

单选题 1分

1. 关于“监督微调（SFT）”的描述，哪项是正确的？

- A 利用无标注数据让模型预测下一 token
- B 利用带标签的输入-输出对训练模型
- C 只训练奖励模型
- D 让模型通过搜索产生更好的推理路径

填空题 2分

2. 对于监督微调来说，他的参数更新方式包括 [填空1] 和 [填空2] 两种。

模型参数微调

➤ 关注模型参数层面的调整

✓ 全面微调

- 在新任务上调整模型的**全部**参数，以使其完全适应新任务。
- 加载预训练模型→在新任务数据集上训练模型，调整所有参数。
- 当新任务与预训练任务差异较大，或者想要充分利用新任务数据集时可以选择全面微调。

✓ 参数高效微调

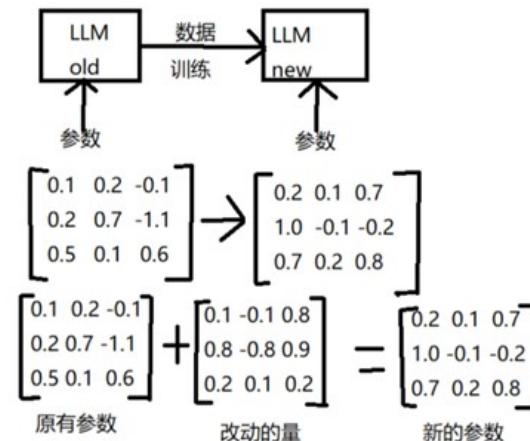
- 仅调整模型的**部分**参数；
- 加载预训练模型→在模型中添加可训练的组件或选择部分参数→在新任务数据集上训练这些组件或参数；
- 当计算资源有限，或者想要快速适应新任务而不影响模型在其他任务上的性能时。

参数高效微调

➤ 参数高效微调 (PEFT)

- ✓ 参数高效微调 (Parameter-Efficient Fine-Tuning, PEFT) 是一系列旨在以较少的计算资源和数据量，实现与全参数微调相近性能的技术。这类方法通常**冻结**预训练模型的**大部分参数**，只**训练少量额外的参数**。

- 提示微调技术
- Adapter
- LoRA系列



填空题 1分

3. Prompt Tuning (软提示微调) 只训练一段 [填空1]，而保持模型主体冻结。

单选题 1分

4. LoRA 的核心思想是什么?

- A 直接替换模型权重
- B 不训练任何参数
- C 将权重更新矩阵用低秩分解表示
- D 训练模型的全部前馈网络层

单选题 1分

5. 现在我准备基于一个预训练的多模态大模型（如 Wan）在自采集的数据集上微调“视频换衣”模型。由于训练资源有限，我应该选择下列哪一种微调方式？

- A 全参数微调 (Full Finetuning)
- B LoRA (参数高效微调)
- C 仅通过提示词控制风格 (Prompt Engineering)

填空题 2分

RLHF 的三个阶段依次是：

- (1) 监督微调,
- (2) [填空1],
- (3) [填空2]。

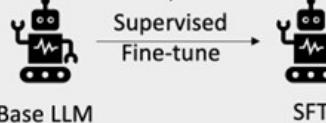
RLHF流程

SFT 模型

奠定基础能力

Step 1 Supervised Fine-Tuning

Collect human demonstration data



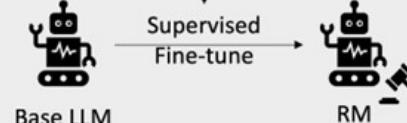
奖励模型 (RM)

学习人类偏好

Step 2 Training a Reward Model

Prompt → SFT → Human → Answer 1 > Answer 2

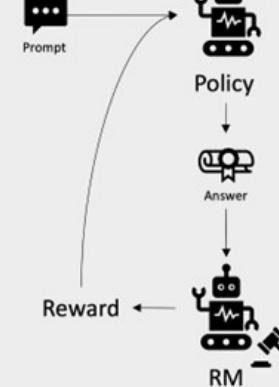
Collect human preference data



PPO 优化

微调策略

Step 3 Optimize Policy

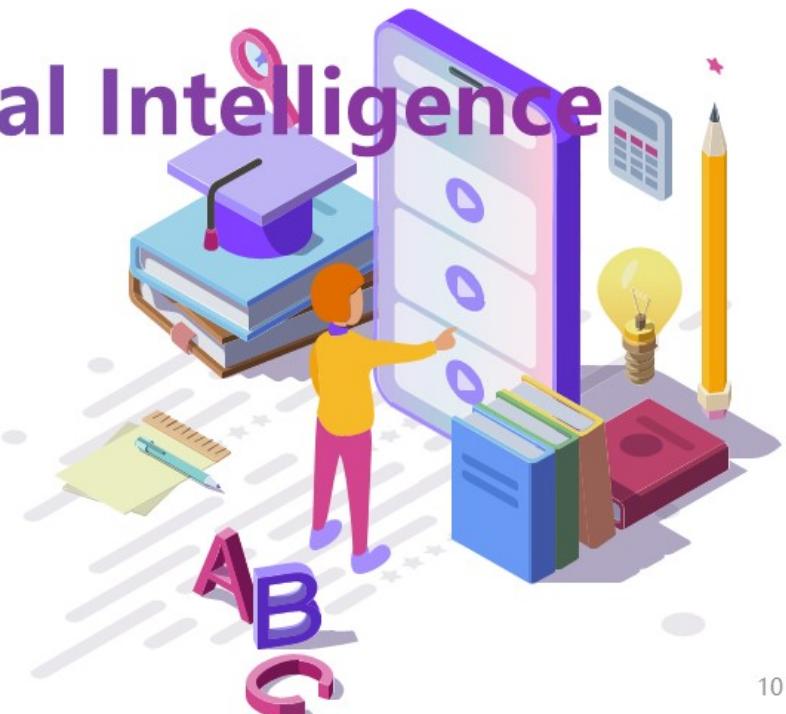


三者形成闭环：SFT提供**初始行为**，RM提供**可微分信号**，PPO实现**策略优化**。

人工智能导论

Introduction to Artificial Intelligence

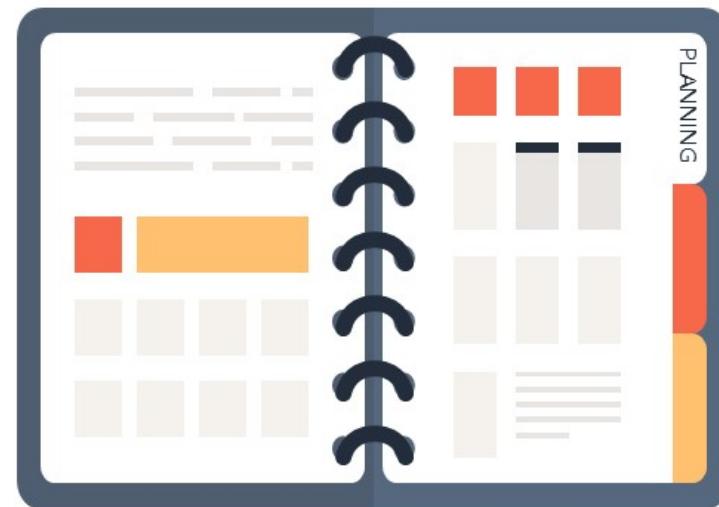
第七章 大模型推理与规划



Lecture Plan

Prompt工程相关技术

- 上下文学习 (In-context Learning)
- 思维链 (CoT)



Prompt工程

➤ 监督微调



- **作用:**

- 让模型学习**特定领域**的知识 (生物医学领域的医疗问答系统)
- 让模型学习任务的**具体特征**, 并适应任务的特殊要求
 - 翻译: sentence-to-sentence
 - 情感分类: sentence-to-label

- **缺点:**

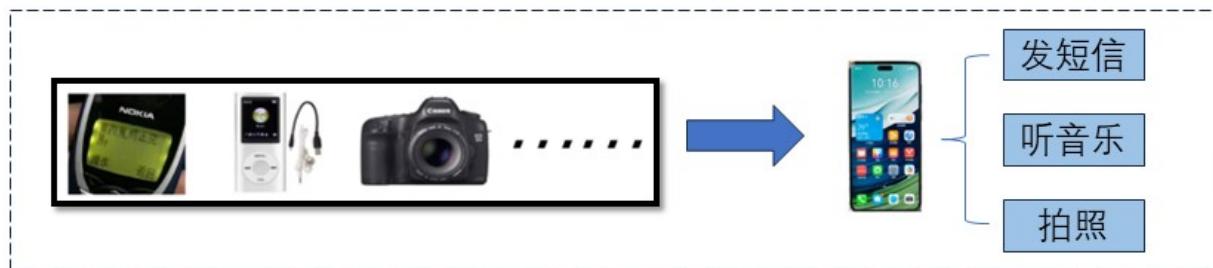
- 高数据成本: 需要大量高质量、标注好的数据准备
- 高资源成本: 需要训练资源 (GPU) 和相应的费用
- 更新不灵活: 每次修改需求/更新知识都需要重新训练

有没有更灵活、个性化、快速的方法?

12

Prompt工程

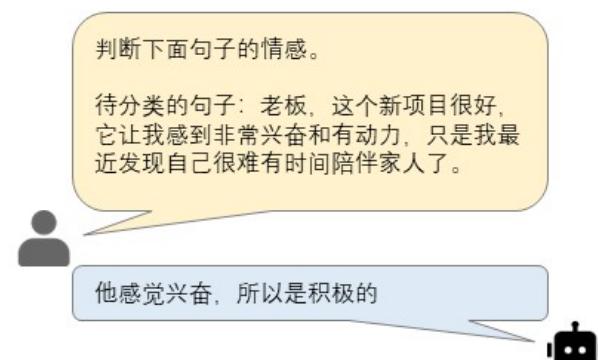
➤ 大模型时代的新范式：预训练+提示预测



13

Prompt工程

Prompt工程（Prompt Engineering），又叫提示工程，是指设计和优化用于与生成式人工智能模型交互的Prompt的过程。Prompt工程作为一门**专注于如何编写有效指令**的技术，成为了连接**模型与任务需求**之间的桥梁。



Prompt 工程

Prompt是指用于**指导**生成式人工
智能模型**执行特定任务的输入指令**，
这些指令通常以自然语言文本的形
式出现。Prompt的核心目的是**清
晰地描述目标任务**，并适当提供解
决该任务所需的**辅助信息**。

Prompt工程

什么才是好的Prompt?



判断下面句子的情感。

待分类的句子：老板，这个新项目很好，它让我感到非常兴奋和有动力，只是我最近发现自己很难有时间陪伴家人了。

他感觉兴奋，所以是积极的

Prompt 工程

**有效提问
激活模型潜力**



判断下面句子的情感为积极还是消极。

待分类的句子：老板，这个新项目很好，它让我感到非常兴奋和有动力，只是我最近发现自己很难有时间陪伴家人了。

针对此问题有如下示例：

示例1：老板，我非常喜欢这个新项目，它让我感到非常兴奋和有动力。
\n积极。

示例2：老板，这个项目的工作量太大了，我感到压力很大。
\n消极。

示例3：老板，虽然项目很有趣，但我最近感到有些疲惫。
\n消极。

以{"结果": ""}的JSON格式返回最终结果。



{"结果": "消极"}

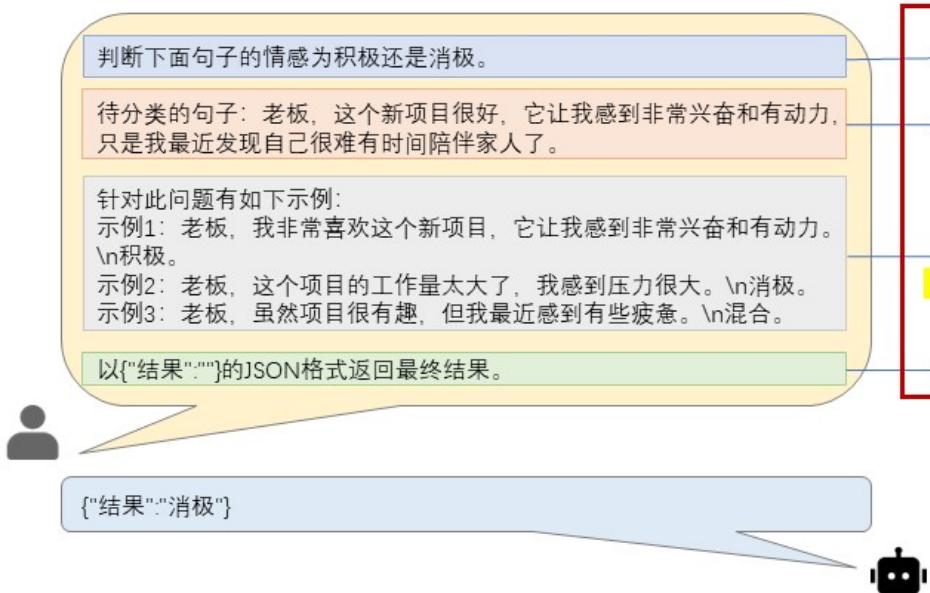
优化后的prompt



Prompt工程

➤ 提示词要素

任务说明、问题、上下文和输出格式四个基本元素



16

Prompt工程

问题+输入

判断下面句子的情感。

待分类的句子：老板，这个新项目很好，它让我感到非常兴奋和有动力，只是我最近发现自己很难有时间陪伴家人了。



他感觉兴奋，所以是积极的



直接Prompt

问题+输入+上下文+输出

判断下面句子的情感为积极还是消极。

待分类的句子：老板，这个新项目很好，它让我感到非常兴奋和有动力，只是我最近发现自己很难有时间陪伴家人了。

针对此问题有如下示例：

示例1：老板，我非常喜欢这个新项目，它让我感到非常兴奋和有动力。
\n积极。

示例2：老板，这个项目的工作量太大了，我感到压力很大。
\n消极。

示例3：老板，虽然项目很有趣，但我最近感到有些疲惫。
\n消极。

以{"结果": ""}的JSON格式返回最终结果。



{"结果": "消极"}



上下文学习

提示方法

➤ 上下文学习 (GPT3)

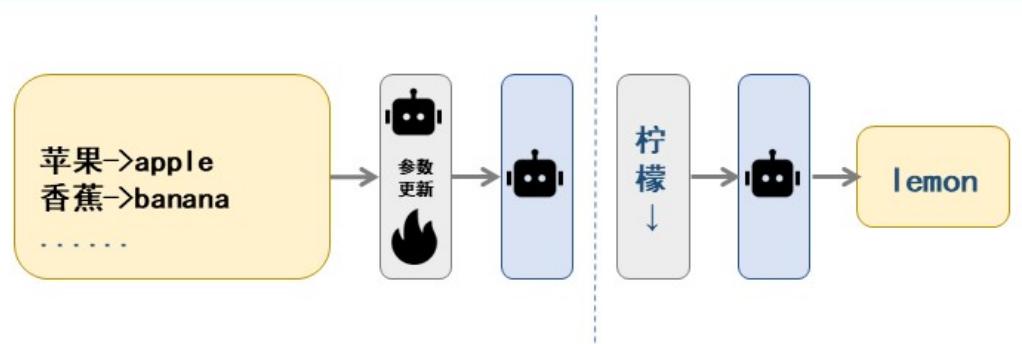
构造包含具体的指令以及相关上下文信息的详细清晰的Prompt，要求大语言模型按照命令输出。

特点：

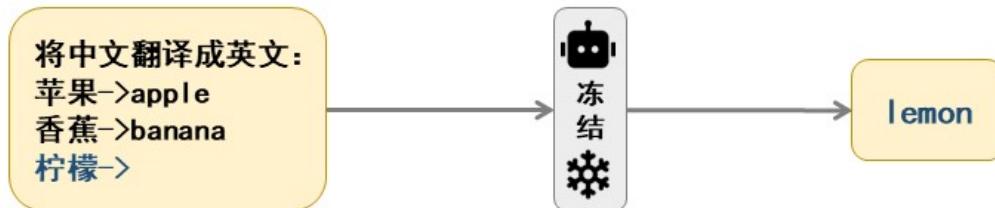
- 不需要进行模型参数微调
- 只有当模型参数量达到一定程度后，上下文学习才有用
- 通过举例子，激发大模型能力上限并不是很高，有些场合下，即使举了很多例子，也无法激活大模型在这方面的能力
- 举例子要有代表性和多样性，不能举太多例子，因为prompt长度受限

上下文学习 vs. 监督微调

监督微调：需要构造训练集，通过更新模型参数来适应下游任务，效果更好，但是成本高。



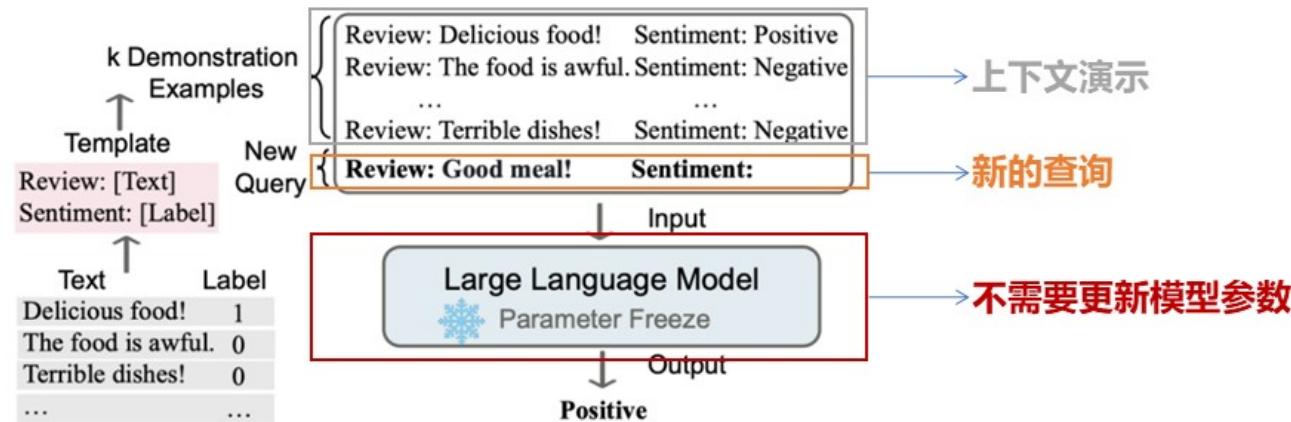
上下文学习：通过任务说明，演示示例等信息引导模型输出，快速适应新任务，使语言模型即服务成为可能。



上下文学习 (In Context Learning, ICL)

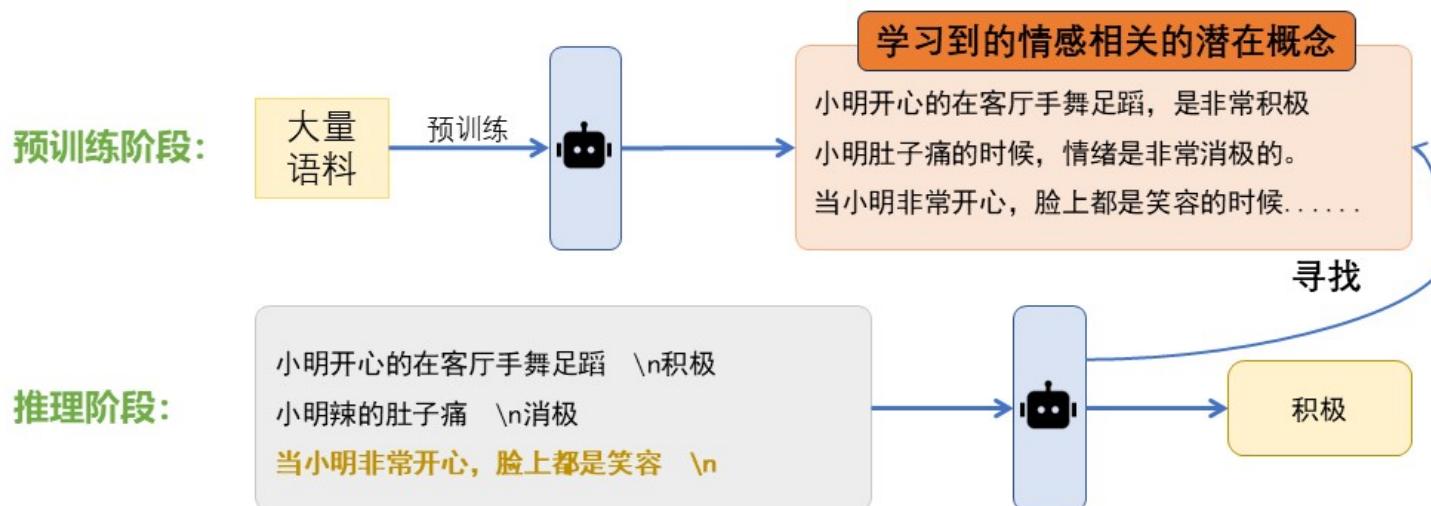
关键思想：从类比中学习

首先，ICL 需要一些示例来形成一个演示上下文。然后 ICL 将查询的问题（即你需要预测标签的 input 和一个上下文演示（一些相关的 cases）连接在一起，形成带有提示的输入，并将其输入到语言模型中进行预测。



为什么上下文学习有效？

大语言模型在预训练阶段从大量文本中学习**潜在的概念**。当运用上下文学习进行推理时，其借助提示词中的任务说明和示例来“**寻找**”大模型在**预训练期间学习的相关的概念**，示例中的输入**文本的分布**和**标签空间**对上下文学习的性能提升起到关键作用，从而引导大模型完成特定的任务。



ICL分类

按照示例数量的不同，上下文学习可以分为三类：零样本（Zero-shot）上下文学习，单样本（One-shot）上下文学习和少样本（Few-shot）上下文学习。

判断下面句子的情感为积极还是消极。

待分类的句子：老板，这个新项目很好，它让我感到非常兴奋和有动力，只是我最近发现自己很难有时间陪伴家人了。



Zero-shot

判断下面句子的情感为积极还是消极。

待分类的句子：老板，这个新项目很好，它让我感到非常兴奋和有动力，只是我最近发现自己很难有时间陪伴家人了。

针对此问题有如下示例：

示例：老板，我非常喜欢这个新项目，它让我感到非常兴奋和有动力。
\\n积极。



One-shot

判断下面句子的情感为积极还是消极。

待分类的句子：老板，这个新项目很好，它让我感到非常兴奋和有动力，只是我最近发现自己很难有时间陪伴家人了。

针对此问题有如下示例：

示例1：老板，我非常喜欢这个新项目，它让我感到非常兴奋和有动力。
\\n积极。

示例2：老板，这个项目的工作量太大了，我感到压力很大。
\\n消极。

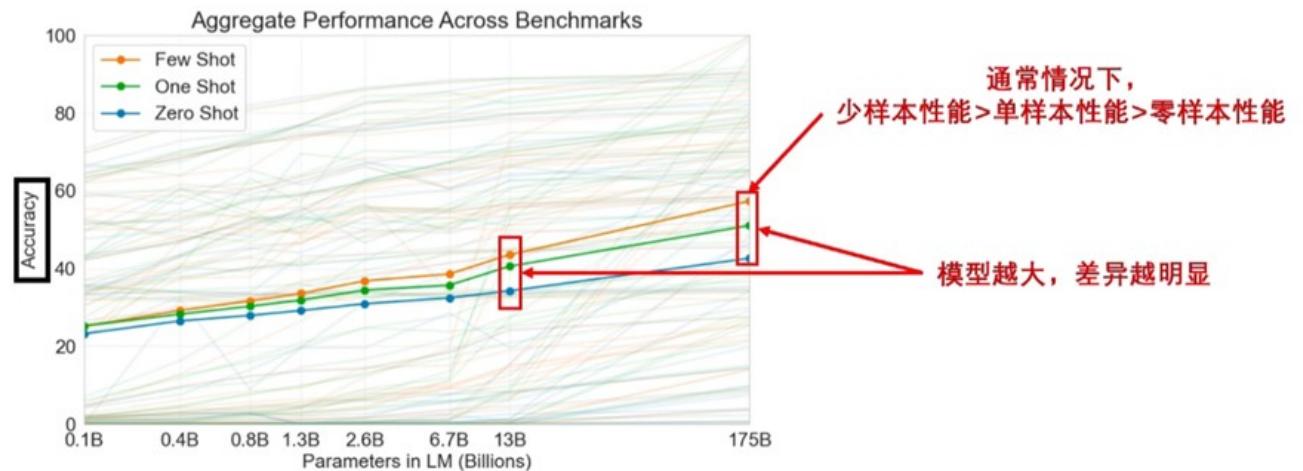
示例3：老板，虽然项目很有趣，但我最近感到有些疲惫。
\\n混合。



Few-shot

由于示例数量的不同，不同类型的上下文学习适用**场景不同**且在性能上存在**差异**。

不同规模的GPT-3在42个基准上的总体表现[1]



[1] Brown T B. Language models are few-shot learners[J]. arXiv preprint arXiv:2005.14165, 2020.

从ICL到思维链

尽管大模型的In-Context few-shot能力极强，传统prompt可以极大地提高大模型在各种下游任务中的灵活性和定制化程度。但在面对**数学计算等任务时效果并不好**，所以我们不只需要模型直接回答，更需要模型**思考和推理**。

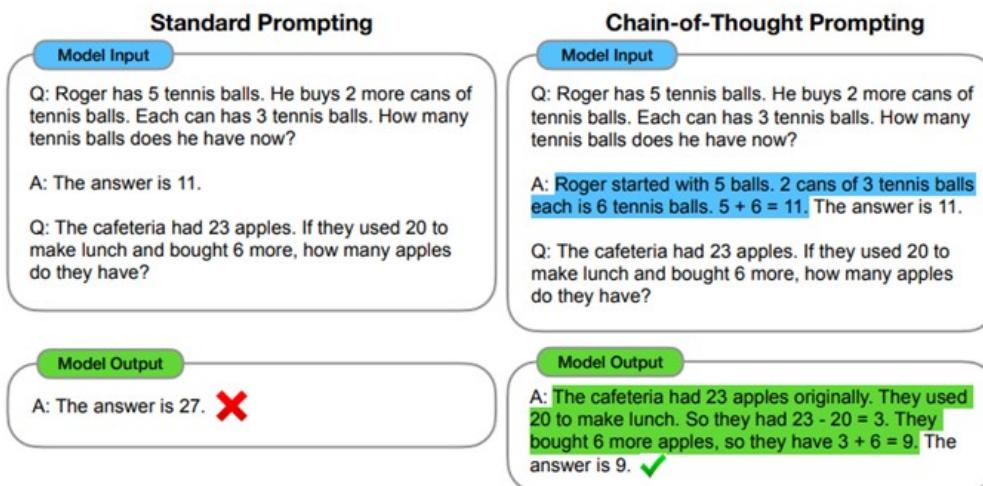


思维链

引导模型写出中间推
理步骤，从而提升逻
辑性与准确性。

思维链

思维链通过在提示中嵌入一系列中间推理步骤，引导大语言模型模拟人类解决问题时的思考过程，以提升模型处理System2任务的能力。



在Google的论文*Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*中被首次提出

Figure 1: Chain-of-thought prompting enables large language models to tackle complex arithmetic, commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted.

ICL vs CoT

判断下面句子的情感为积极还是消极。

待分类的句子：老板，这个新项目很好，它让我感到非常兴奋和有动力，只是我最近发现自己很难有时间陪伴家人了。

针对此问题有如下示例：

示例1：老板，我非常喜欢这个新项目，它让我感到非常兴奋和有动力。\\n积极。



One-shot ICL

Q: 罗杰有5个网球，他又买了两盒网球，每盒有3个网球。他现在有多少个网球？

A: 罗杰一开始有5个网球。他又买了2盒网球，每盒有3个网球，一共买了 $2 \times 3 = 6$ 个网球。所以他现在总共有 $5 + 6 = 11$ 个网球。

Q: 食堂有23个苹果，如果他们用掉20个后又买了6个，他们现在有多少个苹果？

A:

食堂有23个苹果，用掉20个后还剩下 $23 - 20 = 3$ 个苹果，之后又买了6个，他们现在还有 $3 + 6 = 9$ 个苹果。



One-shot CoT

- 增加上下文内容（相似的示例）



不只是告诉模型“要做什么”
还告诉模型如何一步一步“思考”

- 增加中间逻辑推理过程

思维链 (Chain-of-Thought, CoT)

➤ 启发于人类思维

在心理学中，System1任务和System2任务分别代表两种不同的思维方式所处理的任务类型。

System1：

System1的思考过程是快速、自动且无意识的。主要依靠直觉和经验进行判断，不需要你刻意去思考，往往在瞬间完成。

System2：

System2的思考过程比较缓慢，需要集中注意力和耗费精力，运用逻辑分析、计算和有意识的思考来解决问题。

SYSTEM 1

Intuition & instinct



Unconscious
Fast
Associative
Automatic polit



SYSTEM 2

Rational thinking



Conscious
Slow
Logic

思维链

➤ System1与System2的差异

- System1问题，如常识问答、情感分类、意图识别等，随着大模型参数量、算力开销、数据量协同增长，在标准提示下，性能显著增强。
- System2问题，如复杂数学计算、逻辑推理等，大模型表现出了“Flat Scaling Curves”现象，即模型规模增长未带来预期性能提升，性能提升缓慢甚至停滞不前。

System1：

“快速思考”系统，处理日常任务。
解读简单的表情 说母语 驾驶熟悉的路线

System2：

“慢速思考”系统，处理需深思熟虑的任务。
解决数学题 做复杂的决策和计划

Example:

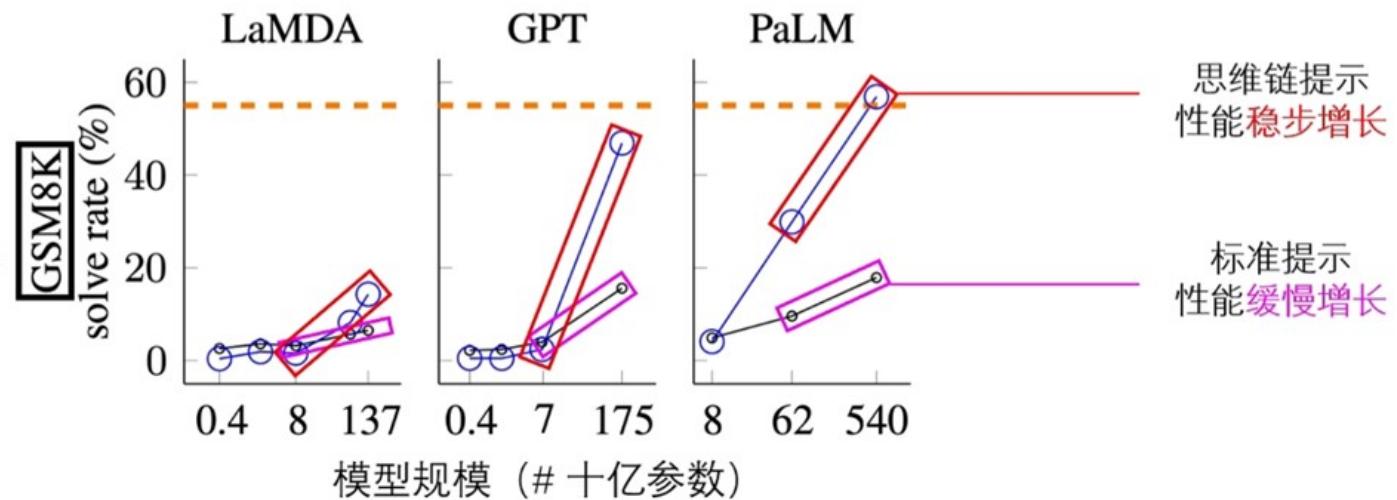
当你开车在熟悉的街道上行驶时，大部分时间你是在使用System 1，你可能同时在听音乐或思考其他事情。但是，如果一个球突然滚到马路上，孩子紧随其后，你的System 2会立即启动，全神贯注于如何迅速而安全地停车。

思维链

➤ 思维链在逻辑推理类任务上的优势

- 标准提示
- 思维链提示
- - - 监督学习微调

GSM8K
复杂数学问题集



思维链

➤ 思维链特点

- ✓ CoT能够“模拟人类在面对问题时一步步思考，最终得到答案的过程”。人类在面对复杂问题时，通常不会立马直接得到答案，而是会经过逐步推理、思考和验证等过程。
- ✓ CoT可以提升大模型的可解释性，让结果可控。通过展示逐步的推理过程，用户可以更容易理解模型是如何得出结论的，让模型由黑盒开始变为白盒。
- ✓ CoT对简单任务的提升有限，而能有效提升复杂任务的性能。思维链是一个能以较小代价而非常显著提升模型智力水平（逻辑能力、解题能力、代码能力）的技术。
- ✓ CoT对小模型的效果提升很小。一般情况下，CoT对小模型作用不大，模型参数至少达到7B才有效果，达到100B效果才明显。
- ✓ CoT推理使用成本高。根据KV cache机制，模型的生成速度和生成 token 数量几乎成正相关，而CoT恰恰又生成了大量的token。

思维链

➤ CoT应用场景

- CoT的用法—推理过程和训练过程。

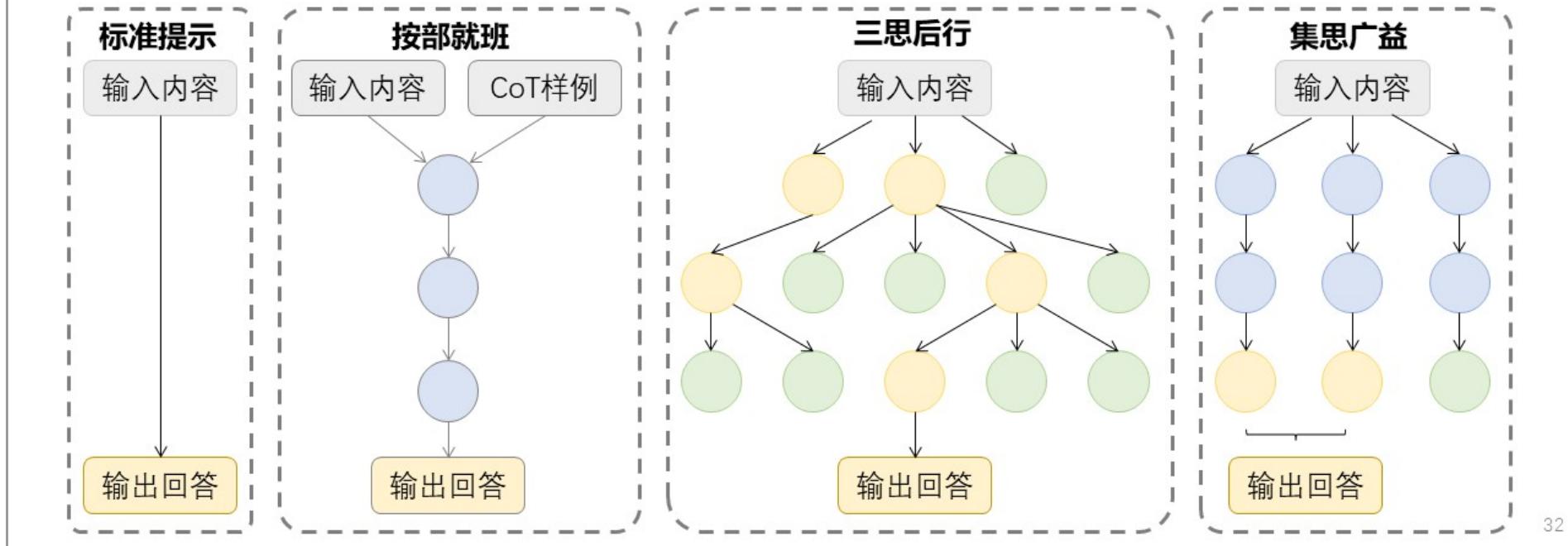
一种是在推理过程中直接做Prompt，另一种是在训练过程中使用CoT。

- 在推理过程中通常使用CoT就会有提升

CoT适用于通常需要逻辑推理的场景，应用场景不仅限于数学问题，还可以应用于逻辑推理、代码编程、法律分析、医疗诊断等领域。逐步展开的推理链条不仅让模型输出的答案更加准确，而且还提升了输出的可解释性。

思维链的分类

在标准Prompt方法上，按照推理方式的不同，可以归纳为三种模式：**按部就班**、**三思后行**和**集思广益**。



按部就班模式

按部就班模式强调的是逻辑的连贯性和步骤的顺序性。在这种模式下，模型像是在遵循一条预设的路径，每一步都紧密依赖于前一步的结论。

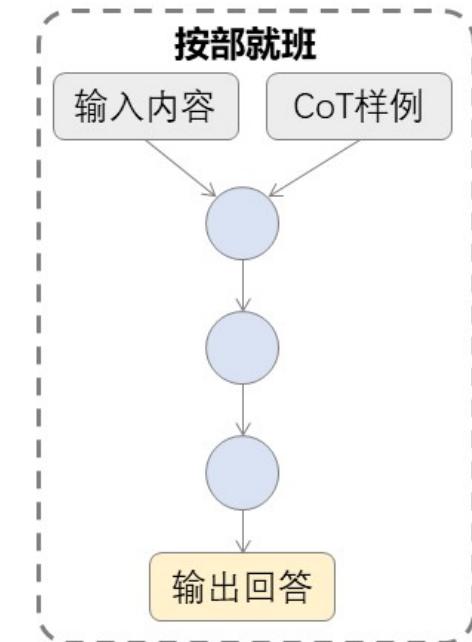
代表性方法：

- CoT[1]：通过手工构造一步一步推理回答问题的例子，作为示例放入 Prompt，来引导模型一步一步生成推理步骤。
- Zero-Shot CoT[2]：通过特定提示，自动化构造一步一步推理回答问题的例子。
- Auto-CoT[3]：从问题库中检索多个相关的问题，并利用Zero-Shot CoT针对每个问题自动化构造推理示例。

[1] Jason Wei et al. "Chain-of-thought prompting elicits reasoning in large language models". In: NeurIPS. 2022.

[2] Takeshi Kojima et al. "Large Language Models are Zero-Shot Reasoners". In: NeurIPS. 2022.

[3] Zhuosheng Zhang et al. "Automatic Chain of Thought Prompting in Large Language Models". In: ICLR. 2023.



标准 CoT

标准的CoT方法在提升模型推理能力方面取得了一定的成功，但是需要**费时费力地手工编写大量的CoT示例，并且过度依赖于CoT的编写质量。**

Q: 罗杰有5个网球，他又买了两盒网球，每盒有3个网球。他现在有多少个网球?
A: 罗杰一开始有5个网球。他又买了2盒网球，每盒有3个网球，一共买了 $2 \times 3 = 6$ 个网球。所以他现在总共有 $5 + 6 = 11$ 个网球。

Q: 食堂有23个苹果，如果他们用掉20个后又买了6个，他们现在有多少个苹果?
A:



食堂有23个苹果，用掉20个后还剩下 $23 - 20 = 3$ 个苹果，之后又买了6个，他们现在还有 $3 + 6 = 9$ 个苹果。



人工编写
费时费力

包含一个样本示例的CoT提示示例

Zero-Shot CoT

Zero-Shot CoT通过简单的提示，如“Let's think step by step”，引导模型自行生成一条推理链。其**无需手工标注的CoT示例**，减少了对人工示例的依赖，多个推理任务上展现出了与原始少样本CoT**相媲美甚至更优**的性能。

第一阶段：推理过程生成

Q: 罗杰有5个网球，他又买了两盒网球，每盒有3个网球。他现在有多少个网球?
A: 让我们一步一步思考。



罗杰一开始有5个网球。他又买了2盒网球，每盒有3个网球，一共买了 $2 \times 3 = 6$ 个网球。所以他现在总共有 $5 + 6 = 11$ 个网球。



第二阶段：答案生成

Q: 罗杰有5个网球，他又买了两盒网球，每盒有3个网球。他现在有多少个网球?
A: 让我们一步一步思考。

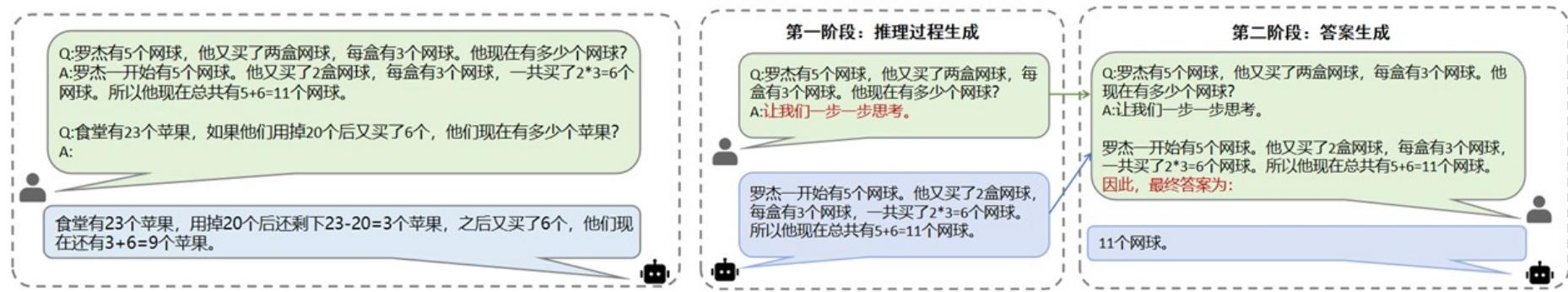
罗杰一开始有5个网球。他又买了2盒网球，每盒有3个网球，一共买了 $2 \times 3 = 6$ 个网球。所以他现在总共有 $5 + 6 = 11$ 个网球。
因此，最终答案为：

11个网球。



标准CoT vs. Zero-Shot CoT

- 标准CoT方法在推理任务上性能优越，但是需要手工编写推理链示例。
- Zero-Shot CoT无需人工编写样本样例，但是效果不如标准的CoT方法。

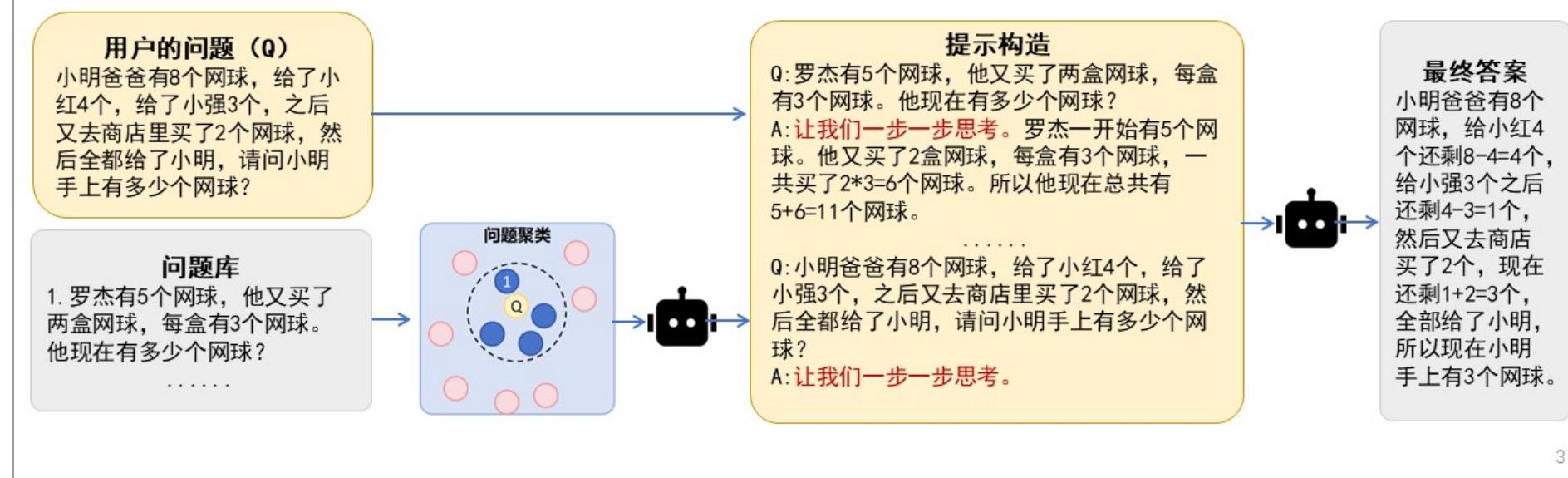


- 标注成本高
- 算法性能好

- 标注成本低
- 算法性能差

Auto-CoT

在Zero-Shot CoT的基础上，Auto-CoT引入与待解决问题相关的问题及其推理链作为示例，以继续提升CoT的效果。Auto-CoT无需人工标注成本，但是性能超过了需要手工标注的COT和无需手工标注的Zero-Shot CoT。



按部就班模式的不足

人类在解决System2类问题时，会有一个反复选择以及回溯的过程。已有的CoT提示方法无法模拟这种过程，从而导致能力受限。

System2任务决策过程：

- 会维护并探索当前选择的各种不同替代方案，而不仅仅挑选其中一个。
- 会评估自身当前状态，并展望未来或回溯以做出更具全局性的决策。

已有CoT提示的问题：

- 在局部上，它们在思维过程中不会探索不同的后续内容。
- 在全局上，大语言模型顺序链式输出，不存在规划、前瞻性思考、自我评估和回溯的过程。

三思后行模式

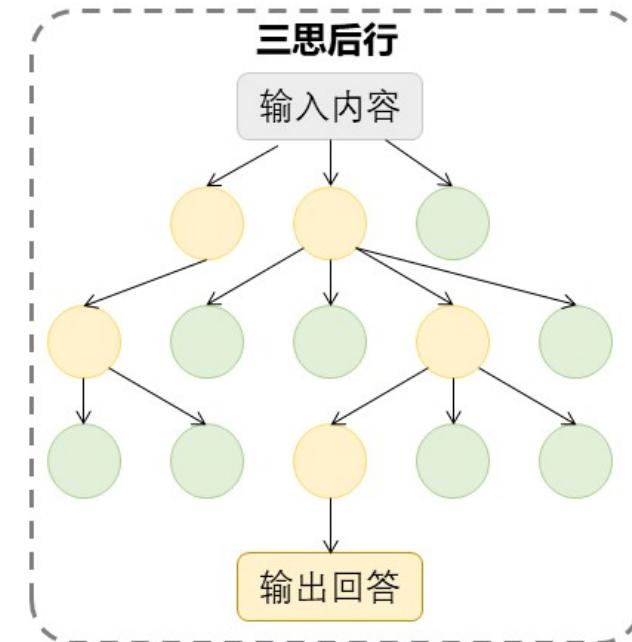
在决策过程中融入**审慎和灵活性**。这种模式下，模型在每一步会停下来评估当前的情况，判断是否需要调整方向。

代表性方法：

- Tree of Thoughts[1]: 将推理过程视为一棵思维树，从拆解、衍生、评估、搜索四个方面进行构造。
 - Graph of Thoughts[2]: 通过特定提示，自动化构造一步一步推理回答问题的例子。

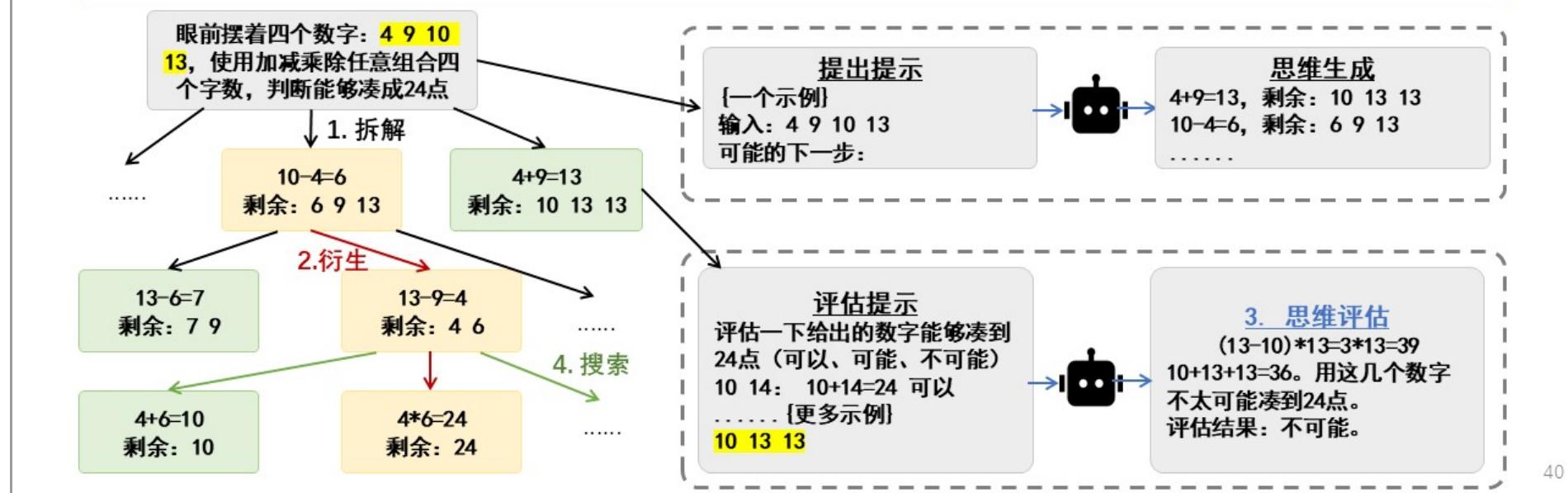
[1] Shunyu Yao et al. "Tree of Thoughts: Deliberate Problem Solving with Large Language Models". In: NeurIPS, 2023.

[2] Maciej Besta et al. "Graph of Thoughts: Solving Elaborate Problems with Large Language Models". In: AAAI. 2024.



Tree of Thoughts

为了模拟人类在做System2任务时**审时度势**的过程，Tree-of-Thought (ToT) 将推理过程构造为一棵思维树，在构造树过程中，允许模型在遇到困难或不确定性时进行**回溯**和**重新选择**。



Tree of Thoughts

给定当前树节点 s , 生成下一步的 k 条 candidate thoughts (思路/中间步骤)。

- Sample (采样) —— 适用于“thought 很长, 多样性很高”的任务
- Propose (顺序生成) —— 适用于“thought 很短, 容易重复”的任务



让语言模型通过 deliberate reasoning (自我推理) 直接评估状态好坏。

- 单步打分;
- 多步比较;

State

Evaluator

决定“往哪条分支走”的启发式函数

Search 决定如何使用这些评价信息来探索树

- BFS: 每个步骤保留 b 个有希望的状态。
- DFS: 首先探索最有希望解决问题的状态, 直到最终的结果被输出; 或者状态评估器认为当前状态不可能解决该问题。

	Game of 24	Creative Writing	5x5 Crosswords
Input	4 numbers (4 9 10 13)	4 random sentences	10 clues (h1.presented;..)
Output	An equation to reach 24 $(13-9)*(10-4)=24$	A passage of 4 paragraphs ending in the 4 sentences	5x5 letters: SHOWN; WIRRA; AVAIL; ...
Thoughts	3 intermediate equations $(13-9=4 \text{ (left } 4,4,10); 10-4=6 \text{ (left } 4,6); 4*6=24)$	A short writing plan (1. Introduce a book that connects...)	Words to fill in for clues: (h1.shown; v5.naled; ...)
#ToT steps	3	1	5-10 (variable)

Table 1: Task overview. Input, output, thought examples are in blue.

Tree of Thoughts

<https://github.com/dave1010/tree-of-thought-prompting>

ToT 提示词的例子如下：

假设三位不同的专家来回答这个问题。

所有专家都写下他们思考这个问题的第一个步骤，然后与大家分享。

然后，所有专家都写下他们思考的下一个步骤并分享。

以此类推，直到所有专家写完他们思考的所有步骤。

只要大家发现有专家的步骤出错了，就让这位专家离开。

请问...

Expert 1: The ball is in the living room.

Expert 2: The ball is in the kitchen.

Expert 3: The ball is in the bedroom.

Expert 1: Bob carries the cup to the bedroom, so the ball must be in the cup.

Expert 2: Oh, I see my mistake. Yes, the ball is in the cup.

Expert 3: Agreed, the ball is in the cup in the bedroom.

Expert 1: Next, Bob turns the cup upside down in the bedroom.

Expert 2: After that, Bob walks to the garden and puts the cup down.

Expert 3: Therefore, the ball must have fallen out of the cup when Bob turned it upside down in the bed

Expert 1: Oh, you're right. I made a mistake. The ball is in the bedroom, not in the cup.

Expert 2: Agreed, the ball is in the bedroom.

Expert 3: Bob then walks to the garage, so the ball remains in the bedroom. It is not in the garden or

Expert 1: Absolutely, the ball is still in the bedroom.

Expert 2: Yes, the ball hasn't moved from the bedroom.

Expert 3: Therefore, the ball is in the bedroom, not in the garden or the garage.

All three experts agree that the ball is in the bedroom.

Graph of Thoughts

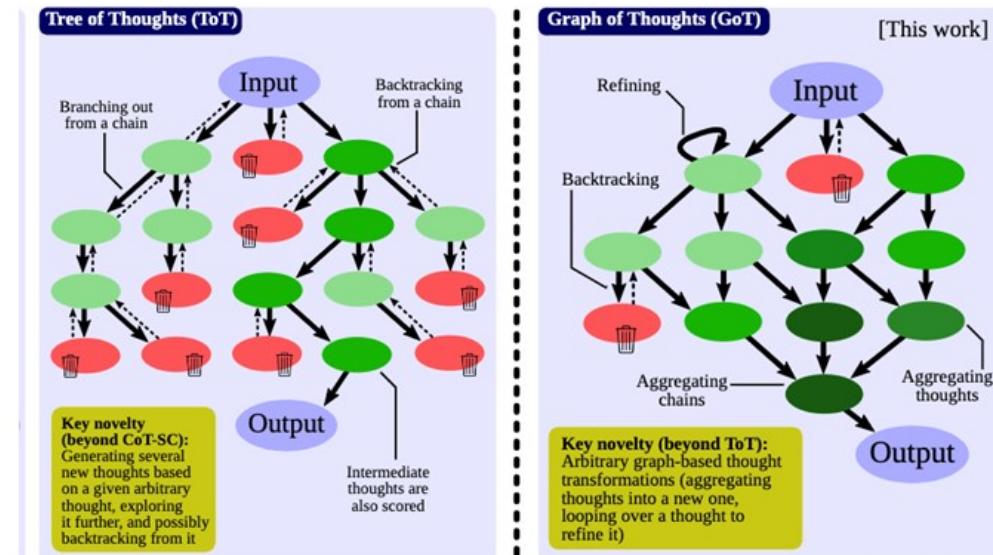
➤ 树有两个问题

- 节点之间不能互相连接，A 分支和 B 分支不能合并。
- 不能重用中间推理，每条路径都是独立的，不共享结果。

➤ ToT不适合

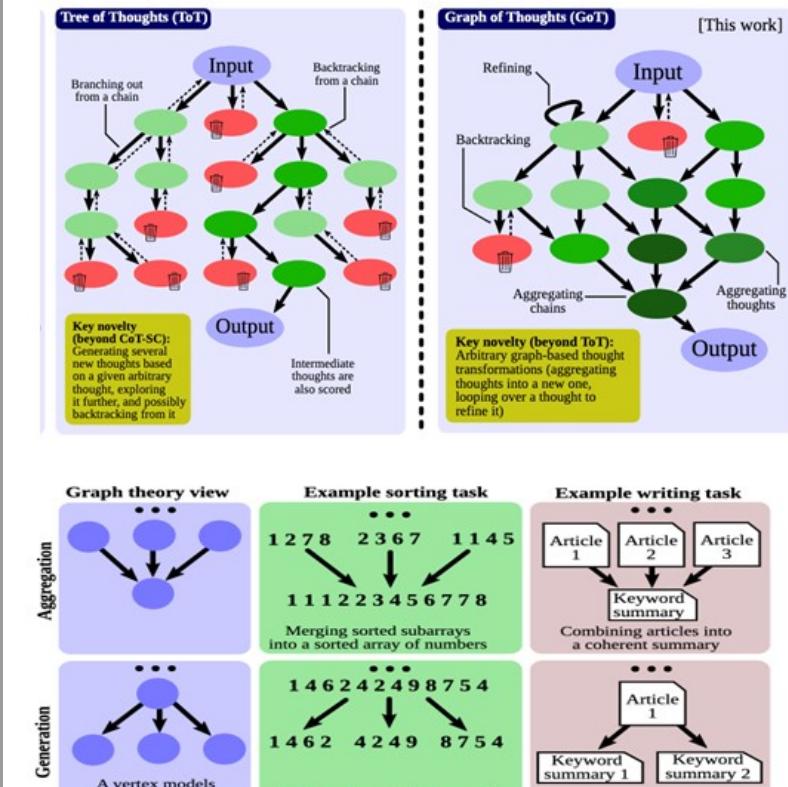
- 多条件整合
- 多子任务协同
- 行程规划（不同路径需要合并）
- 代码调试
- 分解式任务（不同节点依赖共同信息）

于是研究者提出：把推理看成一个图：节点可以合并、分裂、重连。



GoT目前仍处于早期研究阶段

Graph of Thoughts



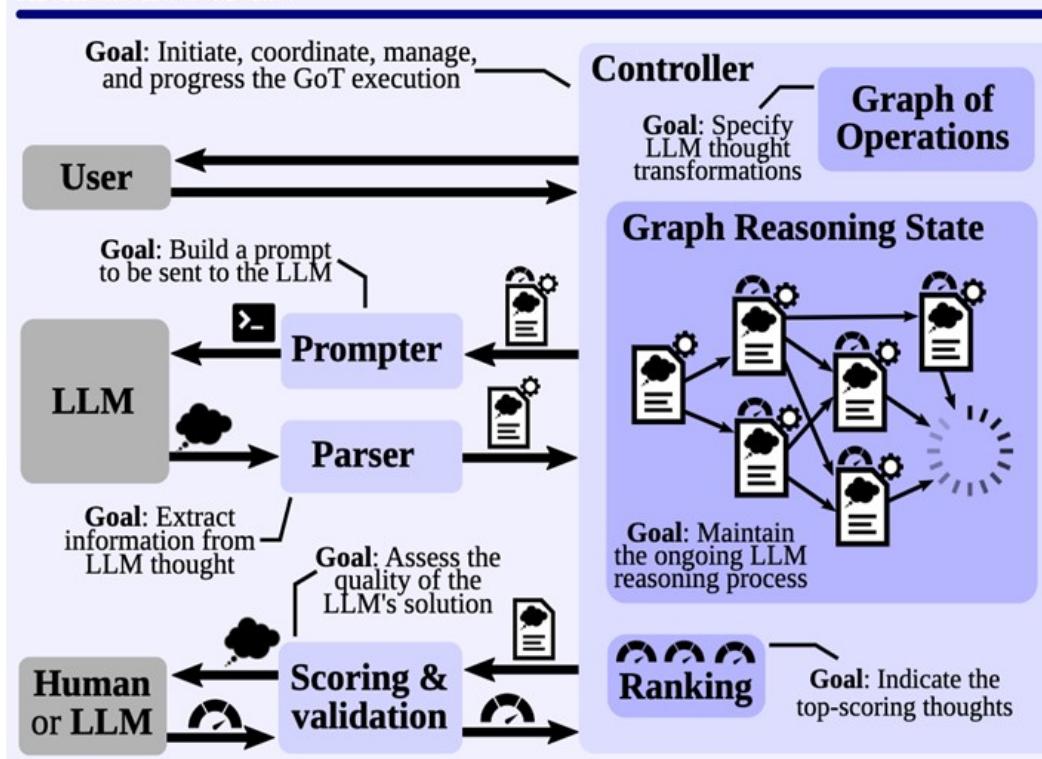
整个Graph是一个有向图，每个节点代表不同类别的thoughts，整个图是一个异构图。而图之间的有向边，则表示节点间的依赖关系。

➤ GoT有三种操作

- **Aggregation**指的是几个想法的整合，变成了一个更好的想法，图上表现为多个节点指向同一个节点；
- **Refining**指节点自身进行反思，不断改善自己本身的内容，图上表现为一个节点重新指向自己；
- **Generation**指通过一个节点，产生了后续1个或者更多新的想法，图上表现为一个节点指向一个或多个节点。

Graph of Thoughts

Architecture overview



「提示器」：为 LLM 准备消息或提示。提示应包含图形结构的编码。

「解析器」：从LLM输出中提取相关信息，从而形成存储在每个想法中的状态。

「评分」：验证思想状态是否满足正确性条件并为其分配分数（来自LLMs或人类注释者）。

「控制器」：协调推理过程并决定如何进行。值得注意的是，控制器选择应应用于底层图的思维转换，将此信息传达给提示器，并根据评分器对生成的思维状态的输出来决定推理过程是否已完成或应继续前进。在整个过程中，控制器维护两条信息：

「操作图」：用户定义的静态结构，在推理过程之前创建，并捕获思维操作的执行计划。

「图推理状态」：跟踪LLM推理过程状态的动态结构，包括所有想法及其状态。

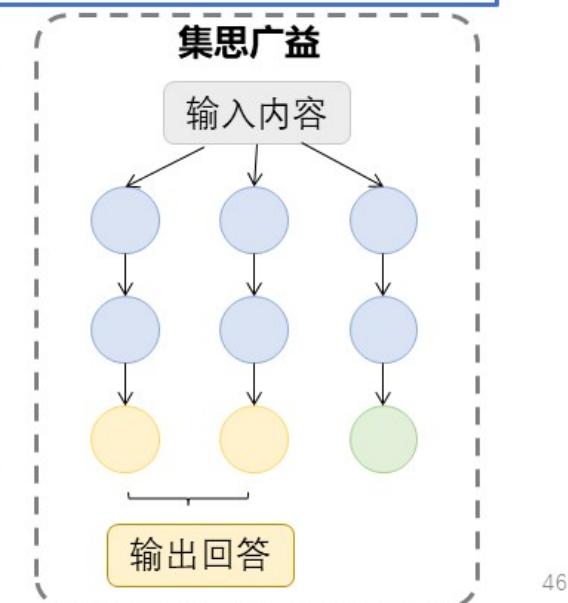
集思广益模式

集思广益模式强调的是通过汇集多种不同的观点和方法来优化决策过程。在这种模式下，模型不仅仅依赖于单一的推理路径，而是通过探索多种可能的解决方案，从中选择最优的答案。

代表性方法：

- Self-Consistency[1]: 引入多样性的推理路径，从中提取并选择最一致的答案，从而提高了模型的推理准确性。
- Universal Self-Consistency[2]: 利用LLMs自身选择最一致的答案，支持更多重任务，无需答案提取过程。在多任务中性能良好且输出匹配度高、对顺序鲁棒。

[1] Xuezhi Wang. Et al. "Self-Consistency Improves Chain of Thought Reasoning in Language Models". In:ICLR. 2023.
[2] Google DeepMind. "Universal Self-Consistency for Large Language Model Generation". In:Arxiv. 2023.



Self-Consistency

Self-Consistency引入多样性的推理路径并从中选择最一致的答案，从而提高了模型的推理准确性。Self-Consistency不依赖于特定的CoT形式，可以与其他CoT方法兼容，共同作用于模型的推理过程。

Self-Consistency的设计源自于以下直觉：

- 复杂推理任务通常允许多种能够得出正确答案的推理路径。
- 对于一个问题，所需的深思熟虑和分析越多，能够得出答案的推理路径的多样性就越大。

Q: 小明爸爸有8个网球，给了小红4个，给了小强3个，之后又去商店里买了2个网球，然后全都给了小明，请问小明手上有多少个网球。
A: 让我们一步一步思考。



小明爸爸有8个网球，给小红4个还剩 $8-4=4$ 个，给小强3个之后还剩 $4-3=1$ 个，然后又去商店买了2个，现在还剩 $1+2=3$ 个，全部给了小明，所以现在小明手上有3个网球。

小明爸爸有8个网球，给小红4个还剩 $8-4=4$ 个，然后又去商店买了2个，现在还剩 $4+2=6$ 个，全部给了小明，所以现在小明手上有6个网球。

小明爸爸一开始有8个网球，给小红4个还剩 $8-4=4$ 个，又给小强3个还剩 $4-3=1$ 个，又去商店买了2个，还剩 $1+2=3$ 个，全部给了小明，所以现在小明手上有3个网球。

基于结果
投票

5个网球

Self-Consistency

Self-Consistency的过程分为三个步骤：推理路径生成、汇总答案和选择答案。

- 推理路径生成：在随机采样策略下，使用CoT或Zero-Shot CoT的方式来引导大语言模型针对待解决问题生成一组多样化的推理路径；
- 汇总答案：针对大语言模型生成的每个推理内容，收集其最终的答案，并统计每个答案在所有推理路径中出现的频率；
- 选择答案：选择出现频率最高的答案作为最终的、最一致的答案。

Q: 小明爸爸有8个网球，给了小红4个，给了小强3个，之后又去商店里买了2个网球，然后全都给了小明，请问小明手上有多少个网球。
A: 让我们一步一步思考。



小明爸爸有8个网球，给了小红4个还剩 $8-4=4$ 个，给了小强3个之后还剩 $4-3=1$ 个，然后又去商店买了2个，现在还剩 $1+2=3$ 个，全部给了小明，所以现在小明手上有3个网球。

小明爸爸有8个网球，给了小红4个还剩 $8-4=4$ 个，然后又去商店买了2个，现在还剩 $4+2=6$ 个，全部给了小明，所以现在小明手上有6个网球。

小明爸爸一开始有8个网球，给了小红4个还剩 $8-4=4$ 个，又给了小强3个还剩 $4-3=1$ 个，又去商店买了2个，还剩 $1+2=3$ 个，全部给了小明，所以现在小明手上有3个网球。

基于结果
投票

5个网球

Universal Self-Consistency

Self-Consistency依赖于答案提取过程来聚合多个解决方案，不适用与自由形式的答案，例如文本摘要等任务。**Universal Self-Consistency**利用大语言模型本身来选择最一致的答案，显著拓宽了Self-Consistency的使用场景。

Q: 小明爸爸有8个网球，给了小红4个，给了小强3个，之后又去商店里买了2个网球，然后全都给了小明，请问小明手上有多少个网球？
A: 让我们一步一步思考。

生成多个回答

小明爸爸有8个网球，给小红4个还剩 $8-4=4$ 个，给小强3个之后还剩 $4-3=1$ 个，然后又去商店买了2个，现在还剩 $1+2=3$ 个，全部给了小明，所以现在小明手上有3个网球。

小明爸爸有8个网球，给小红4个还剩 $8-4=4$ 个，然后又去商店买了2个，现在还剩 $4+2=5$ 个，全部给了小明，所以现在小明手上有5个网球。

小明爸爸一开始有8个网球，给小红4个还剩 $8-4=4$ 个，又给小强3个还剩 $4-3=1$ 个，又去商店买了2个，还剩 $4+2=5$ 个，全部给了小明，所以现在小明手上有5个网球。

根据这个问题：Q
有以下回答：
回答1
回答2
回答3
从中选择最一致性的答案。

利用大模型自身做选择过程

→ 机器人 → 最终答案

单选题 1分

在思维链 (CoT) 的分类中，为了模拟人类在复杂任务中“反复选择和回溯”的过程，将推理过程构建为树状结构来允许模型在不确定时重新选择，属于哪一种模式？

- A 按部就班模式
- B Zero-Shot CoT
- C 三思后行模式
- D 集思广益模式

Prompt工程

➤ 设计提示的通用技巧

- **从简单开始：**设计提示时实际上是一个迭代过程。从简单的提示词开始，并逐渐添加更多元素和上下文。
- **指令：**使用命令来指示模型执行各种简单任务，例如“写入”、“分类”、“总结”、“翻译”、“排序”等，从而为各种简单任务设计有效的提示。可以将指令放在开头，或者使用类似“###”更清晰的注明。

```
### 指令 ###
将以下文本翻译成西班牙语：
文本：“hello! ”
```

Prompt工程

➤ 设计提示的通用技巧

- **具体性:** 非常具体地说明你希望模型执行的指令和任务。提示越具描述性和详细，结果越好。但同时也应注意提示词的长度，包含太多不必要的细节反而会是效果变差。
- **做什么还是不做什么?**: 避免说不要做什么，而应该说要做什么。这样更加的具体，更有利于模型生成良好回复。

提示:
以下是向客户推荐电影的代理程序。不要询问兴趣。不要询问个人信息。
客户: 请根据我的兴趣推荐电影。
代理:
输出:
当然, 我可以根据你的兴趣推荐电影。你想看什么类型的电影? 你喜欢动作片、
喜剧片、爱情片还是其他类型的电影?



VS

提示:
以下是向客户推荐电影的代理程序。代理负责从全球热门电影中推荐电影。它应
该避免询问用户的偏好并避免询问个人信息。如果代理没有电影推荐, 它应该回
答“抱歉, 今天找不到电影推荐。”。
客户: 请根据我的兴趣推荐一部电影。
代理:
输出:
抱歉, 我没有关于你兴趣的任何信息。不过, 这是目前全球热门的电影列表: [电
影列表]。希望你能找到喜欢的电影!



Prompt工程

➤ 设计提示的策略

设定明确的目标

- 使用操作动词指定期望的操作
“撰写项目符号列表，总结所附研究论文的主要发现”
- 定义所需的输出长度和格式
“写一篇 500 字的文章，讨论气候变化对沿海社区的影响。”
- 指定目标受众群体
“为有机护肤新品系列撰写产品说明，受众为关注可持续发展的年轻人。”

提供上下文和背景信息

- 包含相关事实和数据
“考虑到自前工业时代以来，全球温度上升了 1 摄氏度，请讨论海平面上升的潜在影响。”
- 引用特定的来源或文档
“根据随附的财务报告，分析公司过去五年的盈利能力。”
- 当以关键术语和概念
“用简单的词语解释量子计算的概念，适合非技术人员。”

使用少样本提示

- 提供一些输入-输出对的示例
 - 输入：xxx 输出：xxx
- 演示所需的风格或基调
 - 示例1（幽默）xxx。示例2（正式）xxx。
- 展示所需的详细程度
 - 示例1（简单）xxx。示例2（详细）xxx。

<https://cloud.google.com/discover/what-is-prompt-engineering>

53

Prompt工程

➤ 设计提示的策略

具体说明

- 使用确切的语言，避免含糊不清
 - 不要使用“写一些关于气候变化的内容”，而应该使用“写一篇有说服力的文章，主张实施更严格的碳排放法规。”
- 尽可能量化请求
 - 不要使用“写一首长诗”，而应使用“写一首十四行诗，探讨爱与失去主题。”
- 将复杂的任务分解为较小的步骤
 - 不要使用：“创建营销计划”，而应使用“1. 确定目标受众群体。2. 撰写关键营销宣传信息。3. 选择合适的营销渠道。”

迭代和实验

- 尝试使用不同的措辞和关键字
 - 使用同义词或替代性句子结构重新表述提示。
- 调整详细程度和具体程度
 - 添加或移除信息以微调输出。
- 测试不同的提示长度
 - 尝试用较短和长的提示来取得最佳平衡

利用思维链提示

- 鼓励逐步推理
 - “逐步解决这个问题：小强有 5 个苹果，他吃了 2 个。他还剩多少个苹果？第 1 步：xxx。第 2 步：xxx。第 3 步：xxx。回答：小强还剩 3 个苹果。”
- 让模型解释其推理过程
 - “解释你在确定这部电影评论情绪中的思维过程：“表演很棒，但情节可以预料。”
- 引导模型完成符合逻辑的思维顺序
 - “要将此电子邮件归类为垃圾邮件或非垃圾邮件，请考虑以下几点：1. 是已知发送者吗？2. 主题行是否包含可疑关键字？3. 电子邮件里提供的内容是不是太好了，让人难以置信？”

<https://cloud.google.com/discover/what-is-prompt-engineering>

54

提示词设计

CO-STAR

- **C** context 上下文
 - 提供人物的背景信息
- **O** objective 目标
 - 定义希望执行的任务
- **S** Style 风格
 - 希望使用的风格
- **T** tone 语调
 - 设定回应的态度
- **A** audience 受众
 - 确定响应的目标受众
- **R** Response 回复
 - 响应格式

CONTEXT (上下文)

我想推广公司的新产品。我的公司名为 Alpha，新产品名为 Beta，是一款新型超快速吹风机。

OBJECTIVE (目标)

帮我创建一条 Facebook 帖子，目的是吸引人们点击产品链接进行购买。

STYLE (风格)

参照 Dyson 等成功公司的宣传风格，它们在推广类似产品时的文案风格。

TONE (语调)

说服性

AUDIENCE (受众)

我们公司在 Facebook 上的主要受众是老年人。请针对这一群体在选择护发产品时的典型关注点来定制帖子。

RESPONSE (响应)

保持 Facebook 帖子简洁而深具影响力。

提示词设计

➤ 使用分隔符进行分段提示

分隔符是特殊的标记，可帮助 LLM 区分应将提示的哪些部分视为单个含义单位。这很重要，因为整个提示会作为单个长令牌序列到达 LLM。分隔符通过隔离提示的特定部分以进行不同的处理，从而为该标记序列提供结构。值得注意的是，分隔符可能不会影响LLM对简单任务的响应质量。然而，任务越复杂，分段分隔符的使用对 LLM 响应的影响就越大。

分隔符可以是通常不会一起出现的特殊字符的任何序列，例如：

====
>>>

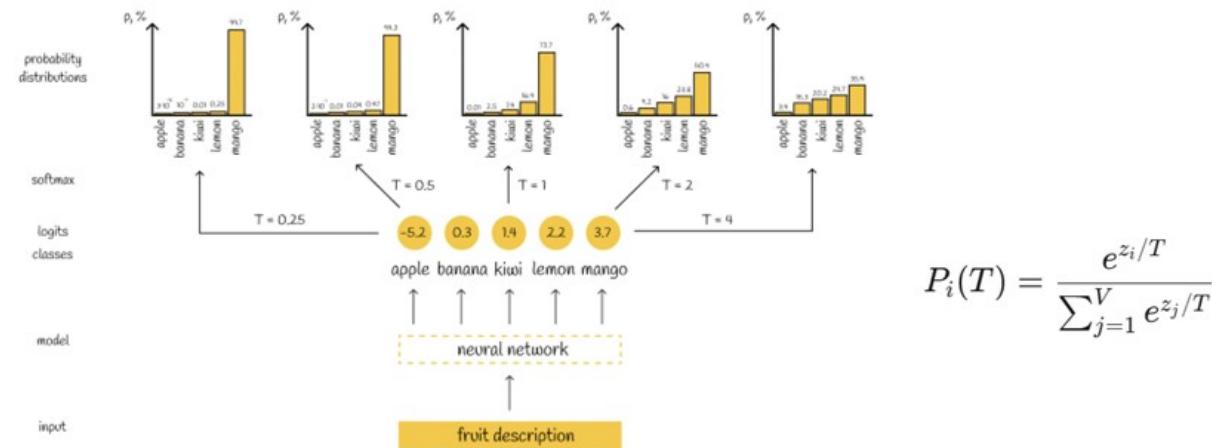
所选择的特殊字符的数量和类型并不重要，只要它们足够独特，以使LLM能够将它们理解为内容分隔符而不是普通标点符号。

模型 (API) 设置

- 除了prompt的设计，模型参数设置也很重要，是提高响应可靠性的关键
- 可能需要进行一些实验才能找出适合的正确设置。

大模型输出的本质，是从所有候选 Token 的概率分布中进行采样的结果；而这一采样过程，则由 Temperature / Top_p 等一系列关键参数来精确控制。

1. Temperature: 高温度使概率分布更平坦（平滑，更均匀），从而缩小了最高概率词与其他词之间的差距，生成答案的随机性和多样性就越高



模型设置

2. Top_k: 限制了模型在每一步中输出的前 K 个最可能的词。

- 在 Top-K 采样中，如果设定 K=5
- 只考虑经过排序后的前 5 个最高概率的词
- 在这 5 个词之间重新归一化概率，使其总和为 1
- 模型从重新归一化的概率分布中对这 5 个词进行采样

采样范围固定，对于不同语境，选取的词要么太多或者太少

3. Top_p: 过滤掉累积概率低于指定阈值 (p) 的词，从而增加输出的多样性，同时避免低概率的词。

候选 Token	概率 (P_i)	累积概率	原始 Logit
A (天气)	0.40	0.40	5.0
B (天空)	0.30	0.70	4.5
C (风景)	0.15	0.85	3.8
D (空气)	0.05	0.90	3.0
E (水流)	0.04	0.94	2.5
F (树木)	0.03	0.97	2.0
G (阳光)	0.02	0.99	1.5
H...	0.01	1.00	1.0

- 若 $\text{top_p}=0.9$ ，则最终的采样范围被限定在 Token A、B、C 和 D 这四个词中

解决了Top_k 采样（只取前k个词）的缺陷：

- 对于尖锐的概率分布：只有 2-3 个词的累积概率就达到0.9。采样范围**自动收缩**。
- 对于平坦的概率分布：可能需要 10-20 个词的累积概率才能达到 0.9。采样范围**自动扩大**。

模型设置

- Temperature/top_k/top_p 三个参数共同决定了模型在生成下一个Token时的**随机性和候选集**
 - Temperature 是最基础且始终作用的参数
 - Top_p 或 Top_k (候选集限定器) 选择其一, top_p被认为是更先进、更灵活的采样方法。
 - 一般建议是改变Temperature 和 Top P 其中一个参数, 不用两个都调整。

4. **Max Length**: 通过调整 max length 来控制大模型生成的token数。
5. **Stop Sequences**: 是一段预定义的文本, 用于指示模型在生成过程中应该停止进一步的文本输出。模型会在每个生成步骤中考虑停止序列作为潜在的输出。当模型预测到停止序列的概率足够高时, 它会结束生成过程。

主观题 1分

当你希望大模型在创作故事时，能够增加更多的随机性和新颖性，但又不希望它陷入对某个词语的过度重复，你应该如何调整参数？

- A 增加 Max Length 和 Stop Sequences
- B 降低 Temperature，同时增加 Frequency Penalty
- C 增加 Temperature，同时增加 Frequency Penalty
- D 增加 Top_k，同时降低 Presence Penalty

豆包



生成照片，男性，亚洲人，20多岁，高度专注并投入于听讲座

环境：大学阶梯教室，傍晚，自然光从左侧的大窗户照射进来，照亮了场景。教室是明亮通风的。背景中隐约可见其他虚化的学生，以暗示课堂环境，但不会分散对主体的注意力。

焦点与细节：*学生坐在一张现代金属课桌前，桌上摆放着打开代码界面的笔记本电脑。学生的目光专注而坚定，眉宇间略微的皱纹显示出深度的思考和集中。旁边放着一个水瓶。画面中可见细微的细节，例如（屏幕上的光线反射 / 笔记本纸张的纹理 / 笔尖留下的轻微压痕）。

光线与氛围：柔和而有方向感的光线营造出深度感，并凸显学生的面部。阴影真实自然，与光源方向一致。整体氛围是（平静而好学 / 紧张而专注 / 宁静而沉思）。浅景深（或插画中的轻微模糊效果）有助于突出学生，强调其专注状态。

Nano Banana



Photo, a male, East Asian, in his 20s, highly focused and engrossed in listening to a lecture.

Environment: A university lecture hall, evening. Natural light streams in from the large windows on the left, illuminating the scene. The classroom is bright and airy. In the background, the faint, blurred figures of other students suggest a classroom setting without distracting from the main subject.

Focus and Details: The student is seated at a modern metal desk with an open laptop displaying a code interface. His gaze is focused and determined, with a slight furrow between his brows indicating deep thought and concentration. A water bottle is placed nearby. Subtle details are visible in the scene, such as (light reflections on the screen / the texture of notebook paper / slight indentations from pen strokes).

Lighting and Atmosphere: Soft, directional light creates a sense of depth and highlights the student's face. Shadows are realistic and natural, consistent with the direction of the light source. The overall atmosphere is (calm and studious / tense and focused / serene and contemplative). A shallow depth of field (or a slight blur effect in illustrations) helps emphasize the student, underscoring his state of concentration.

主观题 10分

Prompt练习：

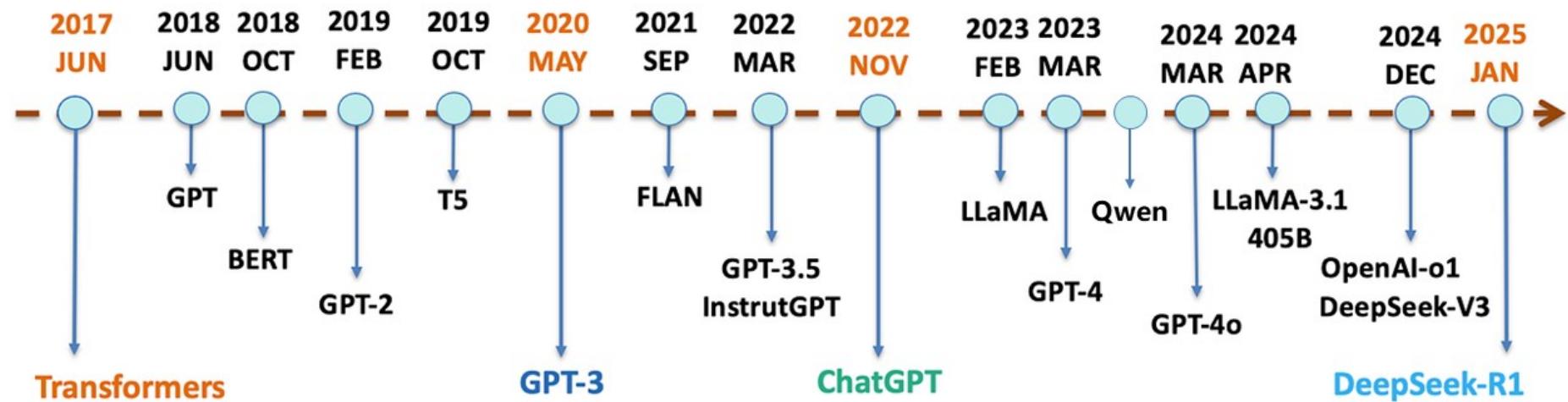
生成一张照片或插画，展现你日常生活中一个的“精彩瞬间”。比如你在明远湖畔的放松时刻，在图书馆的效率时刻，在运动场的活力时刻.....

- 可自由发挥，图片画风不限。
- 平台不限，例如豆包，注意关注图片的物理真实性/场景合理性/细节丰富性/整体氛围等等

需要提交：prompt以及生成的图片

大模型发展

思维链（CoT）极大解锁了模型的复杂推理潜力；与此同时，OpenAI、Meta、通义千问、深度求索等多个公司的大模型也在架构、规模和效率上实现了飞速发展，为高级推理的实现提供了坚实的算力和技术基础。



通用大模型与推理大模型

CoT 运用到大模型过程中产生了很多高质量、带有逻辑路径的数据，我们是否可以把这些数据进一步用到模型的训练中，来内化和增强模型的推理能力呢？

	通用大模型	推理大模型	← 推理增强
GPT系列	<ul style="list-style-type: none">• GPT-3.5/4/5系列• GPT-4o系列	<ul style="list-style-type: none">• o1系列• o3-mini系列	
Deepseek系列	<ul style="list-style-type: none">• DeepSeek-V3	<ul style="list-style-type: none">• DeepSeek-R1	
..... -thinking	

GPT-o1

三种思维链模式都是作用于模型**推理侧**，OpenAI尝试在**训练和推理时深度融合思维链技术**，并提出了GPT-o1。它包括o1, o1-preview和o1-mini三个版本，在回答问题前会**花费更多的时间来思考**，擅长处理**科学、编程、数学**等领域的复杂问题。



GPT-o1 vs. GPT-4o

GPT-o1在通用能力上与GPT-4o持平，推理能力和解决复杂问题方面更胜一筹。值得注意的是，GPT-o1 mini推理能力也显著高于GPT-4o。GPT-o1的推理运行开销显著高于GPT-4o。

资源开销分布：



特性对比：

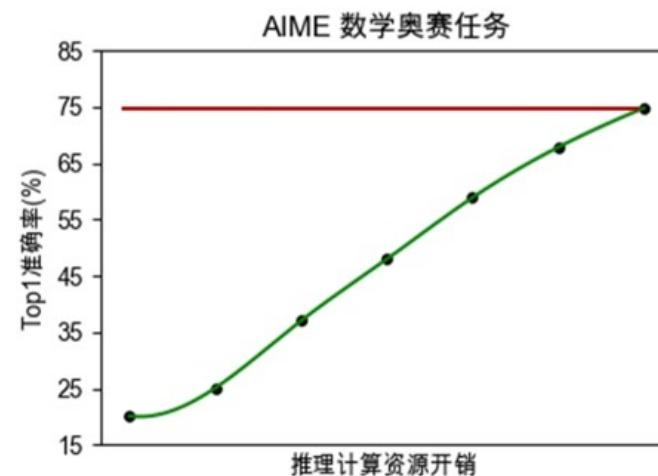
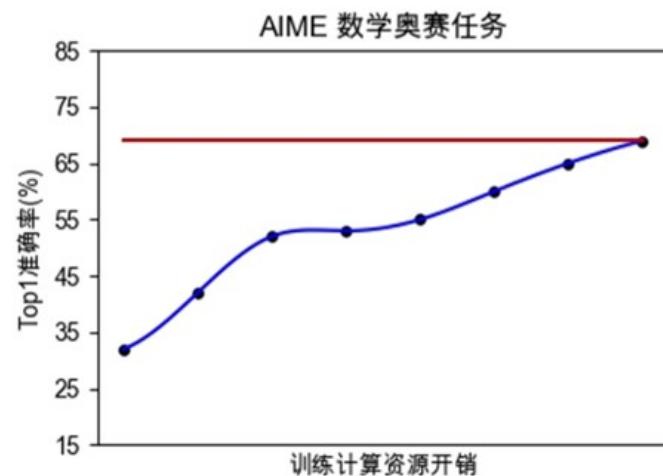
常识能力 GPT-o1 ≈ GPT-4o > GPT-o1 mini

推理能力 GPT-o1 > GPT-o1 mini > GPT-4o

运行开销 GPT-o1 > GPT-o1 mini > GPT-4o

GPT-o1引领新Scaling Law

OpenAI团队利用大规模强化学习算法教会模型如何利用其思维链进行高效思考。并发现o1的性能随着更多的强化学习（训练时的计算）和更多的思考时间（测试时的计算）而不断的提高。

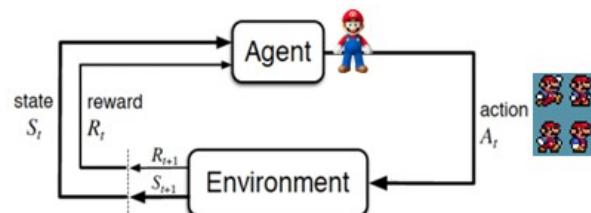


GPT-o1方案复现(猜测)

OpenAI未介绍GPT-o1技术细节。GPT-o1性能提升的关键在于从“**结果监督**”转为“**过程监督**”。训练时，利用**大规模强化学习**增强模型生成优质思维片段能力；测试时，利用**大规模搜索采样**可能的思维片段，并利用**奖励模型**指导生成。

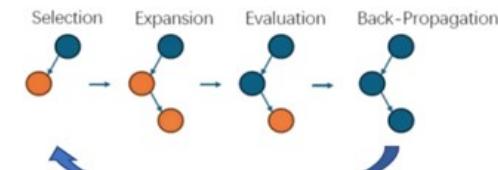
大规模强化学习：

通过强化学习，优化模型策略，鼓励自我纠正行为，从而提升模型在推理任务中的性能。



大规模搜索采样：

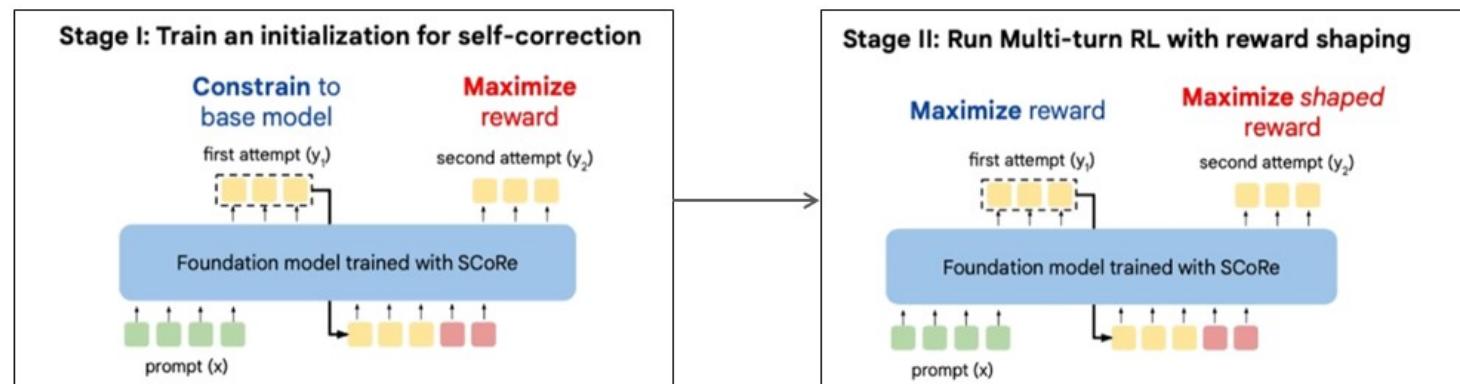
在测试阶段利用大规模搜索，通过**搜索多种解决方案、优化答案选择**，以增强了模型在复杂推理任务中的能力。



训练时大规模强化学习

谷歌提出的SCoRe[1]设计两阶段学习方法，训练语言模型学会自我反思与纠正的方法，通过增大训练时计算，提升模型的思维片段自我纠正能力以及长推理能力。

- 阶段1：针对模型第一次差强人意的回答，进行微调，使其**第二次**回答得更好。（培养纠错意识）
- 阶段2：设计面向自我纠正的**多轮强化学习策略**优化模型，进一步增强模型能力。（培养纠错能力）

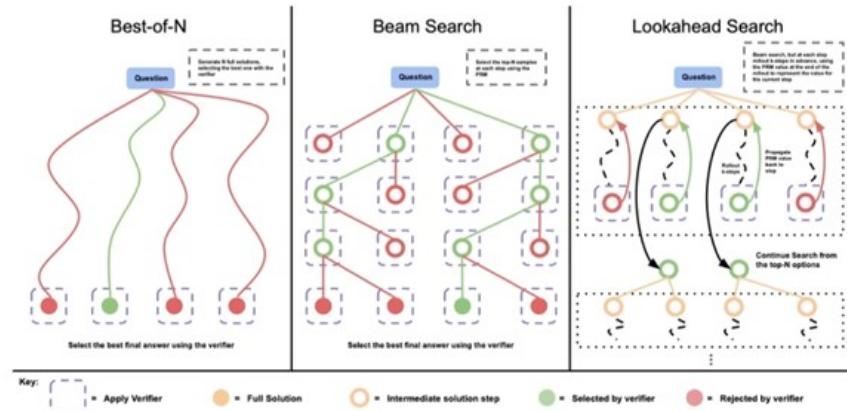


[1] Google DeepMind. "Training Language Models to Self-Correct via Reinforcement Learning". In:Arxiv. 2024.

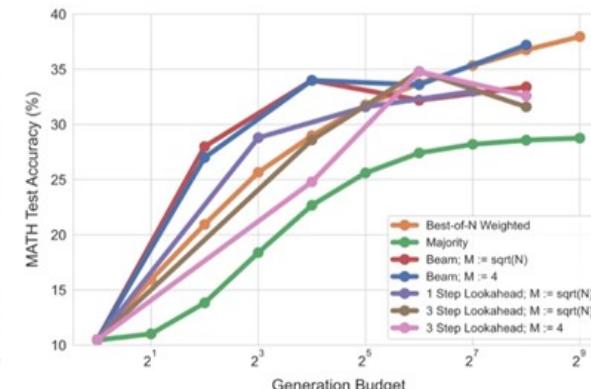
69

测试时大规模搜索

谷歌另一个工作又提出在模型测试时，利用**奖励模型指导大规模搜索**，探索多种思维片段，从而增大测试时计算，显著增强模型复杂推理问题的能力[1]。



多种奖励模型指导的搜索算法对比



不同搜索算法在数学榜单性能对比

[1] Google DeepMind. "Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters". In:Arxiv. 2024.

DeepSeek-V3 与 DeepSeek-R1

尽管两者均基于混合专家（MoE）架构，但它们在设计目标、训练方法、性能表现和适用场景上存在显著差异。

	DeepSeek-V3 深度思考	DeepSeek-R1 深度思考
设计目标	通用型AI助手	高级推理专家
训练方法	<ul style="list-style-type: none">训练阶段：预训练 + SFT + RL + 知识蒸馏（从R1提取推理能力）奖励机制：依赖传统监督数据与部分人类反馈	<ul style="list-style-type: none">训练阶段：纯RL+ 冷启动数据微调 + 多阶段RL对齐人类偏好奖励机制：基于规则奖励（准确性、格式奖励）和语言一致性奖励
性能表现	通用任务能力更优	推理与数学能力更优
适用场景	自然语言处理、知识问答、创意文案生成	数学建模、代码生成、复杂逻辑推理任务

DeepSeek-R1的aha moment

R1不仅推理能力大大增强，还出现了“智能”。

➤ 为什么aha moment现象很有意义？

Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a - \sqrt{a + x}} = x$ is equal to

Response: <think>

To solve the equation $\sqrt{a - \sqrt{a + x}} = x$, let's start by squaring both ...

$$(\sqrt{a - \sqrt{a + x}})^2 = x^2 \implies a - \sqrt{a + x} = x^2.$$

Rearrange to isolate the inner square root term:

$$(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$$

...

Wait, wait. Wait. That's an aha moment I can flag here.

Let's reevaluate this step-by-step to identify if the correct sum can be ...

We started with the equation:

$$\sqrt{a - \sqrt{a + x}} = x$$

First, let's square both sides:

$$a - \sqrt{a + x} = x^2 \implies \sqrt{a + x} = a - x^2$$

Next, I could square both sides again, treating the equation: ...

...

- 模型表现出了拟人化的思维方式
- 模型能够意识到需要重新思考并主动重启解题过程
- 这种行为是通过强化学习自然产生的，而不是预先设定的