AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

AI驱动软件研发
全面进入数字化时代

中国·深圳 11.24-25

AI+
software
Development
Digital
summit

LLM辅助测试脚本代码生成

演讲人 华为
万锐媛

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

K+峰会 深圳站

K+ 全球软件研发行业创新峰会

会议时间：2024.05.24-25

K+峰会 上海站

K+ 全球软件研发行业创新峰会

会议时间：2024.09.20-21

AiDD 峰会

AiDD峰会 深圳站

AI+ 软件研发数字峰会

会议时间：2023.11.24-25

AiDD峰会 北京站

AI+ 软件研发数字峰会

会议时间：2024.07.19-20

AiDD峰会 深圳站

AI+ 软件研发数字峰会

会议时间：2024.11.15-16

演讲嘉宾

万锐媛

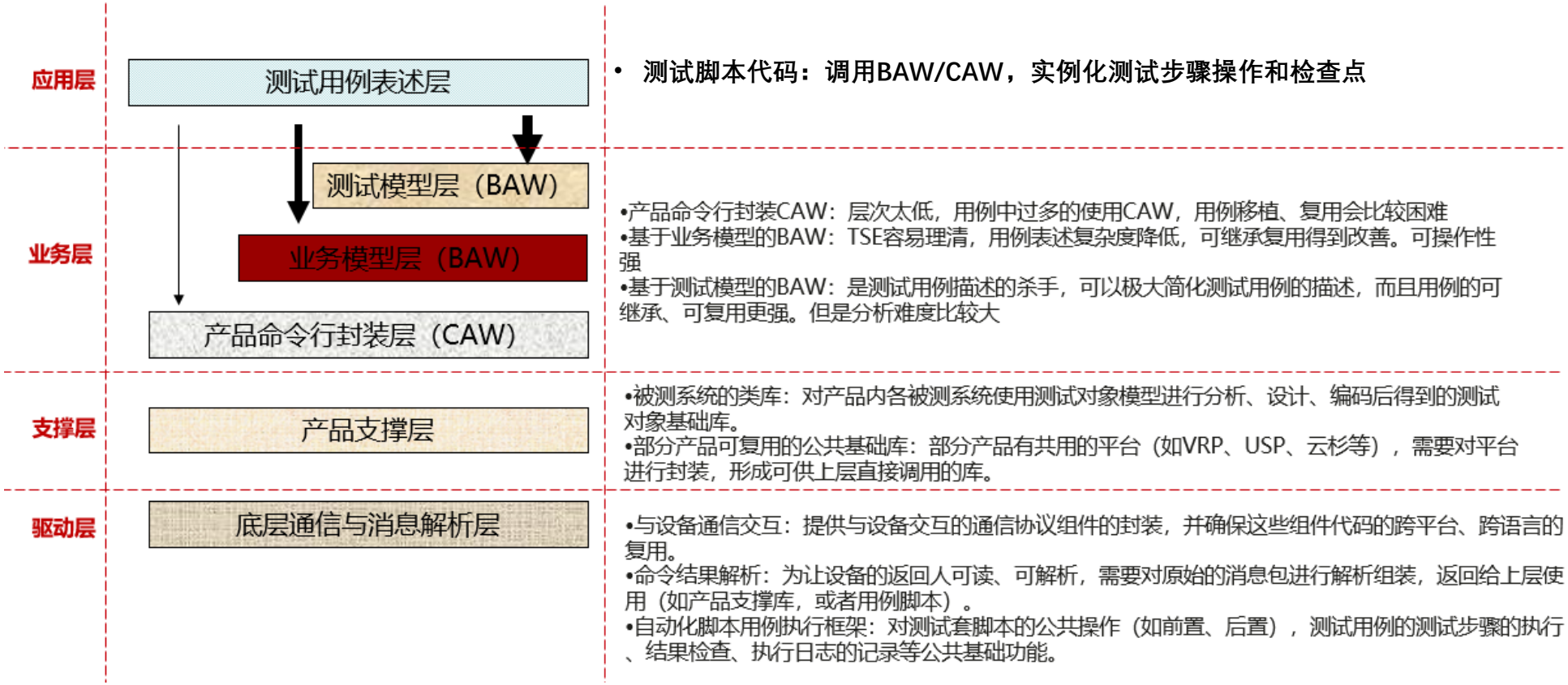
华为测试技术专家，智能化测试C-TMG主任，12年获清华大学EE博士学位，曾赴UC Berkeley EECS访问学者。16年加入华为至今，从事智能辅助测试技术探索、工程工具落地规划和设计，带领团队聚焦智能辅助测试设计、测试自动生成，多目标精准回归，测试失败智能定界等方向，成功孵化多项智能测试服务并规模落地应用，获得华为总裁奖、金牌团队、海盗派重大测试技术突破奖等。获得专利8项，在ICSE、ASE等国际会议发表论文，担任ICST、ISSRE、TiD、NJSD、ChinaSoft等会议演讲嘉宾，AiDD出品人

目录
CONTENTS

PART 01
系统测试脚本代码生成任务背景

系统测试自动化框架复杂性

系统测试框架复杂，特定领域多级测试方法库，系统测试代码生成是一个典型的面向一方，二方，三方库的代码生成任务

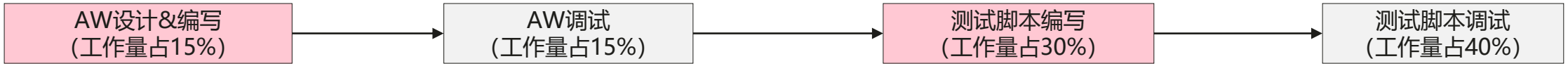


AW: Action Word, 测试方法关键字

系统测试代码生成应用场景

痛点 大部分ICT产品线**自动化开发效率偏低（2-3个/天）**，随着业务测试工作量不断增加，**无法仅靠人力增加解决**自动化问题

AI for Test automation 全码应用场景



场景3：LLM辅助生成AW代码

场景1：防护网补全&特性增强开发场景

场景2（主流场景）：新特性、新业务开发场景

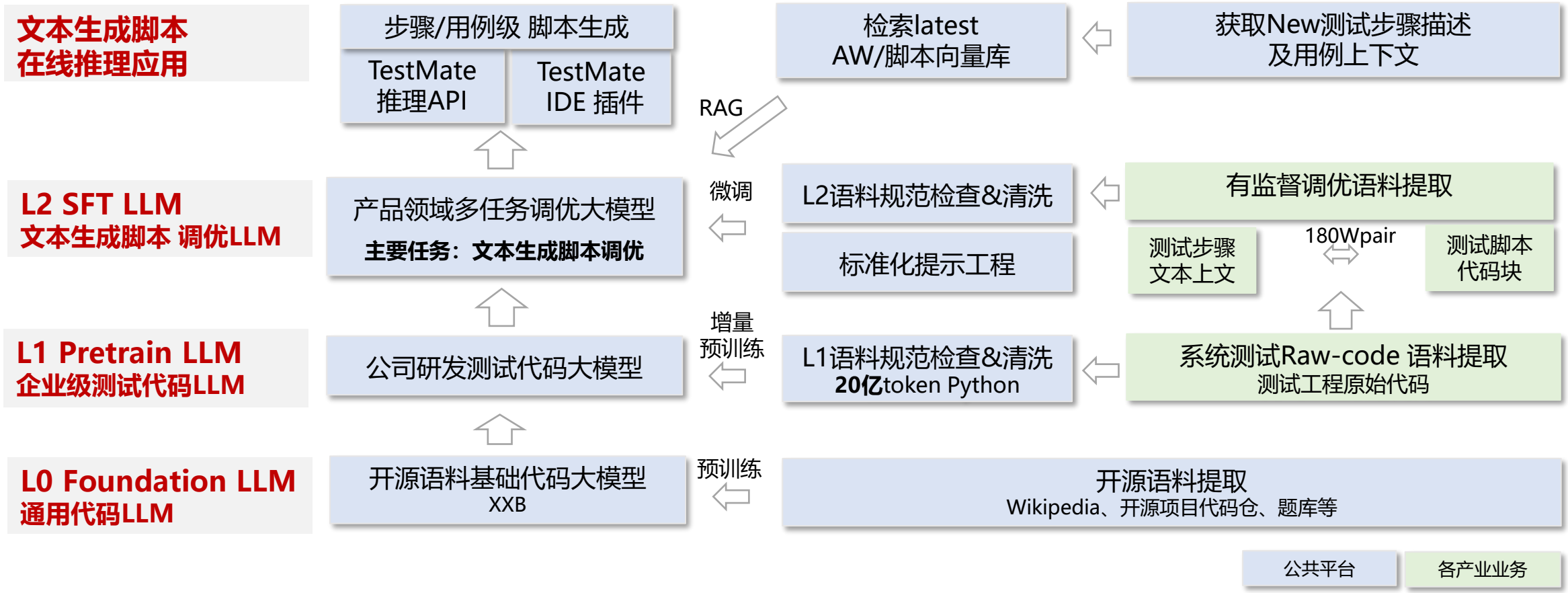
	应用场景
场景1	**防护网补全&特性增强开发场景** (可复用历史接口、流程、功能)
场景2	**全新特性、新业务开发场景** (无历史接口、流程、功能复用，需通过知识库外挂生成脚本)
场景3	**AW辅助生成** (公共接口/库函数封装)

PART 02

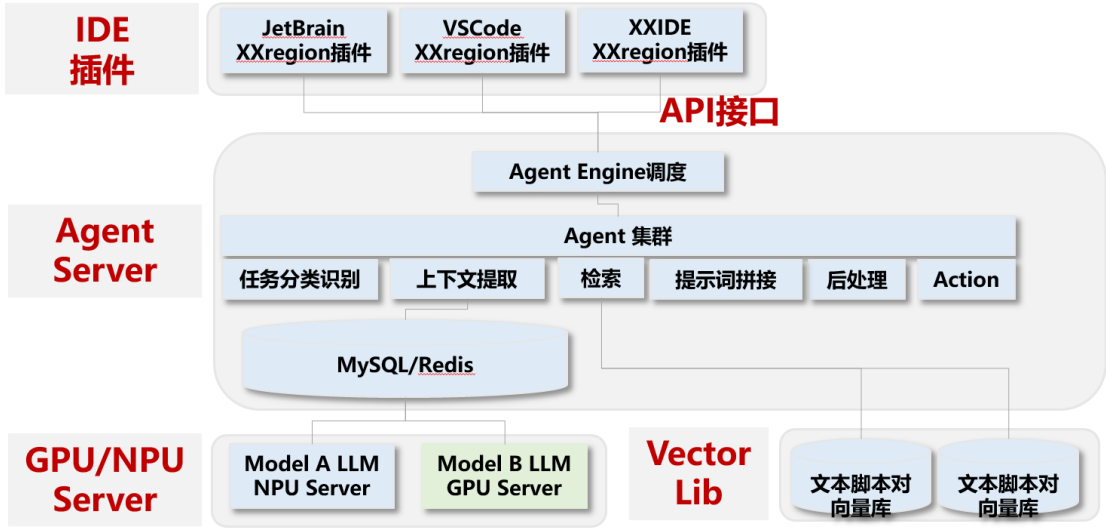
系统测试领域代码大模型训练总体方案

LLM辅助测试脚本生成总体方案

面向**企业级**，各**特定领域业务**上下文的 **系统测试代码**生成 **统一**解决方案



CodeArt Snap-TestMate测试助手（当前华为内部only）

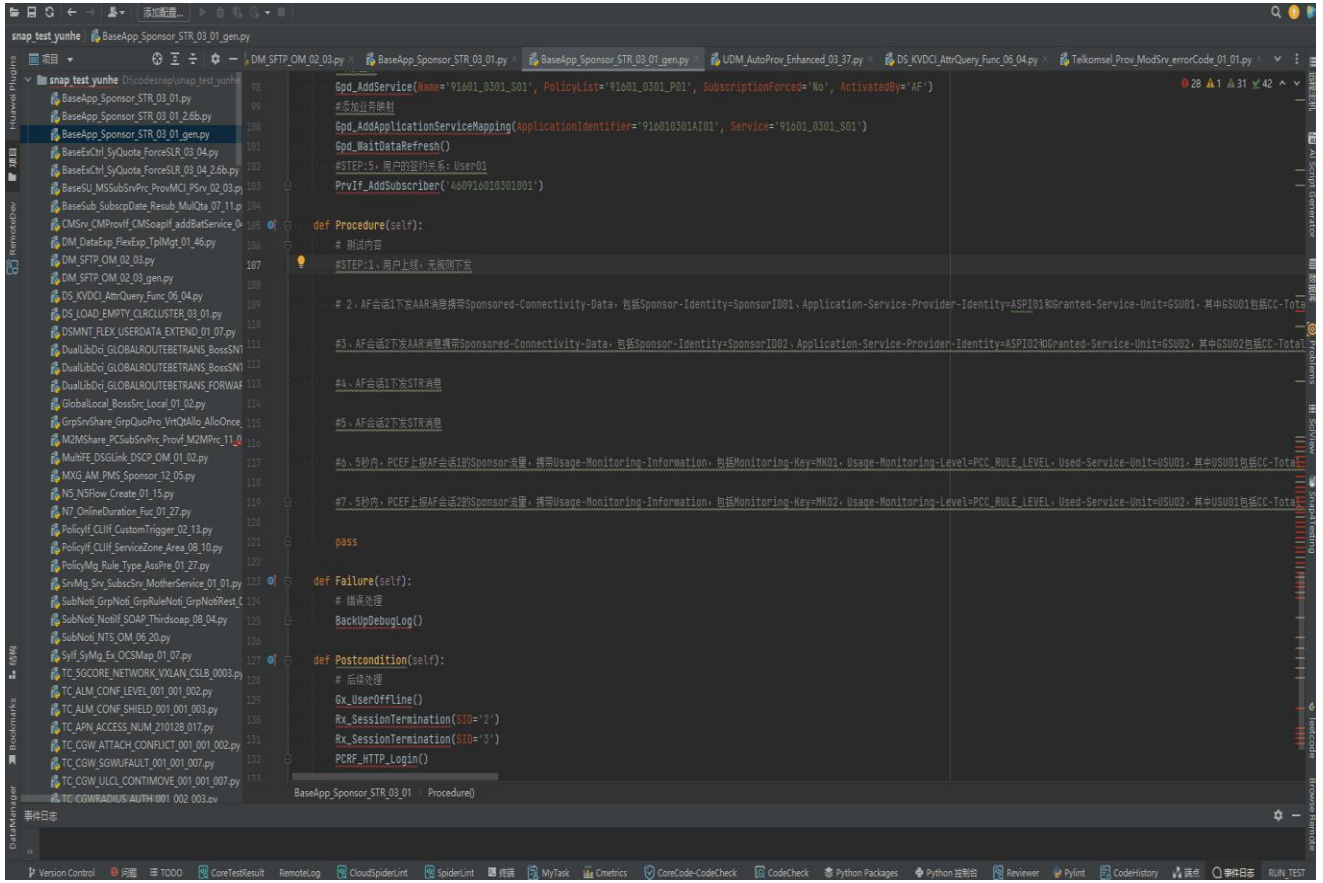


自动识别测试文本语句

流式代码生成

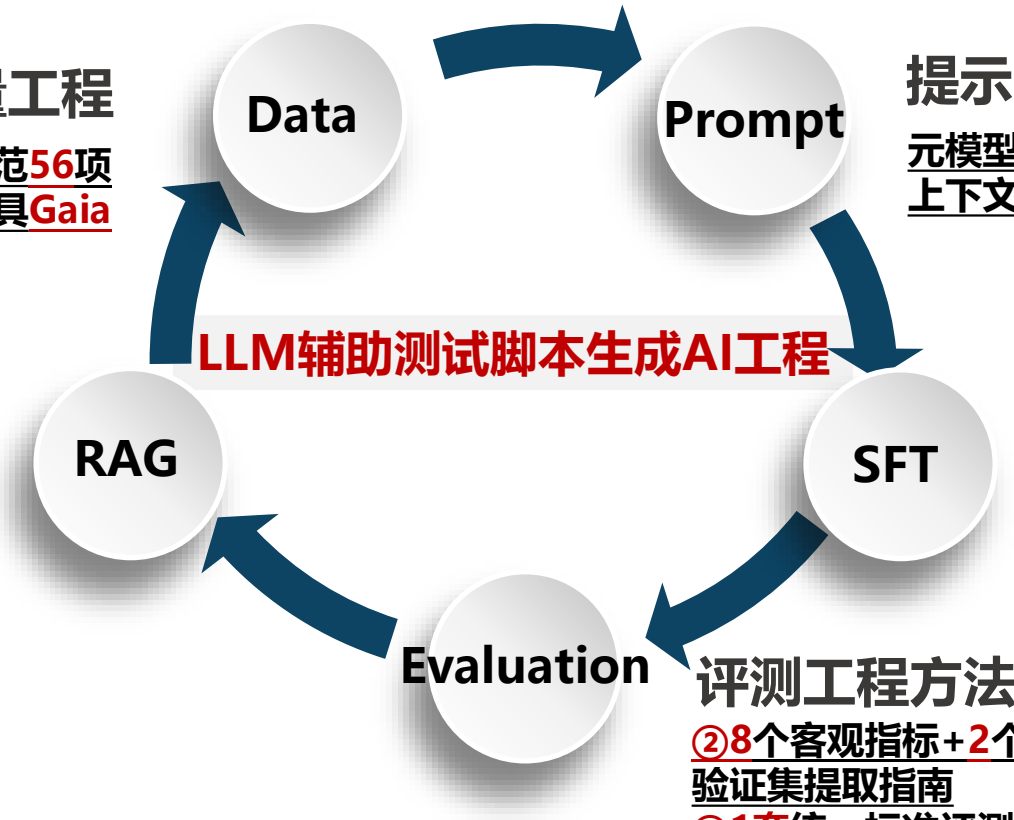
多模型路由

测试步骤级脚本片断代码生成



PART 03
系统测试代码生成任务AI工程

LLM辅助测试代码生成AI工程



数据质量工程

①数据质量检查和清洗规范56项
④1套质量检查&清洗工具Gaia

提示工程

元模型：全局info+用例级info+步骤info+跨用例info
上下文info提取

LLM训练&推理

L1 增量预训练
L2 有监督调优SFT
1X ms 级NPU推理加速及后处理
③ TestMate IDE插件工具

评测工程方法

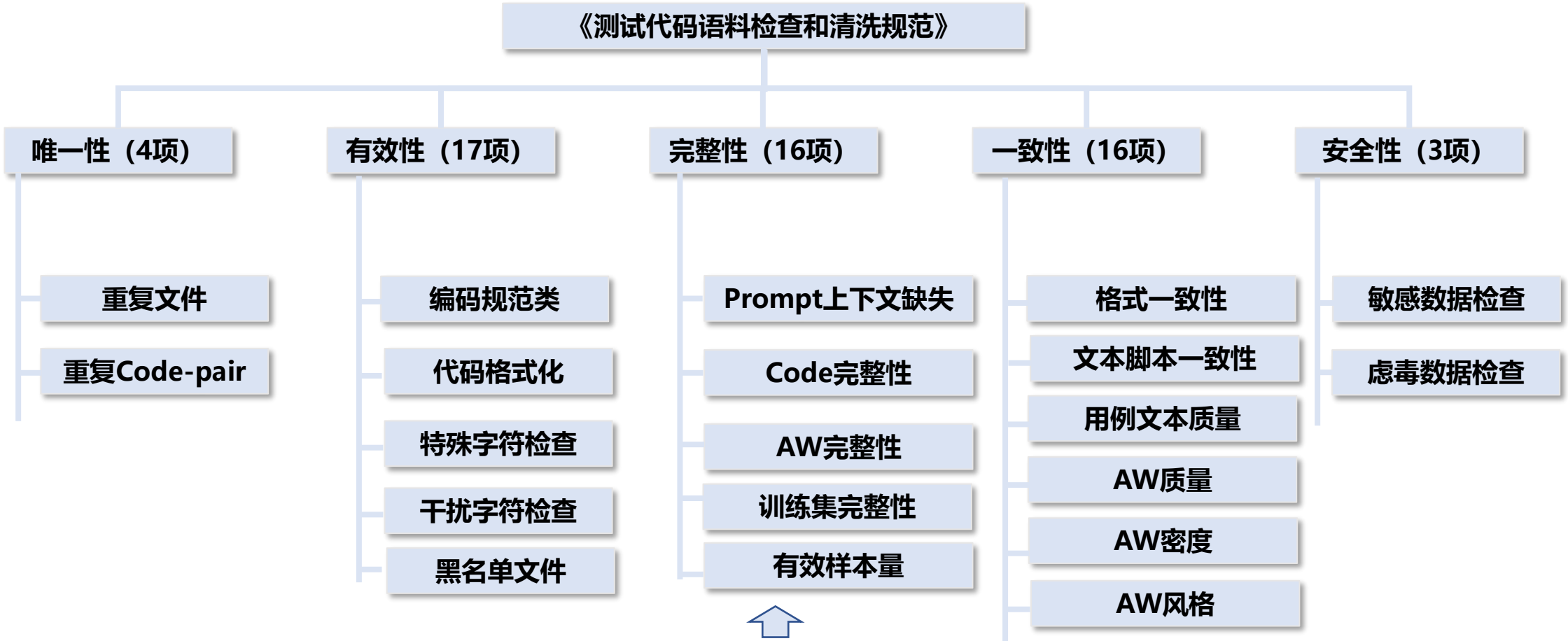
②8个客观指标+2个主观指标
验证集提取指南
⑤1套统一标准评测工具和接口ModelEvaluation

检索增强生成

测试工程代码向量库
相似AW/文本脚本pair检索

数据工程：L1/L2 测试代码语料质量检查和清洗规范

有效性，完整性，一致性，唯一性，安全性 **5性** 关键数据特征共**56条**检查项
云、管、端、车、芯统一规范



Based on 10年来测试脚本工程规范基础

L2测试用例文本脚本pair语料样例

Part1 产品和全局信息

```
product_info: 
{
    product_line_name: "数据存储产品线",
    pdu_name: "分布式存储产品部"
}, 
file_name: "tc_converge_nas_basic_delete_01.py",
language: "python",
```

Part2 Import库信息和测试用例类名

```
dependencies: "import re
from UniAutos.Util import
from ConvergeService.lib import
from ConvergeService.lib import
from ConvergeService.lib.
from ConvergeService.nas_obs_m
",
class_name: "tc_converge_nas_basic_delete_01(Co
Base)",
```

Part3 测试用例上下文

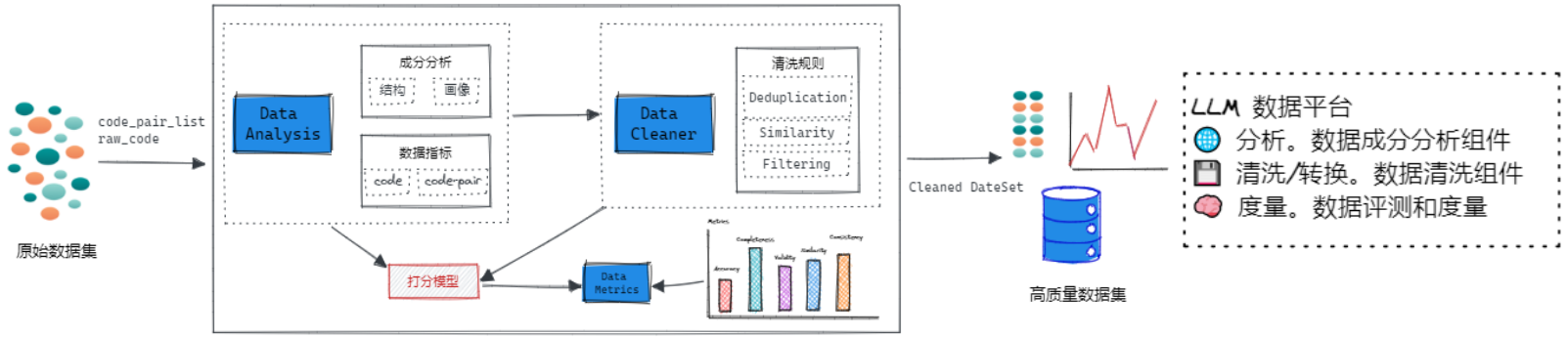
```
tc_info: 
{
    test_case_name: "NAS 侧删除文件",
    test_case_number:    "tc_converge_nas_basic_delete_01",
    test_case_type: "Function test",
    test_activity: "",
    test_feature: "",
    test_environment_type: "",
    pre_condition: 
            "1. 系
            2. 创建
            3. 创建
            4. 对接
            5. nas
    test_step:
            "1、NA
            2、对
            个普通文件
    expected_result
            "1、创
```

Part4 测试步骤文本脚本pair

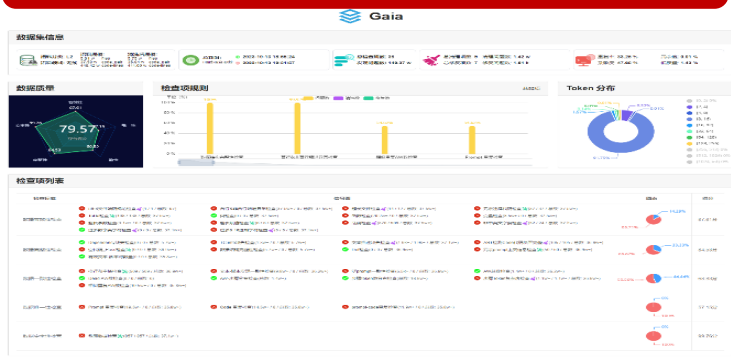
```
code_pair_list: [
{
    prompt: "self.logStep('步骤 1、
    目录并下放一个普通文件')",
    code: "
        file_name = 'nas_file_name'
        nas_path = self.dirPathV3[0] + '/' + file_name
        nas_attr = NasIOB
        )"+
```

数据工程: 测试代码LLM语料数据检查和清洗配置工具-Gaia

数据质量维度5性 共53项语料规则自动检查，19项语料规则自动清洗



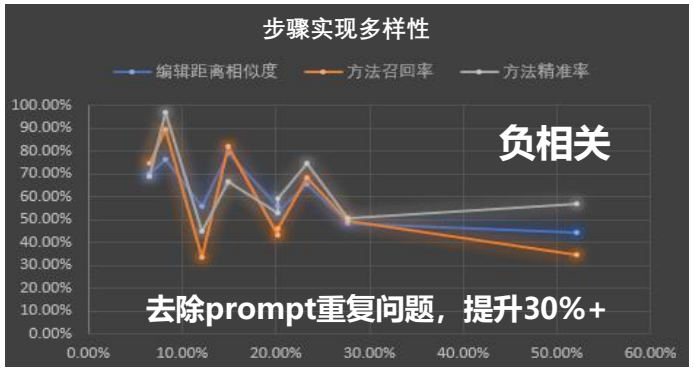
数量质量雷达

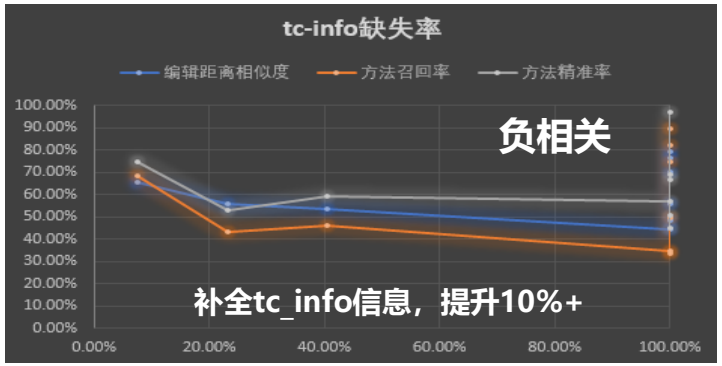


检出数据语料问题XXX万级

产品线	检出问题数	关键问题
数存 分布式/昊天/闪存/智能协作/数据管理/客户化集成	28/36, 100w+	✓ tc_info缺失：问题率 27%+ ✓ 缩进异常：问题率99%+ ✓ prompt重复：重复率54%+
无线 DIS/SRAN/LTE TDD/5G PDU/天馈/TDD JJFA/HERT/FDD JJFA	21/36, 1w+	✓ tc_info缺失：问题率 100.00% ✓ 缩进异常：问题率11.51% ✓ 冗余代码注释：问题率0.86%
云核 分组核心网/云核心网CSIMS/云核心网软件平台部/融合视频产品部	19/36，90w+	✓ tc_info缺失：问题率 100%+ ✓ prompt重复：重复率41%+ ✓ 超长参数检查问题：问题数13%+
华为云 存储	18/36, 3w+	✓ tc_info缺失：问题率 100.00% ✓ 泛prompt一致性：问题率15% ✓ prompt重复：重复率53%+
光 FTTx	16/36，6w+	✓ prompt重复：重复率55%+ ✓ 缩进异常：问题率0.29% ✓ 步骤Token复杂度检查: 问题率0.31%
计算 CANN&Pytorch&Mindstudio/MindX/kunpeng_dpu/kunpeng_boostkit/CC Solution/Solution/云与硬件开发部	10/36，6w+	✓ tc_info：问题率 80%+ ✓ 相似重复AW数检查：重复率90%+ ✓ 文本-脚本实现一致性：问题率5%

数据质量对结果的影响关联

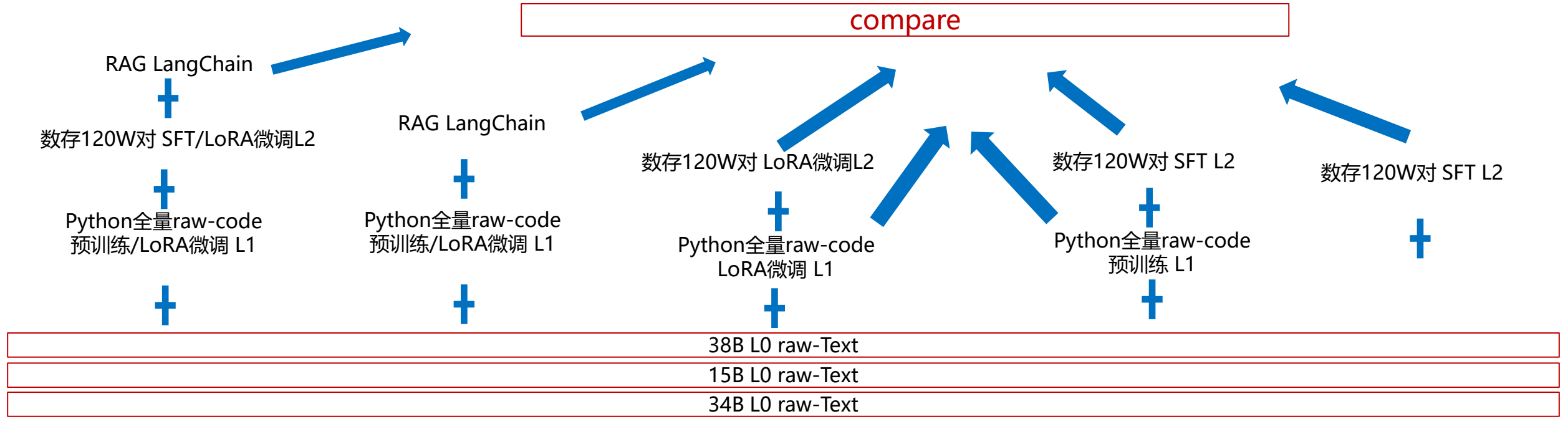




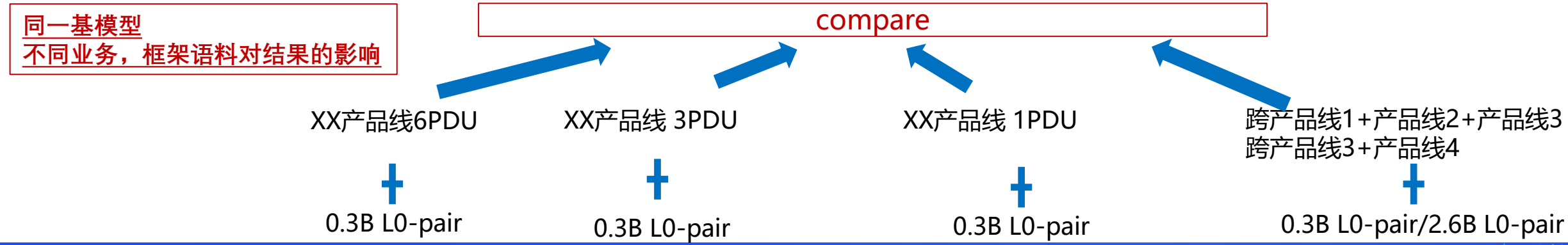
提示工程-prompt元模型

模型训练和推理对比思路

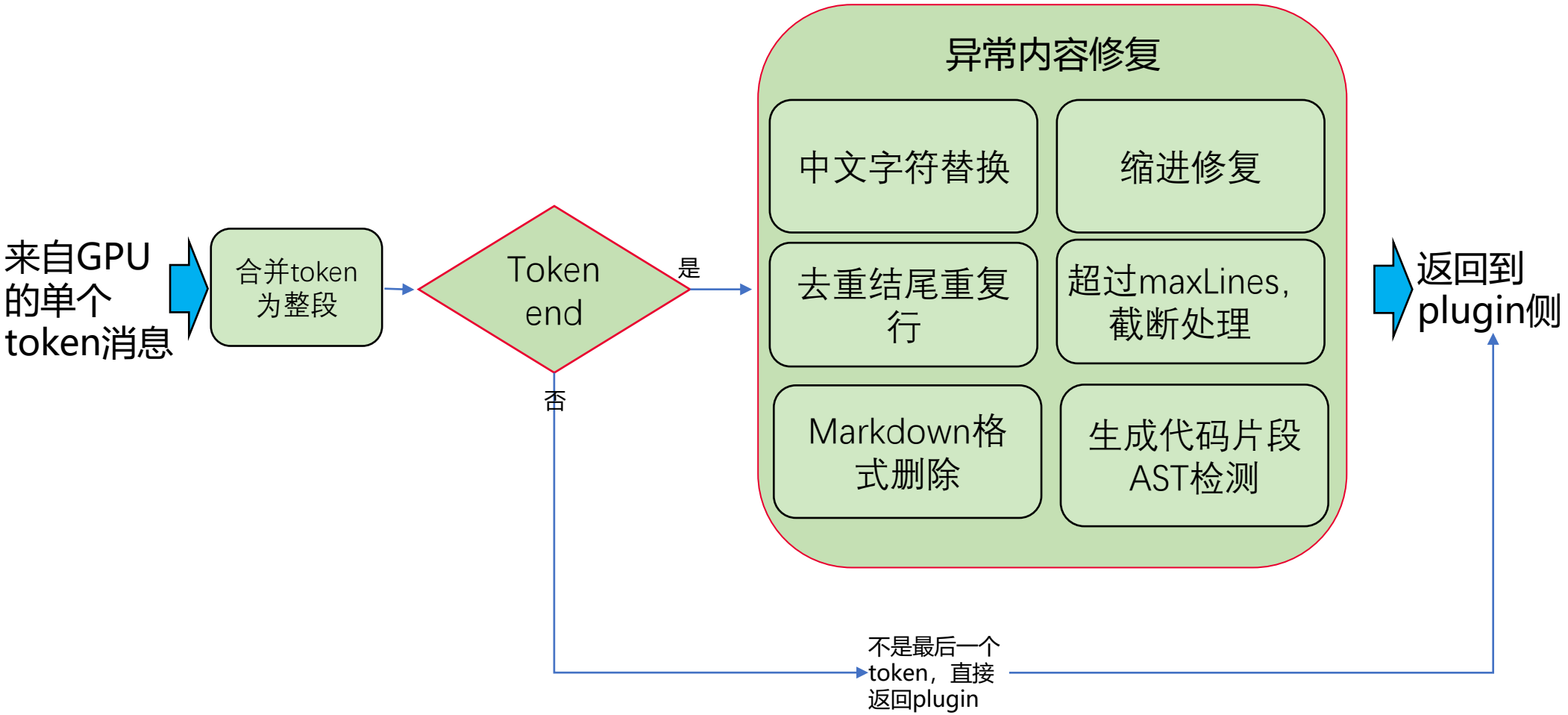
训练手法+基模型参数量对结果的影响



同一基模型
不同业务，框架语料对结果的影响



后处理


来自GPU的单个token消息
合并token为整段
Token end
是
否
异常内容修复
中文字符替换
缩进修复
去重结尾重复行
超过maxLines，截断处理
Markdown格式删除
生成代码片段AST检测
返回到plugin侧
不是最后一个token，直接返回plugin

评测：分层评估指标映射

评估阶段		上线前 模型级验证	运行态IDE 在线反馈评估	运行态代码提交后 离线评估
指标类型		客观	主观+客观	客观
客观指标项1	文本相似度	√		√
客观指标项2.1	测试方法召回率	√		√
客观指标项3.1	测试参数召回率	√		√
客观指标项4.1	参数赋值召回率	√		√
客观指标项2.2	测试方法精准率	√		√
客观指标项3.2	测试参数精准率	√		√
客观指标项4.2	参数赋值精准率	√		√
客观指标项5	代码留存率		√	√
主观指标项1	采纳率		√	
主观指标2	日活/活跃用户数		√	

评测工程方法-客观度量指标

8项客观指标计算公式

客观指标项1 –代码的文本相似度

$$\frac{\text{莱温斯坦距离}\{\text{生成的代码}string,\ \text{真实代码}string\}}{maxlength(\text{生成的代码}string,\ \text{真实代码}string)}$$

客观指标项5 - 行级代码留存率

$$\text{代码行级留存率}=\frac{\text{生成的}line\cap\text{真实}(\text{最终上库})line}{\text{生成的}line}$$

客观指标项2.1 -测试方法召回率(越高代表生成越全)

$$\frac{\text{生成的方法}\cap\text{真实代码中的方法数}}{\text{真实代码中的方法数}}$$

客观指标项2.2 -测试方法精准率（越高，代表误生成越少）

$$\frac{\text{生成的方法}\cap\text{真实代码中的方法数}}{\text{生成代码中的方法总数}}$$

客观指标项3.1 -测试参数召回率(越高代表生成越全)

$$\frac{(\text{生成代码方法}==\text{真实代码方法})\text{and}\ (\text{生成方法中的参数}p\cap\text{真实方法中的参数}p)}{\text{真实代码中的参数}p\text{数量}}$$

客观指标项3.2 -测试参数精准率 （越高，代表误生成越少）

$$\frac{(\text{生成代码方法}==\text{真实代码方法})\text{and}\ (\text{生成方法中的参数}p\cap\text{真实方法中的参数}p)}{\text{生成代码中的参数}p\text{数量}}$$

客观指标项4.1 -测试参数赋值召回率(越高代表生成越全)

$$\frac{(\text{生成代码方法}==\text{真实代码方法})\text{and}\ (\text{生成方法中的参数值}(v)\cap\text{真实方法中的参数值}(v))}{\text{真实代码中的参数赋值}v\text{数量}}$$

客观指标项4.2 -测试参数赋值精准率（越高，代表误生成越少）

$$\frac{(\text{生成代码方法}==\text{真实代码方法})\text{and}\ (\text{生成方法中的参数值}(v)\cap\text{真实方法中的参数值}(v))}{\text{生成代码中的参数赋值}v\text{数量}}$$

特定业务领域验证集收集指南

场景1 测试脚本代码生成

按每个特性（Tcinfo中的TestFeature）抽取验证集
保证评测集中样本的特性分布与全量脚本对一致

↓

从每个特性中取0.5% 测试脚本，
建议保证每特性至少有一个验证用例样例
样本之间保证一定diverse,包括长度diverse、内容diverse
避免重复

↓

从验证用例样本中提取文本
-脚本pair验证集

Raw-code

```
        with TestStep("1、用例准备、发送背景信号"):
            # 用例隔离
            TimeManager.wait_with_msg(10, "用例隔离，
等待样机回到初始状态")
            for _ in range(3):

self.can.send_can_once(self.case_data.can_open_adas)

self.can.send_can_cycle(self.case_data.can_navigation
_signal)
            testsuit.background_on()
            Log.info("发送导航状态背景CAN信号")

self.some_ip.set_json_data(self.case_data.some_ip_nav
igation_signal)

self.some_ip.start(self.case_data.some_ip_id)
            Log.info("发送导航状态背景ETH信号")
```

验证数据pair样例

```
prompt : TestStep('1、用例准备、发送背景信号')

code :      TimeManager.wait_with_msg(10, '用例隔离，等待样机
回到初始状态')
            for _ in range(3):

            self.can.send_can_once(self.case_data.can_open_adas)

            self.can.send_can_cycle(self.case_data.can_navigatio
            n_signal)
            testsuit.background_on()
            Log.info('发送导航状态背景CAN信号')

            self.some_ip.set_json_data(self.case_data.some_ip_na
            vigation_signal)
            self.some_ip.start(self.case_data.some_ip_id)
            Log.info('发送导航状态背景ETH信号')
```

场景3 测试方法AW库代码生成

分层提供AW评测集
覆盖BAW/CAW/仪表等环境配套驱动层AW 多层AW

↓

根据业务真实续写推演时应用的AW变更概率和频率
分配不同层级AW的验证级比例，
例如IF BAW变更更频繁，量更大，则BAW层验证集比例也更高

↓

从验证AW样本中提取AW定义签名+注释-AW代码对

prompt

```
def send_inspection_alarm(self, job_id, location, detail, title="", monitor_feature_link=None,
                          notice_mode="default", add_cc_to=None):
    """
    发送巡检产生的告警，回填到arsenal平台
    Args:
        job_id: arsenal调度传过来的id
        location (str): 告警定位信息
        detail   (str): 告警详情
        title    (str): 告警标题
        add_cc_to (str): 抄送人工号，多个以,分割
        notice_mode (str): 通知模式，配合增加抄送人字段使用，default：按任务默认的通知人抄送，contains：新增a
抄送add_cc_to用户
        monitor_feature_link: 增加抄送人，配合通知模式字段确定抄送人，格式：以,（英文逗号）分割用户
    Returns:
    """
```

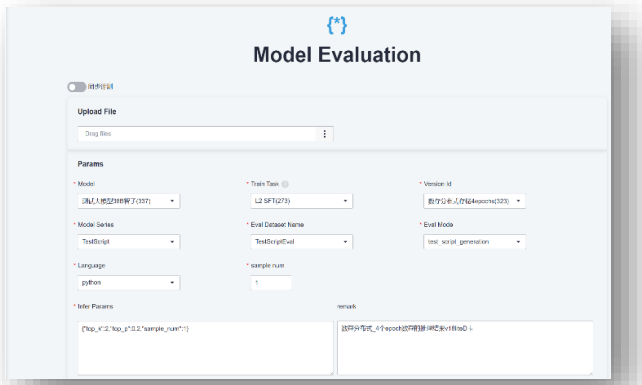
answer

```
    # 告警通知的接口url地址
    url = 'https://taasin.huawei.com/arsenal/api/v1/alarms/campnou_alarm'
    # 发送告警通知请求
    params = {'job_id': job_id, 'alarm_item': location, 'alarm_details': detail, "notice_mode": notice_mode}
    if monitor_feature_link:
        params["monitor_feature_link"] = monitor_feature_link
    if add_cc_to:
        params["add_cc_to"] = add_cc_to
    if title:
        params["monitor_title"] = title
    response = requests.post(url, params=params, verify=False)
    self.log_info(
        "the rsp for arsenal alarm is status_code={} body={}".format(response.status_code, response.text))
    return response.ok
```

华为全公司 *python*测试代码 验证集 1万样本

ModelEvaluation自动评测工具

创建评测任务



- 模型CKPT版本
- 任务类型
- 评测数据集
- 评估模型

评测验证集推理

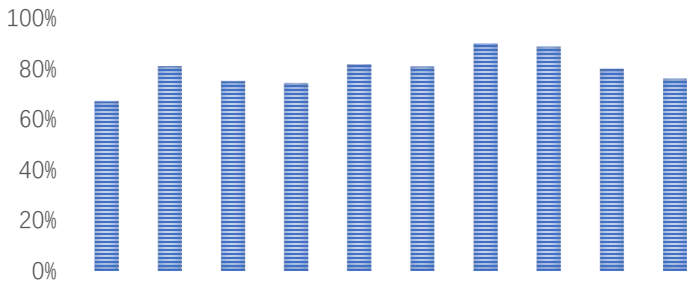
- 模型上线前离线推理

8项评测客观指标自动计算

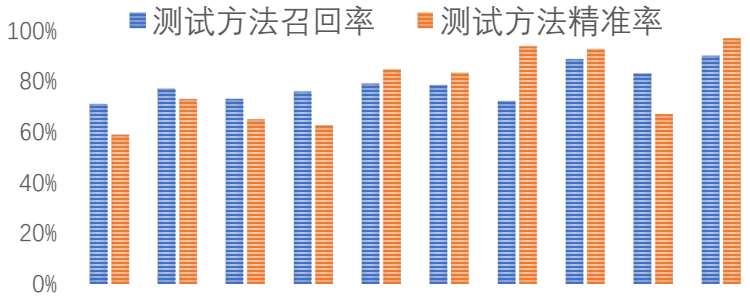
对比评测样本推理结果 vs GroundTruth

数据存储、无线、云核心网部分先锋试点产品客观评测结果

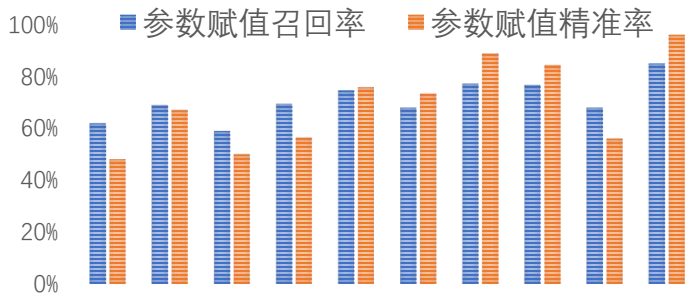
生成代码相似度60%+



生成方法召回率65%+，精准率59%+

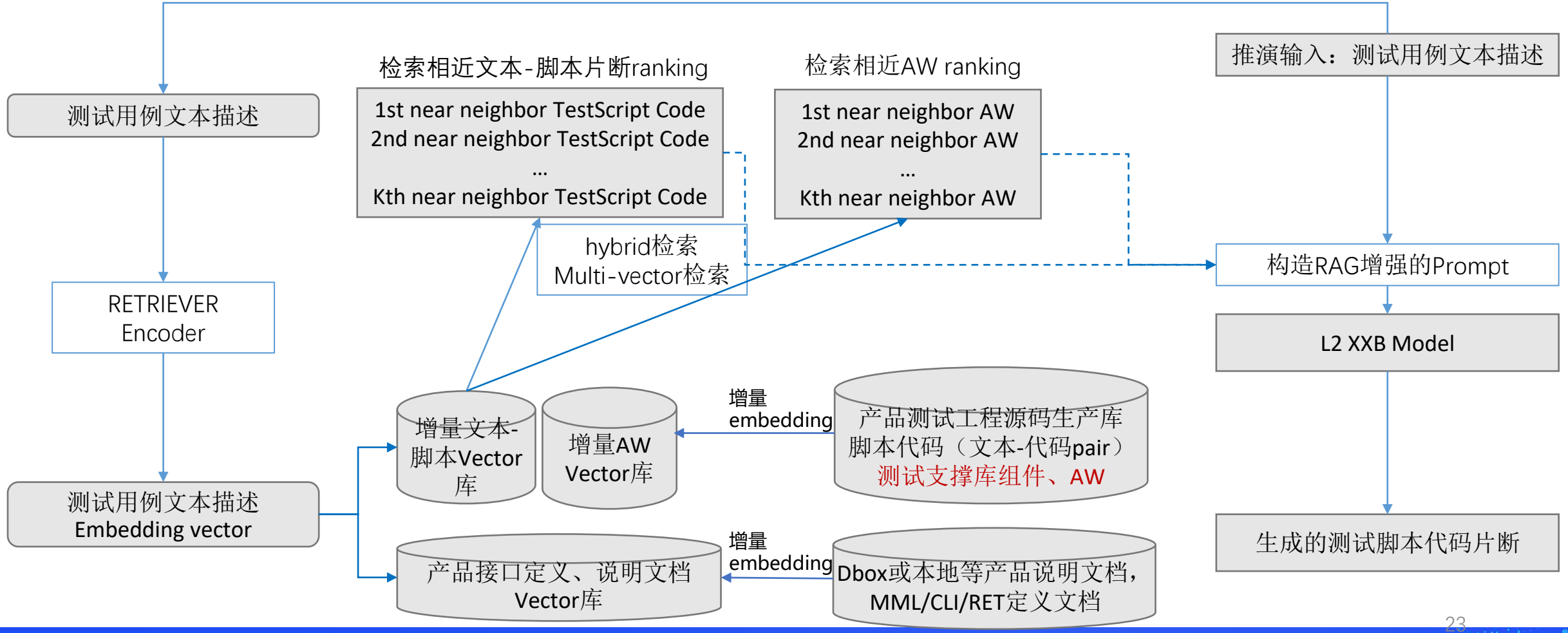


参数赋值召回率59%+，精准率48%+



Retrieve Augmented Generation （RAG）增强生成推理

1. 解决新增的文本-脚本对无法即时 Feed L2 SFT的问题
2. 解决利用新增的AW辅助生成脚本的问题
3. 一定程度上缓解跨产品数据相互干扰

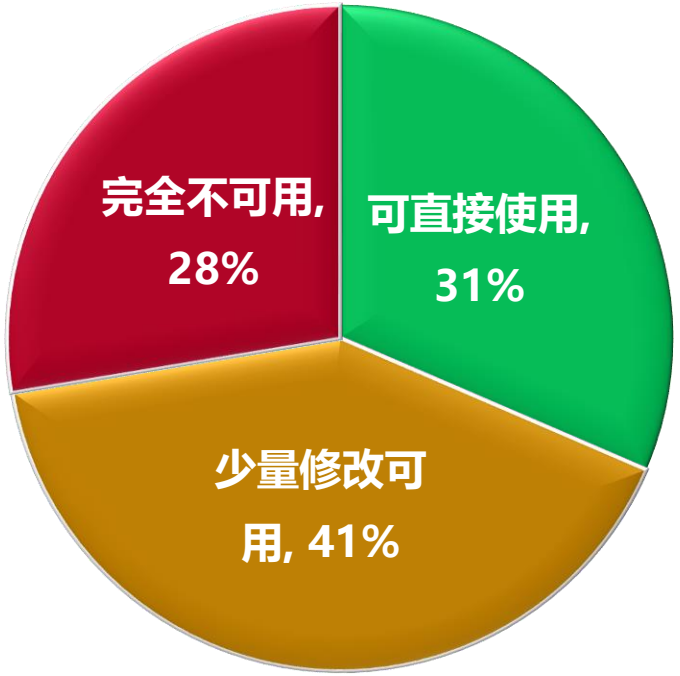


PART 04
华为ICT产品实践效果

无线产品线生产环境试点效果—生成代码主观接纳度~70%

- 无线所有使用python的部门均进入生产环境试点阶段，推理算力受限，白名单开放试点
- 新开发72个用例/861个步骤，主观指标统计： 31%直接可用、41%少量修改可用，28%不可用

业务采纳的分布比例



首批应用范围：

- **8个**自动化工程（所有试用python部门）
- **40+**不同业务专题
- **30+**业务人员
- **72个**新增用例、**861个**步骤

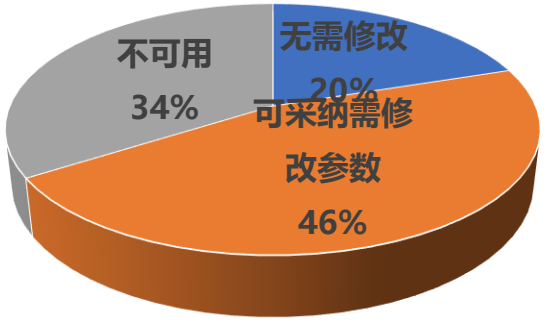
数存产品线试点效果-采纳率66%左右

- 客观评测指标低，不代表推理代码的可用度低；客观评测指标高，代码的可用度一定高。

				客观指标1	客观指标2		客观指标3		客观指标4	
评测日期	产品线	PDU	模型	编辑距离相似度	方法召回率	Adj.方法精准率	参数名召回率	Adj.参数名精准率	参数赋值召回率	Adj.参数精准率
20230906	数存		38B 4epochs L2	63.52	58.96	63.18	62.31	61.36	53.15	52.69
20230913	数存		38B 8epochs L2	75.98	80.1	86.14	78.55	83.31	73.46	78.24
20230920	数存		ChatGLM 5epochs	63.64	59.03	72.16	56.06	64.81	49.84	59.21
20230920	数存		ChatGLM 10epochs	70.13	65	79.74	63.65	74.87	58.64	69.64
20230920	数存		StarCoder 5epochs	40.76	36.01	31.19	29.69	21.96	21.49	20.06
20231019	数存		38B 4epochs	64.901	68.1774	58.575	68.2986	53.9847	59.7744	48.1475
20231019	数存		38B 4epochs	67.3672	70.6525	59.2841	70.7707	56.5088	62.1176	48.0257
20231019	数存		38B 4epochs	20.1277	52.4269	11.9209	54.6777	12.1476	39.9436	9.1286
20231019	数存		38B 4epochs	19.0621	47.6696	10.5811	45.8071	10.1403	36.9061	8.229
20231026	数存		4epochs	53.0736	47.0563	47.5413	47.1668	46.3418	37.9271	37.6814
20231026	数存		4epochs	57.6027	55.863	51.8456	59.2112	49.2042	45.5114	39.7295
20231026	数存		4epochs	54.4534	48.8347	46.0331	49.9272	45.3538	37.2603	33.7063
20231026	数存		4epochs	59.1969	51.7851	50.2068	54.5398	49.0506	40.598	38.1783

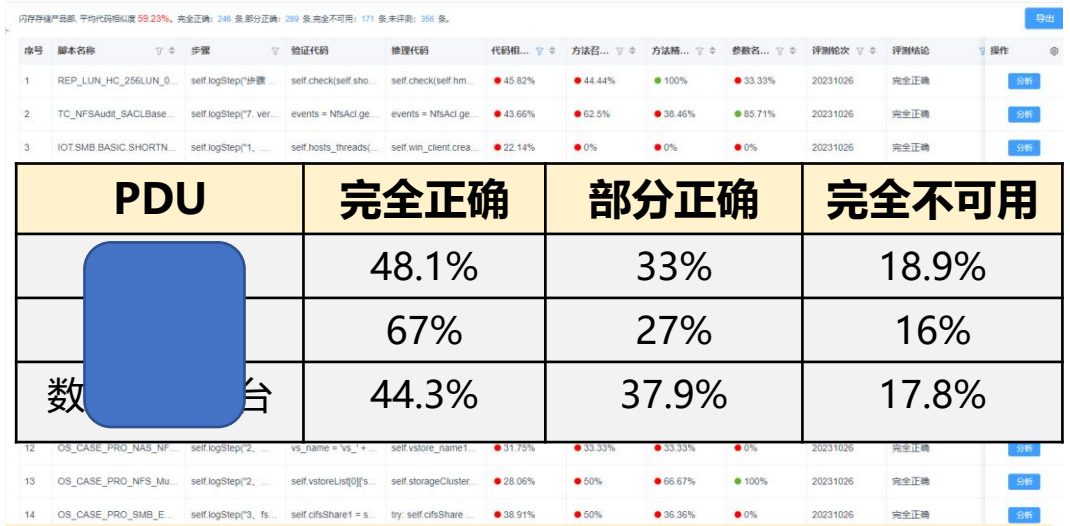
数存分布式存储试用主观评测

新生成200个用例，1800+步骤，采纳率**60%+**，整体脚本写作效率提升**1倍**



抽取其中代码高可用度的68个脚本，采用离线推理结合插件推理，以大步骤为单位，可用占比68.1%，以小步骤为单位，可用占比83.77%

在线化的客观评测平台，一键计算7大客观评测指标，查看相似度，测试方法、测试参数及参数赋值的召回率和准确率评测结果

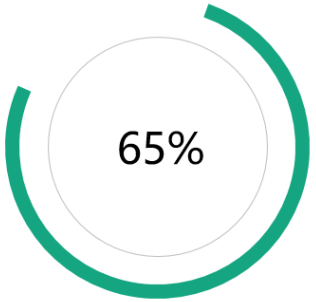


PDU	完全正确	部分正确	完全不可用
	48.1%	33%	18.9%
	67%	27%	16%
数 台	44.3%	37.9%	17.8%

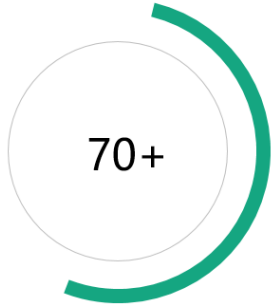
主观与客观评测结果的差异，原因在于语料的多样性导致

云核产品线试点效果-采纳率65%左右

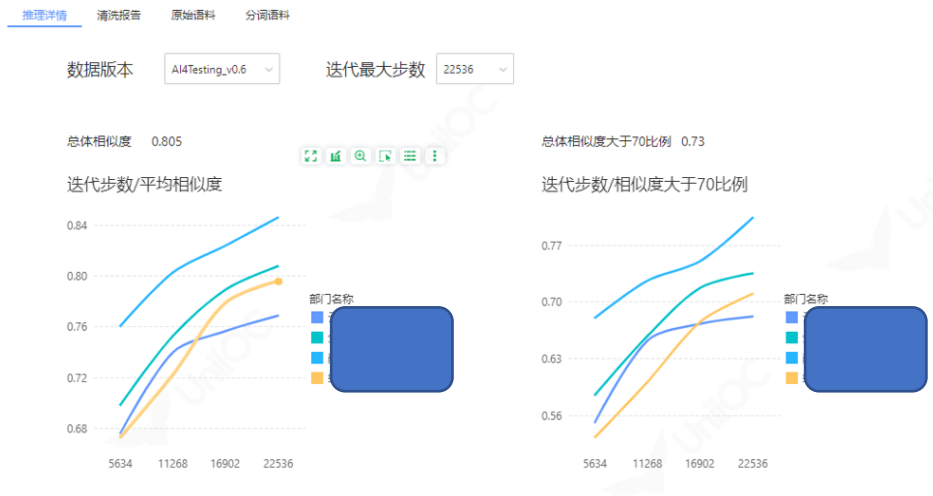
生成代码平均相似度



代码可接纳率



插件试用人数



部门名称	网元	特性名称	用例总数	用例步骤数量	直接可用	少量修改可用	不可用	接纳率（(直接+少量修改)/总数）
[redacted]			13	118	60	29	29	75%
[redacted]			17	195	59	69	67	67%
[redacted]			7	39	7	22	20	74%
[redacted]			7	60	7	19	34	43%
Overall			44	412	133	139	150	65%

经验教训：数据质量对效果影响Top1

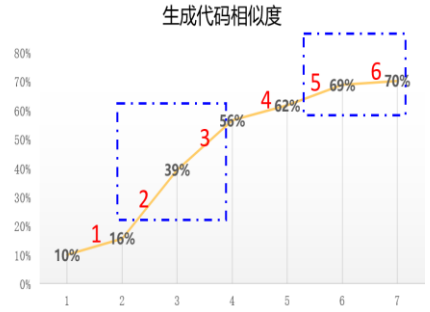
识别影响模型效果的数据语料主要3大特征：1）prompt工程中测试用例TCinfo的完备性 2）文本-脚本 实现一致性 3）测试AW密度

无线FDD JJFA数据质量和prompt信息完整性对结果的影响

清除异常数据+补全TC info信息
可大幅提升准确性

文本生成代码@starcoder试验效果—FDD解决方案

对结果有明显效果的措施：1、数据质量清洗；2、生成代码的后处理



1、增加训练次数:3epochs->10个epochs
2、数据清洗@3epochs：**清除异常数据（空代码等）**
3、完善prompt数据@3epochs：**完善数据集的Tcinfo信息**
4、增加训练次数:3epochs->10个epochs
5、增加后处理，**修复异常缩进**
6、增加后处理，**修复部分语法错误**

主要是提升了精准率和召回率

部门	验证样本Pair数	有效样本Pair数	验证代码缩进错误Pair数	验证代码其它语法错误Pair数	推理代码缩进错误Pair数	推理代码其它语法错误Pair数	编辑距离相似度	方法召回率	方法精准率	参数名召回率	参数名精准率	参数赋值召回率	参数赋值精准率
FD[redacted]案	383	383	0	0	2	3	76.39	74.47	68.57	64.81	54.97	64.80	62.71
[redacted]	185	161	24	0	12	0	63.08	61.02	57.17	50.44	50.23	49.40	45.99
TD[redacted]案	148	148	0	0	4	1	40.93	22.80	34.64	44.94	32.47	14.48	27.10
TD[redacted]案（工具[redacted]洗后）	236	235	1	0	0	2	75.67	68.71	61.85	68.45	55.65	62.33	55.69
[redacted]	259	259	0	0	3	2	34.83	19.16	21.68	8.13	11.92	10.36	12.55
H[redacted]（工具[redacted]洗后）	427	427	0	0	1	0	75.32	71.90	73.84	71.40	71.34	68.82	67.05

数通软件平台AW密度对准确度影响

单测试步骤中AW密度越大
推荐准确率越低

NAAS	AW密度（评测集AW/步骤）	准确率
去重	5.4	
不去重	8	0.18

云杉	AW密度（评测集AW/步骤）	准确率
去重	4.1	
不去重	6.9	0.34

CE1800V	AW密度（评测集AW/步骤）	准确率
去重	2.8	
不去重	4.7	0.71

云核分析数据特性对结果的影响

用例文本信息越具体，用例脚本实现一致性越高，生成结果越好

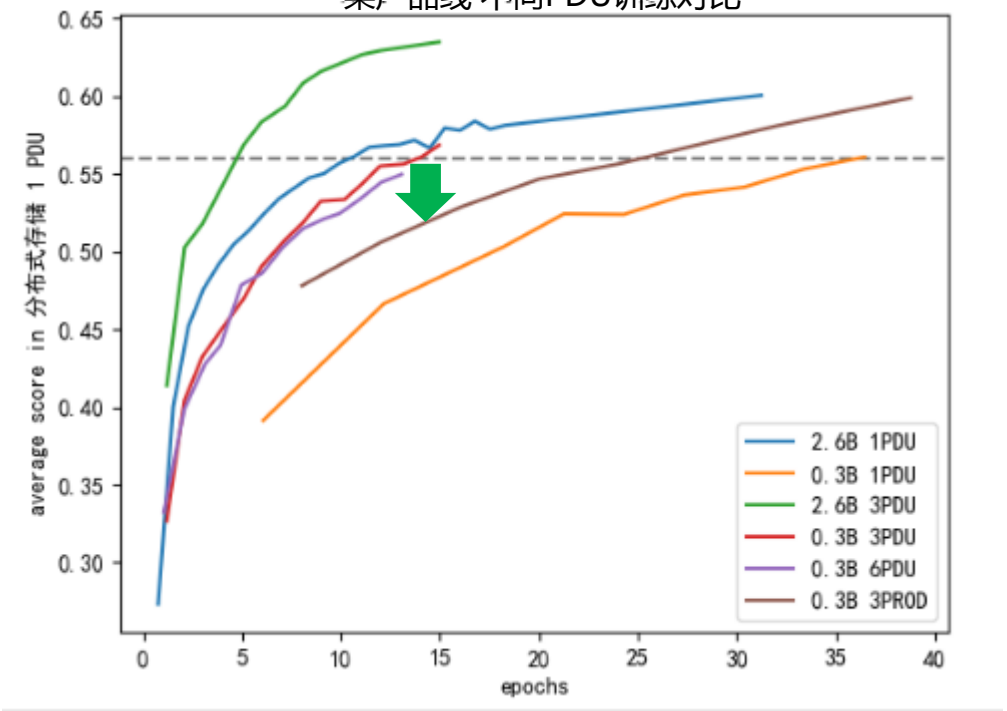
数据去重清洗或标注保留有效训练数据样本，保证测试脚本实现一致性，可提升生成效果

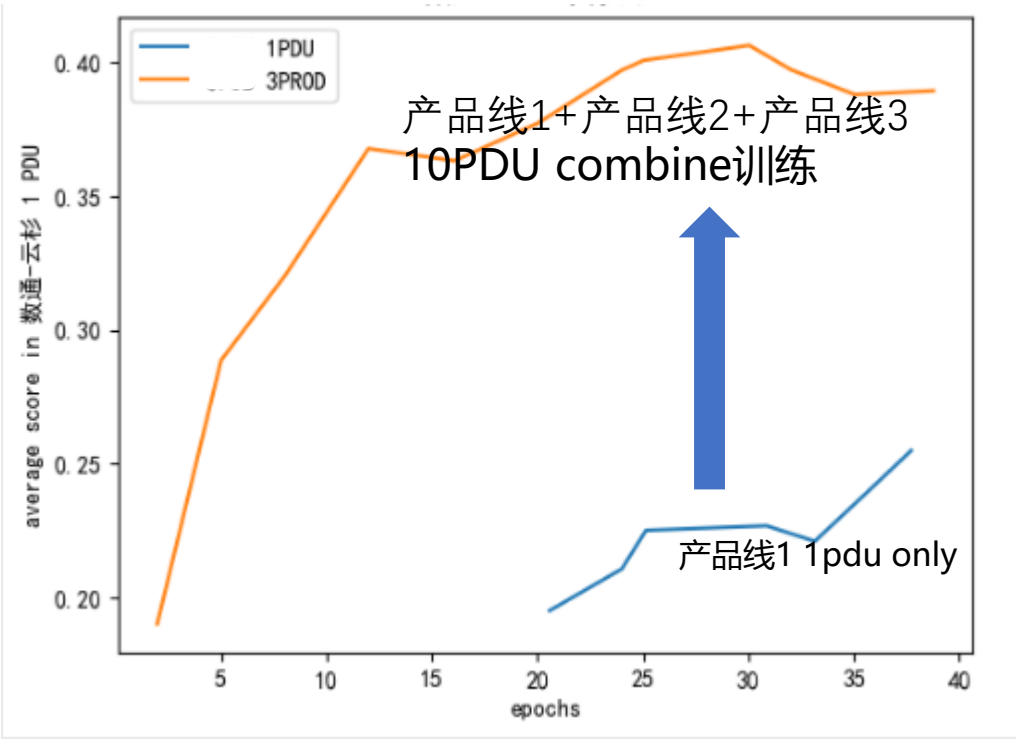
网元样本的数量占比与生成效果关系不大，测试脚本代码按特性相对独立

经验教训：不同产品线的数据一起训是否会相互影响——有可能

产品线1 only VS 产品线1+产品线2+产品线3：相似度提升20%
产品线2 only VS产品线1+产品线2+产品线3：相似度略下降4%
产品线4与产品线3 共享同一软件平台部分特性，但测试代码和方法库不共享，会概率性出现推出其它产品线或PDU测试代码

某产品线 不同PDU训练对比

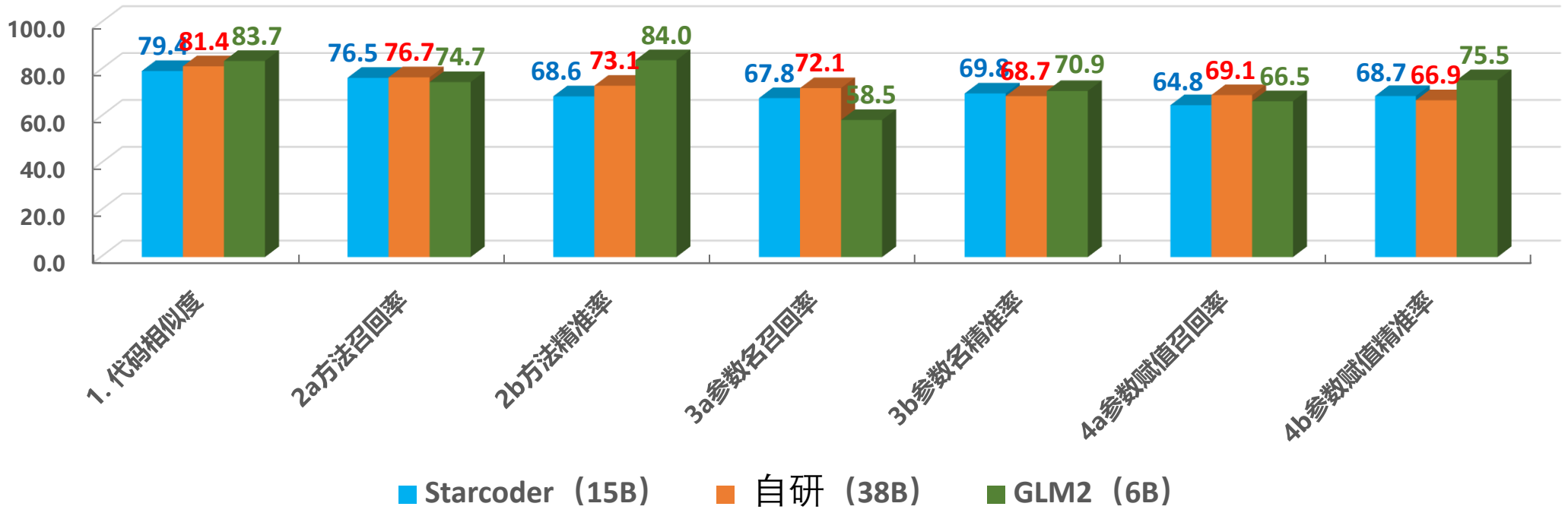




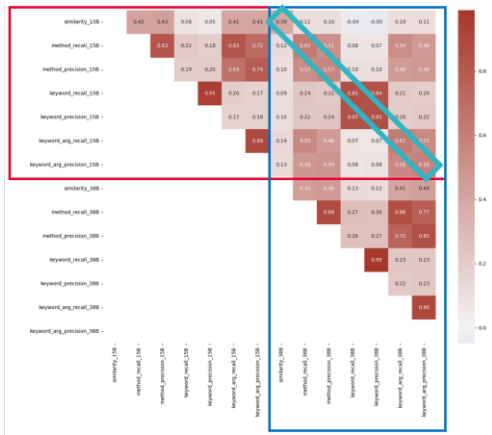
大体经验总结：业务有相近，框架和库不同的产品不要放在一个训练pipeline实例里训

不同基模型对效果的影响：XXB L2 文本生成脚本XX产品线靶场对比结果

结论1 ： XB~XXB模型统计结果差异不大



2模型个体结果相关度55%~60%



结论2：个体结果仍存在不少差异

场景2：全新特性/业务功能的系统测试代码生成

AI for Test automation 全码应用场景

AW设计&编写（工作量占15%） → AW调试（工作量占15%） → 测试脚本编写（工作量占30%） → 测试脚本调试（工作量占40%）

场景3：LLM辅助生成AW代码

场景1：防护网补全&特性增强开发场景

场景2（主流场景）：新特性、新业务开发场景

全新功能/特性系统测试代码生成验证集验证效果，SFT+RAG优于RAG only

		ID	验证样本数	有效样本数	验证代码缩进错误数	验证代码其它语法错误数	推理代码缩进错误数	推理代码其它语法错误数	代码相似度	方法召回率	方法精准率	参数名召回率	参数名精准率	参数赋值召回率	参数赋值精准率
增量场验证集 DPC（全新特性，未参加SFT）	RAG Only	78186afaa853499c9b3384267f85c17e	40	40	0	0	0	5	39.8251	17.9104	18.421	11.9048	20.1923	13.4454	16.0919
	SFT Only	03661845adaf48bcbe0f28b63eabc30c	41	41	0	0	0	0	22.1533	4.3478	1.5625	3.125	3.7736	1.6393	1.0363
	SFT + RAG	03b01ef7fd4c4cfbba7a33abb73fe8e4	41	41	0	0	0	2	69.0463	78.2609	55.8325	67.1875	56.4171	67.2131	50

新旧混合验证集验证效果，SFT+RAG优于RAG only和SFT only

		ID	验证样本数	有效样本数	验证代码缩进错误数	验证代码其它语法错误数	推理代码缩进错误数	推理代码其它语法错误数	代码相似度	方法召回率	方法精准率	参数名召回率	参数名精准率	参数赋值召回率	参数赋值精准率
增量验证集 新旧特性混合	RAG Only	84a07340fa4c40a5af0b2a85dce74c5d	211	211	0	0	4	9	51.916	47.9042	48.5897	42.2188	53.5663	37.6471	39.5631
	SFT Only	a40219efafad460494fb39949fa5753b	211	211	0	0	0	4	42.0676	40.5689	34.7988	28.3513	30.1356	23.2941	21.9985
	SFT + RAG	652a4aeb8ef441768f33d591add81a60	211	211	0	0	0	2	69.5627	80.6886	64.6357	72.265	58.8788	65.2941	50.1129

场景3：LLM辅助测试AW代码续写

- 当前LLM生成AW代码基础指标较差：AW是小众代码，基础模型语料少，内部语料样本数也比较少，预期不比开发生成代码难度低

AI for Test automation 全码应用场景

AW设计&编写（工作量占15%） → AW调试（工作量占15%） → 测试脚本编写（工作量占30%） → 测试脚本调试（工作量占40%）

场景3：LLM辅助生成AW代码

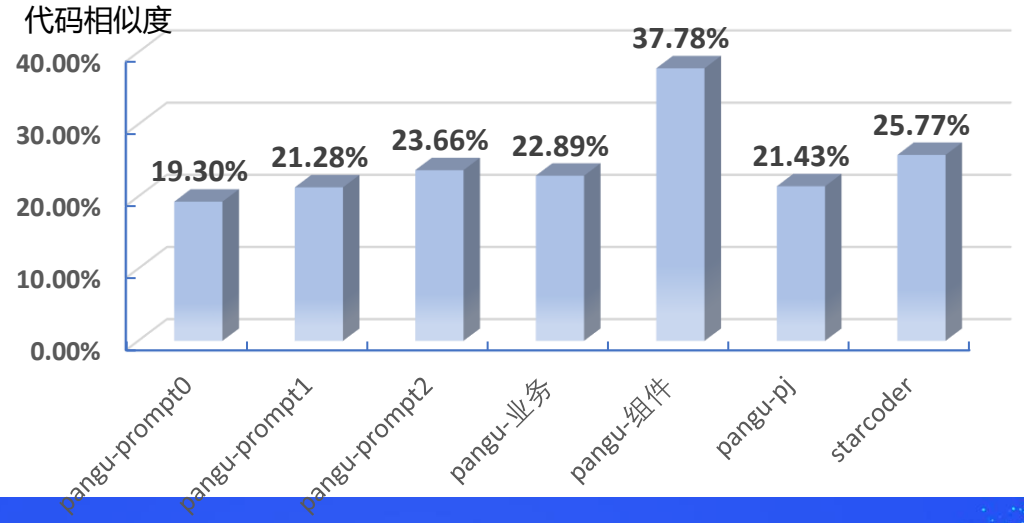
场景1：防护网补全&特性增强开发场景

场景2（主流场景）：新特性、新业务开发场景

主力产品自动化开发工作量分布：

	AW设计&编写	AW调试	脚本编写	脚本调试
产品A	30%	20%	10%	40%
产品B	10%	20%	20%	50%
产品C	30%	20%	10%	40%
产品D	25%	25%	20%	30%
平均	**24%**	**21%**	**15%**	**40%**

尝试LLM续写AW代码效果：



PART 04
总结与展望

总结&future work

QA1 什么因素对模型推理结果影响最大?

首先是调优语料数据质量，其次是prompt上下文完备性，再次是基模型，最后是训练手法

QA2 pretrain,SFT, RAG选哪种?

代码生成类需生成可编译，可执行对象类任务，SFT+RAG或许最佳

QA3 测试人员需做哪些转型?

习惯与测试智能代理共事，学会prompt大法，具备大模型推理结果数据溯源debug能力

Future Work

- 辅助系统测试调试
- AW续写效果提升
- 反馈利用

AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

THANKS