

ART: 大型语言模型的自动多步推理和工具使用

Bhargavi Paranjape¹ Scott Lundberg² Sameer Singh³ Hannaneh Hajishirzi^{1,4}

Luke Zettlemoyer^{1,5} Marco Tulio Ribeiro²

¹华盛顿大学, ²微软研究, ³加利福尼亚大学尔湾分校,

⁴艾伦人工智能研究所, ⁵Meta AI

摘要

大型语言模型 (LLMs) 可以通过生成中间的思维链 (CoT) 推理步骤, 在少量和零-shot设置中执行复杂的推理。此外, 每个推理步骤可以依赖外部工具来支持核心 LLM 能力之外的计算 (例如搜索/运行代码)。以往关于 CoT 提示和工具使用的工作通常需要手工制作特定任务的演示, 并仔细编写模型生成与工具使用的交错。我们引入了自动推理和工具使用 (ART) 框架, 该框架使用冻结的 LLMs 自动生成中间推理步骤作为程序。给定一个新的任务来解决, ART 从任务库中选择多步推理和工具使用的演示。在测试时, ART 在调用外部工具时无缝暂停生成, 并在恢复生成之前集成它们的输出。ART 在 BigBench 和 MMLU 基准测试中对未知任务的少量提示和自动 CoT 的性能有了显著改进, 并在大多数任务上与手工制作的 CoT 提示的性能相匹配。ART 还具有可扩展性, 并且使人们通过纠正任务特定程序中的错误或整合新工具来改善性能变得容易, 我们通过最小的人工干预大大提高了选择任务的性能。

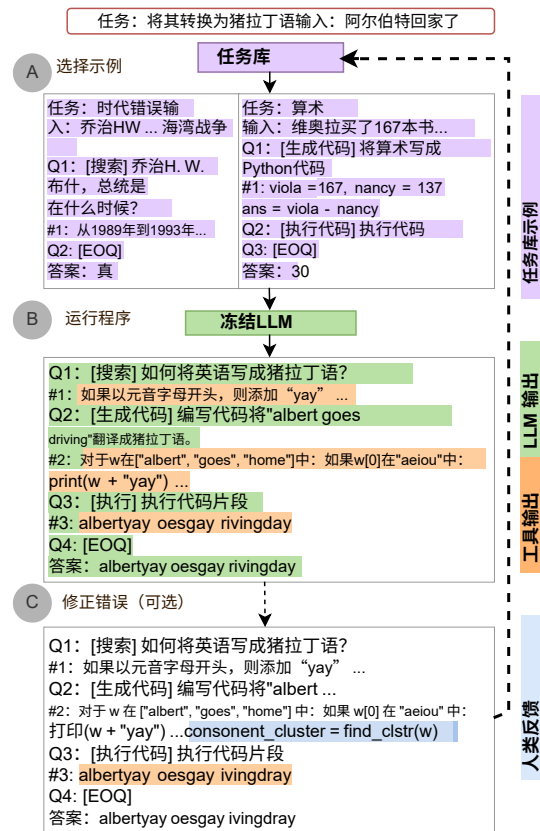


图1: ART通过选择任务库 (A) 中的相关任务的分解和在LLM生成过程中选择和使用工具库 (B) 来为新任务生成自动多步分解。人们可以选择编辑分解 (例如, 纠正和编辑代码) 以提高性能 (C)。

1 引言

上下文学习允许大型语言模型 (LLMs) 通过使用自然语言指令和少量演示作为 LLM 的提示, 快速适应新任务 (Xie 等, 2021; Brown 等, 2020; Chowdhery 等, 2022)。虽然这样可以避免注释大型数据集甚至托管 LLM 本身 (因为许多 LLM 可通过 API 获得), 但在多步推理方面存在严重的性能限制 (Liu 等, 2022)。

数学 (Patel 等, 2021 年), 获取最新信息 (Komeili 等, 2022 年) 等。为了解决这些限制, 最近的研究提出了提示 LLM 模仿思维链 (CoT) 进行多步推理 (Wei 等, 2022 年; Zhou 等, 2022 年; Wang 等, 2022 年; Press 等, 2022 年; Khot 等, 2022 年; Arora 等, 2022 年) 或者为它们提供访问工具 (例如计算器或 QA 模型) 以实现更复杂的推理步骤 (Gao 等, 2022 年) ;

Chen等, 2022年; Press等, 2022年; Wei等; Schick等, 2023年)。然而, 现有的链式推理和工具使用方法很难扩展到新的任务和工具, 需要进行微调或为特定任务定制的提示工程 (Parisi等, 2022年) 或工具 (Schick等, 2023年)。

在本文中, 我们提出了自动推理和工具使用 (ART) 框架, 该框架可以自动生成新任务实例的分解 (多步推理)。该框架还在各个步骤中选择并使用最合适的可用工具 (如搜索引擎和代码执行)。给定一个新任务, ART从任务库中检索相关任务的演示, 以实现少样本分解和工具使用。这些演示遵循一种灵活但结构化的查询语言 (Beurer-Kellner等, 2022年), 因此很容易解析中间步骤, 在调用外部工具时停止生成, 并在包含这些工具的输出后恢复生成 (图1)。ART向LLM提供了如何分解几个相关任务实例以及如何选择和使用来自工具库的任何工具的演示。这鼓励模型从演示中推广, 以零样本方式分解新任务并在适当的位置使用工具。它还使用户能够通过简单更新任务和工具库来修复推理链中的任何错误或添加新工具, 根据需要提供新的演示 (例如, 针对当前任务)。

我们为15个不同的Big-Bench (Srivastava等人, 2022年) 任务构建了一个任务库, 并在19个未见过的Big-Bench测试任务、6个MMLU任务以及相关工作中使用的各种任务 (如SQUAD、TriviaQA、SVAMP、MAWPS) 上评估了ART。ART在32/34个BigBench任务和所有MMLU任务上始终与自动生成的CoT推理链相匹配或超过, 平均提高了超过22个百分点。

特别是工具使用可以将测试任务的性能平均提高12.3个百分点, 与不允许使用工具时相比 (表3)。

在未见过的BigBench和MMLU任务中, ART平均提高了10.8个百分点, 相比直接的少样本提示。改进尤其显著的是在需要算术和算法推理的未见任务上, ART相比直接的少样本提示提高了12.5%, 并且对于使用监督进行分解和/或工具使用的GPT3的先前最佳结果。

6.1% 百分点 (表3)。

最后, ART通过简单地更新任务和工具库中的新演示, 使人类能够干预和改进推理过程, 从而非常容易地改善任何特定任务的性能, 只需进行轻微的人类反馈。在¹²个测试任务中, ART通过额外的人类反馈平均超过GPT3的最佳结果20个百分点 (表6)。¹

2 相关工作

用于低资源适应的规模化微调

最近的研究表明, 在广泛的公共NLP数据集上 (带有预设指令) 对LLM进行微调是一种有效的跨任务泛化技术 (Mishra等, 2021; Sanh等, 2021; Khashabi等, 2020; Wei等, 2021), 在零样本和少样本设置中都适用。Ouyang等人 (2022) 表明, 通过使用人类反馈对期望的模型行为进行微调, 将语言模型与用户意图在各种任务上对齐 (InstructGPT), 进一步改善了复杂NLP任务的上下文学习性能。Chung等人 (2022) 表明, 通过在聚合的任务混合 (T0, CoT, dialog和代码数据集) 上进行微调, 并将模型扩展到540B参数, 可以在诸如Big Bench和MMLU等多个基准测试中实现最先进的上下文学习性能。ART使用InstructGPT和Cotex (在代码上进行微调的LLM (Chen等, 2021)) 的API访问来利用它们的新兴上下文学习能力。未来在LLM的规模化微调方面的改进很可能会提高ART的性能。

使用中间推理步骤进行提示

思维链 (CoT) 提示 (Wei等, 2022年; Suzgun等, 2022年) 是一种流行的无梯度技术, 鼓励LLM在最终答案之前生成中间推理步骤, 具有多个任务特定变体 (例如, 从最少到最多的提示 (Zhou等, 2022年), 自问 (Press等, 2022年), 问我任何事 (Arora等, 2022年), 连续提示 (Dua等, 2022年), 分解提示 (Khot等, 2022年))。虽然最初是手工制作的这些提示, 但最近的研究 (Kojima等, 2022年) 表明, 当以“让我们逐步思考”的前缀提示时, LLM可以以零-shot方式生成CoT风格的多步推理。张等人 (2022年) 使用

¹代码 可以在 <https://github.com/bhargaviparanjape/language-programmes/> 上找到

表1：比较ART与相关方法在多步推理和工具使用方面的差异

特征	CoT自动	CoT前者	工具-	ART
多步推理	✓	✓		✓
有限监督		✓	✓	✓
工具使用			✓	✓
可扩展库				✓
跨任务迁移		✓	✓	✓
人类反馈	✓			✓

LLMs可以自动生成这种CoT风格的提示- Auto CoT-在算术和常识推理任务上的性能与手工制作的提示相媲美。我们在表1中比较了ART、CoT和AutoCoT。ART建立在这一工作基础上，引入了一种通用语言，实现了跨任务演示和灵活可扩展的工具使用，提高了中间推理步骤的准确性。

工具使用越来越多的人对克服LLM的限制提出了兴趣，例如使用搜索引擎、网络浏览器、计算器、翻译系统和Python解释器（Komeili等，2022年；Thoppilan等，2022年；Lazaridou等，2022年；Shuster等，2022年；Nakano等，2021年；Thoppilan等，2022年；Cobbe等，2021年；Thop-pilan等，2022年；Gao等，2022年；Chen等，2022年）。这些方法大多要求大量人工监督（Thoppilan等，2022年；Komeili等，2022年），或者精心构建的针对特定任务和特定工具的提示。最近的另一条研究线路使用自我监督教授LLMs使用搜索、翻译和计算器（Schick等，2023年）- Toolformer。相比之下，由于ART不需要任何额外的训练或特定于工具的提示，它允许用户在替换底层LLM（例如当发布新版本的GPT-3时）和替换或添加新工具（无论是通用工具还是对特定任务至关重要的工具）方面具有灵活性。我们在表1中比较了ART和Toolformer。在第3.4节中，我们展示了人机协作反馈-分析和调试LLM生成和扩展工具使用-如何大幅提升ART的性能，同时还扩展了它的功能。ART的这种内置反馈循环和自适应能力扩展了遵循指令和使用工具的LLMs的能力。

3 ART

通过ART，一个冻结的LLM将新任务的实例分解为多个步骤（在适当的情况下使用外部工具），尽管没有明确的分解或工具使用的监督。在本节中，我们将介绍ART的概述，然后更详细地描述每个单独的组件。我们以物理问题回答（PQA）任务作为一个运行示例，该任务包括高中物理问题。

3.1 概述

在图2中，ART被提供了一个新的任务描述和输入实例。我们还假设可以访问一些输入-输出对（未显示），而无需分解或工具使用监督。

提示构建。ART从任务库中检索相似任务（图2（A）；第3.2节），并将这些任务的实例作为演示添加到提示中。

任务库中的演示以特定格式编写，由自定义解析表达式语法（PeG）（第3.2节）定义。该语法被定义为每个任务实例被分解为一系列子步骤。其中一些子步骤包含与工具库中的工具对应的符号（第3.3节）。我们将这些分解称为程序，因为顺序推理步骤和对工具的符号调用类似于具有函数调用的传统程序。

生成的提示由相关任务的程序组成，并教导LLM如何有效地分解新任务的实例-这些程序中的子步骤和工具可以被LLM用于跨任务的泛化。

在图2（A）中，演示包括对搜索和代码工具的调用。

生成。在生成时（图2（B）），LLM编写自己的程序。ART在生成程序时解析程序，并在生成的文本中遇到工具调用时暂停生成，待工具调用完成并将其输出整合回程序后继续生成。如图所示，使用搜索引擎找到适当的物理公式，然后LLM使用代码生成和执行来替换给定的值并计算答案。

人类反馈（可选）。人们可以向任务库中添加新的分解演示，或者添加/编辑工具库中的工具，以便

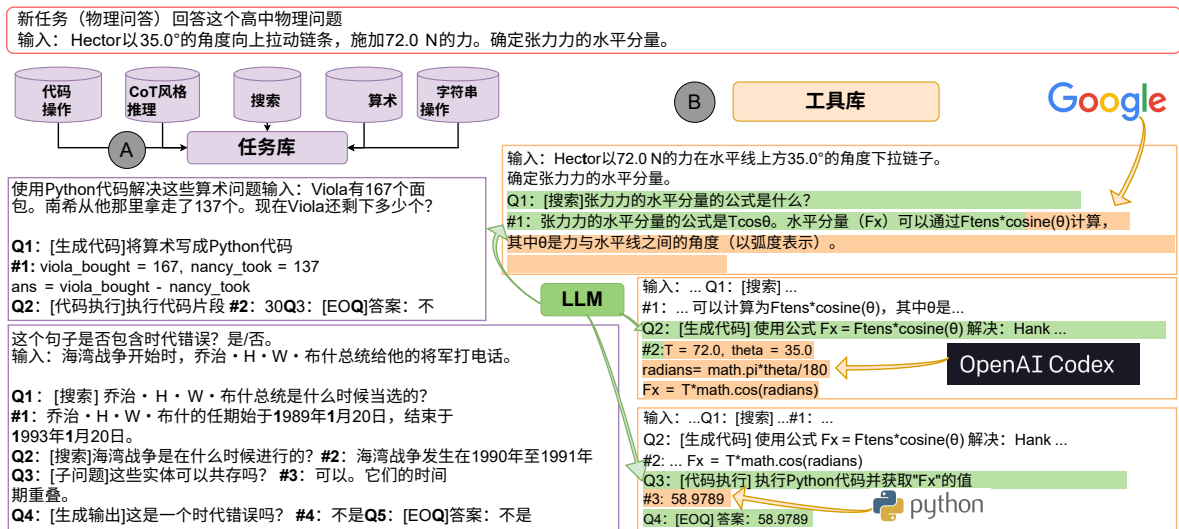


图2：ART在新任务Physics QA上的运行过程。相关任务的程序，如anachronisms和Math QA，为LLM提供少量监督，这些程序中的相关子步骤和工具可以被LLM用于跨任务泛化（显示为紫色）。（B）工具使用：搜索用于找到适当的物理公式，代码生成和执行用于替换给定的值并计算答案（显示为橙色）。

为了改善对特定任务的性能，或者是一般性的改进。在图3（C）中，用户通过包含添加测量单位的步骤来纠正特定程序，并将这个（修改后的）程序添加到任务库中。虽然我们的大多数实验没有使用这样的反馈，但我们表明在任务泛化不自动发生时，它非常有效地显著提高性能。此外，它使用户能够在不重新训练LLM的情况下添加自定义工具。

- 自由形式推理：用自然语言解释逐步推理。
- 字符串操作：重新格式化/编辑字符串，检查字符串蕴含关系等。

然后我们从每个聚类中选择2-4个任务，并为每个任务的几个实例编写程序（分解），包括对外部工具的调用和实际输出。每个聚类中的程序示例在附录A.1中。这些程序遵循特定的语法，如下所述。

3.2 任务库

我们从Big-Bench（Srivastava等，2022）中的一组小型任务中构建了一个程序库，该程序库衡量了语言模型的能力和限制。Big-Bench任务涵盖了传统自然语言处理、数学、常识推理和问答等领域。

构建任务库。我们确定了在BigBench的超过一半任务中都有用的五个技能，这些任务涵盖了英语文本分类或生成简短答案（见A.1）。我们根据这些技能将任务分组到以下类别中：

- 算术：算术和代数问题。
- 代码：生成和执行Python代码。
- 搜索和问题分解：需要搜索的单步或多步问题。

程序语法：程序格式必须在任务输入、步骤和工具调用方面具有灵活性，以涵盖各种自然语言处理任务。为此，我们定义了一个查询语言（Beurer-Kellner等，2022），它扩展了Khot等人（2022）的分解提示格式，因为它可以按顺序表示分解的推理步骤，并且包含对外部工具的函数调用（如其他LLM）。每个程序由一系列节点组成-任务输入节点、多个子步骤节点和一个答案节点。输入节点包含任务名称、描述任务的简单指令以及任务实例的输入：“回答这个高中物理问题。”

输入：Hector yanks...”。输入节点后面是一系列子任务节点，表示为（查询，答案）对“ Q_i : ..., # i : ...”。子任务查询 Q_i 有一个子任务名称和子任务

人类反馈

C

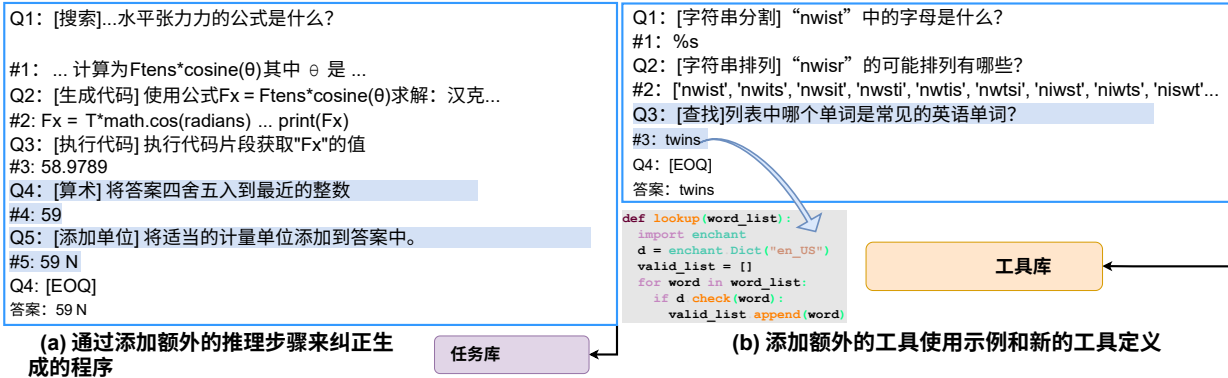


图3: ART的人类反馈, 显示了(a) PQA中添加推理步骤到程序和; (b) 单词解谜中工具库增加了一个新的查找工具。

任务输入 (“Q1: [搜索]公式是什么...”), 而子任务答案 $\#i$ 只是子任务的输出 (“#1: 水平分量(F_x)可以计算...”)。程序以一个虚拟子任务结束 (“Q3: [EOQ]”), 然后是一个最终答案节点 (“答案: 59N”)。图1和图2中的所有示例都遵循这种格式。

任务检索给定一个新任务, ART从任务库中检索 N 个任务来构建一个动态多任务提示。我们探索了两种检索相似任务的策略, 具体取决于可用的数据。如果新任务有少量标记的示例可用 (≈ 50), 我们遍历所有五个任务簇, 并从每个簇中选择几个任务程序来组成提示。

最终, 选择在未标记示例上性能最高的任务集群来预测所有未标记示例时。虽然这种策略需要一个保留集的输入-输出对, 但不需要额外的监督来生成分解程序。

在第二种策略中, 我们创建一个少样本提示 (附录A.2), 其中包含任务对, 每个任务包括一个名称、说明和几个输入-输出示例。对于每一对, 我们提供一个“相似”或“不相似”的标签, 并进行推理 (例如, “它们相关, 因为它们需要解决算术问题”)。在运行时, 我们将测试任务与任务库中的每个任务配对, 并根据“相似”和“不相似”之间的对数概率比选择排名最高的任务。我们在第A.2节中探讨了这两种策略。

3.3 工具库

每当子任务查询名称与任务库中的工具名称匹配时 (例如, “ Q_i : [搜索]”), 生成过程将停止, 并在调用工具并将其输出合并到部分完成的程序后恢复。我们用以下工具初始化工具库 (所有这些工具在任务库中都有演示)。特别地, 我们描述了用于表示这些工具和它们的输入的符号。我们还指定了如何将工具输出合并回程序中。工具特定的实现细节和在反馈过程中添加到ART的其他工具详见附录A.3。

搜索: 我们使用SerpAPI²进行搜索, 该工具提供了一个用于Google搜索的API。搜索的输入是在“ Q_i : [搜索]”之后由LLM生成的序列。当可用时, 我们提取答案框片段, 否则将前两个搜索结果片段组合在一起。对于图2 (B) 中的PQA, 搜索查询是原始输入, 后面跟着“水平张力力量的公式是什么?”, 输出是“... 水平分量 (F_x) 可以通过 $F_{\text{tens}} \cdot \cos(\theta)$ 计算得出...”。

代码生成: 我们使用Codex (Chen等, 2021) 模型进行代码生成。代码生成的输入是在子任务查询符号“ Q_i : [生成Python代码]”之后由LM生成的序列。这个参数是用于代码生成的指令, 并作为Python中的多行注释提示给Codex。例如, 在图2中, Codex被提示了指令““使用公式 $F_x = F_{\text{tens}} \cdot \cos(\theta)$ 来解决...””, 并生成了“ $T = 72.0$,”

²<https://serpapi.com>

`theta = 35.0, ..., Fx = T*math.cos(radians)”`，这被附加到不完整的程序中。

代码执行我们在一个虚拟的Python环境中运行Python代码，预先安装了算术、符号和科学计算包。代码执行的参数是上一个子任务的答案序列“`#(i-1) : ...`”，即要执行的Python代码片段。对于 $i=1$ ，任务的输入被用作参数，因为它可能包含要执行的代码片段。在图2中，执行在上一步生成的代码片段，并将变量“Fx”的值添加到不完整的程序中。

3.4 人类反馈

ART专门设计为适应人类反馈，因为它不需要额外的微调。因此，用户可以立即将反馈纳入ART中，通过编辑任务库和/或工具库。由于ART生成的多步推理程序是可解释的，我们探索以调试的形式进行反馈，即用户编辑现有程序而不是从头创建程序。这些编辑可以是纠正子步骤的输出，添加/删除子步骤（带有适当的输入和答案），添加对新工具的调用等。例如，在图3(a)中，用户通过添加两个子步骤来编辑程序，以便将答案四舍五入到最近的整数，并将适当的单位添加到答案中。这个反馈展示了任务的适当分解，因为这些操作仍然由LLM执行（工具库没有“[算术]”或“[添加单位]”的API）。相比之下，在图3(b)中，用户展示了使用字典“[查找]”并将其实现为工具库中的一个工具。虽然我们的大多数实验不依赖这样的反馈（因此在推理/工具使用方面没有监督的“零-shot”任务转移），但我们展示了这些简单操作可以极大地提高目标任务的性能。

4 实验设置

评估数据集除了 15 个任务在任务库（第3.2节）中，我们还在BigBench上对19个额外的测试任务进行了ART的评估，这些任务也属于第3.2节中确定的五个任务集群。为了检查跨基准泛化，我们还在MMLU基准的随机子集上评估了ART（Hendrycks

et al., 2020）。最后，我们还评估了用于评估Toolformer（Schick et al., 2023）的任务子集，以便将ART与用于工具使用的模型进行比较。

详细信息：我们使用InstructGPT（text-davinci-002）作为冻结的LLM，并使用Codex作为代码生成工具，温度设置为0.3。我们将提示中的种子任务数量设置为 $N=3$ ，并从每个任务中使用2个演示程序。

我们按照Srivastava等人（2022年）的方法，测量每个任务的首选评分指标，并报告5次运行的平均性能。

基线ART提出了一种自动框架，用于生成多步推理分解并在这些分解中使用相关的可用外部工具。我们与以下基线进行比较：

- **Few-shot/Direct**：使用输入-输出对（但没有中间推理）来提示LLMs。我们在BigBench上使用3个示例，在MMLU上使用5个示例，这与之前的工作（Suzgun等人，2022年）相同。我们对GPT-3和Codex两者都评估了这个基线，并报告两者中较高的结果。
- **Auto-CoT**：一种自动生成自然语言多步推理的基线。首先，随机选择5个示例用于引出CoT风格的推理（输入+ 让我们逐步思考。）。这些示例及其生成的输出形成了其他未见过的任务示例的提示。这个基线是自由形式的，不包括工具，因此可以验证我们的查询语言和任务库的有效性。我们对GPT-3评估了这个基线。
- **ART工具**：关闭工具使用的ART，即LLM生成每个子步骤的输出，以验证工具使用的收益。
- **GPT-3最佳**：使用多步分解和/或工具使用的最佳发布的GPT-3/Codex（175B）结果。这些通常包括额外的人工监督来分解推理步骤，并使用外部工具来提高性能（使用精心构建的提示）。

关于基线和GPT-3最佳模型的其他详细信息，请参见附录A.4。

5 结果

我们在任务库（5.1）中评估ART（无人工反馈）的任务，并在各种任务上评估

任务名称 (集群)	Few Shot	AutoCot	ART 无工具使用	ART	GPT-3 最佳
时代错误 (搜索)	71.3 ⁵	51.48	70.87	75.66	-
音乐 (搜索)	2.03 ⁵	12.88	10.04	19.19	15.2 ³
印度教知识 (搜索)	85.02 ⁵	73.03	83.42	87.98	-
已知未知 (搜索)	68.90 ⁵	56.09	80.43	80.43	-
Δ使用ART (搜索)	+9.0	+17.44	+4.6		+4.0
初等数学问答 (算术)	56.40 ⁷	74.52	58.04	68.04	-
水鼠 (算术)	20.54 ⁷	34.41	36.29	54.20	54.1 ⁴
GSM8K (算术)	7.79 ⁷	21.99	53.4	71.00	71.6 ⁴
导航 (算术)	60.7 ⁷	61.7	72.4	72.4	85.90 ¹
Δ与ART (算术)	+30.0	+18.25	+11.4		-4.7
第K个字母连接 (字符串)	3.2 ⁵	0.64	8.19	40.00	98.0 ²
语言游戏 (字符串)	35.14 ⁵	18.58	11.19	23.08	-
日期理解 (字符串)	37.53 ⁵	38.90	52.05	-	70.41 ¹
自动调试 (代码)	62.94 ⁵	38.24	55.29	62.94	-
代码描述 (代码)	97.99 ⁷	88.67	84.67	88.00	-
形式谬误 (CoT)	44.84 ⁵	56.4	64.76	-	58.4 ¹
夸张 (CoT)	62.72 ⁵	55.4	80.80	-	72.4 ¹
Δ与ART (其他)	+9.6	+16.4	+13.7		-15.4
Δ与ART (总体)	+14.90	+17.17	+7.91		-9.0

表2: 任务库中ART的性能。 (¹人工制作的CoT (Wei等, 2022年; Suzgun等, 2022年), ²分解提示 (Khot等, 2022年), ³自问 (Press等, 2022年), ⁴PoT (Chen等, 2022年), ⁵InstructGPT (Ouyang等, 2022年), ⁷Code-davinci-002 (Chen等, 2021年)。) (-) 对于使用CoT推理的任务, 不使用工具。

从BigBench、MMLU和QA基准测试中选择了一些测试任务 (5.2)。然后, 我们展示了ART在更多计算资源 (自一致性) 和人类反馈 (5.3) 的帮助下可以进一步改进。

5.1 任务库上的结果

对于任务库中的任务, 提示中的演示包括任务本身的两个实例, 以及来自同一聚类中其他任务的实例。我们在表2中呈现了结果, 任务按技能聚类进行组织。即使只有两个实例的分解演示, ART的性能大大优于少样本学习 (平均提高14.9个百分点), 与CoT的先前工作一致。在使用代码生成和/或代码编辑模型的语言游戏、代码描述和自动调试等任务上, ART的表现不如其他任务。我们观察到代码生成错误通常会导致推理中的级联错误。

同样, 在大多数任务上, 即使没有使用任何工具, ART的表现也优于AutoCoT (平均提高8个百分点)。我们假设程序格式 (和PeG语法) 比自由形式的CoT更能引出模型的多步推理, 因为它为推理提供了额外的结构。当启用工具使用时, ART在性能上优于AutoCoT。

所有任务 (+17.7%) 减去一个。工具在大约95%的测试实例中被调用, 并且显著提高了性能 (+7.91%)。工具使用带来的收益对于需要将算术问题表示为执行复杂算术的代码的任务尤为显著 (平均增加了21.85)。这也在之前的研究中得到了注意 (Chen等, 2022年; Gao等, 2022年)。

与最佳的GPT-3结果相比, ART在5/8个任务中更强或可比。对于其他任务, 进一步的调查表明Khot等人 (2022年) 和Suzgun等人 (2022年) 提供的演示比我们为这些任务编写的两个程序更有效 (我们在附录A.5中进一步探索了人类反馈)。总之, ART在库任务上比少样本学习和AutoCoT更强 (我们提供了2个标记的分解), 并且与最佳的GPT-3结果相当。

5.2 测试任务 (跨任务迁移)

我们在不使用显式监督进行分解和工具使用的测试任务上测量跨任务泛化。ART根据第3.2节中的第一策略从任务库中检索演示, 该策略使用少量

任务名称 (集群)	Few Shot	AutoCot	ART 无工具使用	ART	GPT-3 最佳
测试任务					
句子歧义 (搜索)	70.67 ⁵	51.47	71.00	73.33	-
策略问答 (搜索)	55.49 ⁵	27.22	59.37	66.44	-
物理学 (搜索)	70.09 ⁵	61.83	59.13	67.55	-
Δ 使用ART (搜索)	+3.7	+22.27	+ 5.9		
物理问题 (算术)	7.02 ⁵	5.56	6.30	20.37	-
运算符 (算术)	71.23 ⁷	75.52	71.80	92.00	-
单位解释 (算术)	58.2 ⁷	41.20	51.4	53.99	-
重复复制逻辑 (算术)	50.01 ⁷	15.63	31.25	44.38	-
对象计数 (算术)	39.2 ⁷	26.80	42.2	87.00	81.20 ¹
表格中的企鹅 (算术)	58.23 ⁷	40.40	68.86	77.85	72.34 ¹
关于对象的推理 (算术)	71.00 ⁷	33.33	45.35	64.34	52.69 ¹
跟踪洗牌对象 (算术)	22.39 ⁷	19.44	18.14	37.67	36.32 ¹
Δ 使用ART (算术)	+19.0	+36.7	+ 23.1		+6.1
单词解谜 (字符串)	40.72 ⁷	32.44	23.03	42.7	-
简单文本编辑 (代码)	35.31 ⁵	30.21	20.74	27.65	-
计算机科学算法 (代码)	73.48 ⁷	0.0	41.59	88.11	-
体育理解 (CoT)	69.74 ⁵	51.47	92.89	-	86.59 ¹
Snarks (CoT)	54.58 ⁵	57.24	57.13	-	65.2 ¹
消歧问答 (自由形式)	55.03 ⁵	48.45	55.89	-	60.62 ¹
时间序列 (CoT)	55.80 ⁷	19.70	49.5	-	81.8 ¹
破坏名字 (CoT)	71.01 ⁵	55.28	60.22	-	-
Δ 使用ART (其他)	2.4	22.5	24.37		-9.4
Δ 使用ART (总体)	+6.9	+24.6	+16.7		-1.7
MMLU					
大学计算机科学 (搜索)	41.00	43.99	63.40	67.80	63.6 ⁶
天文学 (搜索)	62.10	41.48	76.71	79.1	62.5 ⁶
商业道德 (搜索)	61.60	48.8	77.17	81.16	72.7 ⁶
病毒学 (搜索)	50.03	49.52	71.60	71.49	50.72 ⁶
地理学 (搜索)	77.67	57.07	70.30	71.71	81.8 ⁶
数学 (算术)	36.67	33.77	39.50	45.66	34.5 ⁶
Δ 使用ART (MMLU)	+14.6	+23.7	+3.0		+8.5

表3: ART在BigBench任务和MMLU任务上的性能。(¹人工制作的CoT (Wei等, 2022年; Suzgun et al. , 2022年) , ⁵InstructGPT (Ouyang等, 2022年) , ⁶缩放指令微调 (Chung等, 2022年) , ⁷ Code-davinci-002 (Chen等, 2021年)) 。

	SQuAD	T-REx	SVAMP	MAWPS	NQ	TriviaQA
GPT3 (175B)	29.90	39.8	10.0	19.8	22.6	65.9
Toolformer	33.8	53.5	29.4	44.0	17.7	48.8
ART	39.34(+5.5)	50.4(-3.1)	76.2(+46.8)	71.00(+27.0)	33.8(+16.1)	66.13(+17.33)

表4: 比较ART在GPT3 (175B) 模型和 (Schick等, 2023年) 上的结果, 后者是一个经过工具使用微调的较小的GPT-J模型。 结果来自他们的论文 (他们的代码和模型不公开可用) 。

通过标记的输入输出对来选择一个任务簇, 并从该簇中抽样演示程序。³

BigBench测试任务尽管没有分解或工具使用的监督, 但表3中的结果与任务中的结果相似

任务库。ART优于少样本学习 (提高了6.9个百分点)。 特别是在算术任务上, ART有显著改进 (+19.0) , 并且在搜索任务上与少样本性能相当。 在名称中的非语法选择和和时间序列中的非输入选择通常是不正确的, 而少样本基准可能学会忽略这些选择, 而ART则试图

³我们在附录A.2中比较了这两种策略

	简单文本	计算机科学 算法	策略问答	物理学 问题	单位 解释	关于编辑的推理 彩色物体
ART + 自我一致性	27.65 30.67(+3.0)	88.11 90.99(+2.9)	66.44 70.76(+4.3)	20.37 24.07(+3.7)	53.99 57.20(+3.2)	64.34 69.11(+4.8)

表5：通过自我一致性改进ART（Wang等，2022年）。在15次运行中集成模型生成进一步提升性能。

任务	CoT +人类		ART +人类		GPT-3 人类 最佳	人类 反馈
CS算法	0.0	23.0	88.11	92.73	73.48	C: 最长公共子序列代码
关于对象的推理	33.33	67.75	64.34	98.90	71.00	C: 定义对象、颜色、计数数据结构
重复复制逻辑*	15.63	45.22	44.38	80.31	50.01	C: 字符串编辑操作
句子歧义	51.47	72.33	73.33	83.67	70.67	C: 限制查询以提取相关信息。
简单文本编辑*	30.21	35.31	27.65	36.11	35.31	C: 字符串编辑操作
策略问答*	27.22	29.19	66.44	69.15	55.49	C: 限制查询以提取相关信息。
物理*	61.83	68.21	67.55	72.55	70.09	A: [搜索]连接质量的公式, ...
时间序列	19.70	30.22	49.5	88.00	81.8	A: [子问题] X是否可以自由Yam到Zam?
跟踪洗牌的对象。	19.44	36.48	37.67	99.86	36.32	C: 定义对象对数据结构, 交换逻辑
单位解释*	41.2	41.2	53.99	95.0	58.2	A: [添加单位]将正确的单位添加到答案中
单词解谜*	32.44	33.40	42.70	62.11	40.72	T: 在字典中查找排列组合
平均	30.2	43.8	56.0	79.85	58.5	

表6：通过自洽性和人机协同反馈改进ART和自由形式CoT。(*)表示这些任务的CoT推理是在人机协同改进之后进行的。对于ART的反馈包括纠正程序中的子步骤（“C:”），添加额外的子步骤（“A:”），以及定义新的工具（“T:”）。请注意，每个任务仅编辑了五个示例。

明确地对它们进行推理。与库任务一样，我们观察到字符串操作任务（如简单文本编辑、单词解谜和重复复制逻辑）受到代码生成错误的影响。

正如在库任务中观察到的那样，ART在几乎所有任务上都优于AutoCoT（提高了24.6个百分点）。工具再次被频繁调用（89%的实例），并且对于超过基准线的增益贡献了相当大的部分。

与最佳的GPT-3发布结果相比，ART的表现平均较好，尤其在算术任务上（+6.1%）。与之前一样，在需要良好的人类演示如何分解任务的任務中，ART的表现较差（由Suzgun等人（2022）提供）。我们在5.3节对这些任务进行了更多人类反馈的重新评估，但即使没有这些反馈，我们也得出结论，ART在BigBench上具有竞争力，即使我们没有对任务的分解进行监督（即存在跨任务泛化）。

其他基准测试为了确保ART不过度拟合BigBench风格的任务，我们在其他基准测试上评估性能。我们在MMLU基准测试（Hendrycks等人，2020）中报告了随机选择任务的性能。

在表3中，ART在5/6个任务上比所有基线效果更好（平均提高8.5分），尽管没有演示或工具使用的监督。MMLU需要广泛的世界知识，因此大多数任务从搜索工具中获益最多。

在表4中，我们将ART与用于评估Toolformer（Schick等人，2023年）的随机子集任务进行了比较，Toolformer是一个经过微调以使用各种工具的模型。虽然比较不完全，因为Toolformer使用了较小的GPT-J模型，但ART在这些任务的5/6上大幅优于Toolformer是有意义的。为了确保这些收益不仅仅是模型规模的结果，我们还使用了普通的GPT-3作为基准，结果比ART在所有任务上都要差得多。除了性能改进，我们再次注意到，当引入新工具或新的基础LLM时，ART不需要额外的微调，并且还可以通过计算或人类反馈进一步改进。

5.3 改进ART

自一致性 之前的研究已经注意到生成多个LLM输出并选择最频繁的答案（这个过程被称为自一致性）在某些情况下特别有益，尤其是对于具有

多步推理的设置 (Khot等, 2022年; Wang等, 2022年)。在表5中, 我们展示了ART在任务子集上进行自一致性结果 (生成15个输出) 的结果, 并发现它在性能上始终有所提升, 但需要额外的计算成本。

人类反馈 我们还在表6中试点使用任务特定的反馈, 通过让作者之一编辑每个任务中导致错误的模型生成的程序的5个随机实例。在编辑时, 我们纠正子步骤中的错误 (标记为“C:”), 添加缺失的子步骤 (“A:”), 或者定义一个新的工具并演示其使用 (“T:”)

例如, 这涉及到为PQA任务引入了一个“添加单元”子步骤, 并实现了一个用于“单词解谜”任务的字典查找功能 (都由图3所示)。

我们还比较了应用于CoT风格推理的人类反馈。Suzgun等人 (2022) 已经为一些任务提供了参考的CoT风格推理。对于没有人工编写的CoT推理的数据集, 我们根据表6中的指示纠正了自动CoT基线的输出。

同一作者编辑了5个随机实例的Auto-CoT分解, 这些分解在相同的任务上导致错误, 纠正了子步骤中的错误或添加了新的子步骤。作为参考, 编辑包括基线中的35%的标记和ART程序中的15.7%的标记。这包括在所选任务的72%中纠正子-步骤的参数和输出, 并在44%的任务中添加额外的子步骤。为两个任务添加了新的工具定义-单词解谜的字典查找和形式谬误的Prolog引擎。

在这两种情况下, 编辑程序并将它们作为演示添加进去, 会在手头的任务上带来显著的性能提升。然而, 在ART中, 这种提升更加显著, 使其在手头的任务上始终优于最佳的GPT-3基线。此外, 这些修正过的程序和工具可以添加到任务和工具库中, 我们在表3中的先前结果表明它们可能有助于改善ART在其他任务上的表现。这个试点表明, 除了在跨任务泛化上具有竞争力之外, ART在最小的人工干预下也非常适合任务特定的改进。我们在A.5中的任务库中报告了类似的结果。

6 结论

我们介绍了ART, 一种无梯度的方法, 用于自动多步推理生成和

大型黑盒语言模型的自动工具使用。我们的主要贡献包括一个轻量级语法, 用于表示多步推理作为程序 (包括工具调用和参数), 一个可扩展的种子任务库, 用于编写程序, 以及一个工具库, 包括搜索、代码生成和执行等有用的外部工具。可解释的推理框架还允许人类改进任务分解和工具使用, 以提高性能。ART在BigBench和MMLU基准测试中, 在未见过的任务上实现了对少样本提示和自动生成CoT推理的显著改进, 并在人类反馈纳入时大大超过了手工制作的CoT提示的性能。ART还受益于自洽性方法, 或者来自于针对工具使用进行训练的新的更强大的LLM。

参考文献

- Simran Arora, Avaniika Narayan, Mayee F Chen, Luca- J Orr, Neel Guha, Kush Bhatia, Ines Chami, Fred-eric Sala和Christopher Ré. 2022年。问我任何问题: 一种激发语言模型的简单策略。arXiv预印本arXiv:2210.02441。
- Luca Beurer-Kellner, Marc Fischer和Martin Vechev。2022年。提示即编程: 一种用于大型语言模型的查询语言。arXiv预印本arXiv:2212.06094。
- Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell等。2020年。语言模型是少样本学习器。arXiv预印本arXiv:2005.14165。
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman等, 2021年。评估在代码上训练的大型语言模型。arXiv预印本arXiv:2107.03374。
- Wenhu Chen, Xueguang Ma, Xinyi Wang和William W Cohen, 2022年。思维提示程序: 将计算与推理从数值推理任务中分离。arXiv预印本arXiv:2211.12588。
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann等, 2022年。Palm: 通过路径扩展语言建模。arXiv预印本arXiv:2204.02311。
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi

- Wang, Mostafa Dehghani, Siddhartha Brahma, 等。2022年。扩展指令微调语言模型。arXiv预印本arXiv:2210.11416。
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, 等。2021年。训练验证者解决数学文字问题。arXiv预印本arXiv:2110.14168。
- Dheeru Dua, Shivanshu Gupta, Sameer Singh和Matt Gardner。2022年。连续提示来分解复杂问题。arXiv预印本arXiv:2212.04092。
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan和Graham Neubig。2022年。Pal: 程序辅助语言模型。arXiv预印本arXiv:2211.10435。
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song和Jacob Steinhardt。2020年。测量大规模多任务语言理解。arXiv预印本arXiv:2009.03300。
- Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark和Hannaneh Hajishirzi。2020年。Unifiedqa: 用单一的问答系统跨越格式边界。arXiv预印本arXiv:2005.00700。
- Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark和Ashish Sabharwal。2022年。分解提示: 解决复杂任务的模块化方法。arXiv预印本arXiv:2210.02406。
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo和Yusuke Iwasawa。2022年。大型语言模型是零-shot推理器。arXiv预印本arXiv:2205.11916。
- Mojtaba Komeili, Kurt Shuster和Jason Weston。2022年。互联网增强对话生成。在第60届年会的计算语言学协会会议论文集(第1卷: 长论文)中, 第8460-8478页, 爱尔兰都柏林。计算语言学协会。
- Angeliki Lazaridou, Elena Gribovskaya, Wojciech Srokowiec和Nikolai Grigorev。2022年。通过少量样本提示增强的互联网语言模型用于开放领域问答。arXiv预印本arXiv:2203.05115。
- Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal和Colin Raffel。2022年。少样本参数高效微调比上下文学习更好更便宜。arXiv预印本arXiv:2205.05638。
- Swaroop Mishra, Daniel Khashabi, Chitta Baral和Hannaneh Hajishirzi。2021年。通过自然语言众包指令实现跨任务泛化。arXiv预印本arXiv:2104.08773。
- Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders等, 2021年。Webgpt: 带有人类反馈的浏览器辅助问答。arXiv预印本arXiv:2112.09332。
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray等, 2022年。训练语言模型以遵循人类反馈的指令。arXiv预印本arXiv:2203.02155。
- Aaron Parisi, Yao Zhao和Noah Fiedel, 2022年。Talm: 工具增强语言模型。arXiv预印本arXiv:2205.12255。
- Arkil Patel, Satwik Bhattamishra和Navin Goyal。2021年。自然语言处理模型真的能够解决简单的数学问题吗? 在2021年北美计算语言学协会会议: 人类语言技术的论文集中, 页码为2080-2094, 线上举行。计算语言学协会。
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith和Mike Lewis。2022年。测量并缩小语言模型中的组合性差距。arXiv预印本arXiv:2210.03350。
- Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja等人。2021年。多任务提示训练使零-shot任务泛化成为可能。arXiv预印本arXiv:2110.08207。
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda和Thomas Scialom。2023年。工具-形式: 语言模型可以自学使用工具。arXiv预印本arXiv:2302.04761。
- Kurt Shuster, Mojtaba Komeili, Leonard Adolphs, Stephen Roller, Arthur Szlam和Jason Weston。2022年。寻求知识的语言模型: 模块化搜索和生成对话和提示完成。arXiv预印本arXiv:2203.13224。
- Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso等。2022年。超越模仿游戏: 量化和推断语言模型的能力。arXiv预印本arXiv:2206.04615。
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou等, 2022年。挑战性的大型基准任务及其是否可以通过思维链来解决。arXiv预印本arXiv:2210.09261。

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yudu, et al., 2022年。Lamda: 用于对话应用的语言模型。arXiv预印本arXiv:2201.08239。

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi和Denny Zhou, 2022年。自治性改善了语言模型中的思维链推理。arXiv预印本arXiv:2203.11171。

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, An-An Andrew M Dai和Quoc V Le。2021年。精细调整的语言模型是零-shot学习者。arXiv预印本arXiv:2109.01652。

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le和Denny Zhou。2022年。思维链提示引发大型语言模型的推理。arXiv预印本arXiv:2201.11903。

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed H Chi, Quoc V Le, Denny Zhou等。思维链提示引发大型语言模型的推理。在《神经信息处理系统进展》中。

Sang Michael Xie, Aditi Raghunathan, Percy Liang, 和Tengyu Ma。2021年。关于上下文学习的解释作为隐式贝叶斯推理。arXiv预印本 arXiv:2111.02080。

Zhuosheng Zhang, Aston Zhang, Mu Li, 和Alex Smola。2022年。大型语言模型中的自动思维链提示。arXiv预印本 arXiv:2210.03493。

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, 和Ed Chi。2022年。最小到最大提示使大型语言模型能够进行复杂推理。arXiv预印本 arXiv:2205.10625。

附录A

A.1 任务库

库设计 我们分析了BigBench中所有200个任务的输入输出实例，过滤掉了英语文本分类和简短答案生成任务，并创建了一个与解决每个任务相关的推理技能列表。我们不关注本文中的长文本理解、长文本生成和多语言任务。我们发现这些任务大多依赖于以下几个常见的技能：

视觉推理、时间推理、命题逻辑、自然逻辑、机器翻译、网络搜索、知识库或数据库查找、递归子问题分解、长文本

理解、数据库操作、代数和算术、代码生成和编辑、文本标记/注释（语言标记）、专业搜索（例如查找语言知识、科学知识等）、字符串编辑、对多个选择进行递归操作、主题分类、证据提取、条件文本生成/编辑和句子相似度。

在这项工作中，我们选择关注覆盖BigBench任务的五个最常用技能（超过91个任务中剩下的50个任务），用于分类（过滤掉长文本理解、生成和多语言任务）。我们从这5个任务集中随机选择2-4个任务，并为这些任务编写了具有适当工具使用的分解程序。这总共包括15个任务组成的任务库。

- 算术：初等数学问题回答，年级学校数学（GSM8K），关于比率的算术问题（Aqua-Rat），导航
- 代码：自动调试，代码描述
- 搜索和问题分解：首字母缩写，多步问题回答（Musique），印度知识，已知未知
- 自由形式推理：形式谬误，超-度
- 字符串操作：第K个字母连接，语言游戏，日期理解

集群程序在每个任务集群中编写的程序在表7中显示涉及字符串编辑和操作的程序，在表8中涉及算术和代数任务，在表-10中涉及代码生成、编辑和调试任务，在表9中涉及搜索世界知识的任务，在表11中涉及从提示“让我们逐步思考”的任务中获益的思维链推理。

程序格式 我们定义了一个解析表达式语法（PEG）（如图4所示），用于描述编写多步推理程序的语言。这个语法被设计用来解析形式为“输入：... Q_1 :...”的完整程序。

#1:... Q_n : [EOQ] Ans:”。我们使用Python库parsing⁴来构建语法并解析LLMs生成的程序。

⁴<https://pypi.org/project/parsing/>

字符串操作

在这些示例中，您会得到一个任务描述和一个输入。将输入分解为子任务以解决任务。您可以使用字符串操作，如拆分、重新格式化、编辑或合并。您还可以使用其他操作，如算术和逻辑。

描述：（日期理解）使用输入中相关事件和日期的信息找到MM/DD/YYYY格式的所需日期。线索：首先找出今天是星期几。

输入：截止日期为2021年6月1日，距离现在还有2天。24小时后的日期是多少（以MM/DD/YYYY格式）？

问题1：[字符串重新格式化] 2021年6月1日以MM/DD/YYYY格式是多少

答案1：06/01/2021

问题2：[算术] 2021年6月1日距离现在还有2天。今天是什么日期？

答案2：今天是04/01/2021

问题3：[算术] 今天的日期之后24小时是什么日期？

答案3：05/01/2021

问题4：[EOQ]

答案：05/31/2021

描述：（语言游戏）将英语翻译成猪拉丁语。

输入：（英语）Sami made his way across the bar and hugged Layla.

问题1：[字符串拆分] "Sami made his way across the bar and hugged Layla." 中有哪些单词？

#1: ["Sami", "made", "his", "way", "across", "the", "bar", "and", "hugged", "Layla", "."]

Q2: [字符串编辑] 将每个单词的初始辅音移到单词的末尾，并在其后添加"ay"。

#2: ["Amisay", "ademay", "ishay", "ayway", "acrossay", "ethay", "arbay", "andyay", "uggedhay", "Aylalay", "."]

Q3: [字符串合并] 将#2连接成一个完整的句子。

#3: Amisay ademay ishay ayway acrossay ethay arbay andyay ugedhay Aylalay.

Q4: [EOQ]

答案：Amisay ademay ishay ayway acrossay ethay arbay andyay ugedhay Aylalay.

描述：（第K个字母串联）取一个单词列表中单词的第3个位置的字母，并用空格连接起来。

输入：取"Savita Saeed Ramos Sato Yadav"中单词的第3个位置的字母，并用空格连接起来。

问题1：[字符串分割]"Savita Saeed Ramos Sato Yadav"中的单词是什么？

#1: ["Savita", "Saeed", "Ramos", "Sato", "Yadav"]

问题2：[字符串索引]在#1中的列表中，每个单词的第3个字母是什么？

#2: ["v", "e", "m", "t", "d"]

问题3：[字符串合并]用空格连接#2

#3: "v e m t d"

问题4：[EOQ]

答案：v e m t d

描述：%s

输入：%s

问题1：

表7：用于需要字符串操作的任务库中的程序。

算术

在这些示例中，你会得到一个任务描述和一个输入。将输入分解为子任务以解决任务。你可以使用sympy库中的函数生成用于解决算术和代数方程的Python代码。

```
from sympy import Symbol
from sympy import simplify
import math
from sympy import solve_it
# solve_it(equations, variable): 解决方程并返回变量值。
```

描述：（水鼠）解决以下关于比率和分数的算术问题，将中间算术计算写成Python代码。将你的结果存储为名为'ans'的变量。

输入：在一次600公里的飞行中，飞机因恶劣天气而减速。它的平均速度减少了200公里/小时，飞行时间增加了30分钟。飞行的持续时间是：A) 1小时 B) 2小时 C) 3小时 D) 4小时 E) 5小时

Q1：[生成Python代码]编写Python代码来解决问题，使用数学和sympy。

```
#1:
duration = Symbol ('duration', positive=True)
delay = 30 / 60
total_disntace = 600
original_speed = total_disntace / duration
reduced_speed = total_disntace / (duration + delay)
solution = solve_it(original_speed - reduced_speed - 200, duration)
ans = solution[duration]
print(ans)
```

Q2：[代码执行]执行#1中的Python代码并获取“ans”的值

```
#2:
1.0
```

Q3：[比较]在A) 1小时B) 2小时C) 3小时D) 4小时E) 5小时选项中，哪个选项与答案最相似？

```
#3: A
```

Q4: [EOQ]

答案：A

描述：（小学数学）解决以下中学算术问题，将中间算术计算写成Python代码。将结果存储为名为'ans'的变量。

输入：珍妮特的鸭子每天下16个蛋。她每天早餐吃三个，还用四个蛋烤松饼给她的朋友。她每天以每个新鲜鸭蛋2美元的价格在农贸市场卖剩下的蛋。她每天在农贸市场赚多少美元？

问题1：[生成Python代码]将算术或代数方程写成Python代码，将答案存储为'ans'

```
#1:
total_eggs = 16
eaten_eggs = 3
baked_eggs = 4
sold_eggs = total_eggs - eaten_eggs - baked_eggs
dollars_per_egg = 2
ans = sold_eggs * dollars_per_egg
print(ans)
```

问题2：[代码执行]执行#1中的Python代码并获取"ans"的值

```
#2: 18
```

问题3: [EOQ]

答案：18

描述：（小学数学）解决以下中学算术问题，将中间算术计算写成Python代码。将结果存储为名为'ans'的变量。

输入：约瑟夫和盖蒂去买冰淇淋，他们一共买了36个冰淇淋。在回来的路上，约瑟夫吃了12个冰淇淋，现在他还剩下2个冰淇淋。

问题1：[生成Python代码]将算术或代数方程式写成Python代码，并将答案存储为'ans'。

```
num_ice_creams_bought_by_joseph = 2 + 12
total_ice_creams = 36
ans = total_ice_creams - num_ice_creams_bought_by_joseph
print(ans)
```

问题2：[代码执行]执行#1中的Python代码并获取"ans"的值

```
#2: 22
```

问题3: [EOQ]

答案：22

描述： %s

输入： %s

问题1：

表8：需要算术运算的任务库中的程序。

搜索

在这些示例中，你会得到一个任务描述和一个输入。将输入分解为子任务以解决任务。
如果信息不足，您可以在一个或多个子步骤中使用类似Google搜索的搜索功能。还可以使用其他函数，如算术和逻辑运算。

描述：（已知或未知）选择最佳答案的选项。如果问题没有已知的答案，请选择“未知”。

输入：尼尔·阿姆斯特朗登月时头上有多少根头发？

选项：未知

选项：五百万

Q1：【搜索】尼尔·阿姆斯特朗登月时头上有多少根头发？

#1:

阿波罗11号（1969年7月16日至24日）是美国的一次载人航天飞行任务，首次将人类送上月球。指挥官尼尔·阿姆斯特朗和登月舱驾驶员巴兹·奥尔德林。

尼尔·奥尔登·阿姆斯特朗（1930年8月5日-2012年8月25日）是一位美国宇航员和航空工程师，成为第一个在月球上行走的人。

Q2：[子问题] 这些信息有助于回答问题吗？由于问题过于具体，涉及个人细节不在公开记录中，答案尚未知晓，或者问题基于观点，可能没有明确的答案。

#2：不。问题过于具体

Q3：[比较] 最终答案是什么？

#3：未知

Q4：[EOQ]

答案：未知

描述：（时代错误）时代错误是指时间顺序上的错误，或者是一个人、事物或事件在错误的时间出现。这个句子中是否包含时代错误？回答是/否。

输入：乔治·H·W·布什总统在海湾战争开始时将他的将军们召集到椭圆形办公室。

Q1：[标签] 这个句子中的实体是什么？

#1:

乔治·H·W·布什总统

海湾战争

Q2：【搜索】乔治·H·W·布什总统任期是什么时候？

#2：乔治·H·W·布什作为美国第41任总统的任期始于1989年1月20日的就职典礼，并于1993年1月20日结束。

Q3：【搜索】海湾战争是在什么时候进行的？

#3：海湾战争[b]是1990年至1991年间由35个国家的军事联盟发动的一场对伊拉克入侵科威特的武装行动。

#4：【子问题】仅根据时间段，这些实体是否可以共存？

是的。它们的时间段相交。

Q5：【生成输出】这是一个时代错误吗？

#5：不是

Q6：【EOQ】

答：不是

描述：（印度教知识）通过选择最佳答案来回答有关印度神话的问题。

输入：在《摩诃婆罗多》中，卡尔纳被诅咒忘记使用哪种武器所需的咒语？

选择：安贾利卡斯特拉

选择：纳拉亚纳斯特拉

选择：阿格涅亚斯特拉

选择：梵天斯特拉

Q1：【搜索】在《摩诃婆罗多》中，卡尔纳被诅咒忘记使用哪种武器所需的咒语？

#1：结果，他诅咒了卡尔纳，说他的武技，包括使用梵天斯特拉，将在他最需要的时候离他而去。因陀罗，众神之王，以蜜蜂的形式叮咬卡尔纳，使他被帕尔舒拉姆所诅咒。卡尔纳绝望地穿过树林，感到诅咒使他感到沮丧。一个技艺高超且忠诚的战士...

Q2：【比较】第3个选项与答案最相似的是哪个选项？

#2：梵天斯特拉

Q3：【EOQ】

答案：梵天斯特拉

描述： %s

输入： %s

Q1:

表格 9：用于需要搜索世界知识的任务库中的程序。

代码生成和编辑

在这些示例中，你会得到一个任务描述和一个输入。将输入分解为子任务以解决任务。如果需要，您可以在一个或多个子步骤中使用 Python 代码生成和执行函数。还可以使用其他函数如算术和逻辑运算。

描述：（自动调试）通过找到答案或错误消息来调试以下代码片段。

输入：

```

if x < 5:

pass

```

这个程序会出现什么错误？

Q1: [执行] 执行以下 Python 代码片段。

if x < 5:

pass

#1:

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'x' is not defined

Q2: [生成答案] 最终的错误消息是什么？

#2: NameError: name 'x' is not defined

Q3: [EOQ]

答案: NameError: name 'x' is not defined

——

描述：（代码描述）给定一个Python代码片段，选择最佳的代码描述选项片段。

输入：

Python代码：

尝试：

n = int(input())

m = int(input())

integer_sum = int(n) + int(m)

print(integer_sum)

except:

print('error')

选择：打印5和6之间的数字

选择：尝试输入并捕获错误

选择：输入字符串'try'

选择：仅在两个输入数字都是整数时打印它们的和，否则引发错误

Q1: [代码生成] 打印5和6之间的数字

#1:

import random

print(random.uniform(5,6))

Q2: [代码生成] 尝试输入并捕获错误

#2:

尝试：

文件 = open(file_name, "r")

data = file.read()

file.close()

except:

print('error')

Q3: [代码生成] 输入字符串'try'

#3: 打印('try')

Q4: [代码生成] 仅当两个输入数字都是整数时，打印它们的和，否则引发错误

#4:

#!/usr/bin/python

a=raw_input("输入第一个数字：")

b=raw_input("输入第二个数字：")

try:

sum=int(a)+int(b)

print "和为：",sum

except:

print "只能输入整数值"

Q5: [比较] 生成的代码片段中哪个最像原始代码？

#5: 仅当两个输入数字都是整数时，打印它们的和，否则引发错误

Q6: [EOQ]

答案: 仅当两个输入数字都是整数时，打印它们的和，否则引发错误

——

描述： %s

输入： %s

Q1:

表10：用于需要代码生成或编辑的任务的任务库中的程序

CoT 推理

在这些示例中，你会得到一个任务描述和一个输入。将输入分解为子任务以解决任务。

明确思考问题可以是你使用的子步骤之一

描述：（体育理解）确定一个与体育相关的人工构造句子是否合理。

最终答案应该是“是”或“否”。

输入：以下句子是否合理？“桑蒂·卡索拉进了一个触地得分。”

Q1: [逐步思考]

#1: 让我们逐步思考。桑蒂·卡索拉是一名足球运动员。触地得分是美式足球和橄榄球的一部分。所以答案是不。Q2: [EOQ]答案：不—

描述：（夸张法）从两个选择中识别出正确的形容词顺序。这涉及选择一个母语为英语的人会认为更加“直观”的句子。

输入：哪个句子有正确的形容词顺序：

选项：

(A) 令人厌恶的小型巴西锻炼船

(B) 巴西令人厌恶的锻炼小船

Q1: [逐步思考]

#1: 让我们逐步思考。当一个名词前有多个形容词时，形容词需要遵守以下顺序：“[1. 观点] [2. 大小] [3. 年龄] [4. 形状] [5. 颜色] [6. 来源] [7. 材料] [8. 目的] 名词”。

选项 (A)：“令人厌恶的小型巴西锻炼船”。(1) “令人厌恶的”属于意见类别。(2) “小型”属于尺寸类别。(3) “巴西的”属于起源类别。(4) “锻炼”属于目的类别。选项 (A) 的形容词顺序如下：[1. 意见] [2. 尺寸] [6. 起源] [8. 目的]（或者用数字表示，1 2 6 8）。因为 $1 < 2 < 6 < 8$ 是正确的，所以 (A) 有正确的顺序。

选项 (B)：“巴西的令人厌恶的锻炼小型船”。选项 (B) 的形容词顺序如下：[6. 起源] [1. 意见] [8. 目的] [2. 尺寸]（或者用数字表示，6 1 8 2）。因为 $6 < 1 < 8 < 2$ 不正确，所以 (B) 没有正确的顺序。因此答案是 (A)。

Q2: [EOQ]

] 答案：(

A) —

描述：（形式谬误）区分演绎有效的三段论论证和形式谬误，特别注意否定。

输入：“看清谁与谁有亲属关系，以及关系的方式并不总是容易。”以下论证与此问题有关：首先，莱斯利是费尔南多的亲密朋友。此外，成为费尔南多的亲密朋友或洛厄尔的同学足以成为勒罗伊的曾祖父。由此可见，莱斯利是勒罗伊的曾祖父。根据明确陈述的前提，这个论证是演绎有效还是无效的？

选项：-

有效

- 无效 Q1: [逐

步思考] #1:

让我们逐步思考。

(1) 莱斯利是费尔南多的亲密朋友：莱斯利 = 朋友（费尔南多）。

(2) 成为费尔南多的亲密朋友或洛厄尔的同学足以成为勒罗伊的曾祖父：如果 $X =$

friend(费尔南多)或SCHOOLMATE(洛厄尔)，那么 $X = \text{great-grandfather}(\text{勒罗伊})$ 。

假设：勒斯利是勒罗伊的曾祖父吗？勒斯利 = great-grandfather(勒罗伊)？

让我们看看是否可以通过逻辑推理从论据 (1) 和 (2) 中推导出假设？

根据 (1)，我们有勒斯利 = friend(费尔南多)。根据 (2)，我们有如果勒斯利 = friend(费尔南多)，那么勒斯利 = great-grandfather(勒罗伊)。

所以，勒斯利是勒罗伊的曾祖父是真的。所以答案是有效的。

Q2: [EOQ]

答：有效

描述：（关于有色物体的推理）给定文本输入中的一组有色物体，回答输入末尾的问题。

输入：床头柜上有一支红色铅笔，一个紫色马克杯，一个酒红色钥匙扣，一个紫红色泰迪熊，一个黑色盘子和一个蓝色压力球。压力球是什么颜色？

Q1: [逐步思考]

#1: 让我们逐步思考。根据这个问题，压力球的颜色是蓝色。所以答案是蓝色。

Q2: [EOQ]

答案：蓝色

描述：%s

输入：%s

Q1: ""

表11：用于需要自由形式思维推理逻辑和语言的任务的任务库中的程序。

A.2 任务选择

当提供新的任务描述和输入实例时，ART从任务库中检索N个任务来构建动态多任务提示。我们探索了两种任务选择策略。

基于任务聚类的50个示例用于调优，除非有少于100个示例的情况，否则我们将这个数字减少到10个。

我们在库中迭代遍历所有五个任务集群，每次只从一个集群中向LLM提供演示程序。例如，在一个迭代中，我们只使用算术任务中的程序作为提示的演示。选择在保留示例集（50个）上性能最高的任务集群。这种策略需要与任务集群数量相同数量的API调用，并且需要一个新任务的保留输入输出对集合。请注意，对于新任务，不需要额外的监督来生成分解程序。

基于LLM相似性的LLM被提示使用任务对。一些对包含来自同一集群的两个任务，并标记为“相似”，而一些对则不是，并标记为“不相似”。此外，我们还为决策提供推理——“基础数学QA和GSM8K是相关任务，因为它们都需要解决算术问题”。此提示中的任务由其名称、指令和一些输入输出对表示。我们使用表13中的提示来提示LLMs。

对于每个图书馆任务与新任务配对，LLM被提示做出决策。我们选择按照“相似”与“不相似”的对数概率比例排名前N个任务。这种策略需要更少的留存示例，但容易受到每次实验运行中选择的任务而导致性能的高方差影响。对于PQA，基于LLM相似性选择的最相似任务是时代错误和GSM8K。在表12中，我们研究了在ART中改变任务选择策略的影响。我们不再选择在50个示例上具有最高留存

性能的任务集群，而是使用LLM相似性分数为提示选择任务程序。与在留存集上调整性能并且在多次运行中由LLM选择不同任务的平均性能相比，这种策略平均较差，并且具有高方差。选择共享子任务和工具的相似任务（没有任何su-

对于LLMs来说，自动推理和工具使用仍然是一个具有挑战性的任务，并且将在未来的工作中进一步探索这个方向。

A.3 工具使用

代码生成我们使用Codex (Chen et al., 2021)模型进行代码生成。代码生成的论证是前一个子任务的答案序列“`“#i - 1:...”`”和LM在子任务查询符号“`“Qi: [生成Python代码]”`”之后生成的序列。

当 $i = 1$ 时，实例输入被用作第一个参数。我们包括前一个答案/输入，因为它通常包含与生成准确代码相关的信息，比如需要生成代码的算术问题（请参见表8中的示例）。两个参数作为多行Python注释提供给Codex，同时保持它们的原始格式。为了保持答案变量的一致性，我们还附加了一个额外的指令：将最终答案存储在变量'ans'中并打印出来。例如：

珍妮特的鸭子每天下16个蛋。她每天早上吃三个蛋作为早餐，并且每天用四个蛋为她的朋友烤松饼。她每天将剩下的蛋以每个新鲜鸭蛋2美元的价格在农贸市场出售。她每天在农贸市场能赚多少美元？

用于提示Codex的方式如下：

"""

珍妮特的鸭子每天下16个蛋。她每天早上吃三个蛋作为早餐，并且每天用四个蛋为她的朋友烤松饼。她每天将剩下的蛋以每个新鲜鸭蛋2美元的价格在农贸市场出售。她每天在农贸市场能赚多少美元？

将算术或代数方程式以Python代码的形式写下来，将答案存储为'ans'并打印出来。

Codex生成温度设置为0.3，最大长度为500个标记，使用“`print(ans)`”作为停止准则。

代码编辑：我们使用Codex (Chen等, 2021年)模型进行代码生成和代码编辑。两者的参数都包括之前的

	简单文本	计算机科学 算法	策略问答	物理学 问题解释	单位	关于编辑的推理 彩色物体
最佳任务群集	27.65	88.11	66.44	20.37	53.99	64.34
基于LLM的任务模拟	38.30	83.71	60.39	14.06	43.56	62.00

表12：使用两种类似的任务选择策略比较GPT3（175B）模型上的ART结果。基于LLM的相似度平均较选择最佳任务群集更差。

提示LLM选择相似的任务
给出两个任务及其描述，以及任务的输入和输出示例，确定它们是否相似。如果两个任务需要共同的子任务，如字符串操作、网络搜索、翻译、算术、代码执行等，则它们是相似的。——
任务1：[日期理解] 根据输入中相关事件和日期的信息，找到所需的日期（MM/DD/YYYY格式）。输入：截止日期是2021年6月1日，距离现在还有2天。请问24小时后的日期是几月几日（MM/DD/YYYY格式）？ 最终答案是05/01/2021。
任务2：[语言游戏] 将英文翻译成Pig Latin。输入：英文句子为"Sami made his way across the bar and hugged Layla"。最终答案是"Amisay ademay ishay ayway acrossyay ethay arbay andyay uggedhay Aylalay"。它们是否相似？是的。它们都需要以特定的字符串格式回答。
——
任务1：[第K个字母拼接] 取一个单词列表中每个单词的第3个字母，并使用空格将它们连接起来。输入："Savita Saeed Ramos Sato Yadav"中的单词是什么？最终答案是"v e m t d"。
任务2：[语言游戏] 将英语翻译成Pig Latin。输入：英语句子是"Sami made his way across the barand hugged Layla"。最终答案是"Amisay ademay ishay ayway acrossyay ethay arbay andyay uggedhay Aylalay." 这两者相似吗？是的。它们都需要访问和操作字符串中的字符。
——
任务1：[第K个字母拼接] 取一个单词列表中每个单词的第3个字母，并使用空格将它们连接起来。输入："Savita Saeed Ramos Sato Yadav"中的单词是什么？最终答案是"v e m t d"。
任务2：[已知未知] 选择最佳答案。如果问题没有已知答案，选择"未知"。输入：Neil Armstrong登上月球时头发有多少根？最终答案是"未知"。
这些相似吗？不是。任务1需要操作字符串，任务2需要通过可能在网上查找信息来回答问题。
——
任务1：【时代错误】时代错误是指时间顺序上的错误，或者是指不符合其正确时间的人、事物或事件。这个句子包含时代错误吗？输入：库尔特·科本在1980年的电视节目《双峰》中担任主演。最终答案是“是”。
任务2：【已知未知】选择最佳答案的选项来回答问题。如果问题没有已知答案，选择“未知”。输入：马克·吐温出生在哪里？最终答案是密苏里州的佛罗里达。
这些相似吗？是的。它们都需要搜索文本中提到的实体的信息，比如库尔特·科本或马克·吐温。
——
任务1：【印度教知识】通过选择最佳答案的选项来回答有关印度神话的问题。 输入：在《摩诃婆罗多》中，卡尔纳被诅咒忘记使用哪种武器所需的咒语？选项：Anjalikastra, Narayanastra, Agneyastra, Brahmastra。最终答案是Brahmastra。
任务2：[代码调试]通过找到答案或错误消息来调试以下代码片段。输入： if x < 5: pass 最终答案是 Traceback (most recent call last): File "<stdin>", line 1, in <module> NameError: name 'x' is not defined
这些相似吗？不是。任务1是关于回答问题并需要搜索文本中提到的实体信息。任务2是关于调试代码的问题，可能需要Python解释器。
任务1： %s 任务2： %s 这些相似吗？

表格13：任务库中的程序。

```

grammar = parsimonious.grammar.Grammar(
    r"""
    program=program_start*node*partial_command*final_answerp
    program_start=input_start~r"(\s)"text~r"input_start=
    ~r"输入: "
    text=~r"(?<=输入: (\s))(\s| |
    )*(?=Q[0-9]+)"node=command_node~r"output_node~r"
    command_node=command_start~r"(\s)"command_instructionout
    put_node=begin_answer~r"(\s)"output
    command_instruction=~r"(?<=\\s)(\s| | )*(?=\s[0-9]+)"command_s
    tart=~r"Q[0-9]+:\s\[A-Za-z_\s]+\]"begin_answer=~r"\s[0-9]+:"

    output=~r"(?<=\\s[0-9]+:(\s))(\s| | )*(?=Q[0-9]+)"partial_co
    mmand=command_start~r"final_answer=~r"答案: (\s)(\s)*$""")

```

图4：用于解析ART程序的PeG语法

子任务的答案序列“ $\#i-1:\dots$ ”（或输入如果 $i=1$ ），以及子任务查询符号“ Q_i ：[生成Python代码]”后由LM生成的序列。第一个参数是需要编辑的代码片段，第二个参数是用作编辑/生成指令的Python中的多行注释。为了确保后续代码执行产生与变量名无关的答案字符串，编辑指令是打印所需变量。例如，在任务库中的自动调试任务中，以下程序片段：

输入：

```

'''
x = set([1, 1, 2, 3])
'''

```

程序执行后，x的值是多少？

Q1：[代码编辑] 编辑代码以打印x的值

用于提示Codex进入编辑模式，如下所示。
对于代码输入：

```
x = set([1, 1, 2, 3])
```

对于编辑指令：

编辑代码以打印x的值

代码执行我们在预安装了算术、符号和科学计算包的虚拟python环境中运行python代码。代码执行的参数包括前一个子任务的答案序列“ $\#i-1:\dots$ ”，这是需要执行的python代码片段。如果 $i=1$ ，则输入包含代码。另一个参数是子任务查询符号“ Q_i ：[执行代码]”之后生成的序列（

```

from sympy.solvers import solve
from sympy import Symbol, Eq, simplify
import math
import numpy as np
import cvxpy as cp
import statistics

```

```

def solve_it(equation, variable):
    解=solve(equation, variable, dict=True)
    如果 解为空:
        如果 isinstance(variable, list):
            解=变量: None for 变量 in variable
        否则:
            解=variable: None
    返回 解
    否则:
        解 = 解[0]
    返回 解

```

表14：在代码片段执行之前附加的代码前缀。

再次，为了鼓励具有一致变量使用的可执行代码，我们将简单地将代码片段作为注释附加。我们还将序列“将你的最终答案存储在变量'ans'中”附加到注释中。执行调用的结果用于替换语言模型生成的答案序列。

最后，我们在代码片段之前添加了一个包含有用的模块和函数导入的代码片段，以便成功执行调用外部模块（如numpy和scipy）的函数。表14显示了这个代码前缀。我们使用exec原生的Python函数来执行代码片段，并访问'ans'局部变量（如果存在）。

知识库查找 此工具已添加到单词解谜任务中。此函数调用用于在关系知识库中按键查找数据。例如，我们在单词解谜任务中使用字典查找。该函数的输入再次是前一个子任务的答案序列（如果存在，则使用原始输入），以及

函数名称符号后由语言模型生成的序列。第一个参数被解析为Python代码片段，并解释为查找键的列表。第二个参数被解析为代码生成提示，随后被执行。例如， $l = ['yob', 'boy', 'oyb']$ 和第二个参数检查这些单词列表中的哪些是英语单词。将最终答案存储为'ans'并打印出来。结果是以下代码片段和最终答案 'boy'：

```
def lookup(word_list):
    import enchant
    d = enchant.Dict("en_US")
    valid_list = []
    for word in word_list:
        如果 d.check(word):
            valid_list.append(word)
    return valid_list
```

虽然这是对于一般知识库查找或查询的限制性定义，但我们探索了如何利用人机协同反馈来创建自定义查找工具。

Prolog 引擎该工具已添加到形式逻辑谬误任务中。该任务包括用自然语言陈述的一阶逻辑语句，如下所示：

首先，Bonnie 是 Miranda 的同学。此外，任何与 Aubrey 是同事的人都不是 Miranda 的同学。这一切都意味着 Bonnie 不是 Aubrey 的同事。根据明确陈述的前提条件，这个论证是演绎有效还是无效的？

这可以用 Prolog ⁵写成：

```
workmate(X, aubrey) :- \+ schoolmate(X, miranda).
schoolmate(bonnie, miranda).
?- workmate(bonnie, aubrey).
```

人们通过使用一个新的工具符号“[翻译为prolog]”为一些实例编写这样的pro-log语句来提供反馈。然后，他们编写一个新的工具，调用一个Python prolog解析引擎来执行prolog代码并确定最终表达式的二进制值。这被整合回程序中。

A.4 基准

Few-shot基准这是直接提示的基准，其中提示只包含输入-输出对，没有额外的中间推理步骤。根据之前报告使用直接提示的工作（Suzgun等，2022；Wei等，2022），我们选择了3个随机选择的输入-输出实例。我们对Instruct GPT (text-davinci-002)（Ouyang等，2022）和Codex (code-davinci-002)（Chen等，2021）都运行了直接提示，并报告了更高的性能。这是根据（Chung等，2022）得出的结论，他们发现即使在直接提示的情况下，Codex模型在分析任务上也比文本模型更好。

Auto CoT 一个基线模型，可以在自由形式的自然语言中生成自动的CoT风格的多步推理（如AutoCoT（Zhang等，2022）所做的那样）。在数据集中随机选择的子集用于提示LLM引出CoT风格的推理（输入+让我们逐步思考）。由于CoT风格的生成是自由形式的，解析潜在的工具使用符号更加困难，因此我们不在这个基线模型中使用工具。这个基线模型专门衡量了我们用于编写程序和解析工具调用的自定义查询语言（和PeG语法）的有效性；虽然（Zhang等，2022）对训练样例进行聚类以向LLM提供多样化的演示，但我们选择了随机选择的5个样例。对于ART，还可以仔细选择演示样例，我们将这个选择的探索留给未来的工作。我们解析生成的CoT风格推理以提取答案字符串，并将短语“最终答案是”和答案字符串添加到推理的末尾。这种模式用于评估。

最佳GPT-3方法：我们简要描述了在表2和表3中报告的GPT-3最佳结果，这些结果对应于使用多步推理（如CoT）和工具使用的方法，并且在两者都有人类监督的情况下。

- （Suzgun等，2022年）：在BigBench中为几个任务进行人工编写的CoT推理。对他们手工制作的提示进行仔细检查后发现，他们将BigBench任务转化为多项选择任务（选择A、B、C等选项），这与最初提出并在本研究中使用的更具挑战性的格式不同。因此，我们进行了修改。

⁵<https://zh.wikipedia.org/wiki/Prolog>

任务	CoT +人类		ART +人类		人类反馈
第k个字母连接*	0.64	59.40	40.0	100.0	代码C: 提取和合并单词列表的第k个字母
语言游戏*	18.58	26.08	23.08	35.38	代码C: 英文-> Pig Latin和反之亦然
时代错误*	51.48	49.82	75.66	82.91	C: 搜索查询限制提取时间段
自动调试*	38.24	61.18	62.94	67.05	代码 C: 修复代码编辑以打印输入中要求的变量 A: “[生成答案]最终错误消息是什么?”
导航	61.7	85.9	72.4	80.89	代码 C: 正确的前进、后退、右转、左转距离
日期理解	38.9	70.4	52.05	65.45	A: 首先找出今天是什么日期
形式谬误	56.4	56.4	64.76	74.39	T: 翻译为Prolog并添加Prolog引擎

表15: 通过自一致性和人在循环反馈上改进ART和自由形式CoT的图书馆任务。

(*) 表示对这些任务的自动生成CoT推理进行了人在循环改进。

他们的提示在输出空间中生成答案，这与ART进行了公平比较。因此，表2和表3中报告的结果可能与（Suzgun等人，2022）中报告的结果不同。

• 表20用于解开英语单词的混乱，其中在第2步中编辑代码以获得正确的排列，并在第3步中定义了一个新工具来查找英语词典并返回实际单词的排列。

- (Khot等人，2022)：用于列表反转和第k个字母比较的分解提示。
- (Press等人，2022年)：分解的多步音乐问答。
- (Chen等人，2022年)：思维程序化：提示被设计为将算术问题转化为代码并执行。
- (Ouyang等人，2022年)：InstructGPT
- (Chung等人，2022年)：PaLM语言模型。请注意，我们仅考虑此论文中报告的最高文本-davinci-002或代码-davinci-002结果与ART进行比较。
- (Chen等人，2021年)：Codex

在表15中，我们展示了任务库中训练任务的人类反馈结果。

A.5 人类反馈

我们提供了人类对五个BigBench任务的反馈。

- 表16用于推理彩色物体，其中编辑的代码引入了一种一致的数据结构来表示输入。
- 表17用于物理学，其中引入了子步骤，提示给定的物理量、所需的物理量和连接它们的物理公式。
- 表18用于时间序列，其中引入了子-步骤，逐个检查所有选项，以确定输入中的主题在那些时间是否忙于某项活动。
- 表19用于跟踪洗牌后的对象，其中编辑代码引入了一种一致的数据结构来表示输入，并修正了交换逻辑。

关于有色物体的推理
之前
<p>输入：在床头柜上有两本银色教科书和三个银色猫玩具。如果我从床头柜上移走所有的猫玩具，上面还剩下多少个紫红色的东西？</p> <p>Q1: [生成Python代码]将算术或代数方程作为Python代码写下，将答案存储为'ans'</p> <pre> l: num_of_silver_textbooks = 2 num_of_silver_cat_toys = 3 num_of_fuchsia_things = num_of_silver_textbooks num_of_things_on_nightstand = num_of_silver_textbooks + num_of_silver_cat_toys ans = num_of_things_on_nightstand - num_of_silver_cat_toys print(ans) </pre> <p>Q2: [代码执行] 执行Python代码并获取"ans"的值</p> <p>2: 2</p> <p>Q3: [EOQ]</p> <p>答案: 2</p>
之后
<p>输入：在床头柜上有两本银色教科书和三个银色猫玩具。如果我从床头柜上移走所有的猫玩具，上面还剩下多少个紫红色的东西？</p> <p>Q1: [生成Python代码] 编写Python代码，找出如果我从床头柜上移除所有的猫玩具，上面还剩下多少个紫红色的物品#1:</p> <pre> color_object_count = [("silver", "textbooks", 2), ("silver", "cat toys", 3)] removed_objects = [("silver", "cat toys", 3)] def remove_objects(color_object_count, removed_objects): color_object_count_updated = [] for color, object, count in color_object_count: 对于removed_objects中的color2、object2和count2 如果color等于color2且object等于object2: count减去count2 color_object_count_updated.append([color, object, count]) 返回color_object_count_updated color_object_count = remove_objects(color_object_count, removed_objects) ans等于0 对于color_object_count中的color、object和count 如果color等于"fuchsia": ans等于count 打印(ans) </pre> <p>Q2: [执行代码] 执行Python代码片段。</p> <p>#2: 0</p> <p>Q3: [EOQ]</p> <p>答案: 0</p>

表16：人类反馈：关于彩色物体的推理

物理学
之前
<p>输入：识别对于以下每个问题，哪个物理公式对于找到答案最有用。一个质量为6.8×10^{-5}克，电荷为58毫库仑的粒子在一个电场均匀且在x方向上为7.2N/C，在y和z方向上为零的空间区域中运动。如果粒子的初始速度由$v_y = 1.7 \times 10^6$ m/s, $v_x = v_z = 0$给出，那么在0.7秒时粒子的速度是多少？</p> <p>选择：F = m * a 选择：v = dx / dt 选择：E = q / ($\epsilon \cdot A \cdot \cos(\theta)$) 选择：a = dv / dt Q1：[子问题] 这个问题可以逐步回答吗？ #1：是的。 Q2：[搜索] 一个质量为6.8×10^{-5}克，电荷为58毫库仑的粒子在一个电场为7.2 N/C（x方向）且在y和z方向为零的空间区域中运动。如果粒子的初始速度为$v_y = 1.7 \times 10^6$ m/s, $v_x = v_z = 0$，在0.7秒时粒子的速度是多少？</p> <p>#2：v = dx / dt 一个质量为6.6×10^{-5}克，电荷为22毫库仑的粒子在一个电场为5.6 N/C（x方向）且在y和z方向为零的空间区域中运动...</p> <p>Q3：[比较]问题2的答案与哪个选项最相似？ #3: v = dx / dt Q4: [EOQ] 答案: v = dx / dt</p>
之后
<p>输入：识别对于以下每个问题，哪个物理公式对于找到答案最有用。一个质量为6.8×10^{-5}克，电荷为58毫库仑的粒子在一个电场均匀且在x方向上为7.2N/C，在y和z方向上为零的空间区域中运动。如果粒子的初始速度由$v_y = 1.7 \times 10^6$ m/s, $v_x = v_z = 0$给出，那么在0.7秒时粒子的速度是多少？</p> <p>选择：F = m * a 选择：v = dx / dt 选择：E = q / ($\epsilon \cdot A \cdot \cos(\theta)$) 选择：a = dv / dt Q1：[子问题]在"质量为6.8×10^{-5}克，电荷为58毫库仑的粒子在一个电场均匀的空间中运动，电场在x方向上为7.2 N/C，在y和z方向上为零。"中，被询问的是什么数量？如果粒子的初始速度由$v_y = 1.7 \times 10^6$ m/s, $v_x = v_z = 0$给出，在0.7秒时粒子的速度是多少？"#1：粒子的速度</p> <p>Q2：[子问题]在"质量为6.8×10^{-5}克，电荷为58毫库仑的粒子在一个电场均匀的空间中运动，电场在x方向上为7.2 N/C，在y和z方向上为零。"中，还提供了哪些信息？如果粒子的初始速度由$v_y = 1.7 \times 10^6$ m/s, $v_x = v_z = 0$给出，在0.7秒时粒子的速度是多少？"#2：粒子的质量（6.8×10^{-5}克），电荷（58毫库仑），电场（7.2 N/C），初始速度（$v_y = 1.7 \times 10^6$ m/s, $v_x = v_z = 0$）</p> <p>Q3：【搜索】给定粒子的质量、电荷和初始速度，以及电场，我们可以计算出粒子的什么？给出物理公式。</p> <p>#3: 我们可以计算出粒子的加速度，如下所示： $a = (q/m) \cdot E$ 这个公式是从洛伦兹力定律推导出来的，该定律规定了电场中带电粒子的力（F），公式如下所示：F = ma = qE</p> <p>Q4：【子问题】在选择中，与 F = ma = qE, $a = (q/m) \cdot E$ 最接近的物理公式是哪个？ #4: F = m * a Q5：【EOQ】 答案: F = m * a</p>

表17：人类反馈：物理学

时间序列
之前
<p>输入：今天，蒂芙尼去了游泳池。他们可能在什么时间之间去的？</p> <p>我们知道：</p> <p>蒂芙尼早上6点醒来。</p> <p>杰西卡看到蒂芙尼从早上6点到早上10点在机场等候。</p> <p>威廉看到蒂芙尼从下午12点到下午5点在学校上课。</p> <p>大卫看到蒂芙尼从下午5点到下午6点朝自由女神像走去。</p> <p>詹姆斯看到蒂芙尼从下午6点到晚上8点在图书馆阅读。</p> <p>游泳池在晚上8点后关闭。</p> <p>蒂芙尼在什么时间段内可能去游泳池？</p> <p>选择：下午12点到下午5点</p> <p>选择：下午6点到晚上8点</p> <p>选择：上午10点到中午12点</p> <p>选择：早上6点到早上10点</p> <p>Q1: [逐步思考]</p> <p>#1: 让我们逐步思考。蒂芙尼今天去了游泳池。我们需要找出她可能去过的时间。</p> <ul style="list-style-type: none"> - 蒂芙尼早上6点醒来。 - 杰西卡看到蒂芙尼从早上6点到早上10点在机场等候。 - 威廉看到蒂芙尼从下午12点到下午5点在学校上课。 - 大卫从下午5点到6点看到蒂芙尼朝自由女神像走去。 - 詹姆斯从下午6点到8点看到蒂芙尼在图书馆阅读。 - 游泳池在8点后关闭。 <p>游泳池在8点后关闭，所以蒂芙尼不可能在下午6点到8点去过。她在下午6点到8点在图书馆阅读，所以她不可能在上午10点到12点去过。她从早上6点到10点在机场等候，所以她不可能在早上6点到10点去过。那就剩下中午12点到下午5点。所以答案是中午12点到下午5点。</p> <p>Q2: [EOQ]</p> <p>答案：中午12点到下午5点</p>
之后
<p>Q1: [子问题] 蒂芙尼中午12点到下午5点有空吗？</p> <p>#1: 不，她正在学校上课。</p> <p>Q2: [子问题] Tiffany在下午6点到8点有空吗？</p> <p>#2: 不，她正在图书馆看书。</p> <p>Q3: [子问题] Tiffany在上午10点到12点有空吗？</p> <p>#3: 是的，她的日程中没有其他活动。</p> <p>Q4: [子问题] Tiffany在早上6点到10点有空吗？</p> <p>#4: 不，她正在机场等候。</p> <p>Q5: [子问题] 游泳池从上午10点到下午12点开放吗？</p> <p>#5: 是的，游泳池从上午10点到下午8点开放。</p> <p>Q6: [获取答案] 总结一下，她应该什么时候去游泳池？</p> <p>#6: Tiffany应该在上午10点到下午12点去游泳池。</p> <p>Q7: [结束问题]</p> <p>答案：上午10点到下午12点</p>

表18：人类反馈：时间序列

追踪洗牌的物体
之前
<p>输入：爱丽丝、鲍勃、克莱尔、戴夫和伊芙正在玩一个游戏。游戏开始时，他们每个人都拿着一个球：爱丽丝有一个粉色的球，鲍勃有一个白色的球，克莱尔有一个红色的球，戴夫有一个紫色的球，伊芙有一个黄色的球。随着游戏的进行，玩家之间交换球。首先，爱丽丝和戴夫交换球。然后，克莱尔和伊芙交换球。然后，爱丽丝和鲍勃交换球。然后，戴夫和克莱尔交换球。最后，爱丽丝和克莱尔交换球。游戏结束时，克莱尔拥有</p> <p>Q1: [生成Python代码]将算术或代数方程写成Python代码，并将答案存储为'ans'</p> <pre>#1: Alice = "pink" Bob = "white" Claire = "red" Dave = "purple" Eve = "yellow" Alice_and_Dave = [Alice, Dave] Claire_and_Eve = [Claire, Eve] Alice_and_Bob = [Alice, Bob] Dave_and_Claire = [Dave, Claire] Alice_and_Claire = [Alice, Claire] ans = Claire print(ans)</pre> <p>Q2: [代码执行] 执行Python代码并获取"ans"的值</p> <pre>#2: 红色 Q3: [EOQ] 答案: 红色</pre>
之后
<p>问题1: [生成Python代码]将算术或代数方程式写成Python代码，并将答案存储为'ans'。</p> <pre>players = ['Alice', 'Bob', 'Claire', 'Dave', 'Eve'] balls = ['粉色', '白色', '红色', '紫色', '黄色'] swap_list = [[0, 3], [2, 4], [0, 1], [3, 2], [0, 2]] for swap in swap_list: balls[swap[0]], balls[swap[1]] = balls[swap[1]], balls[swap[0]] ans = [players[i] + ' 有一个 ' + balls[i] for i in range(5)] print(ans)</pre> <p>Q2: [代码执行] 执行Python代码并获取"ans"的值</p> <pre>#2: ['Alice 有一个粉色', 'Bob 有一个紫色', 'Claire 有一个白色', 'Dave 有一个黄色', 'Eve 有一个红色'] Q3: [子问题] 最后，克莱尔有什么颜色？ #3: 白色 Q3: [EOQ] 答案: 白色</pre>

表格19：人类反馈：追踪洗牌的物体

追踪洗牌的物体
之前
<p>输入：单词illocnecos是英语单词的一个乱序版本</p> <p>Q1: [字符串分割] “illocnecos” 中的字母是什么？</p> <pre>#1: ["i", "l", "l", "o", "c", "t", "n", "e", "c", "o", "s"] Q2: [字符串排列] #1中字母的可能排列是什么？ #2: ["i", "l", "l", "o", "c", "t", "n", "e", "c", "o", "s"], ["i", "l", "l", "o", "c", "t", "n", "e", "o", "c", "s"], ["i", "l", "l", "o", "c", "t", "e", "n", "c", "o", "s"], ["i", "l", "l", "o", "c", "t", "e", "n", "o", "c", "s"], ["i", "l", "l", "o", "c", "n", "t", "e", "c", "o", "s"], ... Q3: [字符串检查] 在#2中的排列中，哪一个是英语单词？ #3: ["i", "l", "l", "o", "c", "t", "n", "e", "c", "o", "s"] Q4: [EOQ] 答案: ["i", "l", "l", "o", "c", "t", "n", "e", "c", "o", "s"]</pre>
单词解谜：之后
<p>Q1: [字符串拆分] "illocnecos" 中的字母是什么？</p> <pre>#1: ["i", "l", "l", "o", "c", "t", "n", "e", "c", "o", "s"] Q2: [字符串排列] #1中字母的可能排列是什么？ #2: ["illocnecos", "illocneocs", "illocnecos", "illocnecos", "illocnecos", ...] Q3: [单词查找] 在 #2 的排列中，哪一个是英语单词？ #3: collections Q4: [EOQ] 答案: collections</pre>

表格20：人类反馈：单词解谜