在 < 屏幕截图 > 中以组织输入提示。

除了感知的单向解释外，在多轮交互的上下文

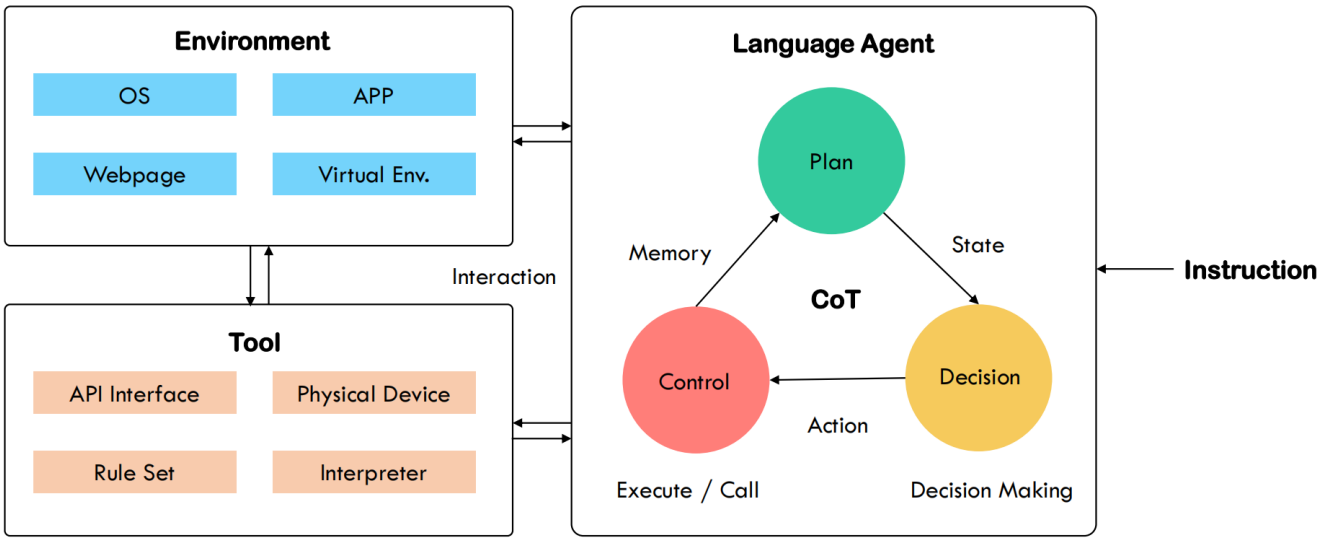

图 5. 语言代理模型框架

中，尤其是在环境可能发生变化的情况下，语言代理可以极大地从整合环境反馈中受益（Chen 等人）。有效整合这种反馈需要实施一个关键方法：具有环境反馈的自我纠正（Xu 等人）。自我纠正包括将模型暴露于复杂的操作序列，包括执行代码、进行操作和控制机器人等任务。这些操作可能导致执行失败并生成错误消息。在这种情况下，代理不仅需要理解这些环境提示，还必须积极参与迭代错误纠正过程，直到达到期望的结果。因此，在这些动态环境中，代理的性能不仅是其自我纠正能力的直接指标，而且展示了代理有效吸收环境反馈的能力。这种反馈的无缝整合不仅提炼了解释能力，还增强了整体功能，使其在先进语言代理领域至关重要。

5.3. CoT 记忆模型

短期记忆. 短时记忆是作为时间信息形成的，可能在 episode 的不同步骤中具有灵活性。短时记忆更加具有时间特定性，提供了明确的、最近的上下文，有助于代理。一方面，短时记忆直接支持并与当前确切状态更密切相关。另一方面，短时记忆对整个环境的影响相对较小。短时记忆可以在多步任务的 episode 中建模，动作历史链，或者在多跳问答的最后几个跳中的 rationales 或子问题中。由于其显著的时间特性，短时记忆引起的存储问题相对较小。

长期记忆. 长时记忆为代理提供了保留和回忆跨 episode 的静态信息的能力。与短时记忆相比，长时记忆更加通用于任务，具有对整个世界的宏观和抽象理解。这可以包括存储制度本身的程序性记忆、存储有关世界的事实的语义记忆以及存储代理过去行为序列的情景记忆（Sumers 等人 [16]）。给定一个目标，即点赞最新的帖子，在不同的环境状态中，观察到了两个完成目标的行动链：(i) [打开 Instagram，转到主页，查看最新的帖子，给最新的帖子点赞] 和 (ii) [转到 HOME 屏幕，打开 Instagram，转到主页，查看一篇帖子，给最新的帖子点赞]。可以发现原子操作 [打开 Instagram，转到主页，查看最新的帖子，给最新的帖子点赞] 可以作为这个目标的长时记忆，即一系列静态记忆。长时记忆可以依赖于参数化和非参数化的知识存储。它们可以来自语言代理的可训练参数，也可以作为通过检索系统利用的外部知识进行维护。以前 episode 的较早跳是来自代理参数的长时记忆，而输出的动作公式是参数化的长时记忆。

5.4. CoT 推理模型

受到引导 LLMs 逐步推理能力成功的启发，CoT 也被应用于通过规划或决策引导代理进行推理。更重要的是，针对语言代理的 CoT 方法需要精心设计，

以处理动作执行和状态观察。

通过结合交替的思考、行动和观察（Yao 等人[16]），弥合了推理和行动之间的差距。通过探索使用 LLMs 以交替方式生成 CoT 痕迹和任务特定的行动，发现推理和行动实现了相互促进。推理痕迹帮助模型制定行动计划和处理异常，而行动允许 LLM 与外部源（如知识库或环境）进行接口，以收集用于知识支持的额外信息。（Xu 等人[17]）将推理过程与外部观察分开，以减少在 CoT 的多个步骤中的令牌消耗。

类似地，AgentBench 迫使语言代理通过 “思考” 和 “行动” 步骤完成任务。此外，Zhang & Zhang 等[18] 提出了一种链式行动技术——利用一系列中间的先前行动历史和未来行动计划——帮助代理决定执行什么动作，从而将决策转化为 CoT 推理问题。

6. 总结与展望

在短短一年多的时间里，CoT 技术大大增强了 LLMs 的推理能力。大语言模型已经超越了 NLP 中的推理任务的局限，CoT 技术极大地促进语言代理的发展并未大模型可解释性提供了全新的解决思路。这些代理模型展示了理解语言指令并在不同环境中执行动作的能力。

当前我们对于大模型的思维机制并不清晰，CoT 机制为我们提供了一种解释方案。我们注意到 CoT 所提供的解决方案中涉及到了对推理过程的分解、组合、链接，在这些推理过程中，必不可少地要涉及到对外界知识的对齐，包括客观知识以及人类表现等 (RLHF)，当模型与外界交互从而更新推理策略时，实际上其逐渐与强化学习中 Agent 概念接近，大语言模型 agent 在引入 CoT 模型当中为模型提供了更强的推理能力和可解释性、可控制性等，CoT 技术将是在 AGI 的建造过程中的关键一环。

参考文献

[1] Adept. Act-1: Transformer for actions. https://www.adept.ai/act, 2022.

[2] Gati V Aher, Rosa I Arriaga, and Adam Tauman Kalai. Using large language models to simulate multiple humans and replicate human subject studies. In International Conference on Machine Learning, pp. 337–371. PMLR, 2023.

[3] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. Advances in Neural Information Processing Systems, 35:23716–23736, 2022.

[4] Hui Bai, Ran Cheng, and Yaochu Jin. Evolutionary reinforcement learning: A survey. Intelligent Computing, 2:0025, 2023a.

[5] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A frontier large vision-language model with versatile abilities. ArXiv preprint, abs/2308.12966, 2023b. URL https://arxiv.org/abs/2308.12966.

[6] Ramakrishna Bairi, Atharv Sonwane, Aditya Kanade, Arun Iyer, Suresh Parthasarathy, Sriram Rajamani, B Ashok, Shashank Shet, et al. Codeplan: Repository-level coding using llms and planning. ArXiv preprint, abs/2309.12499, 2023. URL https://arxiv.org/abs/2309.12499.

[7] Rohan Bavishi, Erich Elsen, Curtis Hawthorne, Maxwell Nye, Augustus Odena, Arushi Somani, and Sa˘gnak Ta¸sırlar. Introducing our multimodal models, 2023. URL https://www.adept.ai/blog/fuyu-8b.

[8] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski,

Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. ArXiv preprint, abs/2308.09687, 2023. URL https://arxiv.org/abs/2308.09687.

[9] Zhen Bi, Ningyu Zhang, Yinuo Jiang, Shumin Deng, Guozhou Zheng, and Huajun Chen. When do program-of-thoughts work for reasoning? ArXiv preprint, abs/2308.15452, 2023. URL https://arxiv.org/abs/2308.15452.

[10] Daniil A Boiko, Robert MacKnight, and Gabe Gomes. Emergent autonomous scientific research capabilities of large language models. ArXiv preprint, abs/2304.05332, 2023. URL https://arxiv.org/abs/2304.05332.

[11] Andres M Bran, Sam Cox, Andrew D White, and Philippe Schwaller. Chemcrow: Augmenting large-language models with chemistry tools. ArXiv preprint, abs/2304.05376, 2023. URL https://arxiv.org/abs/2304.05376.

[12] Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. ArXiv preprint, abs/2305.17126, 2023. URL https://arxiv.org/abs/2305.17126.

[13] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Kaijie Zhu, Hao Chen, Linyi Yang, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. ArXiv preprint, abs/2307.03109, 2023. URL https://arxiv.org/abs/2307.03109.

[14] Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. ArXiv preprint, abs/2310.05915, 2023a. URL https://arxiv.org/abs/2310.05915.

[15] Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. Walking down the memory maze: Beyond context limit through interactive reading. ArXiv preprint, abs/2310.05029, 2023b. URL https://arxiv.org/abs/2310.05029.

[16] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. ArXiv preprint, abs/2211.12588, 2022. URL https://arxiv.org/abs/2211.12588.

[17] Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, et al. Pali-x: On scaling up a multilingual vision and language model. ArXiv preprint, abs/2305.18565, 2023c. URL https://arxiv.org/abs/2305.18565.

[18] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. ArXiv preprint, abs/2304.05128, 2023d. URL https://arxiv.org/abs/2304.05128.