



中国科学院大学

University of Chinese Academy of Sciences

自然语言处理

第10讲 推理(*reasoning*)模型

王石 资康莉 刘瑜

2026年春季课程

<https://ictkc.github.io/teaching/>



第十讲

推理(reasoning)模型



目 录

1

背景与基本概念

2

3

4

什么是推理模型

一个班有 30 人，男生比女生多 6 人，男生和女生各多少人？

男生18人，女生12人

设女生 x 人，男生 $x+6$ 人，总人数 $x+x+6=30$ ，
 $x=12$ ，
所以男生18人，女生12人

什么是推理模型

- 会分解问题
- 会生成中间步骤
- 会检查或修正答案
- 会在推理阶段消耗更多计算量
- 更适合数学、代码、逻辑和复杂决策任务

为什么需要推理模型



oyfdnisd rrtqwainr acxz mynzbhbx → Think step by step

Use the example above to decode:

oyekaijzdf aaptcg suaokybhai ouow aqht
mynznvaatzacdfoulxxz

GPT-4o

A possible division could be:

- oyekaijzdf
- aaptcg
- suaokybhai
- ouow
- aqht
- mynznvaatzacdfoulxxz

However, to make a meaningful decoding, I would need more context about the transformations or letter shifting that might be involved.

Could you provide any additional decoding rules or transformations used in this cipher?

LLM在复杂的推理任务上仍然频繁出错

什么场景需要推理

- 数学
- 逻辑
- 代码
- 科学问答
- 推理决策





为什么LLM在上述场景表现差

- 多步骤
- 强约束
- 中间错误会传播
- 需要验证
- 答案不一定能直接记忆





传统推理模型 VS 大语言推理模型


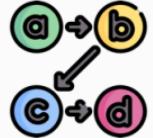
Traditional Reasoning Models

Rules, Logical Deduction  Deterministic Rigid 


Rule-based & Symbolic Logic

Errors from Human Experts  Explicitly Encoded Knowledge 



Manual Knowledge Engineering



Structured Frameworks  Following Logical Premises 



Linear & Deductive Reasoning



Finance or Medical or Law 

Small Scale & Domain-Specific



Training Approach  

Adaptability & Learning  

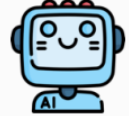

Problem-Solving Strategy  

Generality & Scalability  



Reasoning LLMs

Large-scale Corpus  Probabilistic Reasoning 


Data-driven & Probabilistic Reasoning

Learn Policy Take Action  PPO / DPO / GRPO  Feedbacks from Environment

Self-Improvement & Adaptive Reasoning

Tree Search like MCTS, ToT  Self-correct Mistakes 

Exploratory & Multi-path Reasoning

Math & Code & Chat 

Scalable & Generalize across Tasks

大语言模型 VS 推理模型

模型	普通 LLM	推理模型
输出方式	直接生成答案	先思考再回答
计算方式	单次前向生成	更长推理过程
优势任务	写作、摘要、问答	数学、代码、逻辑、规划
主要问题	容易猜答案	成本更高、速度更慢
能力来源	预训练和指令微调	SFT、RL、验证器、推理时计算

为什么推理模型突然重要

- 预训练规模继续扩大，边际收益变小
- 复杂任务需要过程能力，而不是只靠知识记忆
- 数学、代码、科学任务成为模型能力竞争重点
- 推理时计算成为新的能力提升方向



目 录

1

背景与基本概念

2

核心方法与技术路线

3

4

四条主线

- ❑ Prompt based: 提示词引导
- ❑ Search based: 生成多条候选路径
- ❑ Verification based: 通过结果验证或过程验证
- ❑ Tool augmented: 调用外部工具

Prompt based 基本思想

标准 Prompt (直接回答)

CoT Prompt (思维链)

问题：一家店原有23个苹果，卖出20个，又进货6个，现在有多少个？
(同一个问题，两种 Prompt 策略)

模型内部 (黑箱)

直接从问题→答案
推理步骤被压缩，容易出错

答案：9个 X

(直接跳到答案，算错了)

"让我们一步步思考："

步骤1：原有 23 个

步骤2：卖出 20 个，剩 $23-20 = 3$ 个

步骤3：进货 6 个，共 $3+6 = 9$ 个

等等，让我重新算： $23-20+6 = 9$ 个 ✓

答案：9个 ✓ (正确!)

Zero shot prompt

标准 Prompt (直接回答)

CoT Prompt (思维链)

问题：一家店原有23个苹果，卖出20个，又进货6个，现在有多少个？
(同一个问题，两种 Prompt 策略)

模型内部 (黑箱)

直接从问题→答案
推理步骤被压缩，容易出错

答案：9个 X

(直接跳到答案，算错了)

"让我们一步步思考："

步骤1：原有 23 个

步骤2：卖出 20 个，剩 $23-20 = 3$ 个

步骤3：进货 6 个，共 $3+6 = 9$ 个

等等，让我重新算： $23-20+6 = 9$ 个 ✓

答案：9个 ✓ (正确!)

Few shot prompt

标准 Prompt (直接回答)

Few-shot Prompt (举例引导)

问题：一个长方形的周长为40米，长比宽多4米，求长。
(同一个问题，两种 Prompt 策略)

模型内部 (黑箱)

直接从问题→答案
推理步骤被压缩，容易出错

答案：16 米 x

(直接跳到答案，判断错误)

让模型参考示例，学习解题模式：

exg1:

一根绳子被分成两段，长的一段比短的多6米，总长为30米，
求长的一段。
设长的一段为 x ,

答案：18 米

exg2:

某商品打折后价格为80元，比原价便宜20元，求原价。
设原价为 x ,

答案：100 元

现在问题:

一个长方形的周长为40米，长比宽多4米，求长。
设长为 x ,

答案：12 米 ✓ (正确!)

CoT为什么有效

- 问题分解
- 中间步骤
- 隐式 \rightarrow 显式推理
- 推理稳定性提升
- 多步依赖建模

CoT到推理模型

□ CoT 的局限

可能只是模仿格式

推理步骤可能看起来合理但实际错误

不能保证最终答案正确

需要和 RL、Search、Verifier 结合

Prompt based优缺点

□ 优点

低成本

灵活性强

可解释性好

通用性强

□ 缺点

不稳定

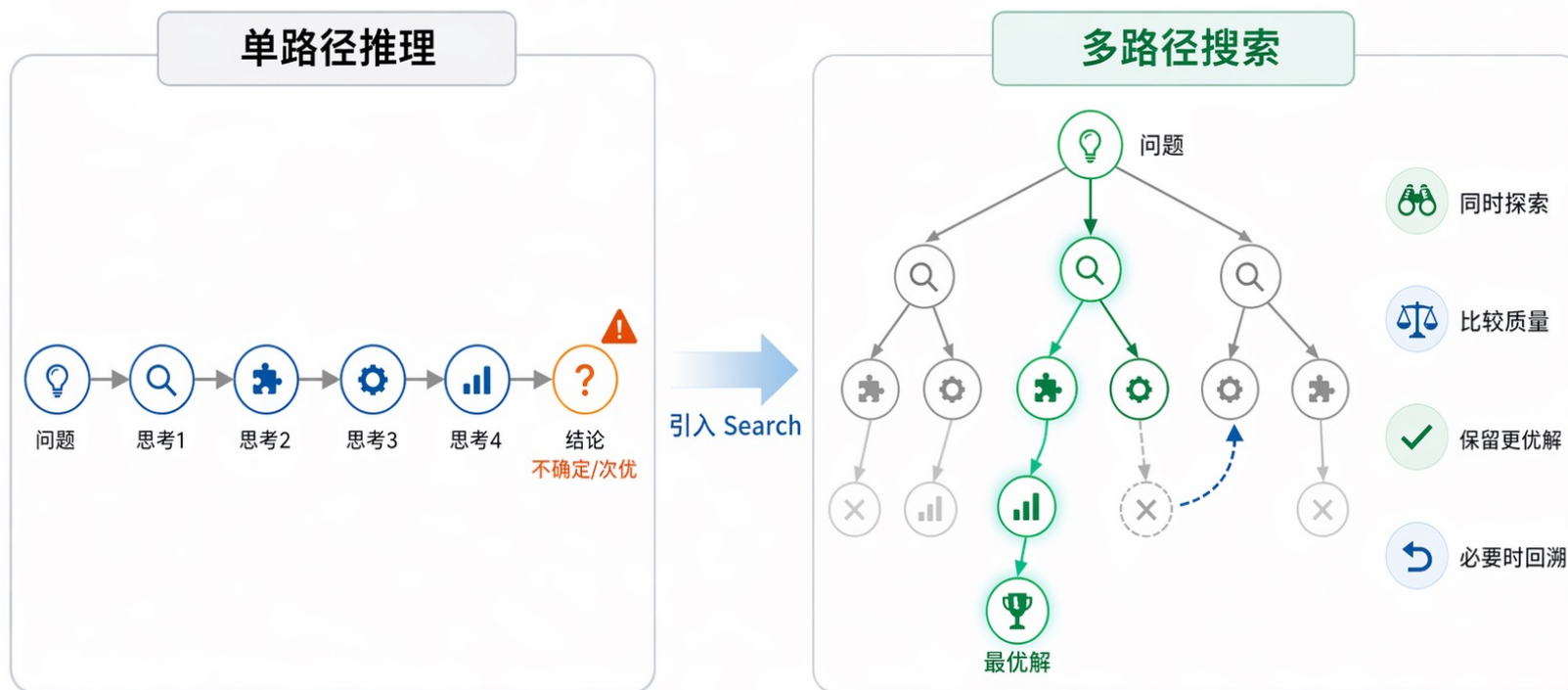
上下文依赖强

难优化

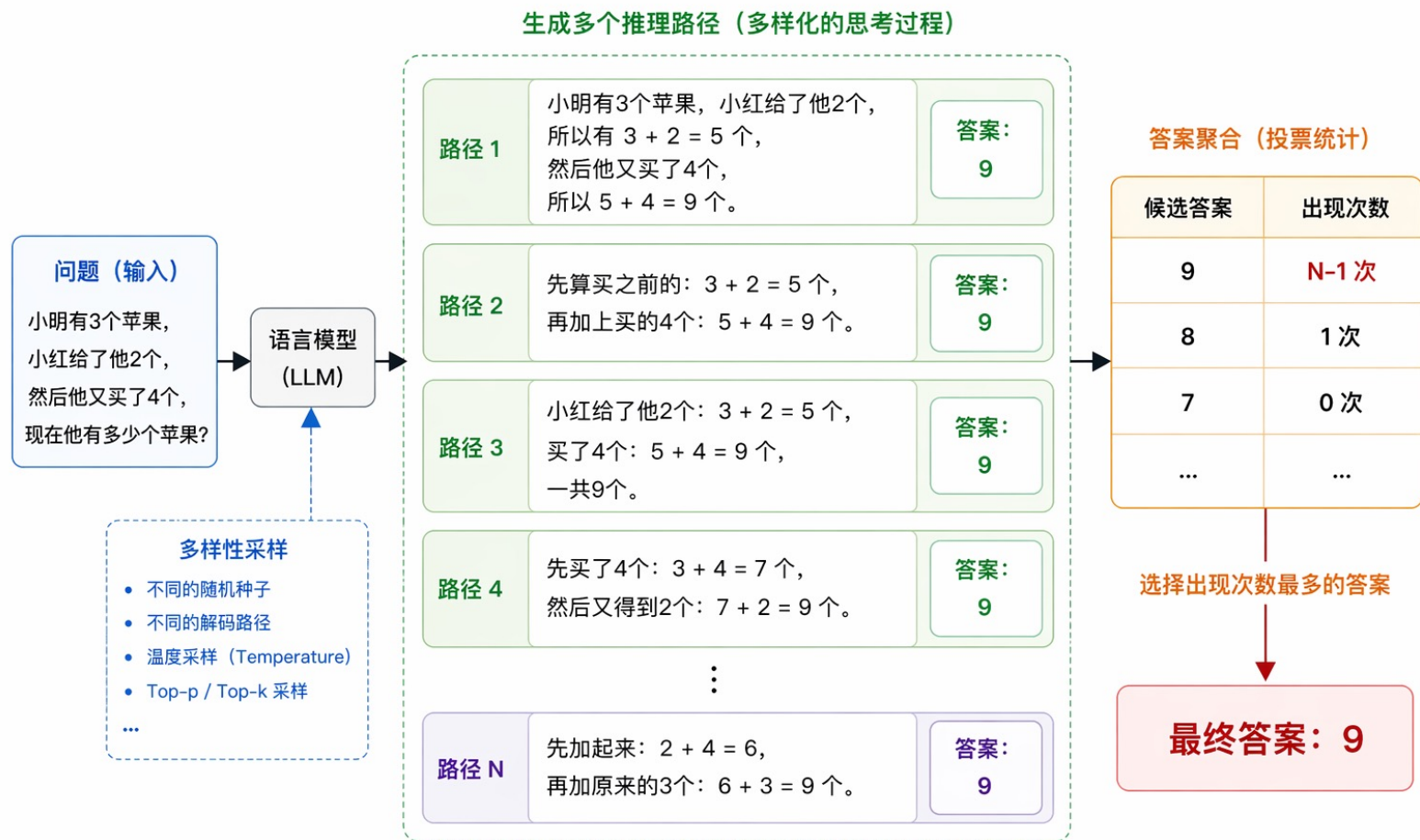
性能上限受限

Search based

□ 推理不再是一条路径，而是一个探索过程



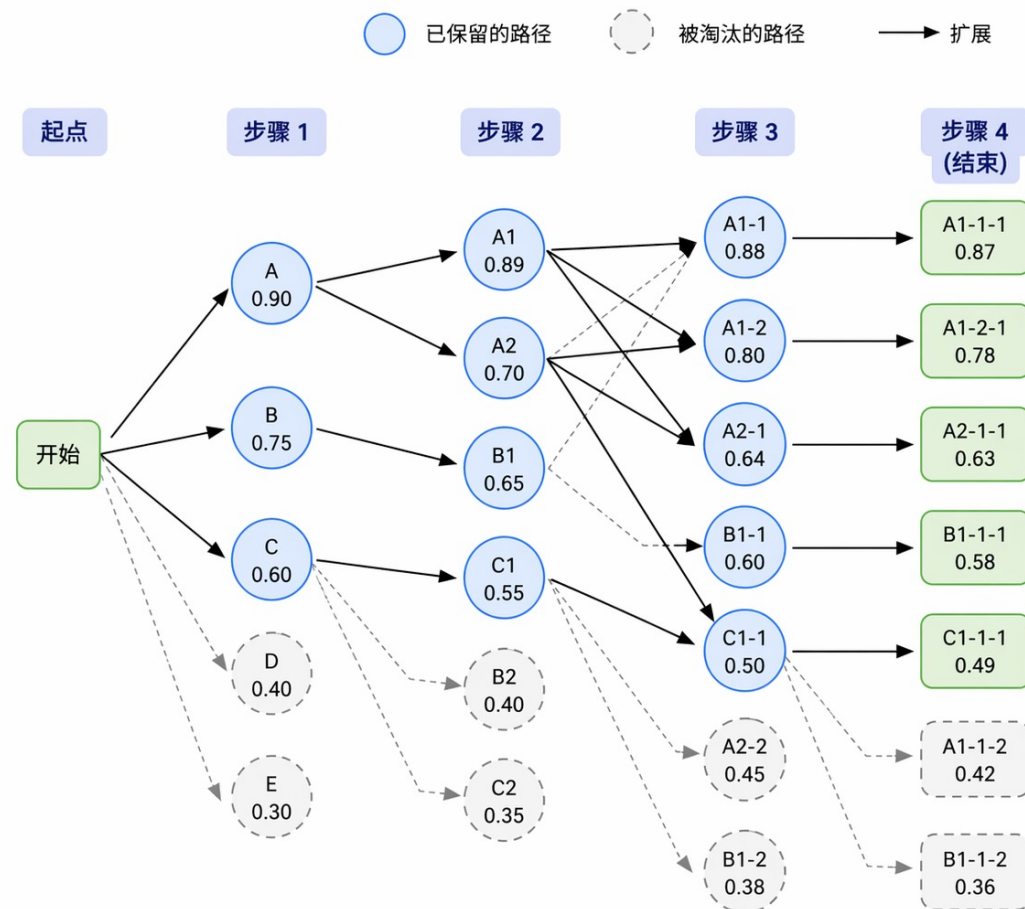
Best of N: 最简单的search



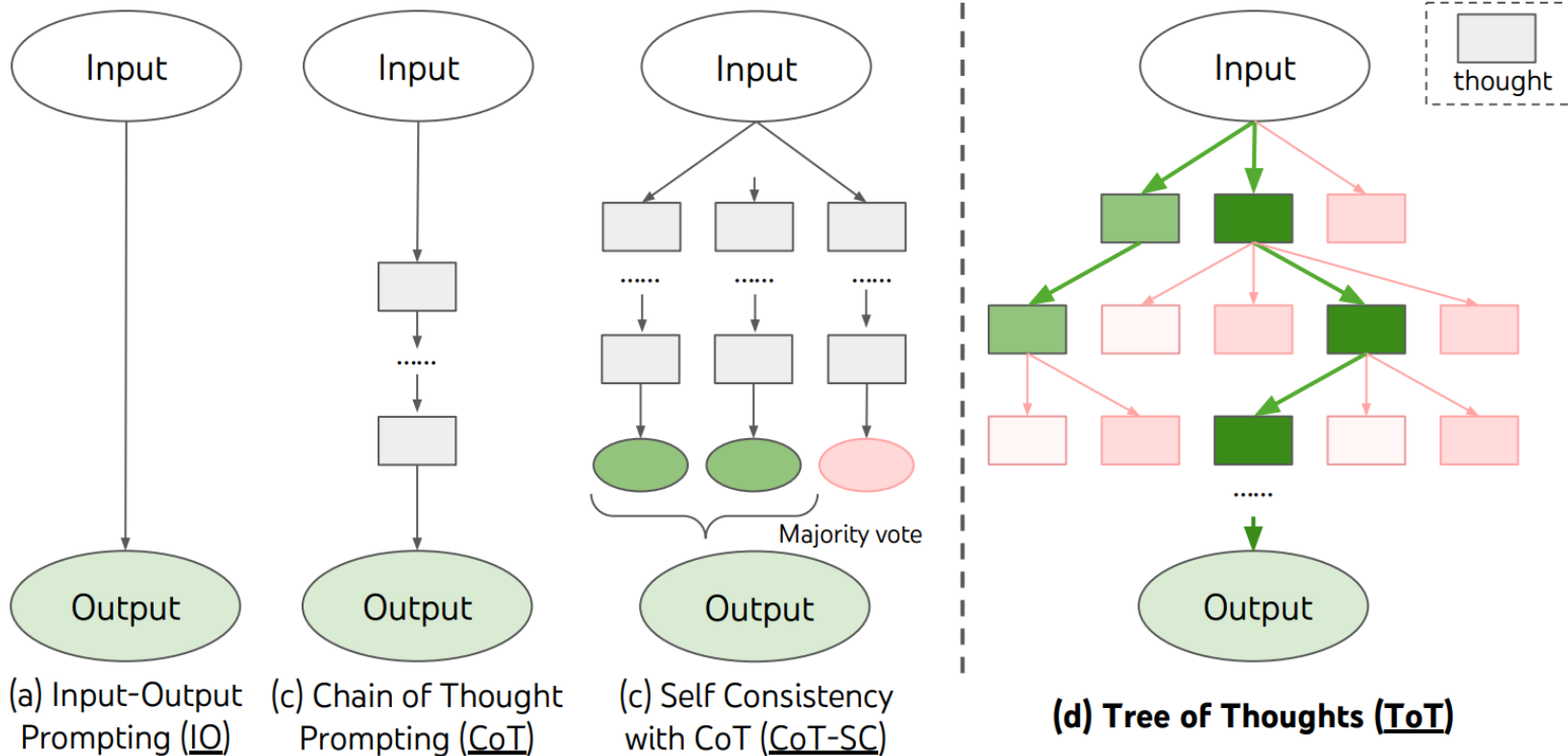
Beam Search与路径保留

□ 同时保留多个高分路径

□ 持续筛选更优候选



Tree of Thoughts: 结构化推理搜索



Search based的局限

- 计算成本高
- 候选路径质量不稳定
- 搜索空间容易爆炸
- 需要额外评分或验证机制
- 不适合所有开放式任务

Outcome vs Process Verification

□ Outcome

只检查最终答案

简单高效

适用于可验证任务

□ Process

检查中间推理步骤

更细粒度

更适合复杂推理


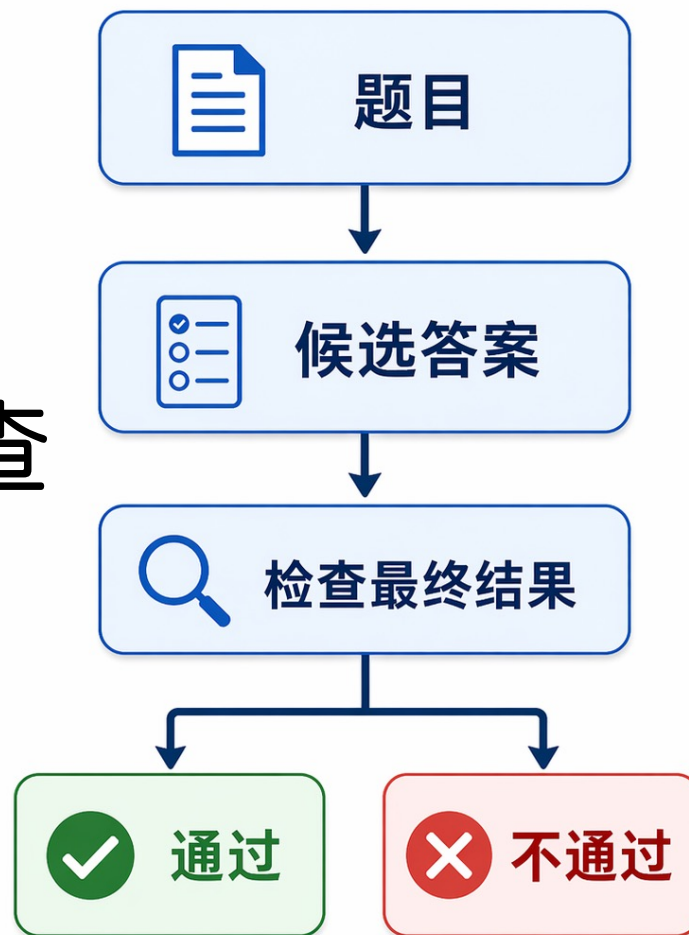
Outcome Verifier: 结果级验证

□ 要求

最终答案是否正确

是否满足题目要求

是否能通过外部规则检查





+


-


×

÷

 题目: $12 \times 8 = ?$

 候选答案: 96

 验证: $12 \times 8 = 96$

 结果: 通过

Process Verifier: 过程级验证

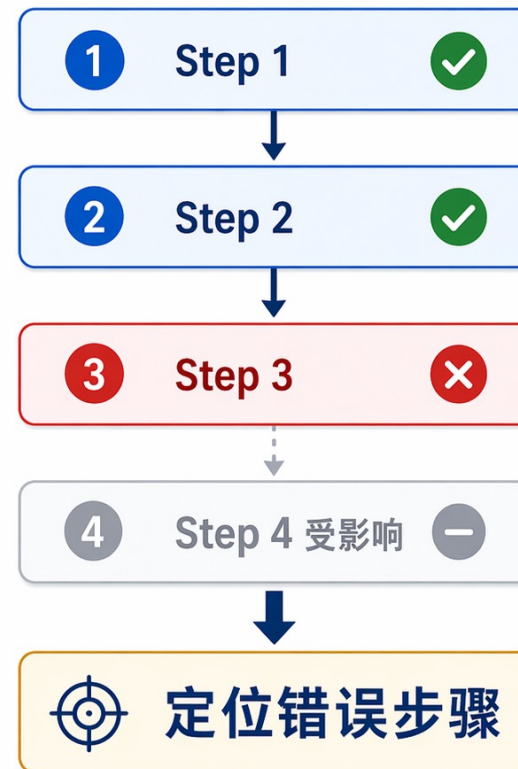
□ 要求

每一步是否合理

推理链是否连贯

是否存在逻辑跳跃

错误从哪一步开始出现



Process Verifier 检查内容



每一步是否合理



推理链是否连贯



是否存在逻辑跳跃



错误从哪一步开始出现

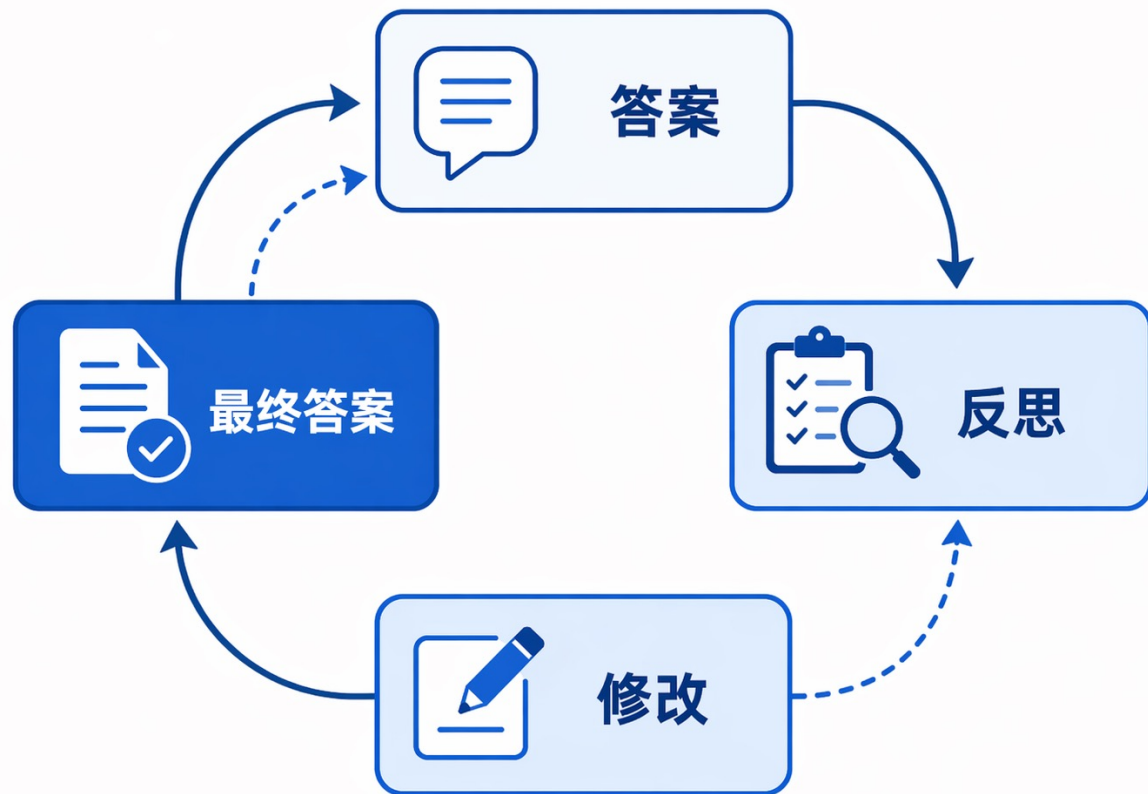
Outcome vs Process Verification

方法	Outcome Verification	Process Verification
检查对象	最终答案	中间步骤和最终答案
成本	较低	较高
粒度	粗	细
优点	简单高效	能定位错误
缺点	不知道错在哪里	验证成本高
适用任务	数学、代码、选择题	多步推理、复杂分析

Self Critique: 自我反思机制

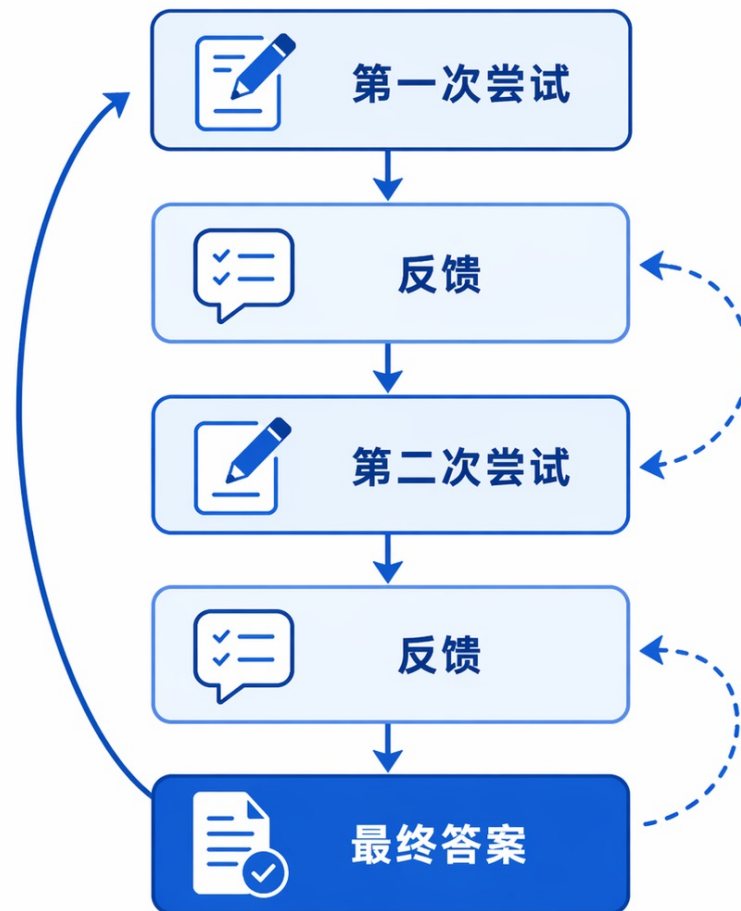
□ 基本流程

先生成一个答案
再检查答案中的问题
找出可能错误
重新修改答案



Reflection: 多轮推理与修正

□ 基本流程
第一次尝试
发现错误
调整策略
再次推理
得到更优答案



反思的关键要点



发现错误



调整策略



再次推理



得到更优答案

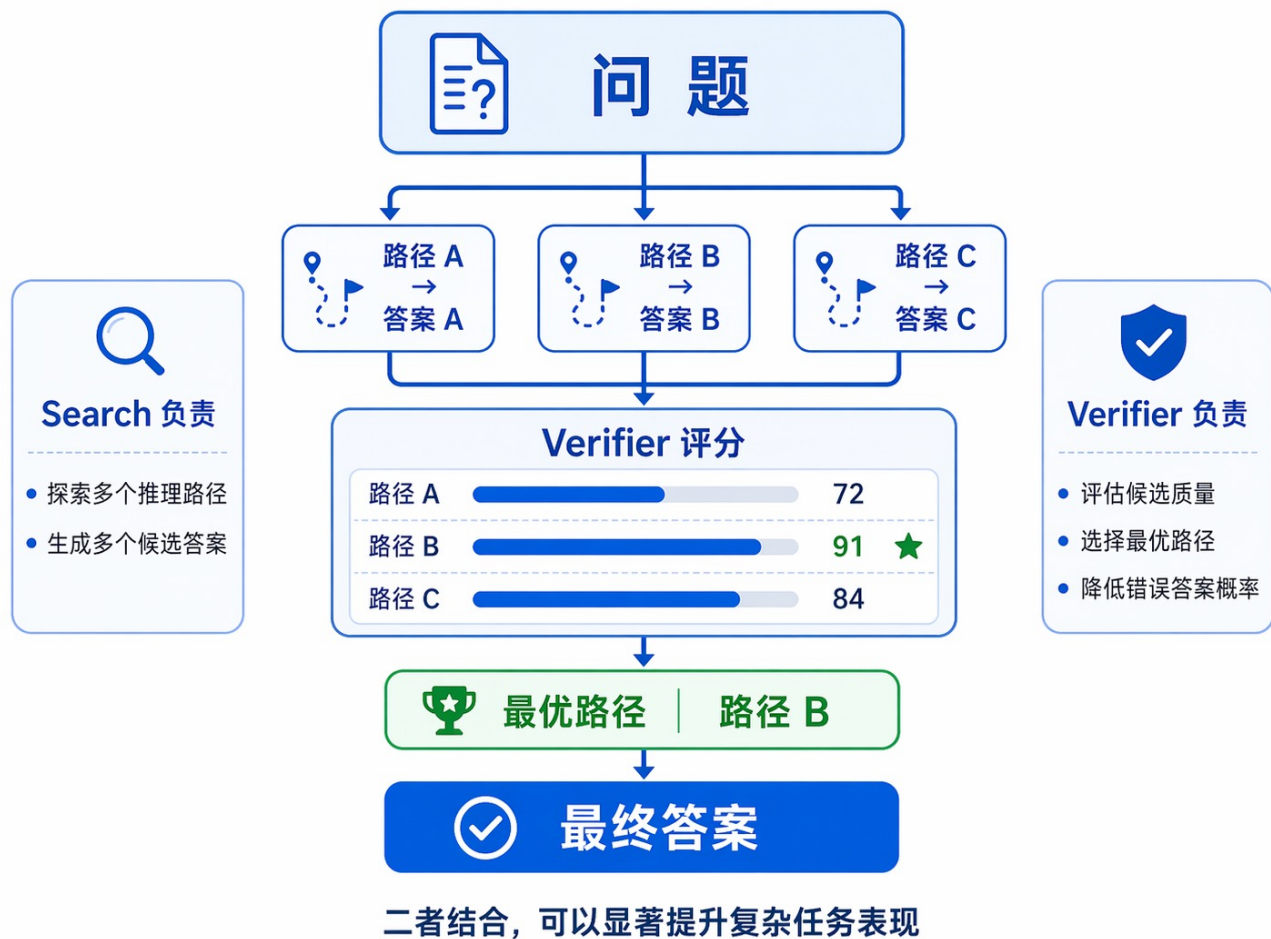
适合任务

- 复杂代码生成
- 长任务规划
- 多步数学问题
- 开放式分析任务

Search + Verifier

□ Search 负责：
探索多个推理路径
生成多个候选答案

□ Verifier 负责：
评估候选质量
选择最优路径
降低错误答案概率



Verification 的局限

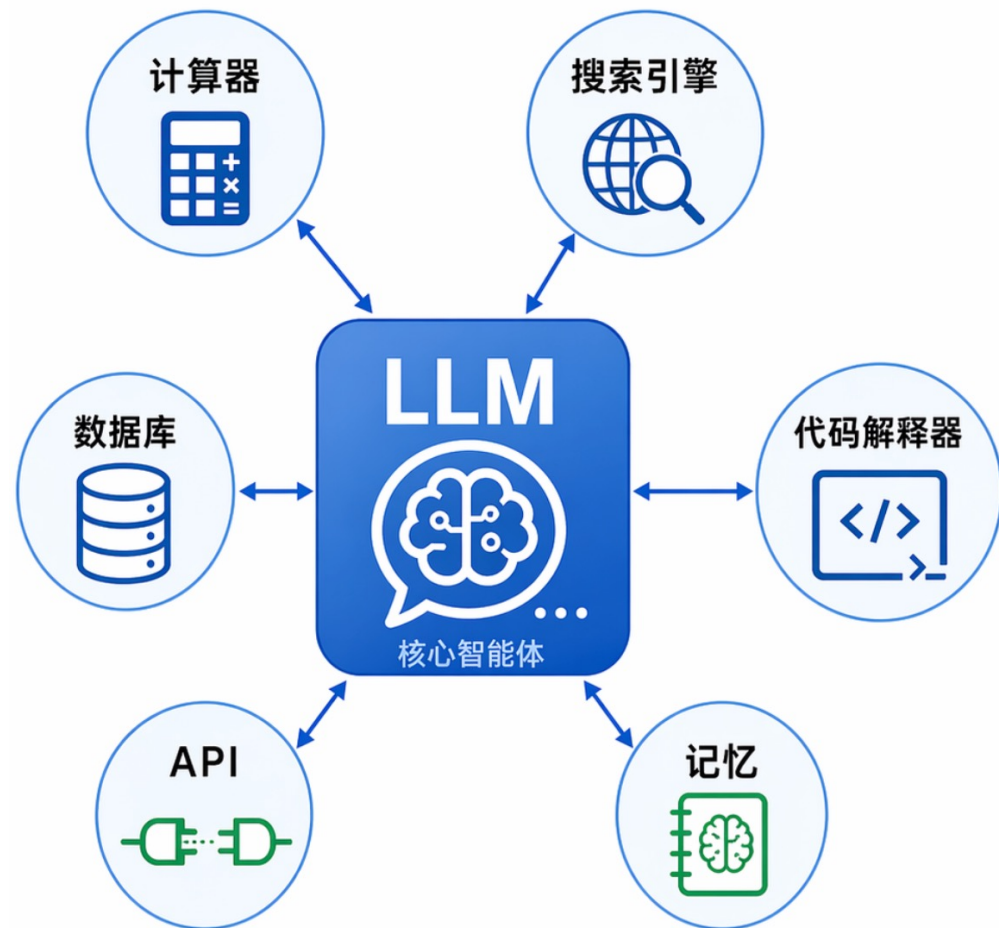
- ❑ verifier 本身可能出错
- ❑ process verification 成本高
- ❑ 对开放任务难以定义“正确”
- ❑ 可能引入额外延迟

Tool augmented

- LLM 只靠内部参数，存在明显限制：
 - 知识可能过时
 - 计算可能出错
 - 无法直接访问外部数据
 - 难以执行真实操作
 - 长任务需要外部记忆和工具支持

Tool augmented 的基本流程

□ 流程：
理解问题
判断是否需要工具
选择合适工具
调用工具获得结果
整合结果生成答案



常见工具类型

工具类型	作用	示例任务
搜索工具	获取最新信息	新闻、论文、价格
计算器	精确计算	数学、汇率、财务
代码执行器	运行程序	数据分析、调试代码
数据库	查询结构化数据	企业数据、知识库
API	执行真实操作	发邮件、订票、查日程
记忆工具	保存上下文	长期任务、个性化助手

代表框架

□ 代表框架：ReAct


Reasoning:


模型先思考下一步

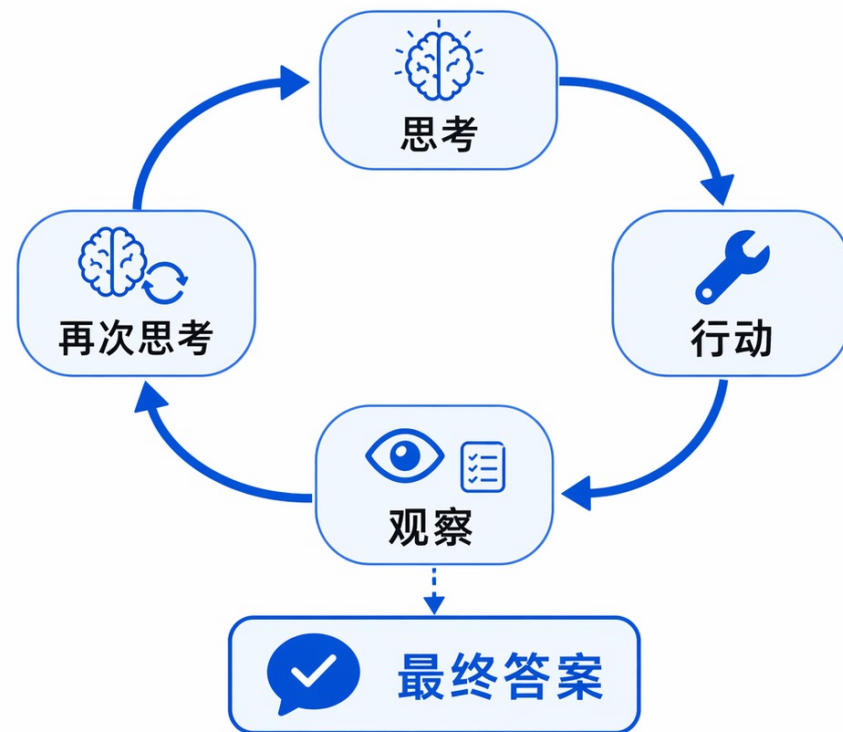
Acting:

模型调用工具执行动作

ReAct = 推理 + 行动

 推理：先思考下一步

 行动：调用工具执行动作



思考 → 行动 → 观察 → 思考 → ... → 最终答案

Tool Augmented的局限

- 知识可能过时
- 内部计算可能出错
- 无法直接访问外部信息
- 难以执行真实操作

方法对比

方法	核心思想	典型方式	优点	局限
Prompt based	通过提示词引导模型产生推理过程	Zero shot CoT、Few shot CoT、Step by step prompting	成本低、使用灵活、容易解释	不稳定，依赖提示词设计，能力上限有限
Search based	不只生成一条路径，而是探索多条推理路径	Best of N、Beam Search、Tree of Thoughts	能提升复杂任务表现，适合多路径问题	计算成本高，候选路径质量不一定好
Verification based	将生成和判断分离，用验证机制筛选答案	Outcome Verifier、Process Verifier、Self Critique、Reflection	能检查答案质量，适合数学、代码、逻辑任务	Verifier 本身可能出错，过程验证成本高
Tool augmented	让模型调用外部工具辅助推理和执行	搜索引擎、计算器、代码执行器、数据库、API、ReAct	能获取最新信息，提高计算精度，支持真实任务	工具选择可能错误，外部信息可能不可靠，系统更复杂



目 录

1

背景与基本概念

2

核心方法与技术路线

3

训练机制与能力提升

4

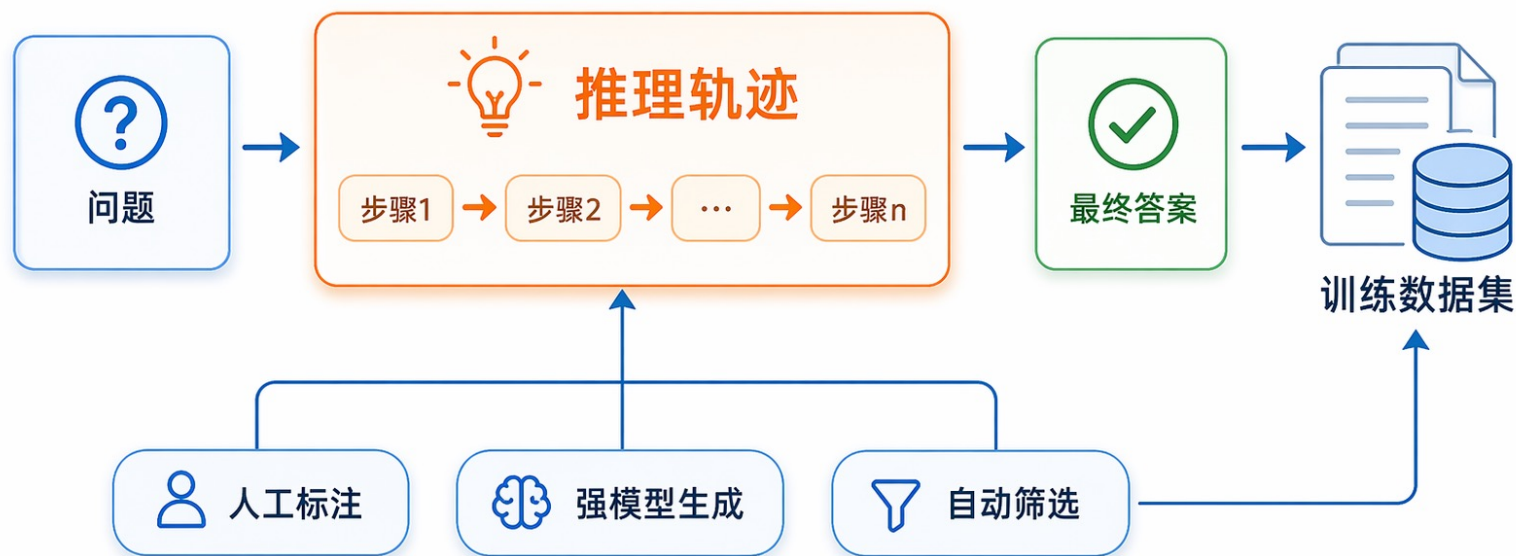
推理能力如何训练？

推理能力如何从无到有？
如何在推理阶段被放大？

模型为什么会推理？

- 推理数据
- 训练方法
- 推理能力的放大和稳定

Step1: 构建推理数据



□ 数据要求

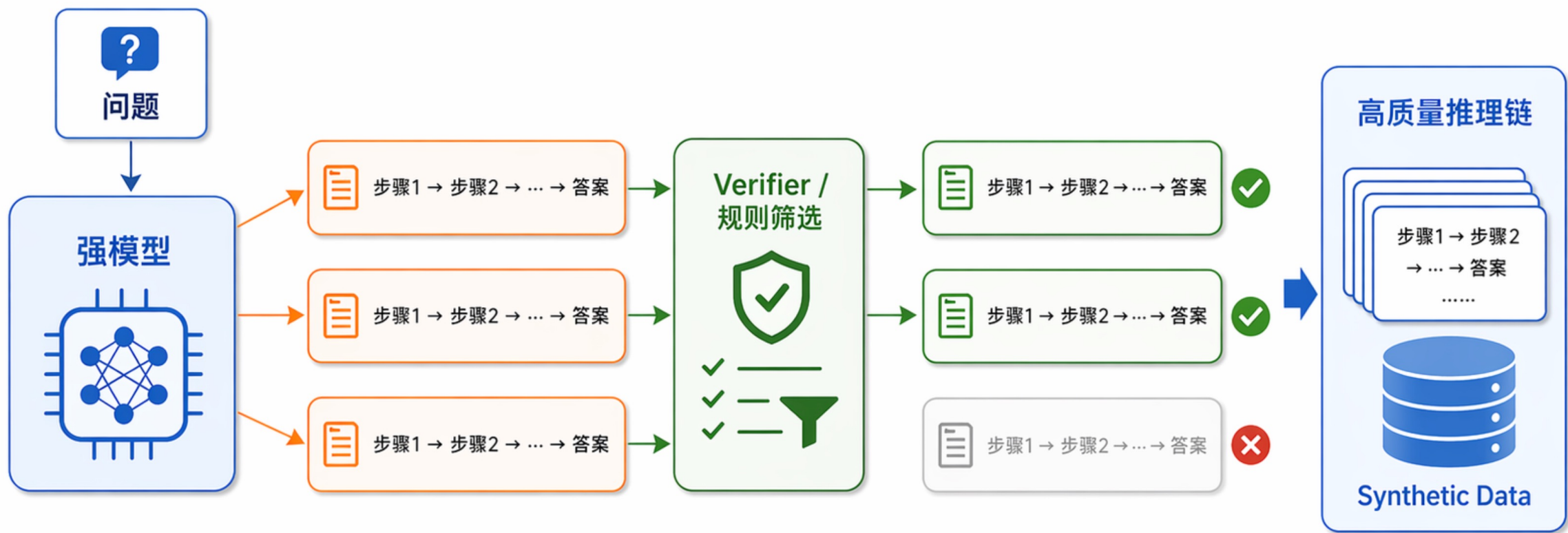
问题-推理步骤-答案

从答案数据到过程数据

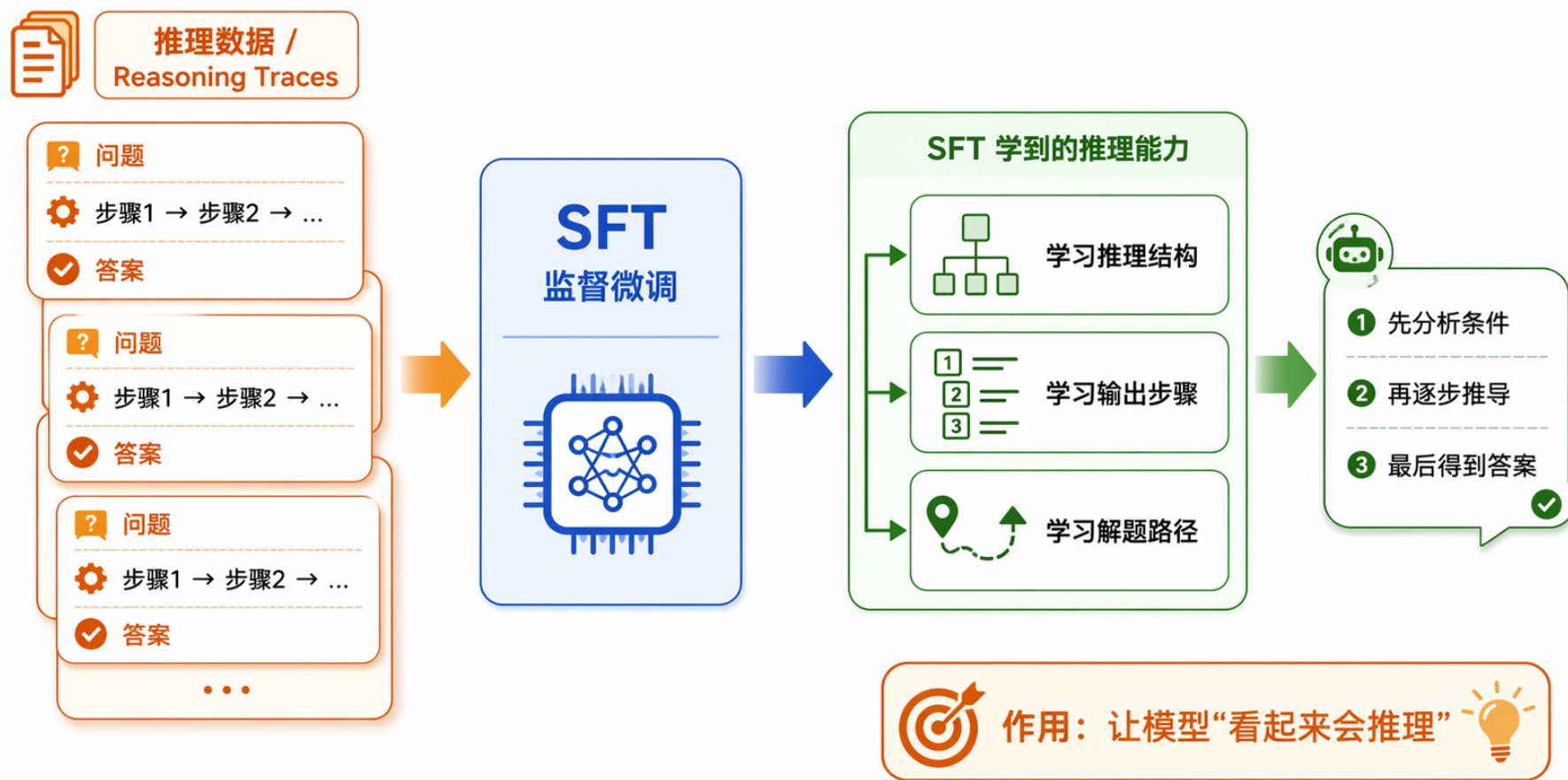
□ 传统数据：
问题 -> 答案

□ 推理数据：
问题 -> 思考步骤 -> 答案

合成数据：用模型教模型推理



Step2: SFT 学习推理模式

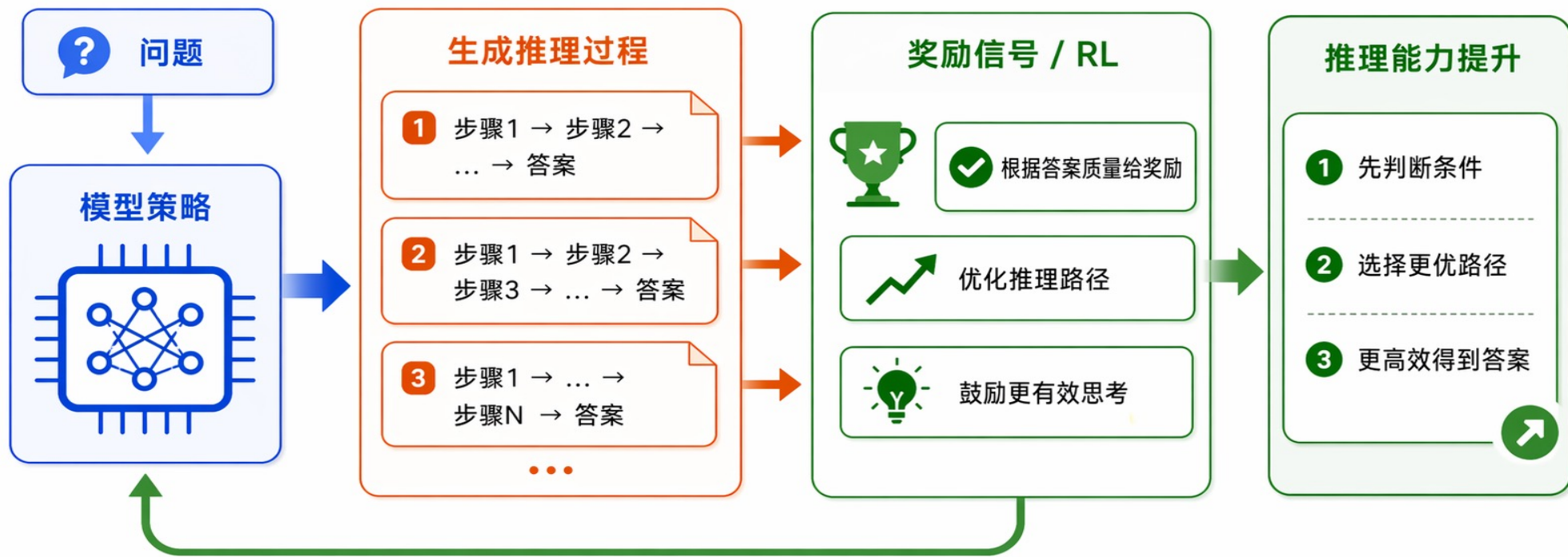


SFT 的问题

- 只是模仿，不是真正推理
- 容易学到格式
- 不一定理解逻辑
- 对复杂任务效果有限

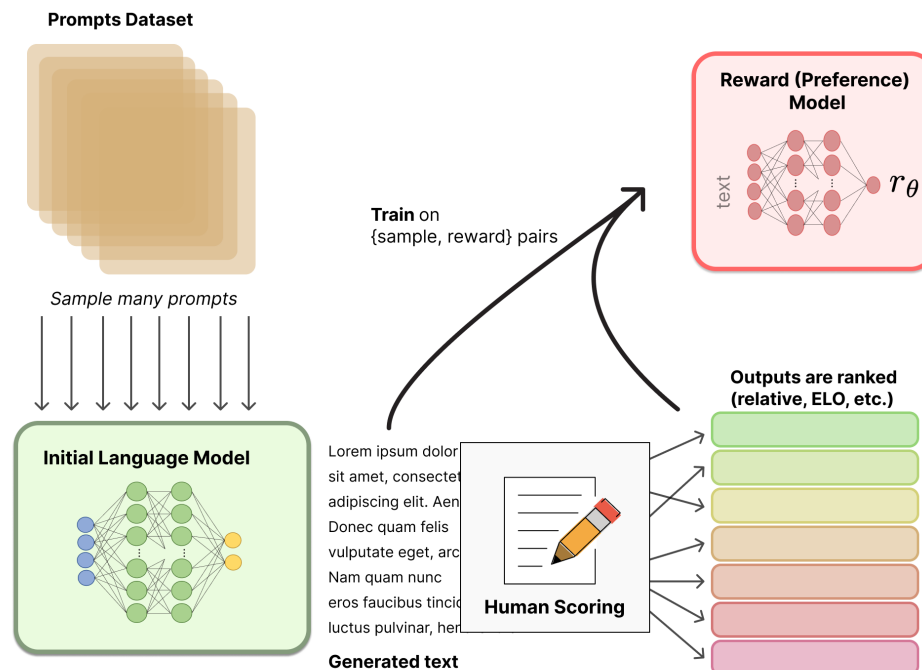
需要更强的训练机制！

Step3: 强化学习激发推理能力



RLHF：从模仿到优化

□ RLHF 通过反馈信号优化模型行为



推理能力可以被“涌现”

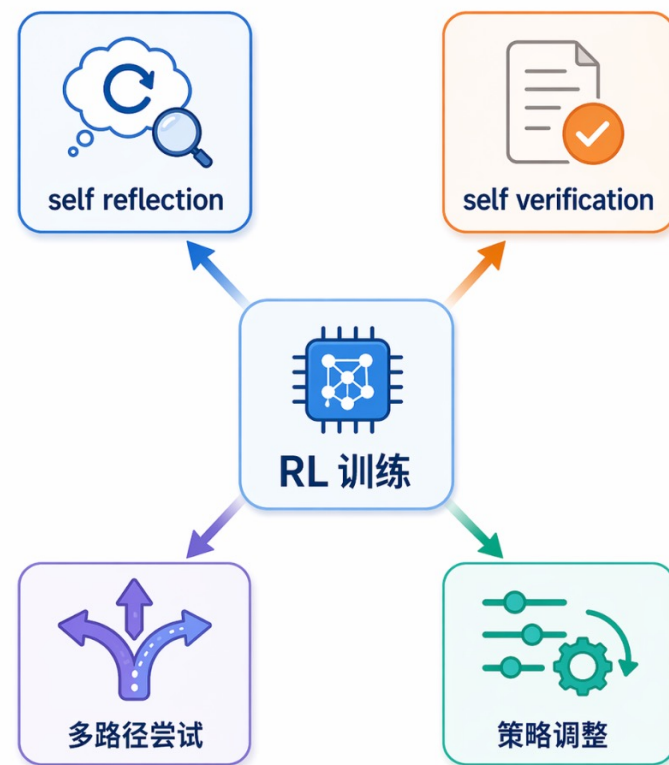
□ 在 RL 过程中，模型会自然出现：

self reflection

self verification

多路径尝试

策略调整



为什么 RL 能产生推理能力？

- 正确答案需要多步推理
- RL 奖励驱动模型寻找有效路径
- 错误路径被惩罚
- 正确路径被强化

Step 5: 推理增强

□ 推理能力不仅来自训练，还来自推理阶段：

更长推理链

多路径采样

search

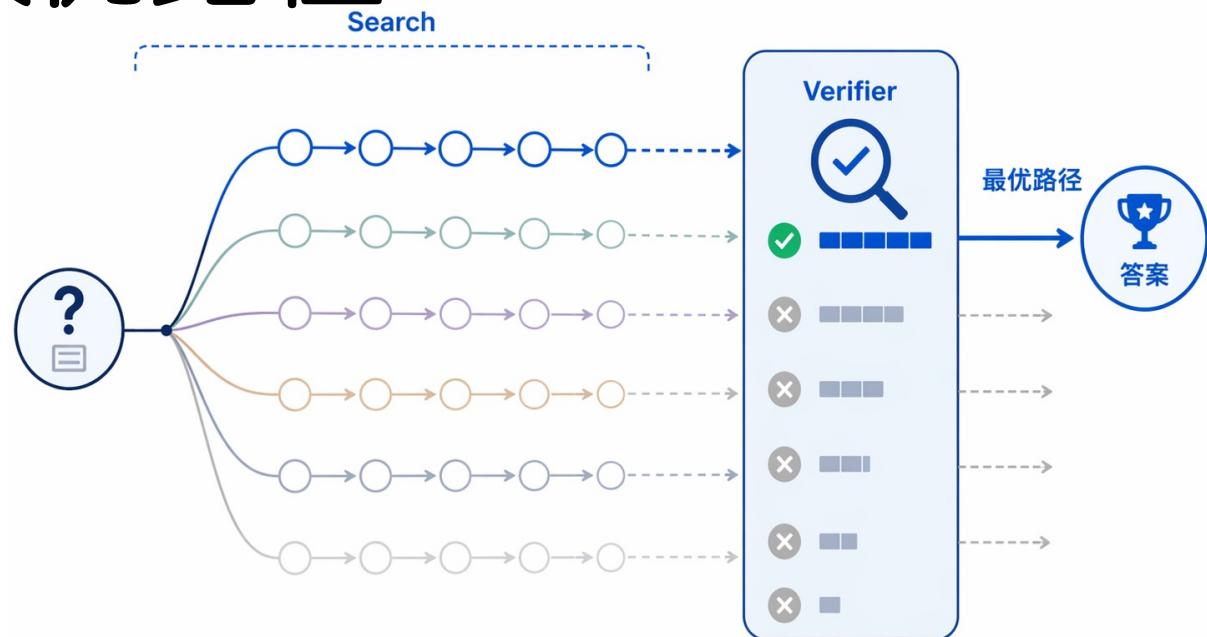
verifier

让模型 “多想一会儿”

- 增加 reasoning token
- 延长思考过程
- 提升复杂任务表现

Search + Verifier: 推理能力放大器

- search: 探索多个路径
- verifier: 选择最优路径



推理模型结构

□ 一个完整 reasoning model 包括：

推理数据

SFT

RL

推理阶段 search

verifier

评价指标

□ 常见任务:

数学推理: GSM8K、MATH、AIME

代码推理: HumanEval、Codeforces

科学问答: GPQA

通用知识与推理: MMLU

逻辑推理: FOLIO、LogiQA

评价指标

- 评价重点：
 - 最终答案是否正确
 - 推理过程是否合理
 - 复杂任务上是否稳定
 - 多轮修正后是否变好
 - 成本和速度是否可接受



目 录

1

背景与基本概念

2

核心方法与技术路线

3

训练机制与能力提升

4

代表模型

代表模型

□ ChatGPT O1



□ DeepSeek R1



□ Gemini Thinking



ChatGPT O1

□ 特点:

大规模强化学习训练

长链推理 (long CoT)

test time thinking

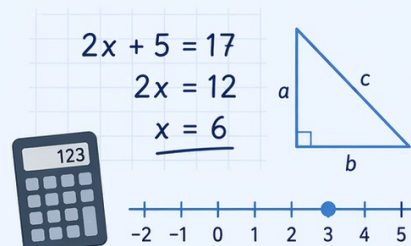
在数学、代码、科学任务表现强



ChatGPT 01

- 适合任务:
- 数学
- 代码
- 科学问答
- 复杂规划

数学



例：已知 $2x + 5 = 17$ ，求 x

代码



```

1 def is_palindrome(s):
2     s = s.replace(" ", "")
3     s = s.lower()
4     return s == s[::-1]
5
6 # 示例
7 print(is_palindrome("上海自来水来自海上"))
8

```

例：写一个 Python 函数判断字符串是否为回文

科学问答



例：为什么天空是蓝色的？

复杂规划



	第一天	第二天	第三天
确定目的地	城市探索	自然景区	文化体验
预订交通	上午/下午	全天	上午/下午
安排住宿			
景点与活动			
打包行李			

例：安排一次三天旅行的路线与时间

ChatGPT 01

□ 优点

强推理

复杂任务表现好

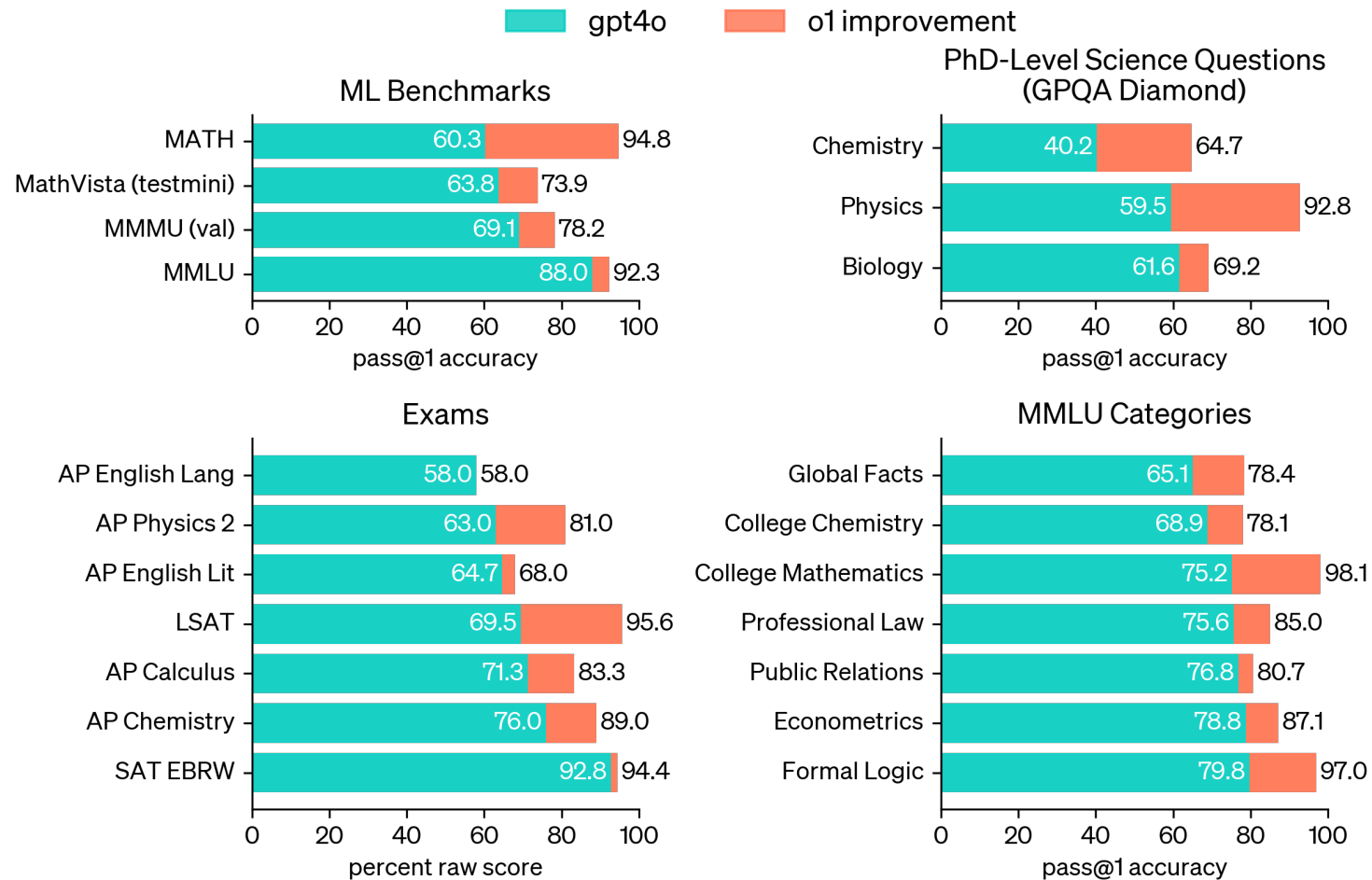
□ 缺点

推理速度慢

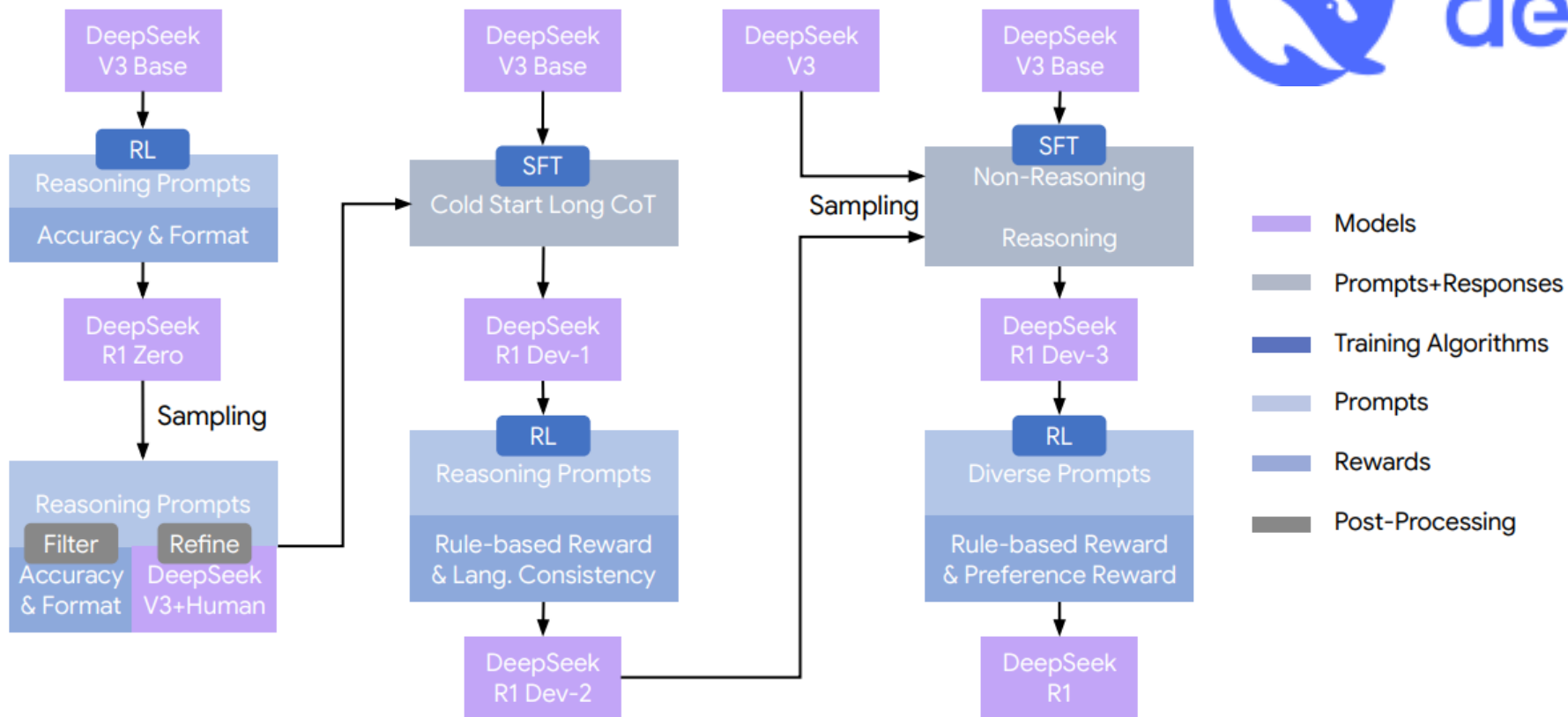
闭源

Api贵

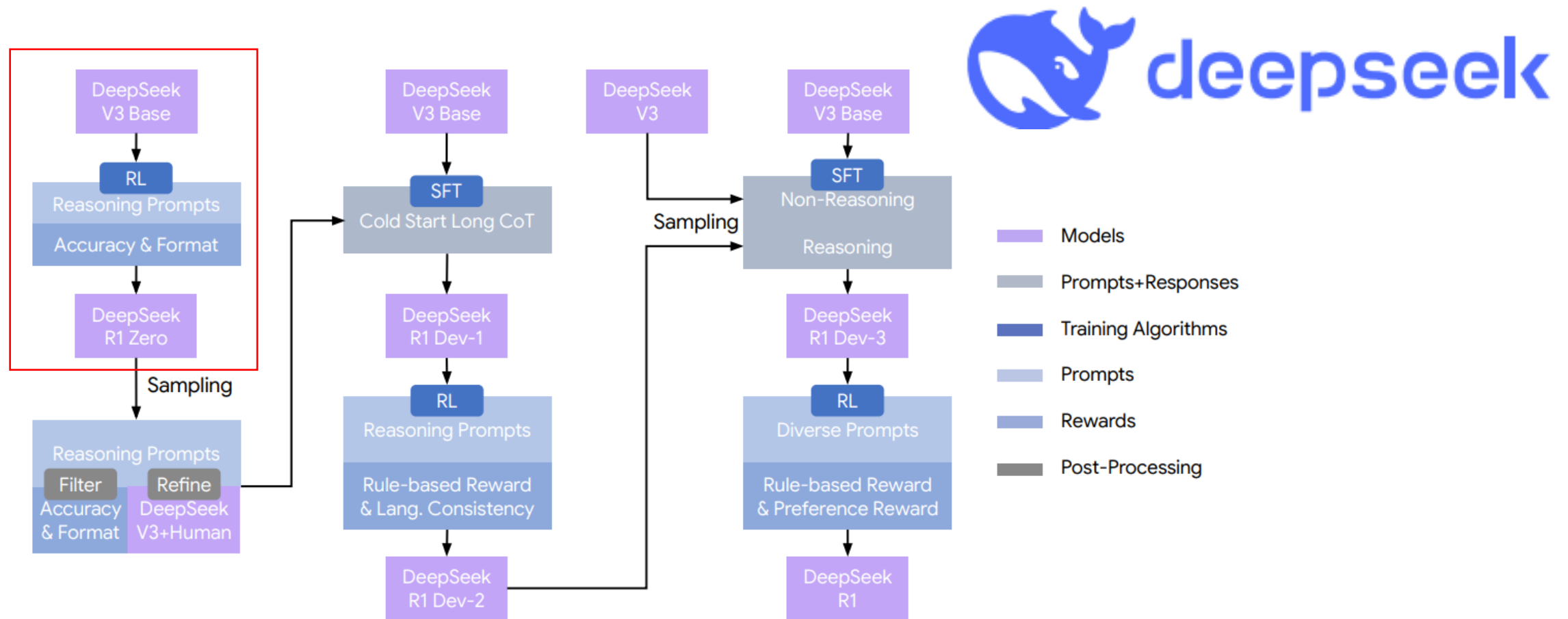
ChatGPT O1



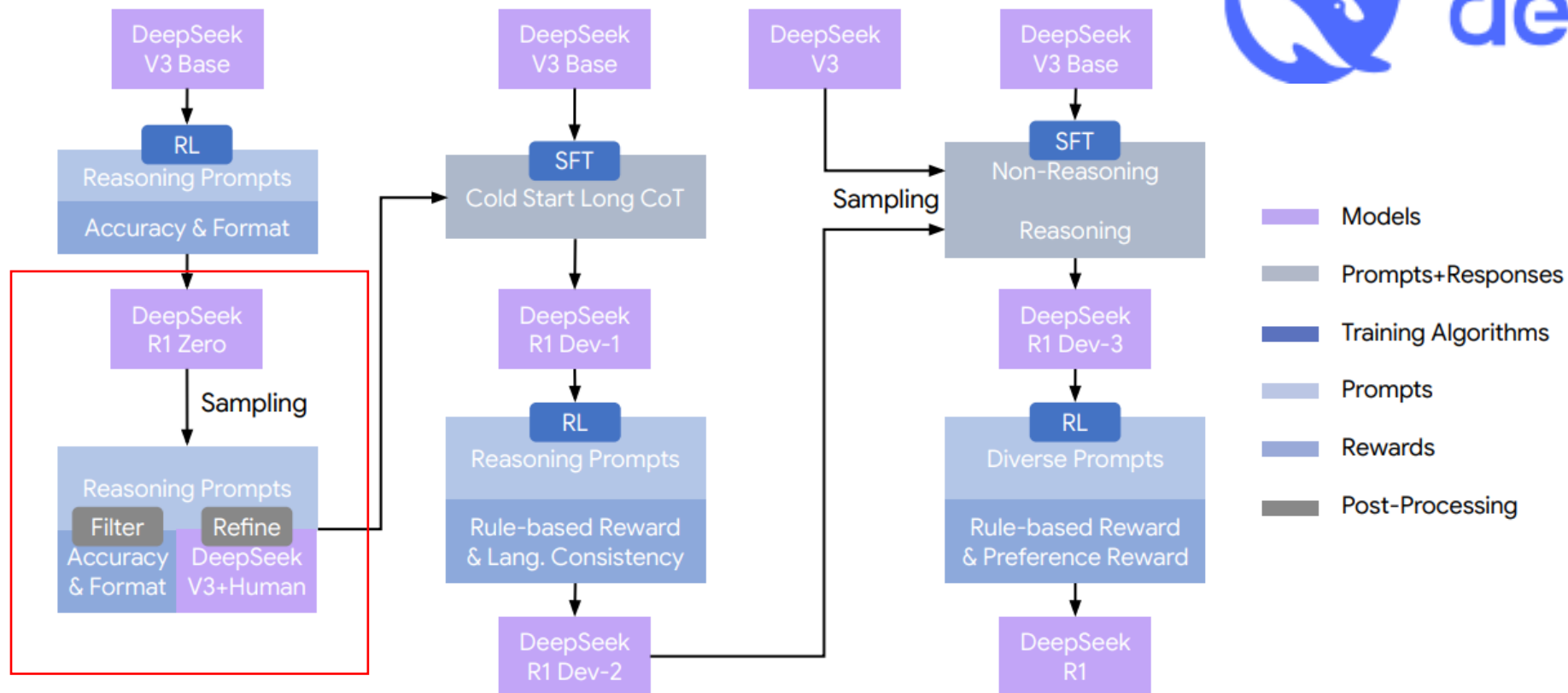
DeepSeek R1



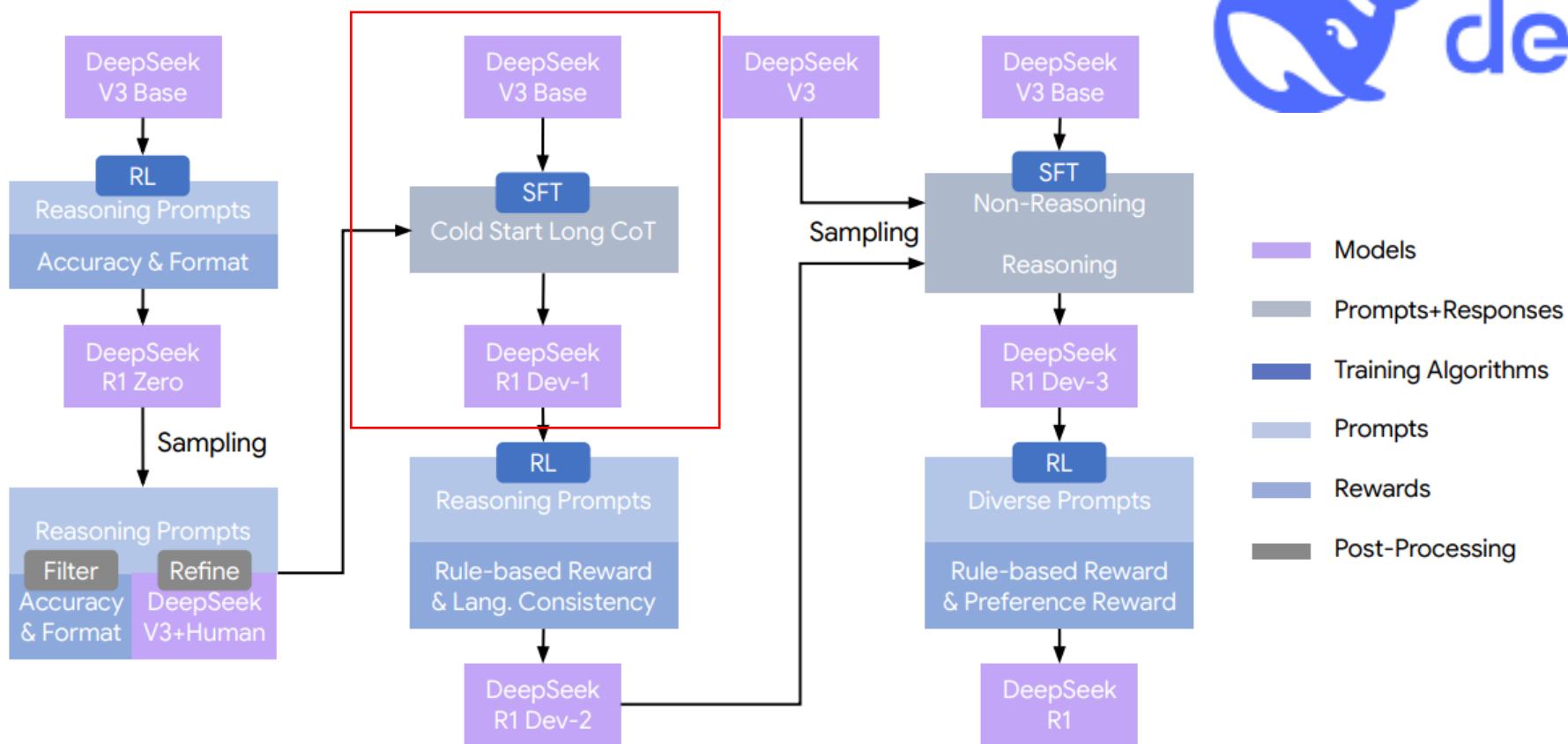
DeepSeek R1: R1 Zero 阶段



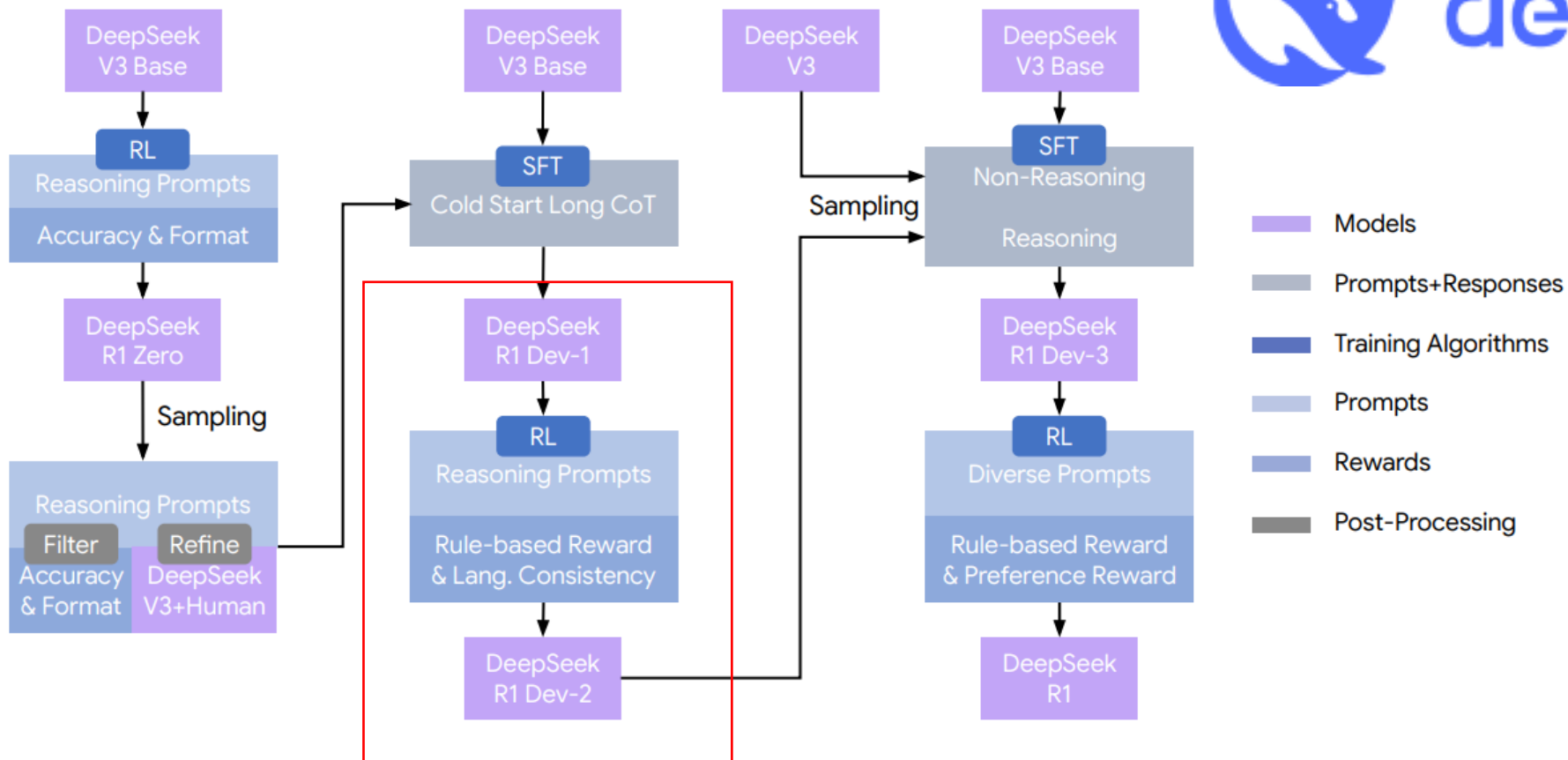
DeepSeek R1: 采样



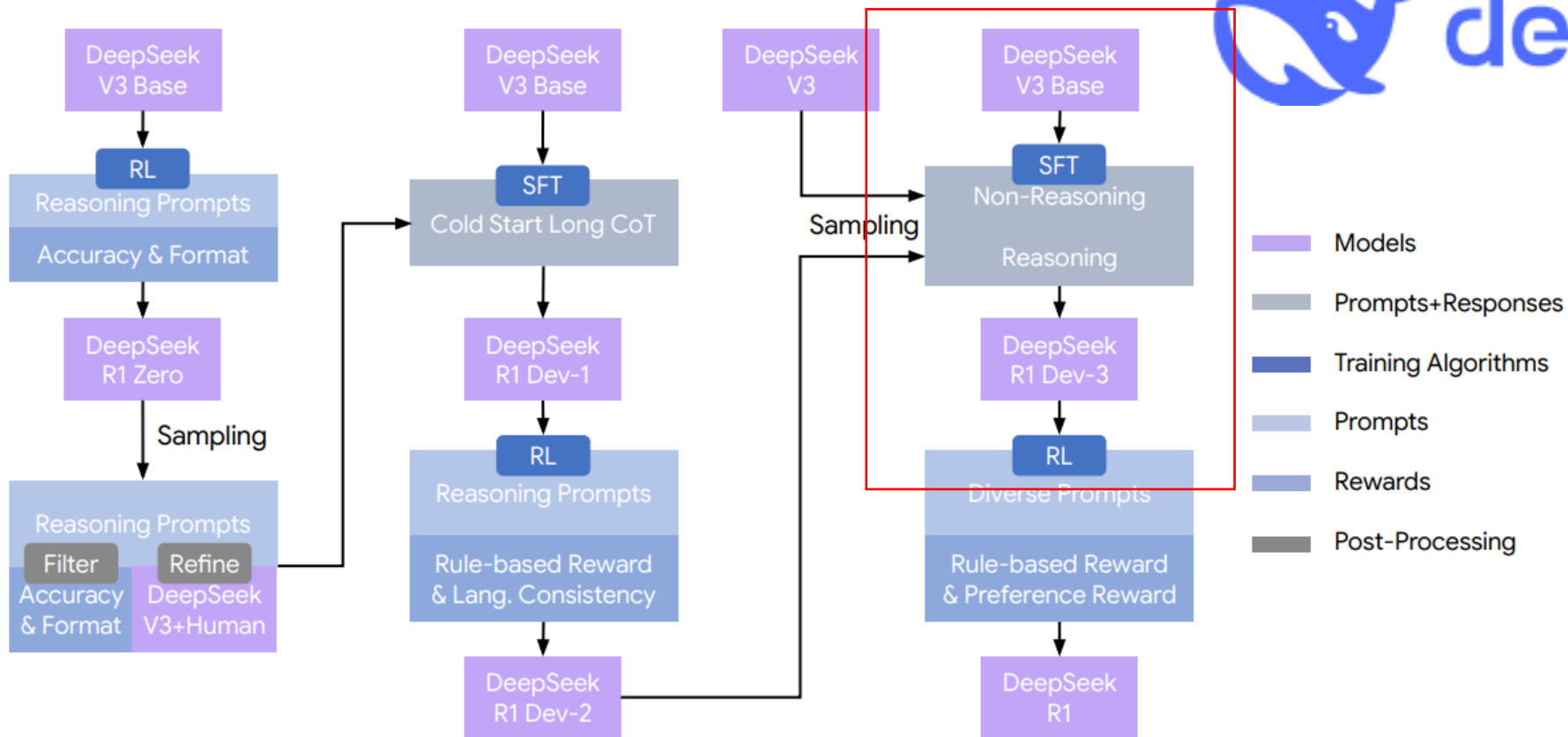
DeepSeek R1: 冷启动



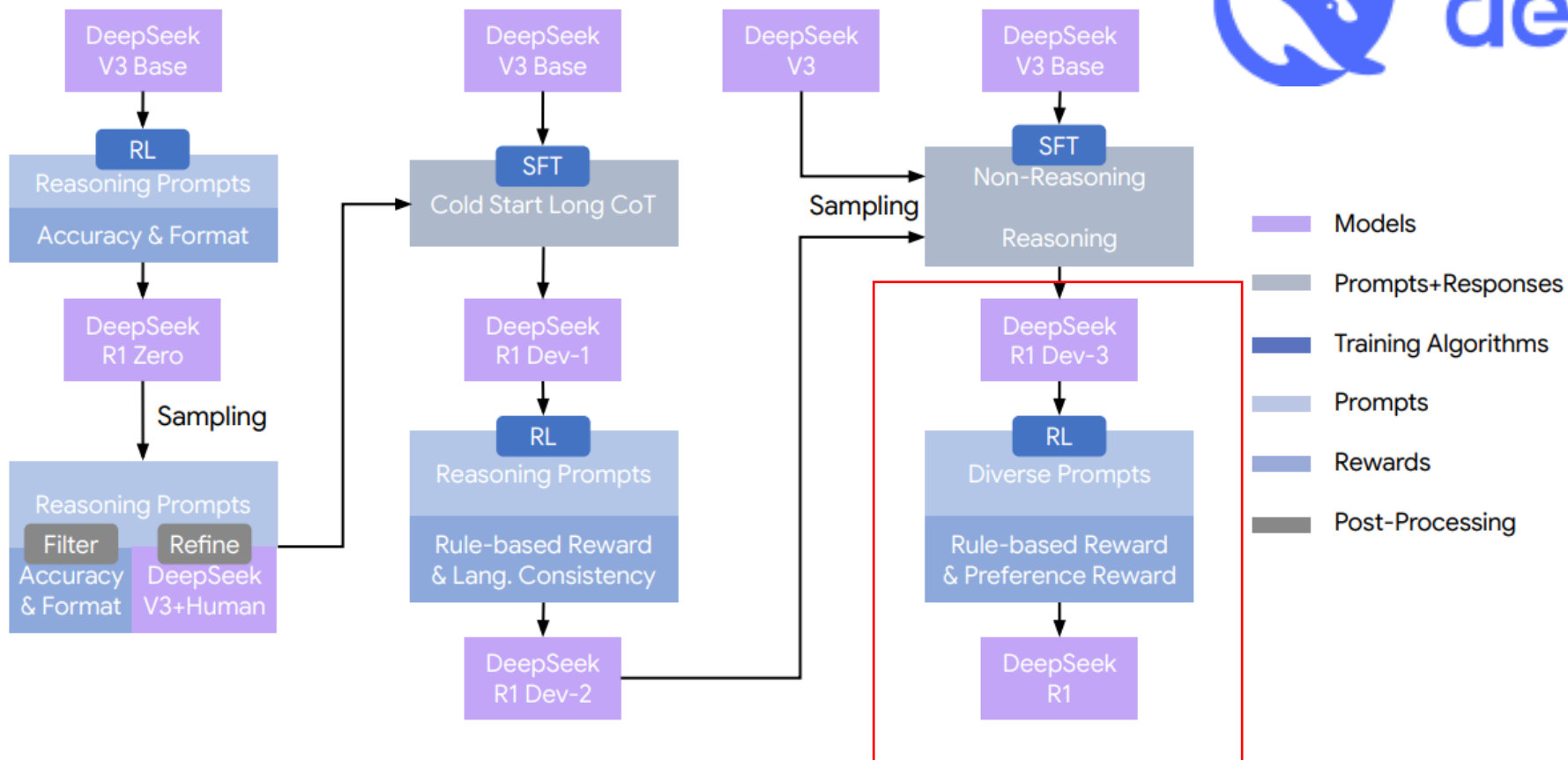
DeepSeek R1: 推理增强



DeepSeek R1: 联合SFT



DeepSeek R1: 对齐



DeepSeek R1

□ 优点

开源

复杂任务表现好

□ 缺点

推理速度慢

训练成本高

DeepSeek R1

	Benchmark (Metric)	R1-Zero	R1-Dev1	R1-Dev2	R1-Dev3	R1
English	MMLU (EM)	88.8	89.1	91.2	91.0	90.8
	MMLU-Redux (EM)	85.6	90.0	93.0	93.1	92.9
	MMLU-Pro (EM)	68.9	74.1	83.8	83.1	84.0
	DROP (3-shot F1)	89.1	89.8	91.1	88.7	92.2
	IF-Eval (Prompt Strict)	46.6	71.7	72.0	78.1	83.3
	GPQA Diamond (Pass@1)	75.8	66.1	70.7	71.2	71.5
	SimpleQA (Correct)	30.3	17.8	28.2	24.9	30.1
	FRAMES (Acc.)	82.3	78.5	81.8	81.9	82.5
	AlpacaEval2.0 (LC-winrate)	24.7	50.1	55.8	62.1	87.6
	ArenaHard (GPT-4-1106)	53.6	77.0	73.2	75.6	92.3
Code	LiveCodeBench (Pass@1-COT)	50.0	57.5	63.5	64.6	65.9
	Codeforces (Percentile)	80.4	84.5	90.5	92.1	96.3
	Codeforces (Rating)	1444	1534	1687	1746	2029
	SWE Verified (Resolved)	43.2	39.6	44.6	45.6	49.2
	Aider-Polyglot (Acc.)	12.2	6.7	25.6	44.8	53.3
Math	AIME 2024 (Pass@1)	77.9	59.0	74.0	78.1	79.8
	MATH-500 (Pass@1)	95.9	94.2	95.9	95.4	97.3
	CNMO 2024 (Pass@1)	88.1	58.0	73.9	77.3	78.8
Chinese	CLUEWSC (EM)	93.1	92.8	92.6	91.6	92.8
	C-Eval (EM)	92.8	85.7	91.9	86.4	91.8
	C-SimpleQA (Correct)	66.4	58.8	64.2	66.9	63.7



Gemini Thinking



□ 特点：
自适应推理深度
面向产品优化

Gemini Thinking



Benchmark		Gemini 3.1 Deep Think Feb 2026	Gemini 3 Pro Preview Thinking (High)	Opus 4.6 Thinking (Max)	GPT-5.2 Thinking (xhigh)
ARC-AGI-2 Abstract reasoning puzzles	ARC prize verified	84.6%	31.1%	68.8%	52.9%
Humanity's Last Exam Academic reasoning (full set, text + MM)	No tools	48.4%	37.5%	40.0%	34.5%
	Search + code execution	53.4%	45.8%	53.1%	45.5%
MMMU-Pro Multimodal understanding and reasoning	No tools	81.5%	81.0%	73.9%	79.5%
International Math Olympiad 2025 Mathematics		81.5%	14.3%	—	71.4%
Codeforces Coding and algorithms	No tools, Elo	3455	2512	2352	—
International Physics Olympiad 2025 (theory) Physics		87.7%	76.3%	71.6%	70.5%
CMT-Benchmark Condensed matter theory		50.5%	39.5%	17.1%	41.0%
International Chemistry Olympiad 2025 (theory) Chemistry		82.8%	69.6%	—	72.0%

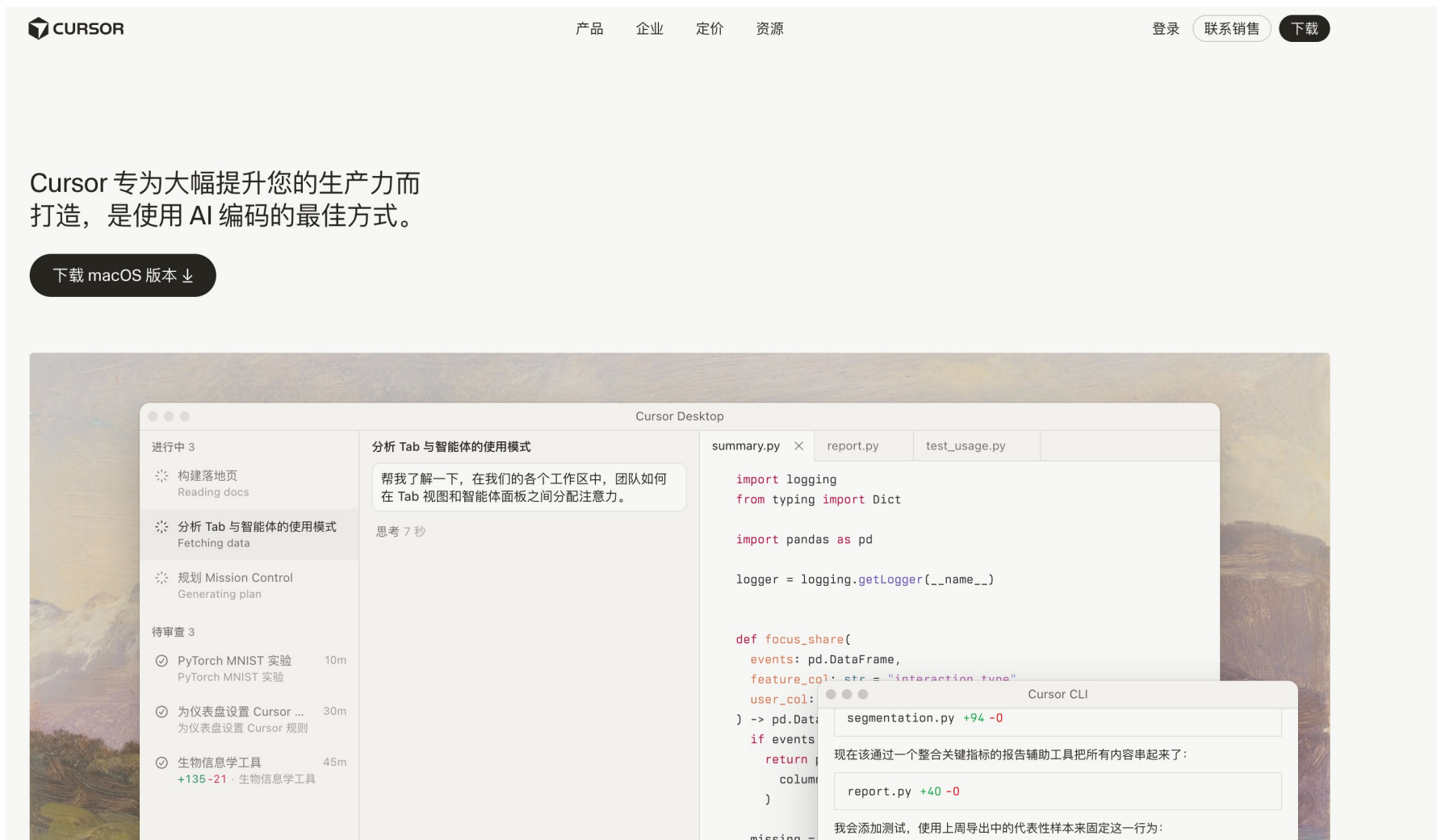
模型差异

模型	O1	R1	Gemini
路线	闭源	开源	闭源
核心方法	RL + 推理 thinking	RL + 整理	动态thinking
推理方式	长链推理	RL驱动推理	自适应推理
特点	强性能	可复现	面向产品
可控性	较弱	中等	强

应用场景

- 数学与科学计算
- 代码生成与调试
- 搜索与研究
- 数据分析
- 复杂决策与规划
- 智能 Agent 执行任务

代码补全



The screenshot displays the Cursor AI code editor interface. At the top, the CURSOR logo is on the left, and navigation links for '产品' (Product), '企业' (Enterprise), '定价' (Pricing), and '资源' (Resources) are in the center. On the right, there are buttons for '登录' (Login), '联系销售' (Contact Sales), and '下载' (Download).

The main content area features the text: 'Cursor 专为大幅提升您的生产力而打造，是使用 AI 编码的最佳方式。' (Cursor is specifically designed to significantly increase your productivity, making it the best way to use AI for coding.) Below this text is a button labeled '下载 macOS 版本 ↓' (Download macOS version ↓).

The interface shows a 'Cursor Desktop' window with several open files: 'summary.py', 'report.py', and 'test_usage.py'. The 'report.py' file is active, displaying Python code for logging and data analysis. A 'Cursor CLI' window is overlaid on the code, showing file change statistics: 'segmentation.py +94 -0' and 'report.py +40 -0'. A chat window on the left contains the prompt: '帮我了解一下，在我们的各个工作区中，团队如何在 Tab 视图和智能体面板之间分配注意力。' (Help me understand, in our various workspaces, how the team allocates attention between Tab views and the AI agent panel.) and the response: '思考 7 秒' (Thinking for 7 seconds).

探索研究

ChatGPT ▾

我们先从哪里开始呢？

有问题，尽管问

+ 思考 ▾

添加照片和文件

近期文件 >

创建图片

深度研究

网页搜索

更多 >

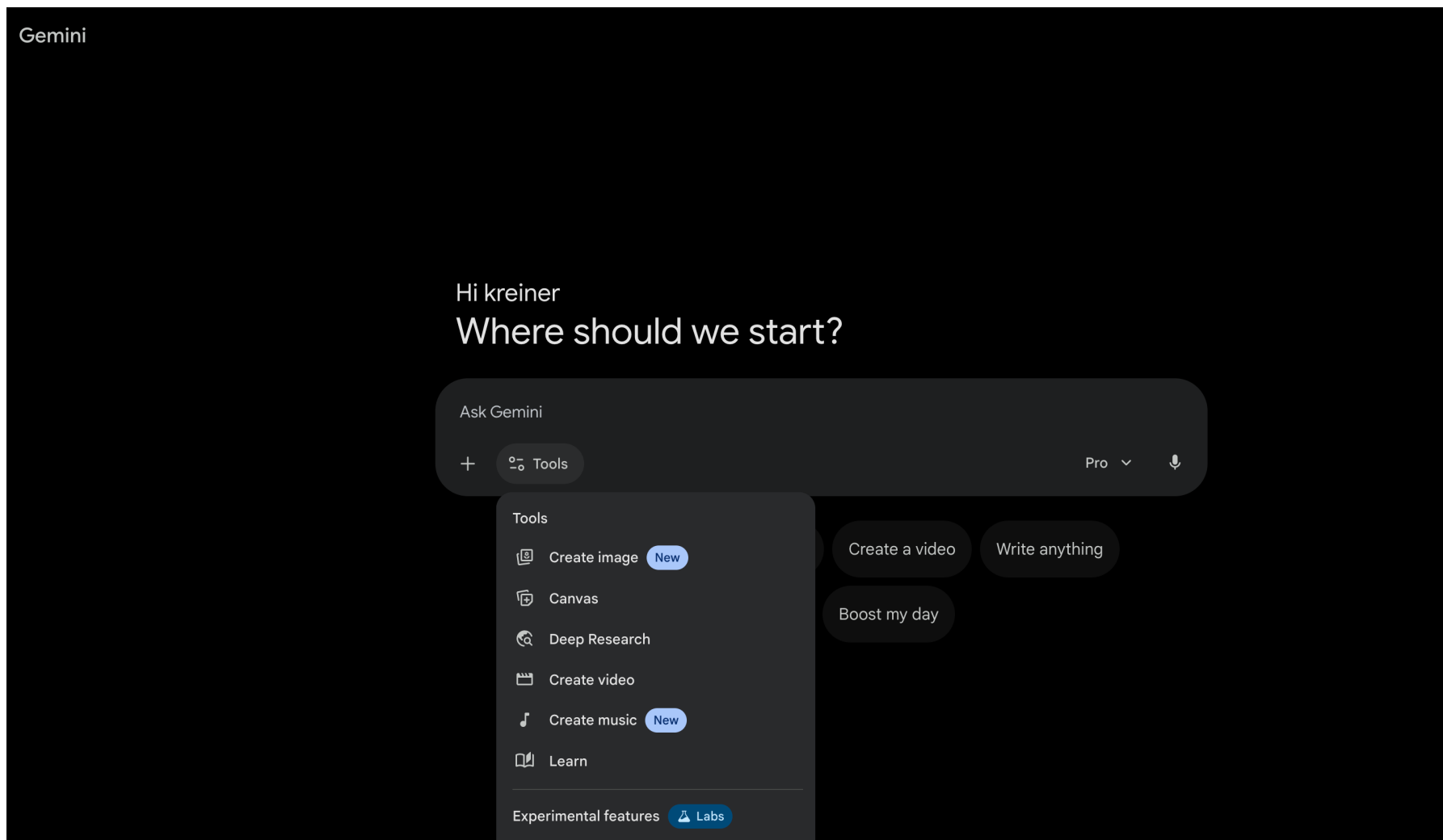
项目 >

生成图片

撰写或编辑

查找资料

探索研究



局限性

- 推理过程可能仍然是错的
- 中间步骤看起来合理，但可能只是伪解
- 长推理会增加成本和延迟
- 开放式任务很难定义唯一正确答案

本节复习

- 什么是推理模型
- 为什么需要推理模型
- 推理能力如何提升
- 代表模型有什么

参考文献

- Hua, Wenyue, and Yongfeng Zhang. "System 1+ system 2= better world: Neural-symbolic chain of logic reasoning." Findings of the Association for Computational Linguistics: EMNLP 2022. 2022.
- Guo, Daya, et al. "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning." arXiv preprint arXiv:2501.12948 (2025).



THANKS

<https://ictkc.github.io/teaching/2026spring-nlp>