质量大模型及其在接口测试场景下的实践

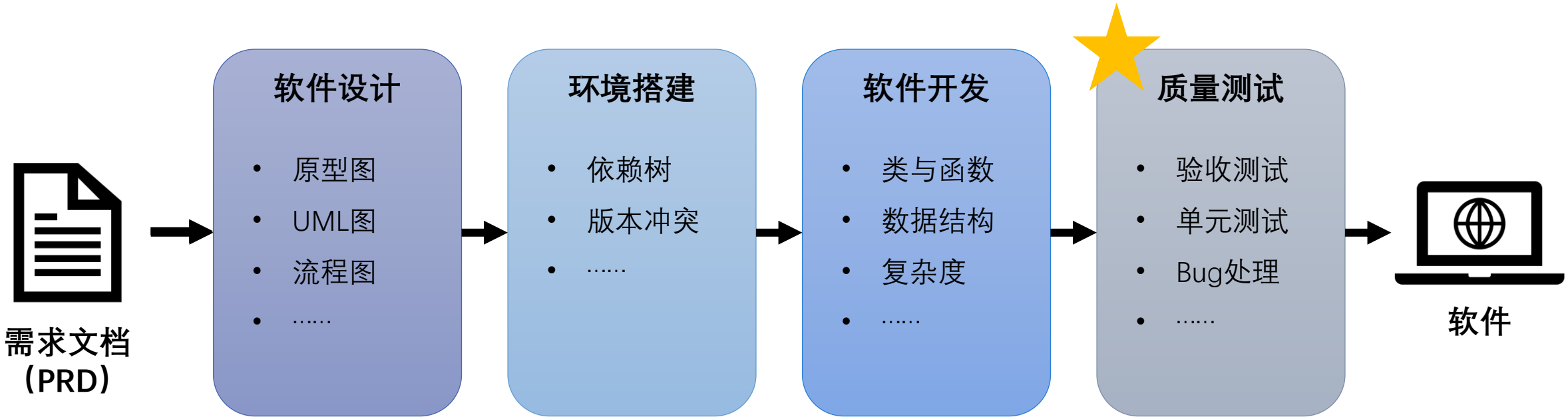
李庆泉 蚂蚁集团

目录
CONTENTS

PART 01

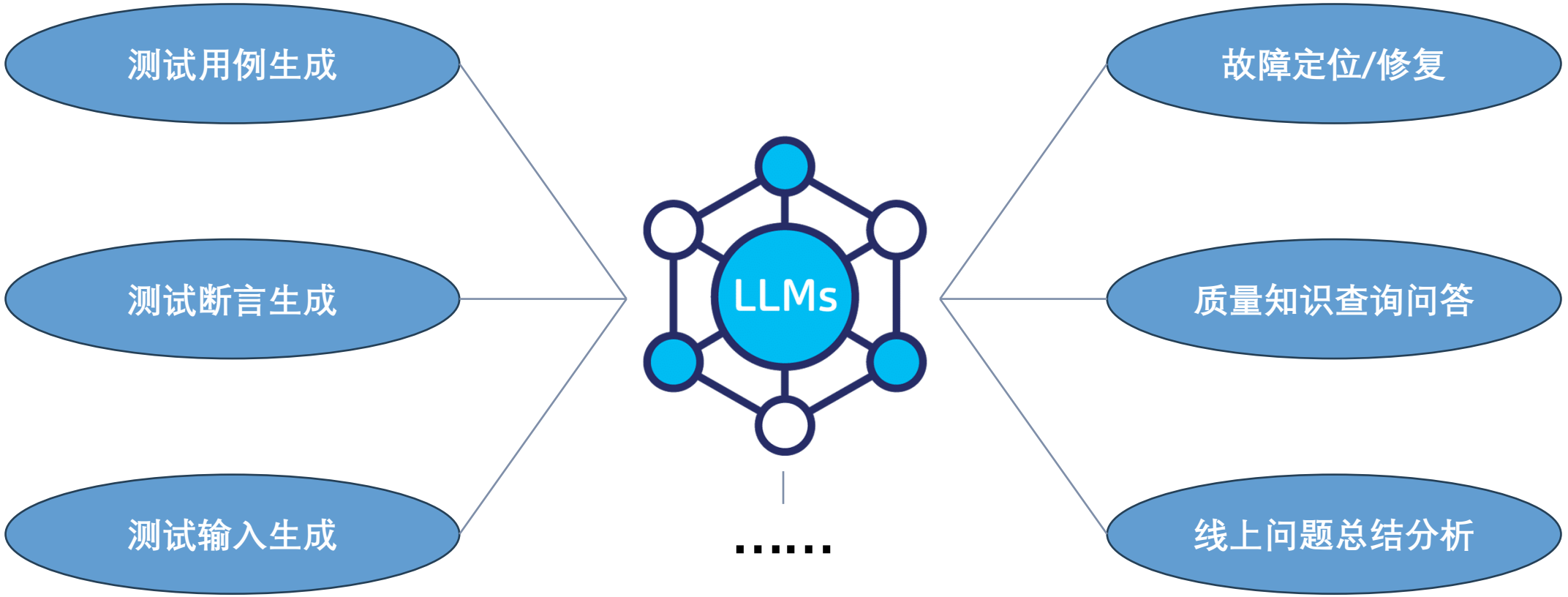
大模型如何解决质量域问题

大模型具有赋能软件研发全流程的能力



Li B, Wu W, Tang Z, et al. Devbench: A comprehensive benchmark for software development[J]. arXiv preprint arXiv:2403.08604, 2024.

大模型在质量域下的应用场景



Wang J, Huang Y, Chen C, et al. Software testing with large language models: Survey, landscape, and vision[J]. IEEE Transactions on Software Engineering, 2024.

大模型能够解决质量测试领域的哪些问题?

质量测试

覆盖率要求高

全面、准确地发现各种特殊情况与边界问题

自动化难度大

经验密度大、业务特异性高，难以自动化或手段难复用

人工成本高

测试用例撰写、质量问题分析等工作需要大量人工

1 **严谨、全面**

2 **知识、经验**

阶段	继续预训练	微调/对齐	对话/服务
能力	领域知识注入	多任务能力获取	拓宽应用边界

质量测试领域呼唤质量大模型

知识中心

语言理解

质量知识

业务经验

……

LLMs as Brains

执行中心

基础质量概念问答

端到端任务执行

……

服务中心

质量工具调度

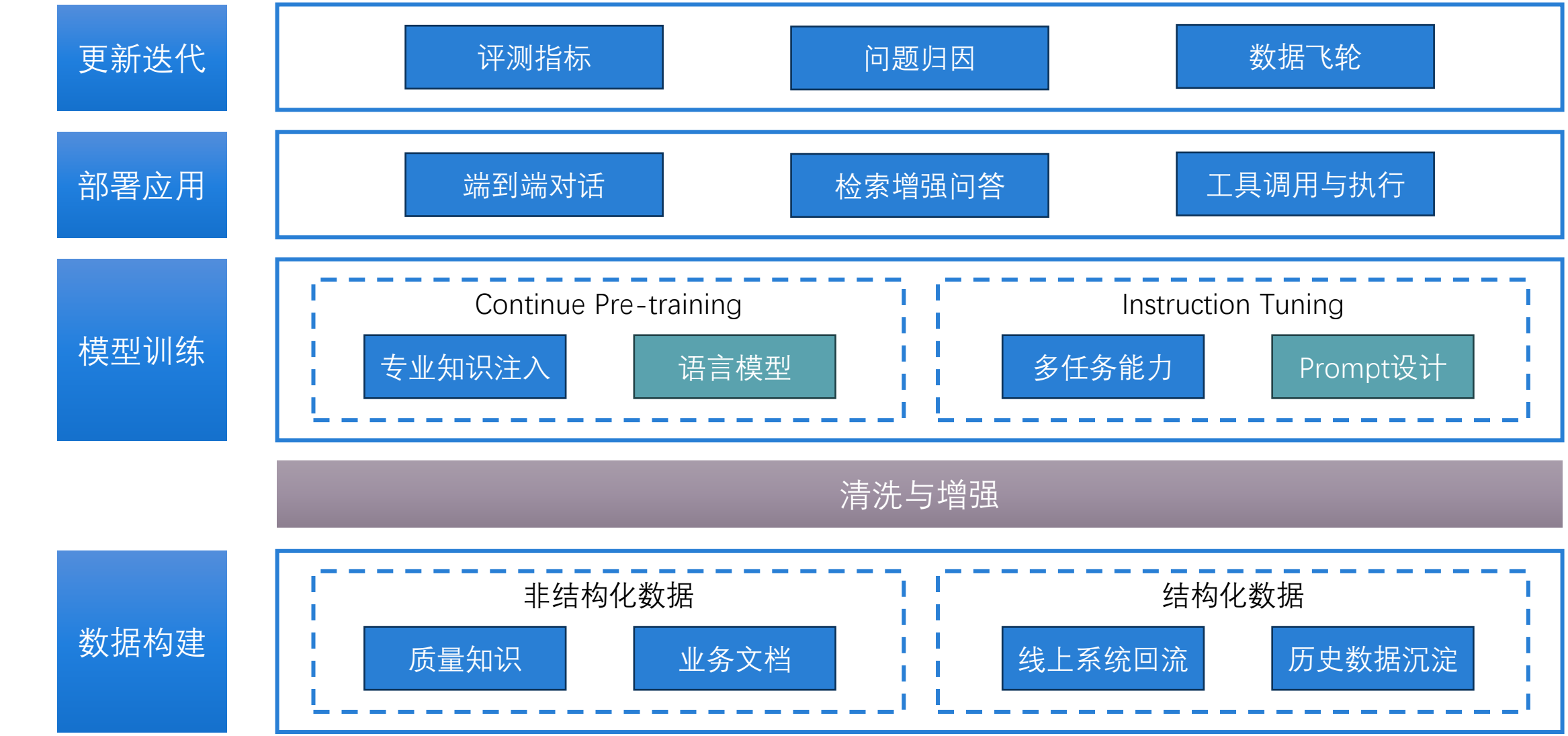
外部知识增强问答

……

PART 02

质量大模型的构建路径

质量大模型的构建路径


更新迭代
评测指标
问题归因
数据飞轮
部署应用
端到端对话
检索增强问答
工具调用与执行
模型训练
Continue Pre-training
专业知识注入
语言模型
Instruction Tuning
多任务能力
Prompt设计
清洗与增强
数据构建
非结构化数据
质量知识
业务文档
结构化数据
线上系统回流
历史数据沉淀

继续预训练——知识注入

训练材料：知识构建

知识来源

- 质量域
- 业务域
- 其他背景

知识处理

- 切片
- 聚类
- 数据增强

知识质量

- 筛选
- 清洗
- 数据分析

训练任务：语言模型

训练对象：开源模型底座



* 不完全列举

指令微调——技能培训

关键问题：质量域中的许多任务都是以结构化数据为基础，如何用基于非结构化对话形式的大模型完成结构化数据任务？——解法在于**如何针对特定任务设计指令并进行微调**

流程：任务选取 → 数据获取 → 指令构建 → 微调

两大类任务选取：

- 质量/业务知识问答
  - 向模型提供基础的质量问答能力
- 特定质量任务执行
  - 以端到端形式执行的质量任务，如测试用例生成等

两大类数据来源：

- 基础知识
  - 包括**外部**基础知识与**内部**的文档积累
- 任务数据
  - 来源于真实场景的特定任务数据

指令构建技巧：

- 明确任务需求
  - 清晰描述任务的推理逻辑、执行步骤
- 格式规约
  - 对格式进行严格规约，降低工程难度
- 指令泛化
  - 增强鲁棒性

微调注意事项：

- 数据分布
  - 平衡各任务的数据分布，避免过拟合
- 训练成本
  - 多卡训练、LoRA微调等，节省时间与显存成本

应用方式——作为端到端Bot

以端到端对话形式执行质量任务

上下文学习

Input

Code Demonstrations

```
### Fix ESLint error in the following JavaScript code:
### Buggy JavaScript
  return date.getFullYear();
  break;
case 'month':

"rule_id" : no-unreachable
"evidence": Unreachable code.
"warning_line": break

### Fixed JavaScript
  return date.getFullYear();
case 'month':

END_OF_DEMO
```

Demonstration 1

Demonstration 2

...

Demonstration N

Query

Instruction

```
### Fix ESLint error in the following JavaScript code:
### Buggy JavaScript
  return 'local';
  break;
case 'mapbox:':

"rule_id" : no-unreachable
"evidence": Unreachable code.
"warning_line": break;

### Fixed JavaScript
```

Error Snippet

Error Context

Context

Output

```
  return 'local';
case 'mapbox:':
```

逻辑构建

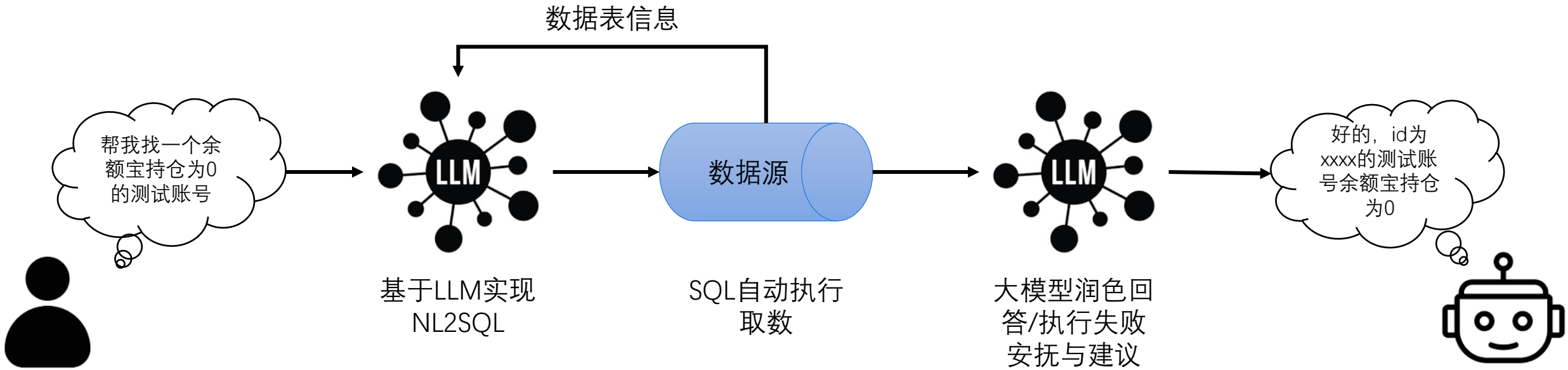
	Bug ID:	**JacksonDatabind-110**	
	Issue Title:	Inconsistent handling of `Collections$UnmodifiableList` VS `Collections$UnmodifiableRandomAccessList`	
	Issue Description:	I'm sorry to bring that one up again, but I'm under the impression that the issue about unmodifiable collections {...} is still not solved completely. {...} Currently `java.util.Collections.unmodifiableList( )` can only return these 2 types of unmodifiable lists, so I believe it is safe for now to just hardcode yet another special case for this class. This can currently be solved on user side by adding a mixin, bug since `Collections$UnmodifiableRandomAccessList` is supported, I would find it natural to also support non-random access variant.	
Prompt 1	Issue title + description	<buggy function>	Identify which line(s) are buggy in the code
	The buggy line in the code is : `if ( type.hasRawClass ( CLASS_UNMODIFIABLE_LIST ) ) { conv = converter ( TYPE_UNMODIFIABLE_LIST , type , List . class ) ; }` *Response 1*		
Prompt 2	Previous context + response		Explain why these lines are buggy
	These lines are buggy because they only handle `Collections$UnmodifiableRandomAccessList`, which is not only possible type of unmodifiable list that can be returned by `Collections.unmodifiableList( )`, as there is also `Collections$UnmodifiableList` .{...} This results in a failure to deserialize non-random access unmodifiable lists, such as those returned by `Collections.unmodifiableList(new LinkedList<>(Arrays.asList("f", "s")))` *Response 2*		
Prompt 3	Previous context + response		Please provide a fixed version of the code with minimal changes.

[1] Kang S, Yoon J, Yoo S. Large language models are few-shot testers: Exploring llm-based general bug reproduction[C]//2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). IEEE, 2023: 2312-2323.

[2] Fakhoury S, Chakraborty S, Musuvathi M, et al. Towards generating functionally correct code edits from natural language issue descriptions[J]. arXiv preprint arXiv:2304.03816, 2023.

应用方式——作为工作流组件

大模型作为质量工作流当中的组件，为质量工具提供了丰富的泛化性和强大的生成能力，提升了质量工具的能力上限。以一个测试账号查询工作流为例：



大模型通过接受用户输入与数据表（表名、字段名称、字段描述、取值范围等）信息既可实现NL2SQL，节省了小模型训练的成本

PART 03

质量大模型在接口测试中的实践

接口测试的特点与难题

专业性

对专家知识和经验要求高

场景化

需要结合特定业务逻辑

复杂性

人工成本和时间成本较高

质量大模型应用——校验点生成

输入

```
{
  "接口名": "interface_1",
  "入参": {
    "param_1": "xxx",
    "param_2": 1234,
    "param_3": true
  },
  "出参": {
    "status": 200,
    "result": {
      "success": true,
      "param_4": "xxx",
      "param_5": 1234,
    },
    "description": ""
  }
}
```

质量大模型

输出

```
{
  "check_list": {
    {
      "key": "$.result.success",
      "value": true,
      "type": "boolean",
      "compareType": "equ",
      "desc": ""
    },
    {
      "key": "$.result.param_4",
      "value": "",
      "type": "string",
      "compareType": "notBlank",
      "desc": ""
    },
    {
      "key": "$.result.param_5",
      "value": 4321,
      "type": "int",
      "compareType": "equ",
      "desc": ""
    },
  }
}
```

质量大模型应用——校验点生成

Prompt组装

输入信息

任务所需的必要信息

接口名称 ✚ 入参 ✚ 出参

推理步骤

根据专家经验总结的执行该任务所需的推理过程

识别出参中的关键字段 → 根据业务要求确定字段的取值范围和校验方式 → 撰写校验点列表

规范要求

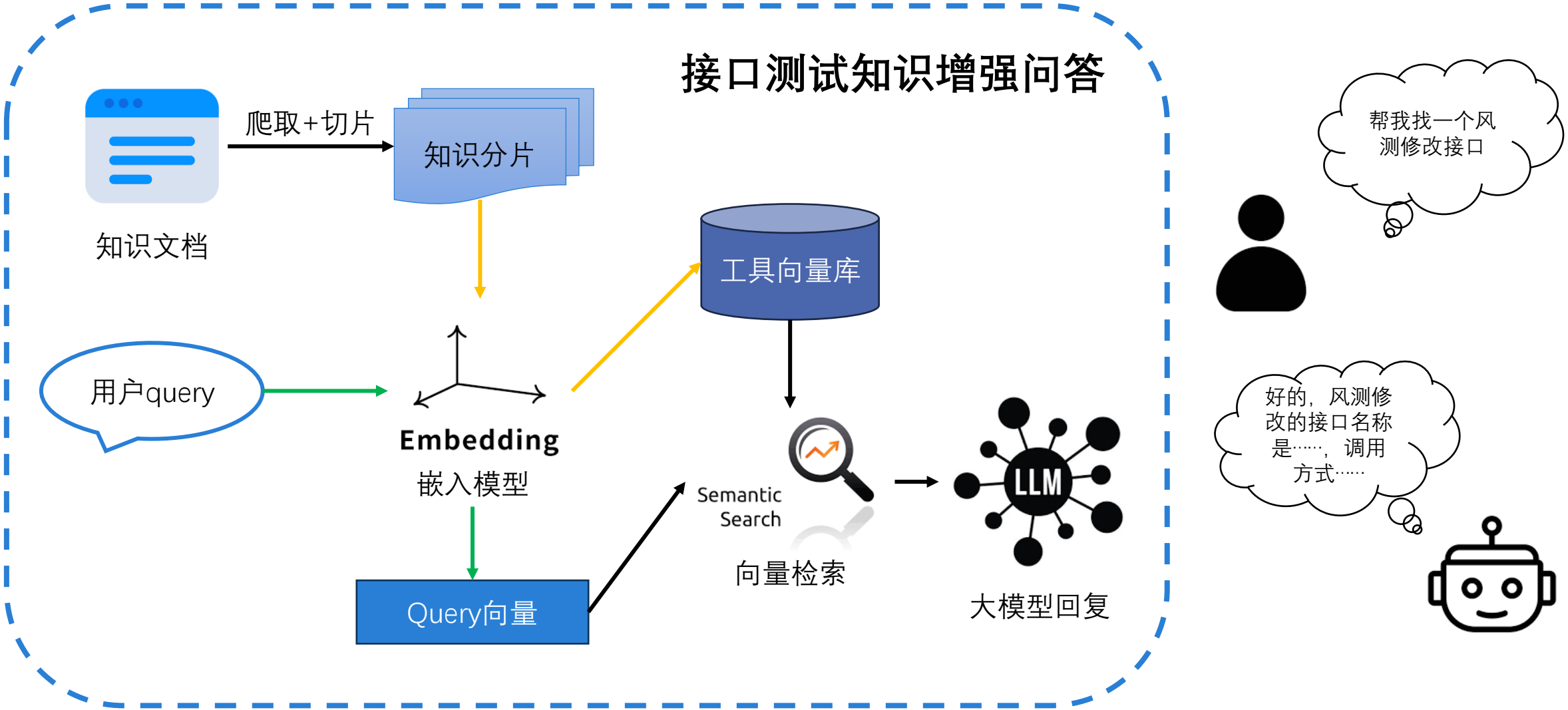
强调针对任务输出格式、形式等规范的要求

输出必须遵循json格式

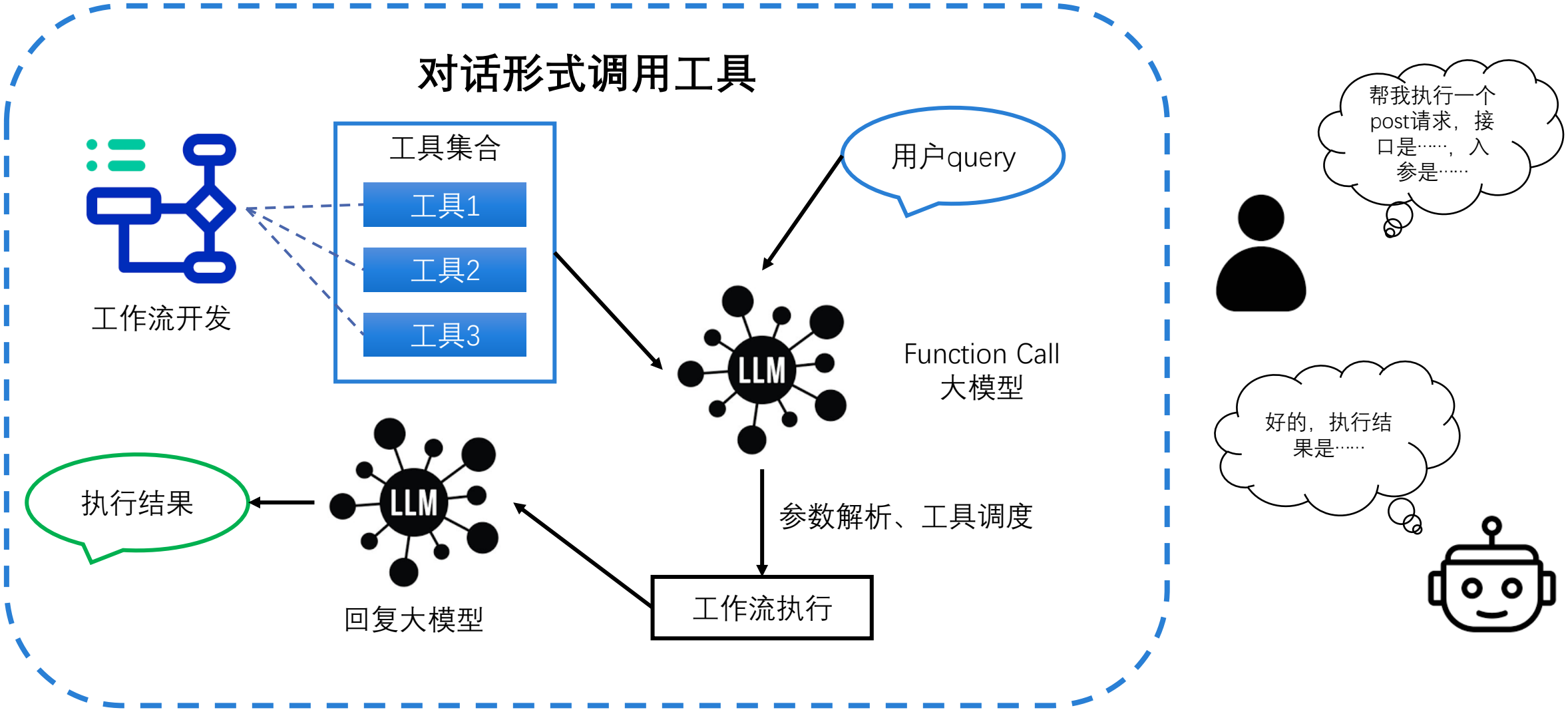
每个校验点必须包含的特定字段

……

质量大模型应用——业务知识问答



质量大模型应用——工具执行



PART 04
未来展望

总结

1 质量 + 大模型

产品内容上围绕质量 + 大模型的提效工作

2 垂类解决方案

产品设计上是一套较通用的垂直领域的大模型解决方案

3 大模型行业赋能框架

算法设计上是一套通用的垂类领域大模型提效框架

质量大模型技术趋势

- 浩如烟海的质量知识与业务经验
- 多样的质量任务
- 前置的问题发现
- 代码能力与问答能力的兼顾

AiDD

THANKS