

《Chameleon: Plug-and-Play Compositional Reasoning with Large Language Models》

复现报告

摘要

大语言模型 (LLMs) 由于新兴的推理能力, 在解决各种自然语言处理任务方面取得了显著进展。然而, LLMs 由于固有的限制, 无法访问最新信息 (存储在网络上或任务特定的知识库中), 使用外部工具以及进行精确的数学和逻辑推理。在本文中提出了 Chameleon, 这是一个通过为 LLMs 增加即插即用的组合推理模块来缓解这些限制的 AI 系统。Chameleon 通过组合各种工具 (例如 LLMs、现成的视觉模型、网络搜索引擎、Python 函数和基于启发式的模块) 来合成程序, 以完成复杂的推理任务。Chameleon 的核心是基于 LLM 的规划器, 它组装一系列工具来执行并生成最终的响应。本文展示了 Chameleon 在两个多模态知识密集型推理任务 ScienceQA 和 TabMWP 上的有效性。由 GPT-4 驱动的 Chameleon 在 ScienceQA 上实现了 86.54% 的整体准确率, 比已发表的少样本结果提高了 11.37%。在 TabMWP 上, 由 GPT-4 驱动的 Chameleon 将准确性提高了 17.0%, 将最先进技术推向了 98.78%。本文的分析还显示, 与 ChatGPT 驱动的计划者相比, GPT-4 驱动的计划者通过从指令中推断潜在约束来展示更一致和理性的工具选择。

关键词: 大语言模型; 组合工具; 自然语言处理;

1 引言

近期, 对于各种自然语言处理任务, 如 GPT-3 [2]、PaLM [3]、LLaMA [18]、ChatGPT [12] 以及最近开发的 GPT-4 [13] 等, 大型语言模型 (LLMs) 取得了显著的进展。LLMs 展示了出现的能力, 包括上下文学习和思维链推理 [19]。这些模型能够以 zero-shot 方式 [7] 或在 few-shot 的帮助下 [20] 解决各种任务, 并展示了在规划和决策方面与人类类似的巨大潜力 [5]。尽管具备这些能力, LLMs 仍然面临固有的限制, 例如无法获取最新信息、进行精确的数学推理或利用专门的模型。因此, 增强当前的 LLMs 以具备自动组合外部工具来解决现实世界任务的能力对于解决这些缺点至关重要。

2 相关工作

2.1 神经模块化推理方法与挑战

组合推理神经模块化和组合方法已被广泛研究，用于自动执行所需的子任务分解，这增强了在各种推理任务中的可解释性和适应性。早期的工作认为复杂的推理任务基本上是组合的 [1]，并提出了神经模块网络，将其分解为子任务。然而，这些方法依赖于脆弱的现成解析器，并受到模块配置的限制。一些后续工作通过使用强化学习 [21] 和弱监督学习，进一步改进了这一方法，通过端到端预测特定实例的网络布局，而不依赖于解析器。在视觉和数学推理领域，提出了结合深度表示学习和符号程序执行的模型 [6]，这些方法在某些任务中展示了可解释性和高效性。

2.2 工具增强的大语言模型

随着大型语言模型取得显著进展，尤其是在提示学习 [20] 和指导学习 [18] 方面，尽管其性能令人印象深刻，但 LLMs 仍存在固有的局限性，如无法访问最新信息 [8]、无法利用外部工具 [16] 或进行精确的数学推理 [14]。为了增强 LLMs 的能力，研究者们尝试通过外部工具和模块化方法进行增强，尤其是通过网络搜索引擎 [11] 和外部资源的领域特定知识 [23] 来弥补这些缺陷。相比于传统的工具增强方法，Chameleon 通过自然语言指令引导 LLMs，并能够以即插即用的方式扩展新模块，避免了在不同工具组合 [16] 时需要额外训练或工具提示的需求。

2.3 Chameleon 方法的优势

如表 1 所示在与其他工具增强的语言模型进行比较时，Chameleon 展现出显著的灵活性和适应性。许多现有方法要么受限于一小组工具，要么仅限于特定任务，这限制了它们在多任务和新任务中的泛化能力 [16]。与这些方法不同，Chameleon 能够通过简单的自然语言指令，提供灵活的模块组合，并支持不断更新底层 LLMs 和添加新工具，从而适应不同的任务。本文的工作与 AutoGPT [15] 有相似的精神，后者是一个旨在整合多种工具以实现用户定义目标的自主代理，而 Chameleon 是首次实例化这一思想并在基准测试中验证其有效性。

Model	Tool Use						Skill Dimension					Inference & Extension		
	Size						Image	Web	Know.	Math	Table	Composition	Planning	Plug-n-Play
CoT [57]	1	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
Lila [39]	1	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
PoT [6]	2	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
Code4Struct [55]	1	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
PAL [10]	2	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
MathPrompter [18]	2	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
ART [43]	4	✓	✗	✗	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓
Toolformer [49]	5	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	natural lang.	✗
WebGPT [40]	10	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	program	✗
MM-ReAct [60]	>10	✓	✗	✗	✓	✗	✓	✓	✓	✓	✓	✓	word match	✓
Visual ChatGPT [59]	>10	✓	-	-	✗	✗	✓	✗	✗	✗	✗	✓	natural lang.	✓
ViperGPT [52]	>10	✓	-	-	✗	✗	✓	✗	✓	✓	✗	✓	program	✓
VisProg [13]	>10	✓	-	-	✗	✗	✓	✗	✗	✗	✗	✓	program	✓
HuggingGPT [50]	>10	✓	✓	✗	✗	✗	✓	✗	-	✗	-	✓	natural lang.	✓
Chameleon (ours)	>10	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	natural lang.	✓

表 1. Chameleon 使用的工具与其他方法的比较

3 本文方法

3.1 本文方法概述

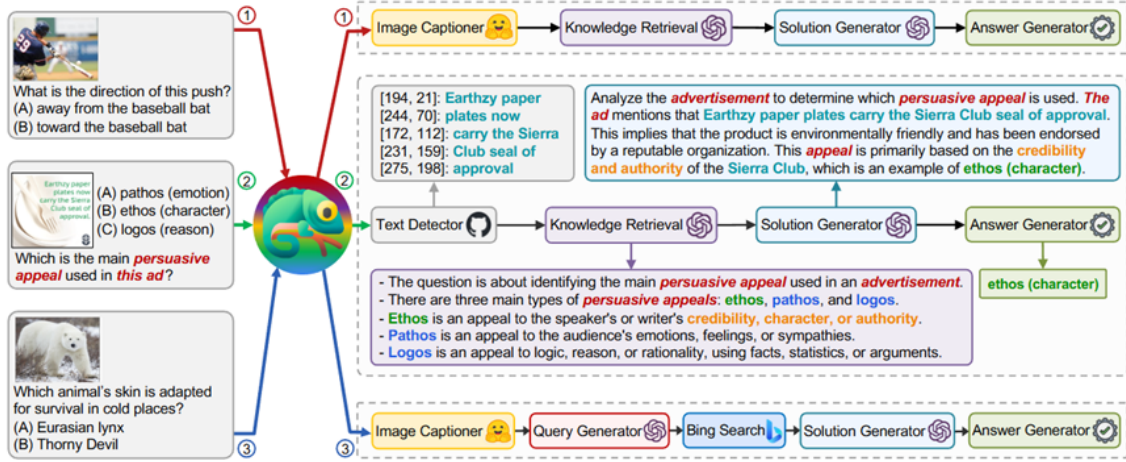


图 1. ScienceQA 数据集 [9] 上的 Chameleon 方法示例

本文在两种不同类型的任务上使用了两种不同的方法，针对基于科学常识的问答数据集 ScienceQA [9] 使用了如图 1 的方法；针对具有表格数据的问答数据集 TabMWP [10] 使用了如图 2 的方法。如图中 1 的例子②：这则广告中使用的主要说服手法是什么？回答这个问题需要：

- 1) 推断存在一个包含文本上下文的广告图像，并调用文本解码器来理解语义；
- 2) 检索有关说服性诉求以及三种说服性诉求之间的差异的背景知识；
- 3) 根据输入查询和先前步骤的中间结果生成解决方案；
- 4) 最后以任务特定的格式生成答案。另一方面，当回答“哪种动物的皮肤适应在寒冷地方生存”时，可能需要调用模块，如图像字幕生成器来解读图像信息，以及网络搜索引擎来检索领域知识以理解科学术语。

然而，当前的工具增强 LLMs 在处理这些真实世界的查询时仍面临挑战，涉及各种场景。大多数现有方法要么仅限于少量工具 [4, 16, 17, 22]，要么依赖于特定领域的工具，因此不容易推广到新领域的查询。在这项工作中，研究如何使 LLMs 能够合成程序以捕捉组合异构工具的逻辑。为了应对现有工作的挑战，引入了 Chameleon，一个即插即用的组合推理框架，利用 LLMs 来合成程序并组合各种工具，以解决广泛的任务。与现有的工具增强 LLMs 不同 [4, 11]，Chameleon 使用了更丰富的工具集，包括 LLMs、现成的视觉模型、网络搜索引擎、Python 函数和基于启发式的模块。此外，Chameleon 利用了 LLMs 的上下文学习能力，并建立在一个 LLM 上作为自然语言规划器，无需任何训练或精心策划的规则。在工具描述和使用示例的推动下，规划器推断出一个由一系列工具组成的程序，按顺序执行以生成用户查询的最终响应。

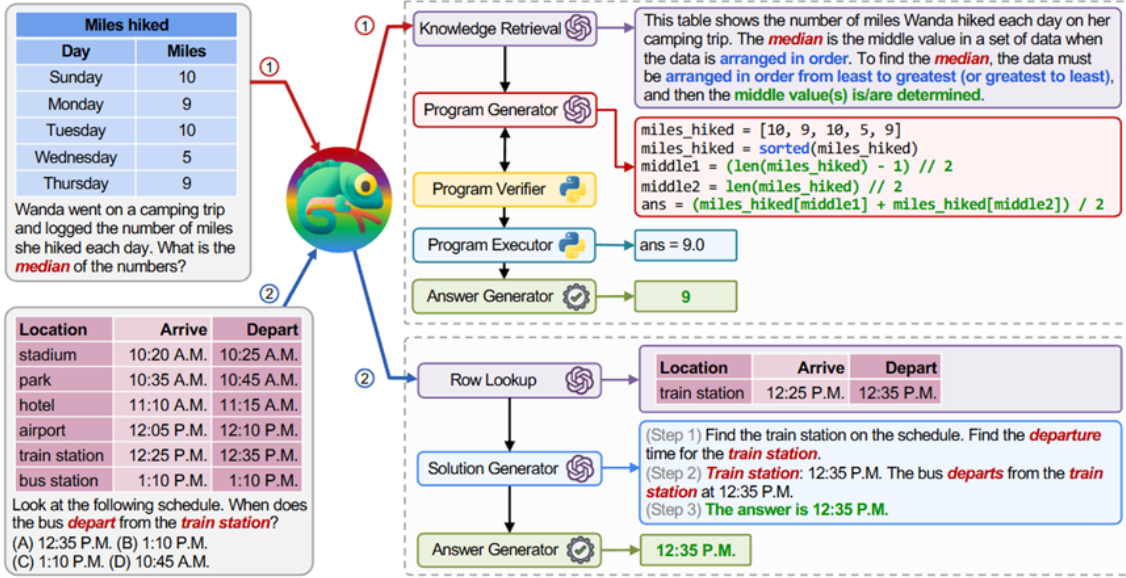


图 2. TabMWP 数据集 [10] 上的 Chameleon 方法示例

3.2 规划器

Chameleon 不像生成领域特定程序那样，而是采用基于 LLM 的规划器创建类似自然语言的程序，遵循自然语言指令，这样做的好处是减少错误，易于扩展到新模块，并且用户友好。规划器形式化如下：

给定输入查询 x_0 ，模块库存 M 和约束 G ，自然语言规划器 P 选择一组可以按顺序执行以生成一个类似自然语言的程序来回答查询的模块。模块库存 M 包括一组预先构建的模块： $\{M^i\}$ ，每个模块对应于各种类型的工具（表 2）。 G 是计划生成的约束，例如，并发关系和模块的顺序限制。

在工作中，规划者 P 是一个 LLM，被要求在少量样本的设置中生成一系列模块名称。规划者以自然语言的形式接收规划任务指令 I ，模块 M 的描述以及相应的约束 G ，还有一些示范示例 D 。

从 P 中采样得到的长度为 T 的计划可以表示为 $p = M^0, \dots, M^t$ ，其中 M^t 表示生成计划中的第 t 个元素， $M^t \in M$ 。形式上，给定一个输入查询（问题陈述） x_0 ，计划 p 的生成如下：

$$p \leftarrow P(x_0; I, M, G, D) \quad (1)$$

给定生成的计划，然后按顺序执行每个步骤的相应模块。该计划是一个自然语言程序，其中每个模块仅通过字符串匹配进行绑定。在时间步骤 t 评估模块 M^t 时，执行的输出 y^t 被计算为：

$$y^t \leftarrow M^t(x^{(t-1)}; c^{(t-1)}) \quad (2)$$

其中 $x^{(t-1)}$ 是当前模块 M^t 的输入， $c^{(t-1)}$ 是缓存信息（例如图像语义、检索到的知识、生成的程序），它们是由模块执行历史所产生的。下一个模块 $M^{(t+1)}$ 的问题输入 x^t 和缓存 c^t 分别通过以下方式进行更新：

$$x^t \leftarrow \text{update_input}(x^{(t-1)}, y^t) \quad (3)$$

$$c^t \leftarrow \text{update_cache}(c^{(t-1)}, y^t) \quad (4)$$

`update_input` 和 `update_cache` 函数是为每个 M^i 手动设计的。具体而言, `update_input` 应用于输入查询中的元素, 包括问题、表上下文和图像。这些元素在模块执行后进行更新。`update_cache` 对应于生成新信息, 例如输入图像的描述或从外部资源检索到的知识。

最后, 查询的响应 r 由最后一个模块 M^T 生成:

$$r = y^T \leftarrow M^T(x^{(T-1)}; c^{(T-1)}) \quad (5)$$

3.3 通用模块

为了适应对各种查询的不同推理能力, 系统利用了各种外部工具的丰富模块库。这里提供了对这个库的高级概述, 并在具体实验中提供了详细的实现。完整的模块库 M 在表 2 中呈现。







Tool Types	Tools
 OpenAI	Knowledge Retrieval, Query Generator, Row Lookup, Column Lookup, Table Verbalizer, Program Generator, Solution Generator
 Hugging Face	Image Captioner
 Github	Text Detector
 Web Search	Bing Search
 Python	Program Verifier, Program Executor
 Rule-based	Answer Generator

表 2. Chameleon 的模块库

4 复现细节

4.1 与已有开源代码对比

本次工作在复现该论文的过程中, 参考了原作者发布的代码, 原代码可在 <https://chameleon-llm.github.io/> 获取。原代码整体架构与功能实现为本次工作提供了重要基础, 大部分基础的代码逻辑与框架结构均来源于此。首先, 将原代码中调用 ChatGPT 的 api 部分, 全部替换为 Azure OpenAI 的 api。这一过程并非简单的接口更换, 需要深入理解 Azure OpenAI 的 api 规范、调用方式以及参数设置, 确保与原代码的逻辑无缝衔接。例如, 在处理模型选择、请求参数传递、响应解析等方面, 都需要依据 Azure OpenAI 的特点进行重新设计与调试。

其次, 本次实验把原代码中使用 bing search 的部分替换为 google search。这不仅涉及到更换搜索引擎的调用接口, 还需要重新适配搜索结果的解析逻辑。因为 bing search 和 google

search 返回的结果格式、数据结构存在差异，需要编写新的代码逻辑来提取所需的信息，如搜索结果的标题、链接、摘要等，以满足整体项目的功能需求。

除此以外，在代码运行前，还需针对数据集进行预处理，比如针对测试集中存在图片的问题，需要利用光学识别 ocr 扫描技术对图片进行处理，预先提取出所有文字信息，并且将这些信息转为一个独立 json 文件。如此处理之后在代码运行到文本检测模块可以直接读取 json 文件中的对应信息，加快数据处理的速度。

4.2 实验环境搭建

实验环境采用如表 3 的依赖进行配置即可

依赖项	版本要求
Python	3.8.10
huggingface - hub	-
numpy	1.23.2
openai	0.27.0
pandas	1.4.3
transformers	4.21.1
requests	2.28.1

表 3. 实验环境依赖信息

4.3 创新点

原框架在生成解决方案后，直接将其作为最终输出，没有对生成的解决方案进行检查和验证的机制。这可能导致所生成的解决方案存在各种潜在的问题，例如在处理复杂问题时，生成的解决方案可能会出现逻辑错误、数据不准确、信息不完整或未能完全满足需求等情况。

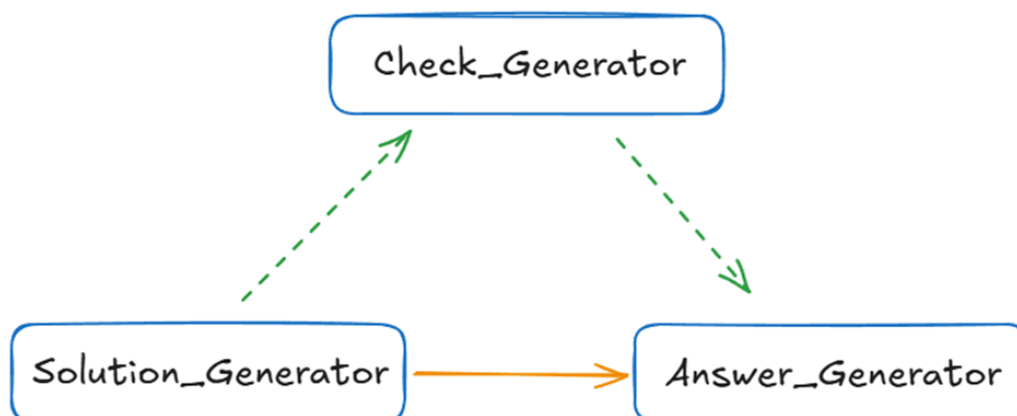


图 3. 增加 check 模块示意图

本次工作对论文在现有框架的基础上进行了改进，如图 3 所示，主要是增加了一个检查模块：当生成解决方案后，将其作为输入传递给 LLM。LLM 凭借其强大的语言理解和推理

能力，会对输入的解决方案进行多方面的检查，包括但不限于逻辑连贯性、内容完整性、信息准确性等方面。在检查完成后，LLM 会根据检查结果输出一个正确的答案，这个正确答案可能是对原解决方案进行修改、补充，甚至是重新生成，以纠正原解决方案中存在的各种错误和不足。

随后，将 LLM 输出的正确答案传递给答案生成器，答案生成器会对正确答案进行处理，提取出符合要求的格式化答案，确保最终输出的答案不仅内容正确，而且格式规范，便于后续统计准确率。

方法的伪代码如下：

```
1  定义 check_generator 方法：
2      获取模块输入：test_prompt, full_prompt = 构建提示词()
3      准备消息：messages = [{ "角色": "用户", "内容": full_prompt }]
4
5      初始化 success 为 False
6      初始化耐心值 patience = sg_patience
7      初始化计数器 count = 0
8
9      当 count 小于耐心值且 success 为 False 时：
10         如果 sg_temperature 小于 0.1 且 count 大于 0：
11             设置 _temperature 为 min(sg_temperature + 0.1, 1.0)
12         否则：
13             设置 _temperature 为 sg_temperature
14         调用 get_azure_chat_response 获取响应 check
15         使用正则表达式从响应中提取答案：
16             定义模式 pattern = re.compile(r"[Tt]he answer is ([A-Z])")
17             提取匹配结果 res
18         如果 提取结果 res 非空：
19             设置 success 为 True
20         计数器 count 增加 1
21
22     更新缓存：
23         缓存["check"] = check
24         缓存["check_generator:input"] = test_prompt
25         缓存["check_generator:output"] = check
26     返回 test_prompt 和 check
```

设计的模块提示词如下，命令 LLM 仔细阅读并理解问题、选项以及所提供的解答内容，接着去核查解答中的推理过程，通过与相关背景信息（比如元数据体现的一些条件等）进行对照，判断解答的逻辑步骤以及事实准确性是否正确，最后根据判断结果明确答案是正确还是错误，若错误要指出问题所在、给出正确推理及正确答案，并按照规定格式给出最终结论。该提示词采用 few-shot 方式 [20] 为 LLM 提供两个具体示例作为回答参考（篇幅有限下面的示例中只展示了一个具体示例），确保模型生成的回答紧密围绕实际问题，避免偏离主题。

check_generator_prompt

Your task is to verify the correctness of a given solution to a question. Carefully analyze the solution and determine whether it is correct based on the provided context and reasoning. Follow these steps:

1. Read and understand the question, options, and provided solution carefully.
2. Verify the reasoning: Cross-check the logical steps and factual accuracy of the solution against the context (including metadata, retrieved knowledge, or images, if any).
3. Determine correctness: - If the solution is ****correct****, confirm it explicitly without making any changes. - If the solution is ****incorrect****, identify the mistake, explain the correct reasoning, and provide the correct answer.
4. Finally, conclude the answer in the format of "the answer is [ANSWER]", where [ANSWER] is one from the options ["A", "B", "C", "D", "E"]. For example, "the answer is A", "the answer is B", "the answer is C", "the answer is D", or "the answer is E". If the answer is not in the options, select the most possible option.

Question: Would you find the word dozen on a dictionary page with the following guide words?daddy - definition

Options: (A) no (B) yes

Metadata: {'has_image': False, 'grade': 7, 'subject': 'language science', 'topic': 'reference-skills', 'category': 'Reference skills', 'skill': 'Use guide words'}

Solution: Guide words in a dictionary are the words that appear at the top of each page and help you find words quickly. They usually show the first and last words on the page. To determine if the word "dozen" would be found on a dictionary page with the guide words "daddy - definition", we need to compare the alphabetical order of the guide words with the alphabetical order of the word "dozen".The word "dozen" comes after "daddy" in alphabetical order. Therefore, it would be found on a dictionary page with the guide words "daddy - definition".Therefore, the answer is B.

Check: The original answer incorrectly determined that the word "dozen" falls within the alphabetical range of the guide words "daddy - definition." By correctly comparing the alphabetical order, we see that while "dozen" comes after "daddy," it also comes after "definition." Therefore, it would not appear on this page. The answer is A.

5 实验结果分析

所有实验使用的 LLM 是 ChatGPT3.5。

5.1 ScienceQA 数据集实验结果分析

本次实验在 ScienceQA 数据集 [9] 的实验结果如表 4 所示。本工作进行了两次实验，一次是基础的复现工作的结果，一次是添加改进创新后的结果，复现的结果与原文有细微偏差，可能是由于 bing search 模块替换成了 google search 模块。控制变量，表中加粗的比较只针对复现结果和改进结果。改进后的结果并不是所有指标都能得到提升，这可能是因为 LLM 在检查过程中其自身的预训练知识体系与推理机制存在固有局限性。尽管它能对各类信息进行整合分析，但面对复杂且专业的科学问题，其对语义的理解可能产生偏差。例如，在处理一些涉及多学科交叉的概念时，由于预训练数据中相关内容占比有限，导致在推理时难以准确把握关键信息。这使得改进后的实验在部分依赖精准语义理解的指标上，未能获得预期提升。

Model	ALL	NAT	SOC	LAN	TXT	IMG	NO	G1-6	G7-12
Chameleon	79.93	81.62	70.64	84.00	79.77	70.80	86.62	81.86	76.53
Implementation	79.90	82.33	69.89	83.04	80.67	71.89	85.75	82.28	75.58
Innovation	78.69	83.69	65.52	78.51	83.33	71.63	80.41	80.29	75.62

表 4. Chameleon 在 ScienceQA 数据集 [9] 上的实验结果

5.2 TabMWP 数据集实验结果分析

在 TabMWP 数据集 [10] 上因为全量数据消耗的资源较多，本次工作在复现过程中只测试了 3000 条数据，从结果表 5 上看基本和原方法接近。

Model	ALL	FREE	MC	INT	DEC	EXTR	BOOL	OTH	G1-6	G7-8
Chameleon	93.28	93.13	93.72	92.71	94.76	91.29	98.11	78.85	93.37	93.17
Implementation	93.8	93.76	93.93	93.09	96.32	91.84	97.91	81.40	93.51	94.19

表 5. Chameleon 在 TabMWP 数据集 [10] 上的实验结果

6 总结与展望

本次实验是原有框架基础上进行的复现与创新，包括替换 API、调整搜索引擎以及增加检查模块等改进。然而，当前实现过程中仍存在一些不足，尤其是在处理复杂、多学科的推理任务时，LLMs 的预训练知识和推理机制可能导致语义理解的偏差，且检索结果的准确性和多模态数据的处理仍需进一步优化。未来的研究方向可以聚焦于提升模型的语义理解能力、优化检查模块的智能推理、增强检索和多模态数据的处理能力，并在更大规模和复杂度的任务中验证系统的扩展性与实时性。

参考文献

- [1] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1545–1554, 2016.
- [2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [3] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.
- [4] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14953–14962, 2023.
- [5] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In *6th Annual Conference on Robot Learning*, 2022.
- [6] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 2989–2998, 2017.
- [7] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.
- [8] Mojtaba Komeili, Kurt Shuster, and Jason Weston. Internet-augmented dialogue generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8460–8478, 2022.
- [9] Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

- [10] Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *International Conference on Learning Representations (ICLR)*, 2023.
- [11] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.
- [12] OpenAI. Chatgpt, 2022.
- [13] OpenAI. GPT-4 technical report. *ArXiv*, abs/2303.08774, 2023.
- [14] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, 2021.
- [15] Toran Bruce Richards. Auto-GPT: An experimental open-source attempt to make GPT-4 fully autonomous. <https://github.com/Significant-Gravitas/Auto-GPT>, 2023.
- [16] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.
- [17] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*, 2023.
- [18] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [19] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.
- [20] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.
- [21] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, pages 5–32, 1992.

- [22] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual ChatGPT: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023.
- [23] Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. Generate rather than retrieve: Large language models are strong context generators. In *International Conference on Learning Representations (ICLR)*, 2023.