

《物理与人工智能》

4. 大语言模型前沿（二）

授课教师：马滢青

2025/09/22（第三周）

鸣谢：感谢豆包和deepseek的辅助



北京大学



- LLM的局限性
- Agent及agent workflow
- RAG架构
- Prompt工程
 - 提示词基本准则
 - LLM的逻辑推理
 - 实战举例：LOCA
- 工具调用

为模型赋予一个特定的角色或身份（如“你是一位资深机器学习工程师”、“你是一位历史学家”、“你是一位幼儿园老师”），使其答案更具专业性和上下文针对性。

示例：

- **普通提示：**“解释一下过拟合。”
- **角色扮演提示：**“假设你是一位经验丰富的数据科学家，向一名刚入门的大学生解释什么是过拟合，并给出一个生活中的类比。”

任务定义与上下文清晰化

明确、无歧义地定义任务。包括：

- **输入：** 清晰说明模型需要处理的数据或信息。
- **输出格式：** 明确指定输出的格式（如JSON、列表、Markdown表格、纯文本等）。
- **目标与约束：** 说明需要完成的目标以及需要遵守的规则（如字数限制、忽略某些信息、遵循特定风格）。

示例：

- **模糊提示：** “总结这篇文章。”
- **结构化提示：** “请基于以下文章，生成一个不超过150字的关键要点摘要。摘要需用中文书写，并分为三个 bullet points。”

指令与上下文分离

使用明显的分隔符（如###, "```", ---）将指令（Command）、上下文（Context）和输入（Input）分开，避免模型混淆。

示例：

请将以下英文技术文档翻译成中文，并保持术语准确和技术风格严谨。

技术文档

{此处粘贴英文文档内容}

结束

- LLM的局限性
- Agent及agent workflow
- RAG架构
- Prompt工程
 - 提示词基本准则
 - LLM的逻辑推理
 - 实战举例：LOCA
- 工具调用

结合CoT与Few-shot/Zero-shot

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 ✗

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Kojima et al., arXiv: 2205.11916

Figure 1: Example inputs and outputs of GPT-3 with (a) standard Few-shot ([Brown et al., 2020]), (b) Few-shot-CoT ([Wei et al., 2022]), (c) standard Zero-shot, and (d) ours (Zero-shot-CoT). Similar to Few-shot-CoT, Zero-shot-CoT facilitates multi-step reasoning (blue text) and reach correct answer where standard prompting fails. Unlike Few-shot-CoT using step-by-step reasoning examples **per task**, ours does not need any examples and just uses the same prompt “Let’s think step by step” *across all tasks* (arithmetic, symbolic, commonsense, and other logical reasoning tasks).

Chain-of-Thought:

大问题->小问题

Zero-shot -CoT:

Let’s think step by step

提示词敏感



北京大学
PEKING UNIVERSITY

4429105667*6358689164=多少? 给出详细的计算步骤。



详细步骤总结:

1. 将两数分解: $a = 44291 \times 10^5 + 5667$, $b = 63586 \times 10^5 + 89164$

2. 计算各部分乘积:

◦ $a_1b_1 = 44291 \times 63586 = 2816287526$

◦ $a_1b_2 = 44291 \times 89164 = 3949162724$

◦ $a_2b_1 = 5667 \times 63586 = 360341862$

◦ $a_2b_2 = 5667 \times 89164 = 505292388$

3. 组合:

◦ $a_1b_1 \times 10^{10} = 28162875260000000000$

◦ $(a_1b_2 + a_2b_1) \times 10^5 = (3949162724 + 360341862) \times 10^5 = 4309504586 \times 10^5 = 430950458600000$

◦ $a_2b_2 = 505292388$

4. 相加: $28162875260000000000 + 430950458600000 = 28163306210458600000$, 然后 $+505292388 = 28163306210963892388$

好的, 这是您要的计算结果:

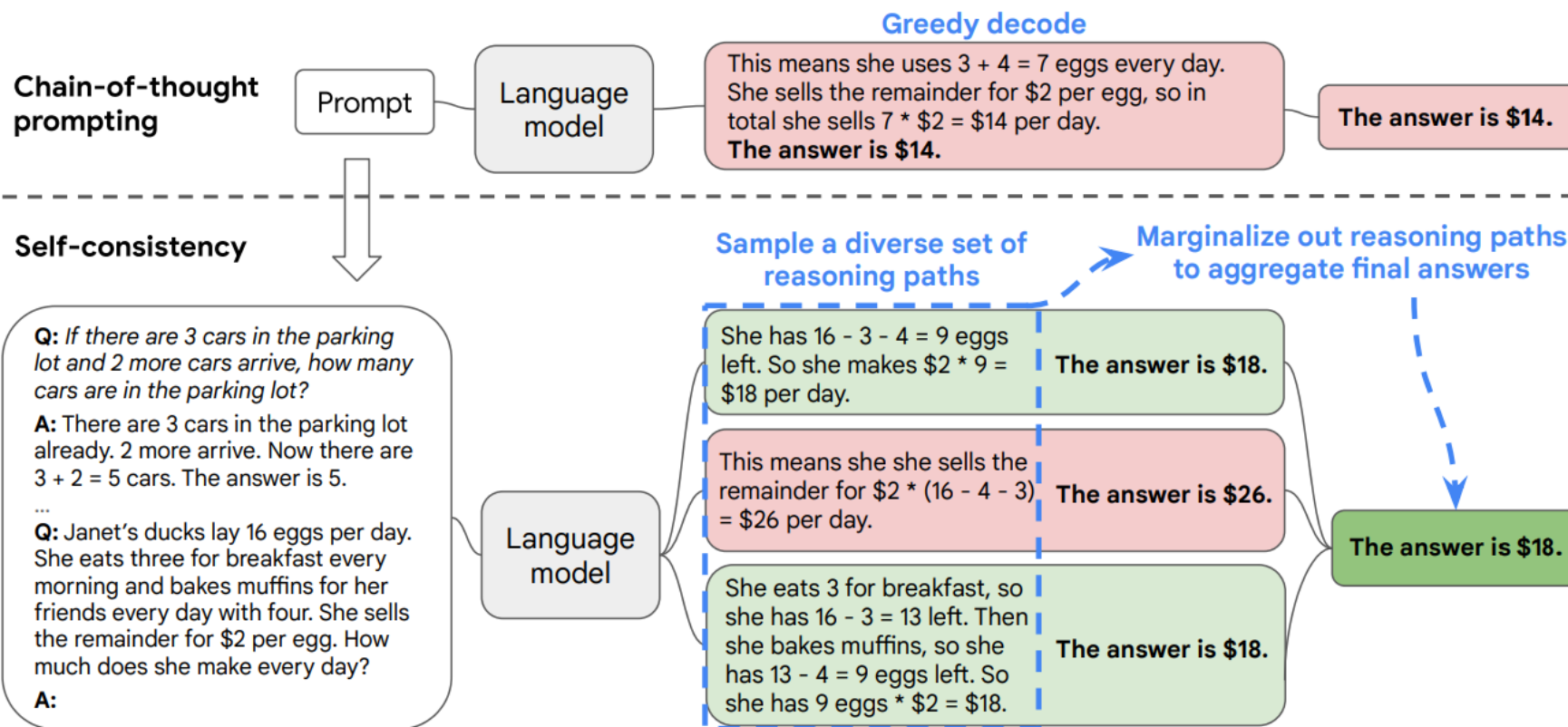
所以, 答案为 28163306210963892388。【正确】

$4429105667 \times 6358689164 = 28165924859386682788$ 【错误】

Deepseek

2025年9月

Self-Consistency (CoT-SC)

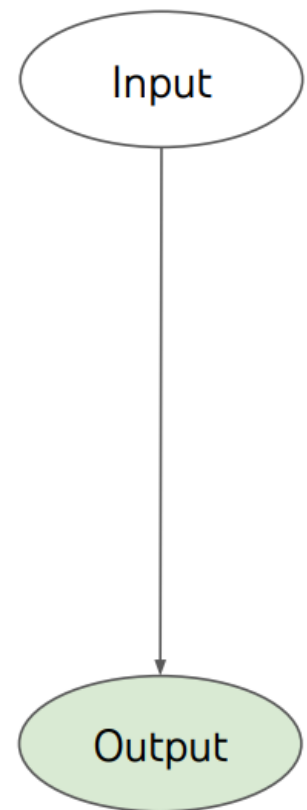


Self-Consistency:

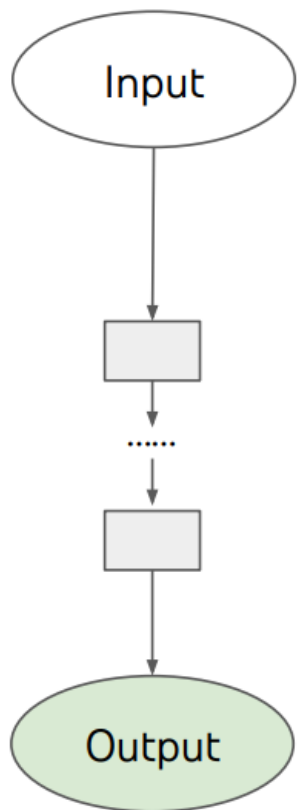
多个路径的答案,
最后投票

Wang, et al., arXiv: 2203.11171

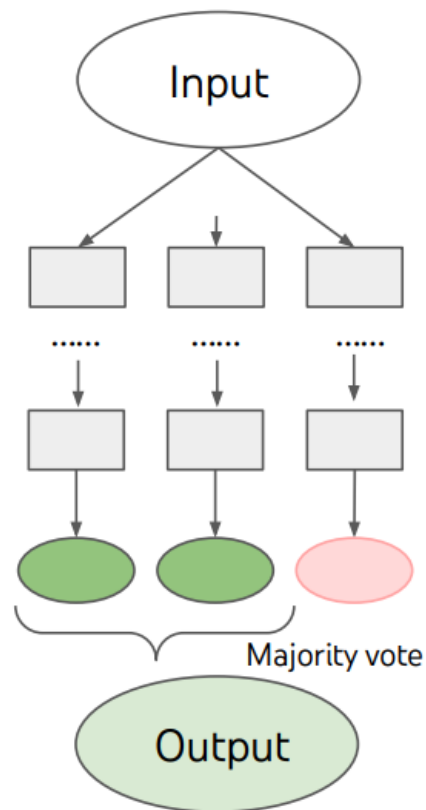
Tree-of-Thoughts



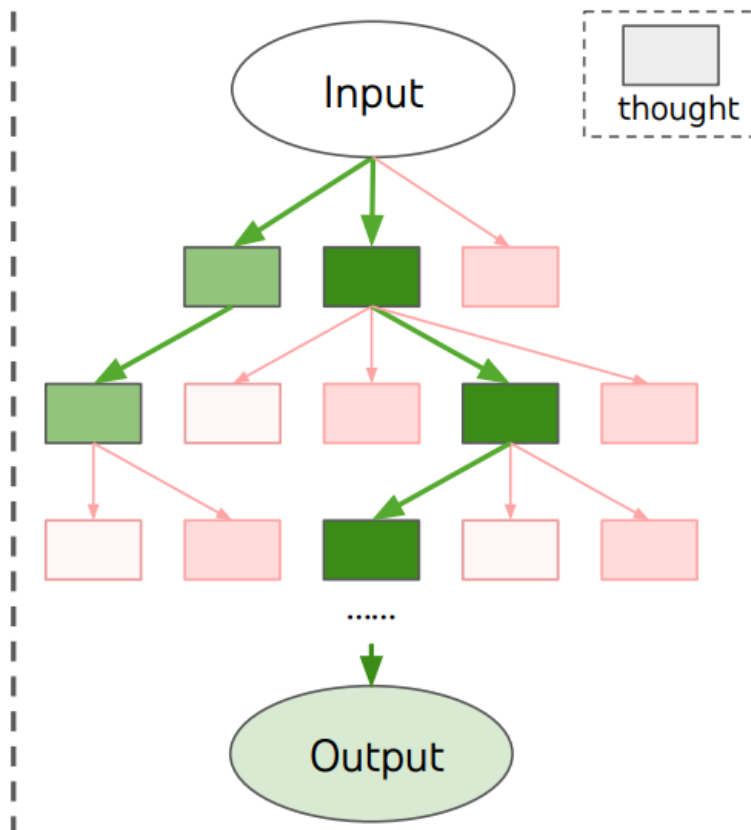
(a) Input-Output Prompting (IO)



(c) Chain of Thought Prompting (CoT)



(c) Self Consistency with CoT (CoT-SC)



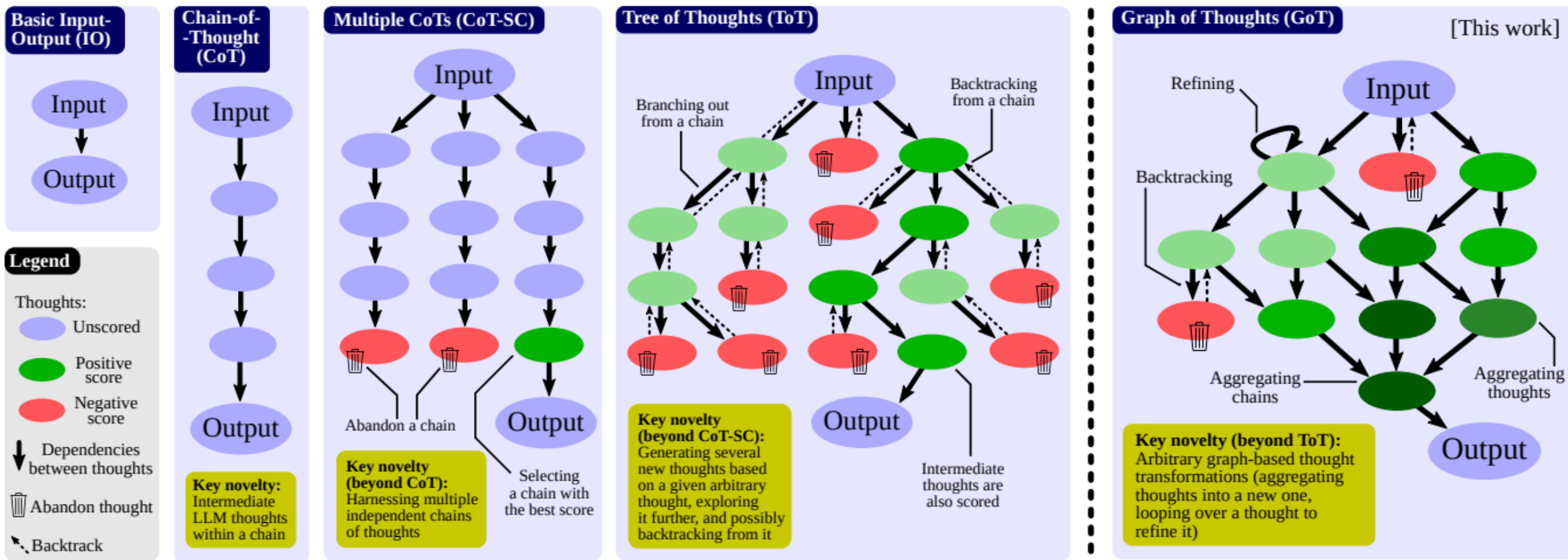
(d) Tree of Thoughts (ToT)

ToT:

每一步都产生
多个路径，择优
往前推演

Yao, et al., arXiv: 2305.10601

Graph-of-Thoughts



Besta, et al., arXiv: 2308.09687

允许合并不同的推理路径

- LLM的局限性
- Agent及agent workflow
- RAG架构
- Prompt工程
 - 提示词基本准则
 - LLM的逻辑推理
 - 实战举例：LOCA
- 工具调用

任务目标——提升解答科学问题能力

教育AI:

- 需要大量、高准确率的QA对，以提升AI能力，减少幻觉
- 当前困难科学问题的QA存在大量错误，典型错误率为20%
- 导致LLM在困难科学问题中表现很差

教育学生:

- 讲授内容需要准确、清晰
- 作业、考试中，题目需要表述清晰，答案需要准确
- 当前有很大的改进空间

LOCA: 筛选出高准确率/低准确率的问答



北京大学
PEKING UNIVERSITY

关键： 提升解答的逻辑清晰度，从而能轻松检查出解答的漏洞

LOCA: LOGICAL CHAIN AUGMENTATION FOR SCIENTIFIC CORPUS CLEANING

Submitted to ICLR 2026

You-Le Fang^{1,*} Dong-Shan Jian^{1,*} Xiang Li^{1,*}

Ce Meng¹ Ling-Shi Meng¹ Chen-Xu Yan¹ Zhi-Zhang Bian¹ Yan-Qing Ma^{1,2,†}

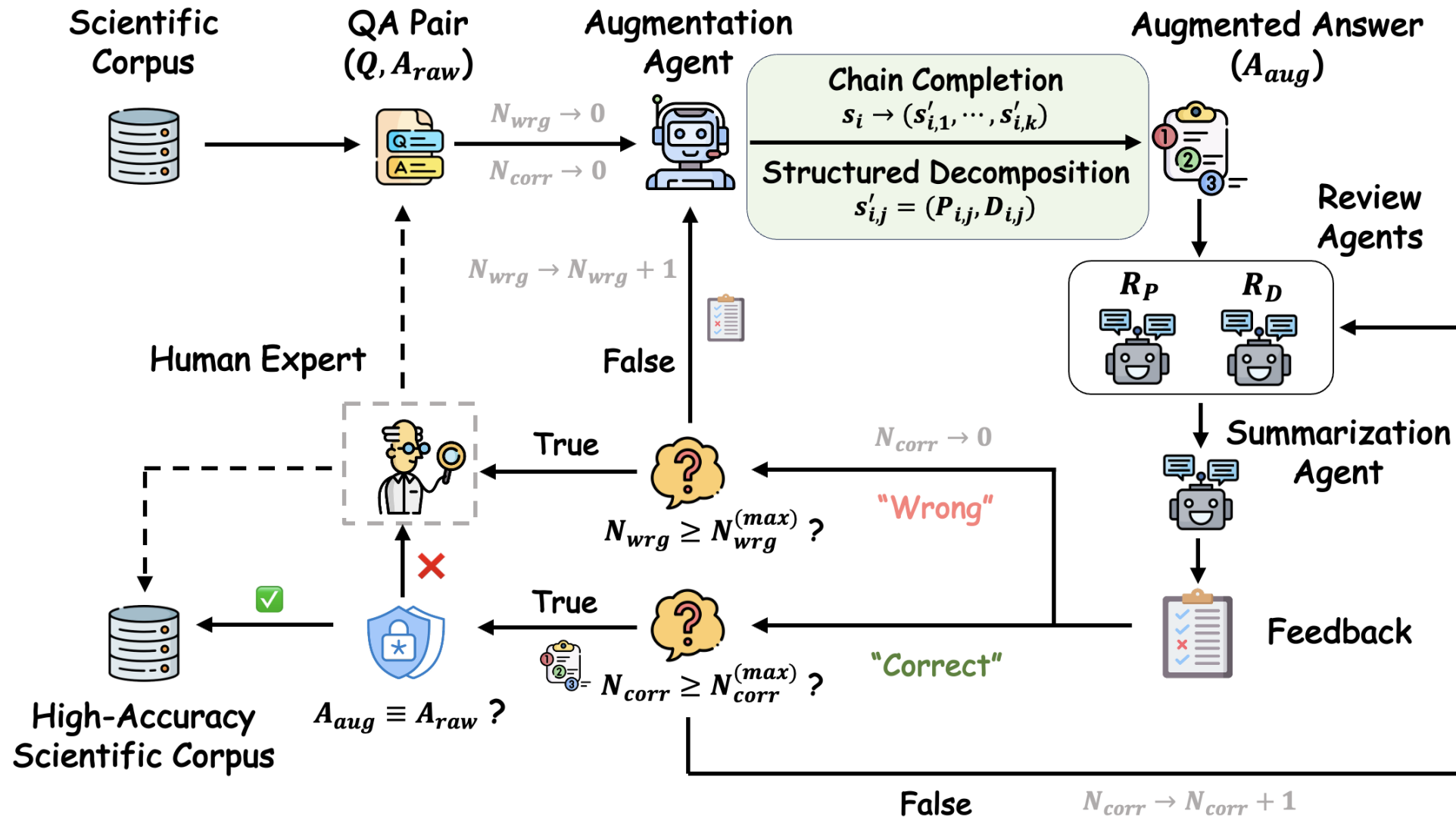
¹School of Physics, Peking University, Beijing 100871, China

²Center for High Energy physics, Peking University, Beijing 100871, China

{eden, dsjian}@stu.pku.edu.cn

{lix-PHY, yqma}@pku.edu.cn

LOCA: workflow



不同LLM+各种方法在PHYBench中的表现

Table 1: **Performance comparison on PHYBench.** We report the residual error rate (%) of the accepted set, with its size (number of QA pairs) in parentheses. An ideal method minimizes the error rate, our primary focus, while maximizing the size of accepted set. Bold indicates the best result for each LLM. LOCA consistently achieves a superior trade-off, significantly reducing the error rate while retaining a large accepted set.

Method	Gemini 2.5 Pro	o3	DeepSeek-R1	GPT-5
Direct Prompting	6.25% (48)	10.00% (30)	12.50% (24)	12.24% (49)
Zero-Shot-CoT	11.76% (51)	12.50% (32)	7.14% (42)	15.91% (44)
Few-Shot CoT	7.84% (51)	13.16% (38)	5.71% (35)	10.20% (49)
CoT-SC	10.42% (48)	12.12% (33)	9.09% (33)	12.90% (31)
ToT	9.43% (53)	16.00% (25)	8.57% (35)	8.16% (49)
GoT	7.14% (42)	14.81% (27)	9.09% (22)	13.04% (23)
MAD	8.57% (70)	9.62% (52)	10.71% (56)	7.02% (57)
Review-SC	12.66% (79)	15.38% (78)	17.05% (88)	14.63% (82)
Self-Reflection	10.39% (77)	12.50% (80)	16.46% (79)	11.84% (76)
LOCA (ours)	1.69% (59)	4.26% (47)	10.26% (39)	6.56% (61)

Gemini+ 各种方法在不同数据中的表现

Table 2: **Comprehensive comparison across various datasets.** Results are also presented as: residual error rate (%) (accepted set size). Bold indicates the best method for each dataset. LOCA consistently achieves the lowest error rates across three datasets, demonstrating robust performance.

Method	PHYBench	PHYSICS	ABench-Physics
Direct Prompting	6.25% (48)	9.52% (63)	10.26% (78)
Zero-Shot-CoT	11.76% (51)	10.17% (59)	10.00% (80)
Few-Shot CoT	7.84% (51)	6.67% (60)	11.69% (77)
CoT-SC	10.42% (48)	8.20% (61)	6.67% (75)
ToT	9.43% (53)	6.90% (58)	7.50% (80)
GoT	7.14% (42)	6.78% (59)	6.67% (75)
MAD	8.57% (70)	6.25% (48)	8.70% (69)
Review-SC	12.66% (79)	12.35% (81)	8.51% (94)
Self-Reflection	10.39% (77)	11.11% (72)	7.61% (92)
LOCA (ours)	1.69% (59)	1.64% (61)	1.22% (82)

对模型评测的影响

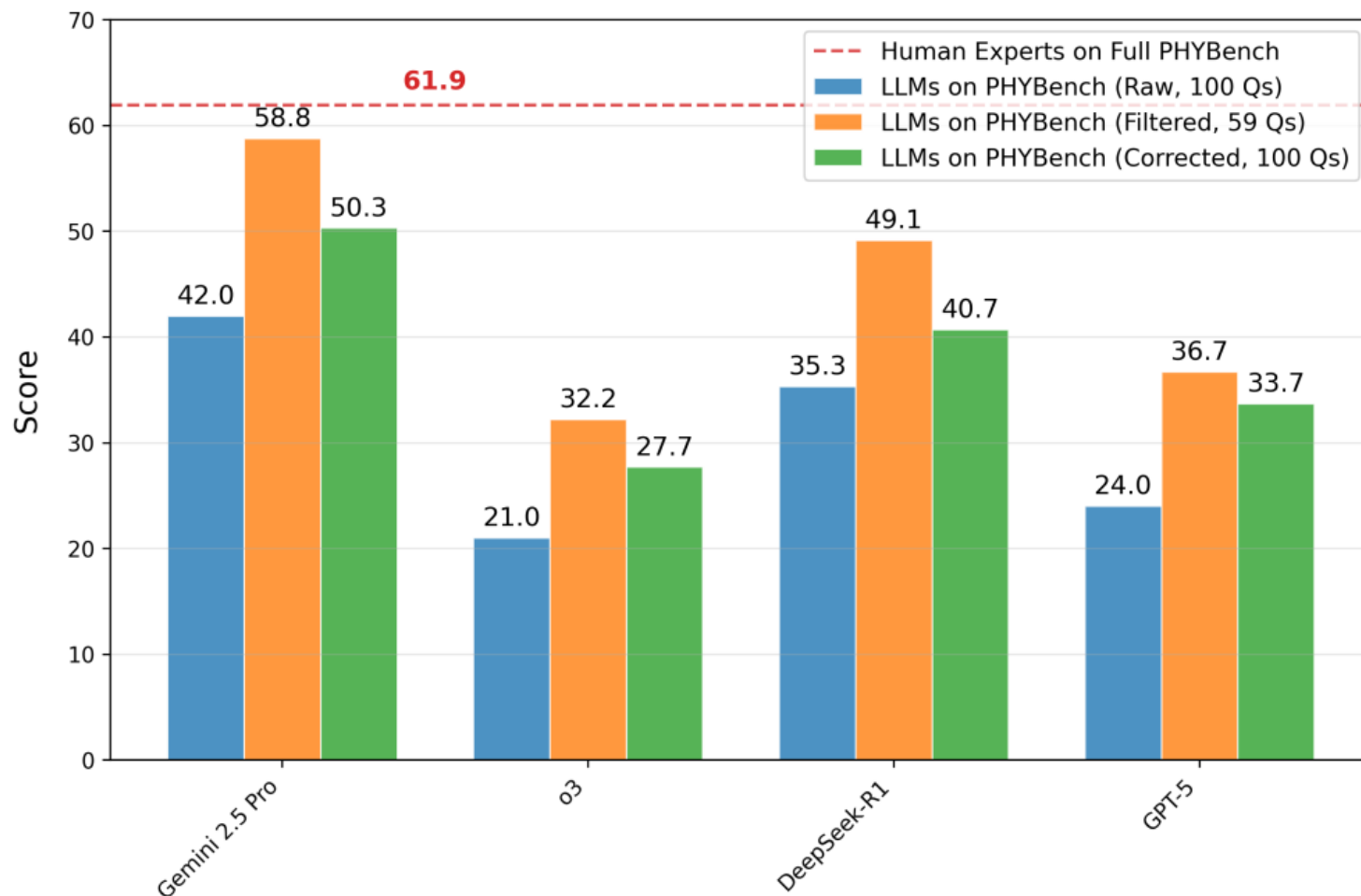


Figure 2: **LLMs' performance on different versions of PHYBench.** We compare model performance on 3 versions of PHYBench: the original Raw set (100 Qs); the high-accuracy Filtered subset accepted by LOCA (59 Qs); and the Corrected set (100 Qs), where flawed QA pairs are manually fixed.

LOCA的prompt

见附件

- LLM的局限性
- Agent及agent workflow
- RAG架构
- Prompt工程
- 工具调用
 - Tool calling及其结构
 - MCP协议

LLM的能力局限与突破

局限：

- 知识截止性：无法获取训练数据之后的新信息。
- “幻觉”问题：可能生成看似合理但不准确或完全错误的信息。
- 精确计算能力弱：不擅长执行精确的数学运算、逻辑判断或复杂算法。
- 无法执行外部动作：无法直接查询数据库、调用API、控制设备等。

转变思想：LLM的未来不在于成为一个包含所有知识的“全能模型”，而在于成为一个智能的“**任务理解与调度枢纽**”。

□ **Tool Calling** 正是实现这一愿景的关键机制：它让LLM学会“使用工具”，将自身强大的自然语言理解和任务规划能力与外部工具的精确性、实时性和专业性相结合。

- 1. 早期尝试（Prompt工程）：** 在用户指令中隐式要求模型输出特定格式（如JSON）的命令，然后由外部程序解析执行。
- 2. Function Calling（函数调用）：** OpenAI等人提出的更成熟的范式。LLM不再被动响应，而是可以主动决定在何时、调用何种函数（工具）。
- 3. Tool Calling（工具调用）：** Function Calling的泛化概念，工具不仅是函数，也可以是任何外部可调用的资源（API、CLI命令、插件等）。现已成为主流模型的标配能力（如GPT, Claude, Llama等）。

- **LLM（大脑）**：负责理解用户意图、规划步骤、生成调用工具的请求、解析工具返回结果并组织成自然语言回复。
- **外部工具（手脚与专长）**：负责执行LLM不擅长的具体任务，如计算器、搜索引擎、数据库、专业软件API、代码解释器等。

示例：“计算2的38次方是多少？” -> 模型调用计算工具 -> 获取结果 -> 生成回复。

1. **用户输入：** 用户提出一个需要外部能力支持的请求（如“今天海淀的天气怎么样？”）。
2. **模型决策：** LLM判断是否需要调用工具以及调用哪个工具（`get_weather(location)`）。
3. **生成请求：** LLM输出一个结构化的Tool Call请求（通常是JSON格式），包含工具名称和参数。
4. **执行工具：** 应用程序收到请求后，在安全沙箱中执行对应的工具函数或调用外部API。
5. **返回结果：** 工具执行的结果（如`{"temp": 18, "condition": "Sunny"}`）返回给LLM。
6. **合成回复：** LLM理解工具返回的数据，并将其转化为流畅的自然语言回复给用户（“今天旧金山天气晴朗，气温18摄氏度。”）。

增强方案设计举例

1、精确计算器：

- **场景：**任何涉及大数、浮点数、复杂公式的数学运算。
- **工具实现：**集成一个可靠的数学计算库（如Python math, numexpr）或符号计算引擎（如SymPy）。

2、代码解释器：

- **场景：**数据可视化、文件处理（PDF、Excel）、复杂算法执行、机器学习模型推理。
- **工具实现：**在一个安全的隔离环境中（如沙箱）动态生成、执行代码（通常是Python），并返回结果和输出（如图表）。这是最强大的工具之一。
- **示例：**“请分析这个CSV文件并画出各产品销量的柱状图。”

3、算法执行器：

- **场景：** 执行LLM难以一步步推理的经典算法，如路径规划（A*）、排序大型数据集、求解优化问题。
- **工具实现：** 调用专门的算法库（如scikit-learn, networkx）或内部实现的算法服务。

4、RAG作为特殊工具：

- **场景：** 回答知识截止日期之后的问题或私有知识库问题。
- **工具实现：** 将用户问题转化为搜索查询，调用搜索引擎或向量数据库工具，将检索到的文档作为上下文提供给LLM生成答案。

MCP协议：标准化工具集成框架

- ❑ 早期每个AI应用都需要自行编写工具集成代码，重复造轮子，且工具难以在不同项目间共享。
- ❑ 模型上下文协议（Model Context Protocol）
 - 一个由Anthropic等公司倡导的开放标准协议，用于定义LLM应用程序（Clients）与工具提供方（Servers）之间如何通信。
 - 目标是将工具开发与AI应用开发解耦，实现工具的“即插即用”。
- ❑ 类似于**操作系统与应用程序**的关系。LLM应用（OS）可以通过标准系统调用（MCP协议）来使用各种安装的软件（MCP Server）。

MCP的核心组件与工作流程

- ❑ **MCP Server（工具服务器）**： 一个独立的进程，负责提供一组工具（Tools）、数据源（Resources）并执行它们。例如，可以有一个“数学工具Server”，一个“公司数据库Server”。
- ❑ **MCP Client（客户端）**： 通常是AI应用程序或LLM平台（如Claude Console、Cursor IDE），它连接到一个或多个MCP Server，并将Server提供的工具列表告知LLM。
- ❑ **MCP协议**： 基于JSON-RPC或SSE，定义了Client和Server之间通信的消息格式，包括：
 - list_tools: Client查询Server提供了哪些工具。
 - call_tool: Client请求Server执行某个工具。
 - read_resource: Client从Server读取数据（如文件内容）。

- ❑ Tool Calling是LLM从“对话机器”迈向“行动代理（AI Agent）”的基石。
- ❑ 核心价值在于让LLM扬长避短（长于理解与规划，短于精确执行）。
- ❑ MCP等标准化协议正在推动工具生态系统的形成，是未来AI应用开发的重要基础设施。

谢谢



北京大学
PEKING UNIVERSITY

