

同一套系统，两个场合



演示 Demo

一切顺利 · 老板点头，同事叫好

✓ Demo 日

上线第一周

开始出各种怪问题

上线第一周 ✗

今天这期做的是一次事故复盘

× ~~不讲：怎么写 Prompt~~

× ~~不讲：怎么选模型~~

✓ **要讲：一次事故复盘**

现场 ① 公开榜单

现场 ② 连着跑

现场 ③ 连报错都没有

翻车的原因，多半不在模型身上

《Agentic AI 工程五讲》

第 1 讲

翻车现场挑了三个，一个比一个隐蔽

第一个，在一张公开榜单上

现场 ①
公开榜单上

现场 ②

现场 ③

?

隐蔽程度  越往右越隐蔽

BIRD

Text-to-SQL 领域的权威评测 —— 考的是把自然语言问题翻译成数据库查询

自然语言问题

「上个月华东区
销售额多少」



转译

那条正确的 SQL

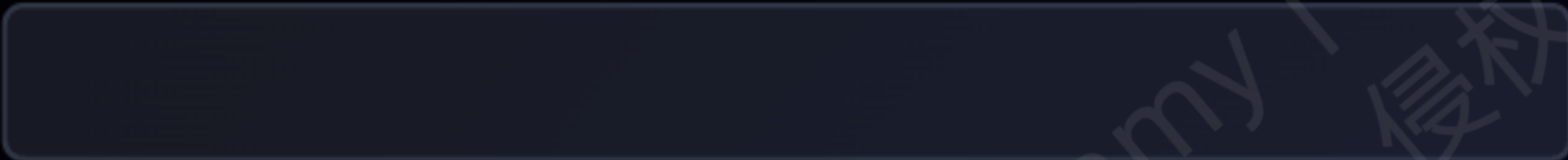
```
SELECT SUM(amount) FROM sales WHERE region = '华东'  
AND month = '2026-07';
```

直觉：这件事纯拼模型能力 —— 模型越强，SQL 写得越好？

BIRD · Text-to-SQL 榜单

同一张卷子 · 同一类任务

#1



#2



#3



小米：模型 + 一整套工程闭环

⋮

中间还有二十几名

#30+



纯模型选手（哪怕用的是最前沿的模型）

差距不在模型本身，在模型外面包的那套东西上

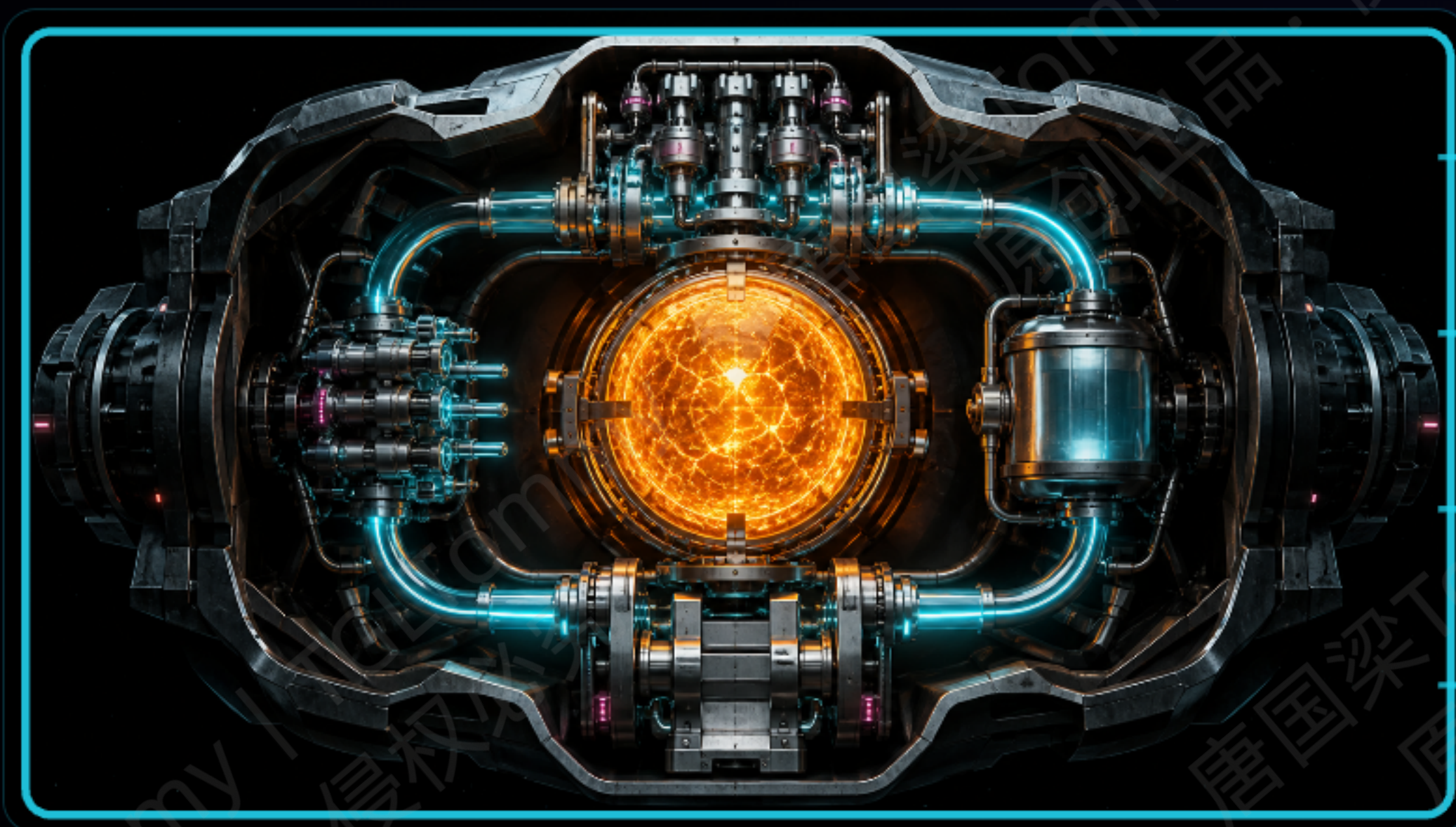
小米包的那套 →

① 生成之后有校验

② 多路互相比对

③ 答不准主动拒答

这套东西现在有个通用的名称 **Harness**



包住模型的那层工程外壳

这期后面会专门讲它到底管哪几件事 →

? 状态怎么保存?

? 工具怎么调用?

? 结果怎么验证?

? 出问题怎么恢复?

小米具体包了些什么，这个系列后面会讲到

这里先记住现象就够了

SWE-bench · 编程评测

同一个模型 · 一个字没动



42 → 78 分 (接近翻倍)

cyan 段 = 换一套执行 Harness 涨出来的 +36 分

amber 段 = 模型本身, 与左柱等高同色

换的只是它干活时的环境和流程

两个不同领域，同一个信号

Text-to-SQL (BIRD)

工程闭环 → 第3名

编程 (SWE-bench)

换 Harness → +36 分

决定成绩的，除了模型，
还有包在模型外面的那层工程

比直觉里以为的大得多

第二个现场：藏在「成功率」这个词的歧义里

tau-bench · 模拟客服任务

现场 ①

公开榜单上

现场 ②

连着跑八遍

现场 ③

隐蔽程度



越往右越隐蔽

读法一：跑一次能不能成

衡量的是模型能力

读法二：连着跑能不能都成

衡量的是生产系统

	1	2	3	4	5	6	7	8	八遍全过?
任务 A	✓	✓	✓	✓	✓	✓	✓	✓	✓
任务 B	✓	✓	✗	✓	✓	✓	✓	✓	✗
任务 C	✓	✗	✓	✓	✗	✓	✓	✓	✗
任务 D	✓	✓	✓	✓	✓	✓	✓	✓	✓
任务 E	✓	✓	✓	✗	✓	✓	✓	✓	✗



八道全开才算过 —— 任何一道卡住，前面七道白开

单次通过率 **61.2%** ↓ 掉了一多半 八遍全过 **25%**

同一个模型 · 同一批任务，只是把「跑一次」换成「连着跑」

同一个「成功率」，两个场合量的不是一回事



Demo（演示） 比的是「跑一次」

- ▶ 跑 1 次
- ▶ 可以重录十遍，剪最顺的一遍
- ▶ 失败的尝试观众永远看不到

生产（线上） 比的是「连着跑」

- ▶ 每天跑成千上万次
- ▶ 用户不会挑你成功的那次来用
- ▶ 任何一次失败都是真实的损失

两个数字，量的是两件事

落差里住着无数「Demo 很惊艳、上线就翻车」

61.2% ▲ 上限 ceiling · 单次成功率

翻车就住在这段落差里

25% ▼ 下限 floor · 重复稳定性

单次成功率 = 模型能力的上限

重复稳定性 = 生产系统的下限

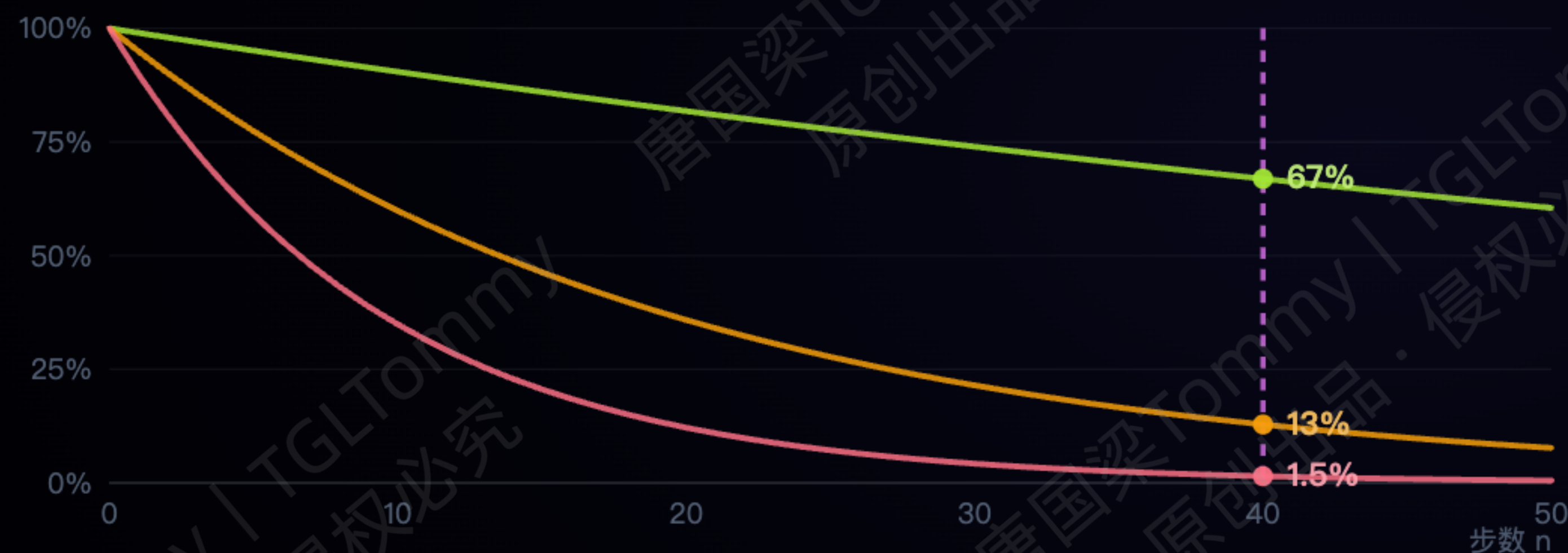
Agent 的「跑一次」，本身就不止一步



一个任务连着做几十个决定 —— 每一步都对，整条链才算对

中间错一步 → 整条链失效

单步再高，一路乘下来也会缩水得厉害



$$P_{\text{端到端}} = p_{\text{单步}}^n$$

单步 99% → 40 步后 67%

单步 95% → 40 步后 13%

单步 90% → 40 步后 1.5%

连续做决定本身的代价 · 跟模型聪不聪明关系不大

步子越长，缩水越狠

第三个现场最隐蔽 —— 它连报错都没有

腾讯 TEG · 数据平台团队

现场 ①

公开榜单上

现场 ②

连着跑八遍

现场 ③

连报错都没有

隐蔽程度



越往右越隐蔽

数据任务的失败，是静默的

一条 SQL 平稳运行、零报错、执行成功

```
SELECT SUM(amount) FROM sales WHERE ...
```

运行状态 **✓ 成功**

报错 **0 条**

执行耗时 **1.2s**

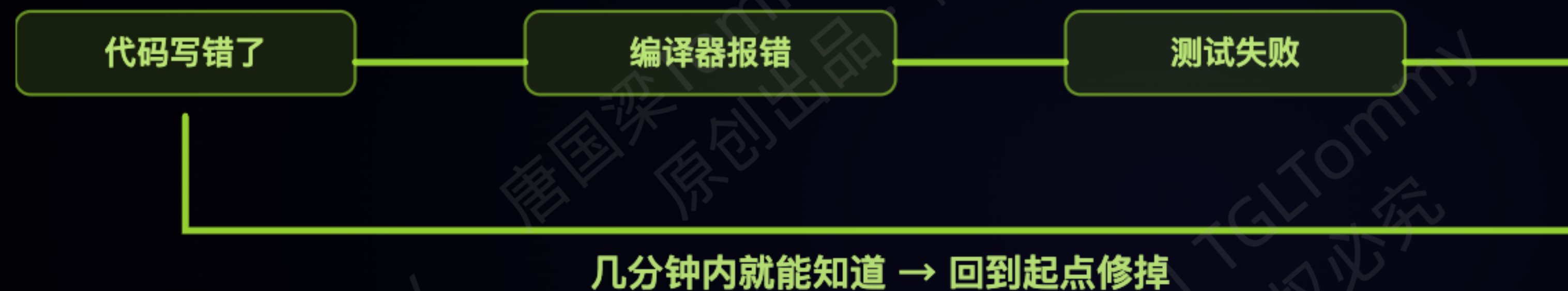
返回结果 **¥ 8,412,900** ← 这个数字是错的



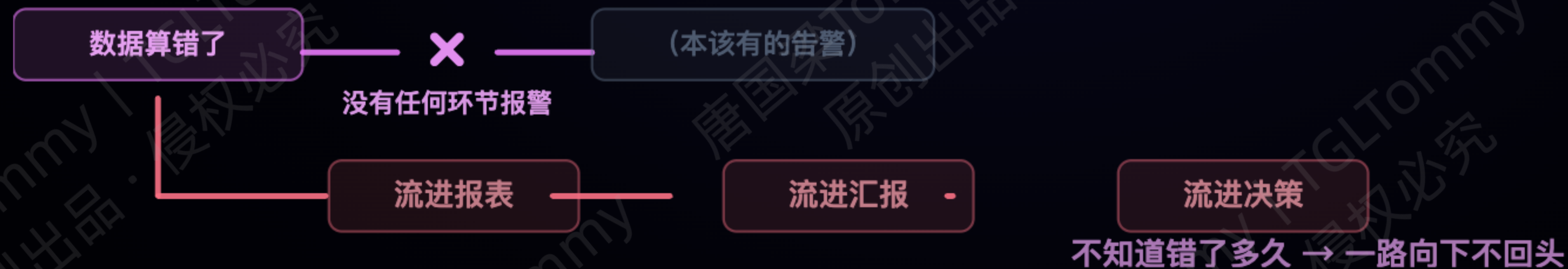
指示灯全绿，下面在无声渗漏 —— 没有任何告警

错误能不能被发现，看这个环能不能闭上

写代码：闭合的反馈回路



数据：断开的回路



同一个起点，两条命运线

崩溃（显式失败）—— 反而是好事

失败



触发告警



有人立刻来修



修好

静默失败 —— 没人修

失败



(无告警)



下游当成正确结果消费



继续扩散

报表 ▸ 汇报 ▸ 决策 ? 不知道错了多久、影响了多少判断

文档理解这边，是同一个故事

阿里 · DocScope 评测基准



答案是对的，但说不清从文档哪里得出来的

答案 ✓ 对的

引证线全部断在半空

文档里的出处

注意分母：是在「答对的那些回答」里再往下看

答对的那些回答

100% (分母)

↓ 再往里剖一层

里面再分

70% 给不出可信的证据链

30% 有据可查

换句话说：蒙对的

今天蒙对了 ✓ → 明天换一份文档？ → 没人知道

演示阶段的「质量」，是人眼免费提供的

演示阶段

输出 →

人眼裁判

盯着每一条输出



生产环境

输出 →

空·无人

没人盯着了



裁判席空了，错误就一点点积累起来



现场 ①
公开榜单上

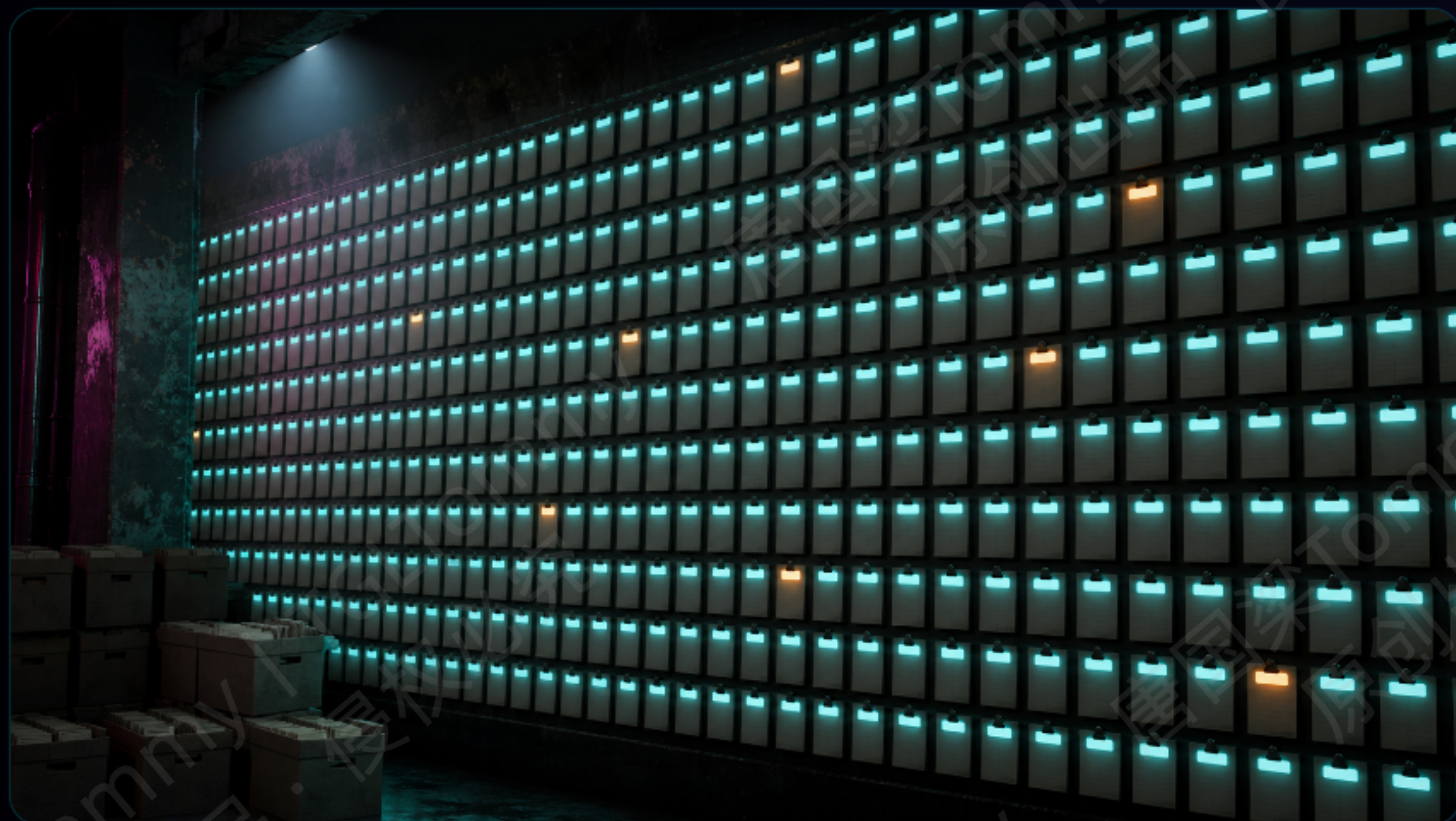
现场 ②
连着跑八遍

现场 ③
连报错都没有

你的监控大屏可以全绿，而业务已经错了

三个现场看完，可以做归因了

1688 团队 · 线上真实任务复盘



× 凭感觉猜

✓ 把线上真实任务拉出来，一个一个人工复盘

288

个任务 逐个人工复盘，不是抽样估算

每一个都追问同一个问题：
这个任务没做好，问题到底出在哪一环？

288 个失败任务的首因分布

71.6% 系统 & 基础设施

28.4%

其余（含模型说错话、理解跑偏）—— 反而不是大头

具体是这一类 →

Agent 压根没响应

调用额度出了问题

该交付的内容缺了一块

归因结果落在轴的哪一头？

61.2% ▲ 上限 · 模型能力

翻车就住在这段落差里

- ↘ 71.6% 首因 = 系统 & 基础设施
- ↘ Agent 没响应 / 额度 / 少交付
- ↗ 模型说错话 —— 不是大头

25% ▼ 下限 · 生产系统

问题最多的那一环，恰恰是我们最后才去查的那一环

线上一出问题，该按什么顺序查？

很多人的顺序

自上而下 ↓

① Prompt

④

② 模型能力

③

③ 数据

②

④ 系统 & 基础设施

71.6% 首因在这一层

①

正确的顺序

自下而上 ↑

先查系统 → 再查数据 → 然后才看模型能力 → 最后才轮到 Prompt

把顺序搞反的代价很具体



在屋顶刷漆，而裂缝在地基

✗ 你花好几天打磨提示词

→ 而真正的问题是某个接口一直在超时

这个结论还有一层组织含义

团队精力分布（很多公司的现状）

优化模型 / 写提示词

后端 / 基础设施

很少参与

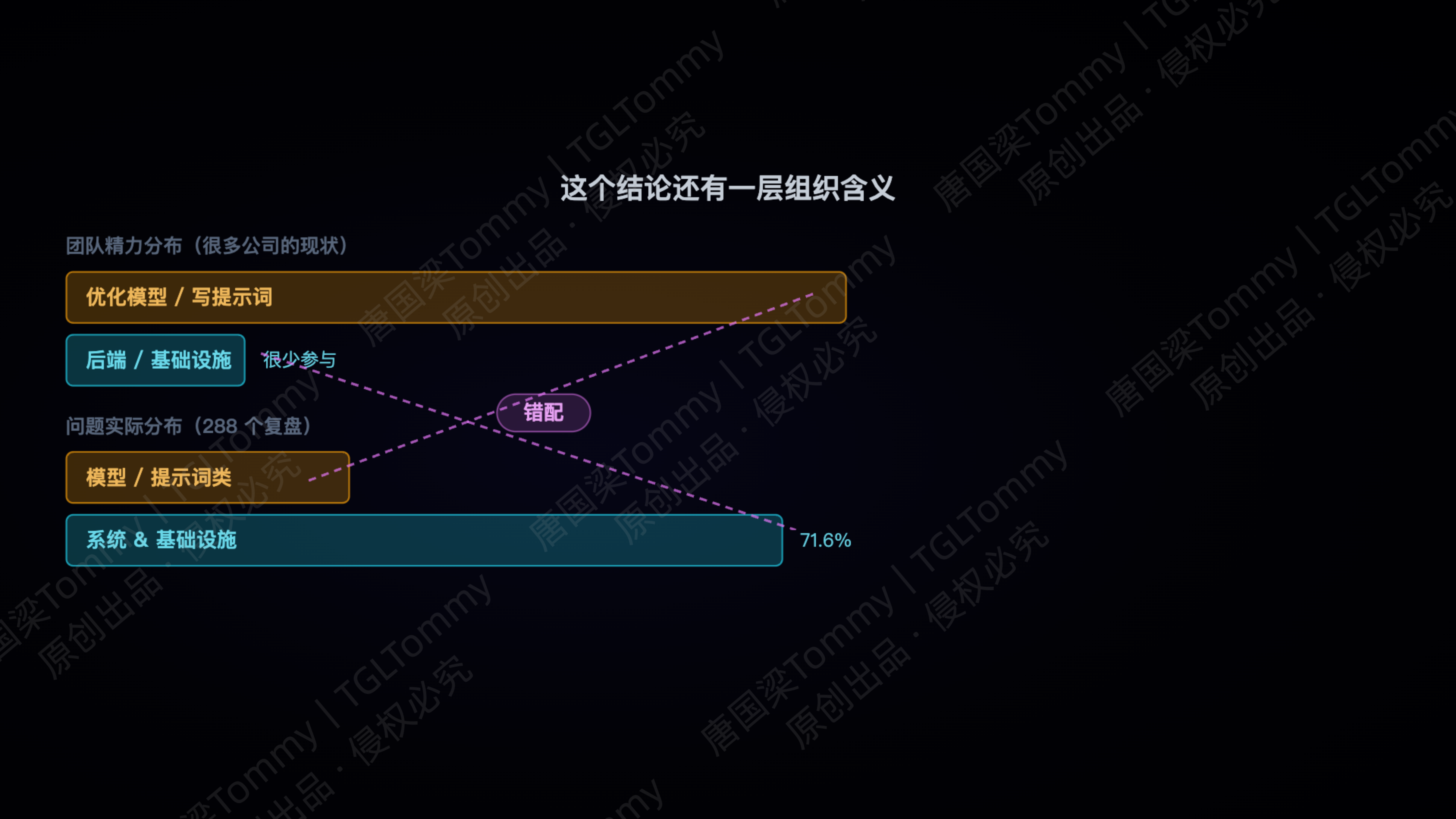
问题实际分布（288 个复盘）

模型 / 提示词类

系统 & 基础设施

71.6%

错配



精力在这头



× 错配

问题在那头 (71.6%)

它首先是一个系统工程

要的是把这些环节做扎实的工程能力 →

状态

调用

验证

不是说 Prompt 不重要，是说它

排在队尾

三个现场，可以用一个框架串起来

Agent =

Model

+

Harness

最聪明的时候能做到什么（上限）

最差的时候会差成什么样（下限）

整个系列最重要的一个等式

现场 ① 公开榜单上

现场 ② 连着跑八遍

现场 ③ 连报错都没有



都是 Harness 这一侧的事



Agent =

Model

+

Harness

最聪明的时候能做到什么（上限）

最差的时候会差成什么样（下限）

下面这四件事，全在 Harness 这一侧

☰ ① 状态怎么保存

从断点接着跑 / 还是一切重来

↺ ② 工具怎么调用

失败了怎么重试 / 重试会不会做两遍

✓ ③ 结果怎么验证

确认是真完成了 / 还是模型自己说完成了

不可信 → 质量那期

↺ ④ 出了问题怎么恢复

回到一个干净的状态 / 还是带着半截烂摊子跑

模型和 Harness 的分工，一句话

生产环境真正在乎的，是下限

61.2% ▲ 上限 ceiling

← Model 决定

翻车就住在这段落差里

↘ 小米第3 = 工程闭环把下限抬起来

↘ SWE-bench +36 分 = 涨的全是下限的分

↗ 61.2% 是上限，25% 才是下限

25% ▼ 下限 floor

← Harness 决定

★ 生产比的是这条

演示比的是上限

生产比的是下限

—— 同一套东西在两个场合表现割裂的根本原因

这个等式还有一个很实际的用法：下次立项的时候

常见立项

换更强模型



补工程



▲ 上限

顶天花板

▼ 下限

✗ 给上限加码，而你真正缺的是下限

应该的做法

换更强模型



补工程



▲ 上限

垫高地板

▼ 下限

✓ 把下限抬起来

⌚ 等下一代模型来救场

vs

⚡ 先把手里这套系统的工程补齐 —— 见效更快，也便宜得多

把 Harness 这层再往下看 —— 从 Demo 走到生产，一共有九道坎要过



九道坎

前面那些现场，其实都是这九道坎里的

这个系列只挑最要命的几道，一道一道讲透

01 可靠性

从单次成功到重复稳定

已见过

02 验证

谁来判定任务真完成了

已见过

03 权限

它能碰什么、不能碰什么

04 语义

业务黑话怎么翻译给它

05 状态

断了怎么接上

已见过

06 评测

怎么持续知道它做得好不好

07 成本

算力账单撑不撑得住

08 运营

喂给它的知识会过期

09 组织

个人提效怎么变团队提效

每一道坎背后都有真实团队踩过的坑

有些坑的形态相当出人意料

权限那道坎

以为 把安全规则写进提示词就行

实际 根本拦不住任何东西



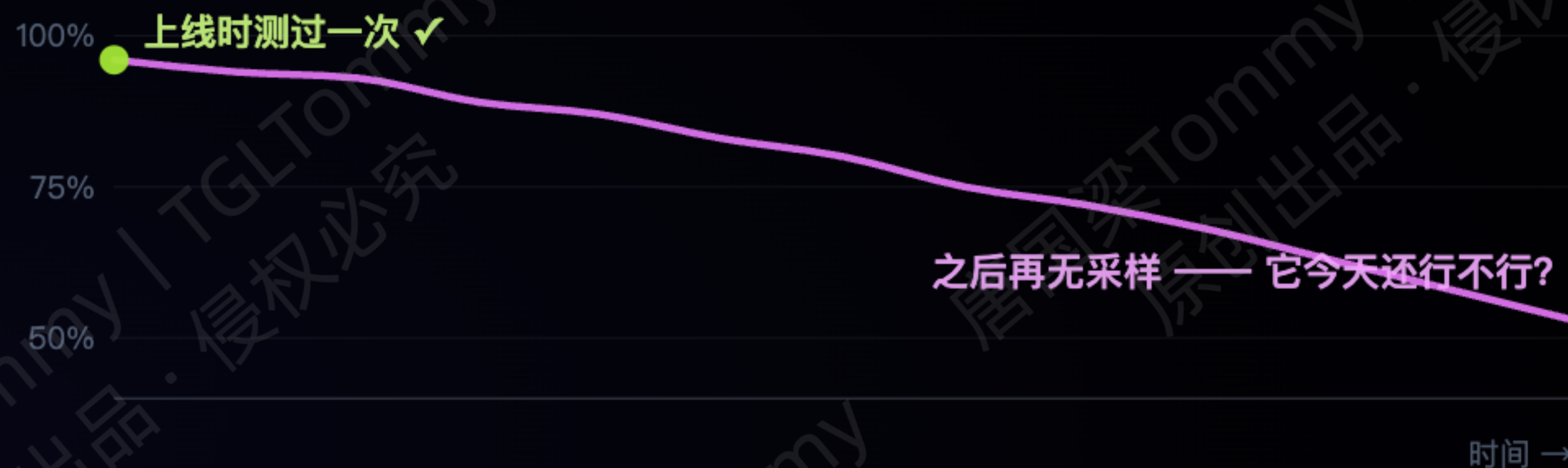
便签贴在柜门上，柜内电源与网线照旧可插拔

评测 ≠ 上线前跑一遍

以为 上线前跑一遍测试集就算评测了

实际 能力会随数据、知识和外部系统的变化持续漂移

评测那道坎



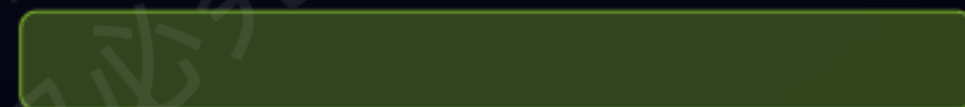
评测不做成一件持续的事，你根本不知道它今天还行不行

运营那道坎：喂给它的知识会过期



精心维护的知识库，
停止维护一个月

刚建好



停维护 1 个
月



↓ 准确率明显往下掉

九道坎不会全讲

只挑最要命的几道，一道一道讲透

接下来几期，沿三条主线走

① 执行

让 AI 只负责【想】，不负责【跑】

高频执行 → 交给确定性的系统

对应：可靠性 · 成本

② 安全

让 Agent【想犯错也做不到】

× 靠 Prompt 里那句「请不要删库」

对应：权限

③ 质量和数据

怎么【证明】它真的做对了

怎么把公司的知识，喂成它能用的形态

对应：验证 · 评测 · 语义 · 运营

① 执行

让 AI 只负责【想】，不负责【跑】

高频执行 → 交给确定性的系统

对应：可靠性 · 成本

② 安全

让 Agent【想犯错也做不到】

× 靠 Prompt 里那句「请不要删库」

对应：权限

③ 质量和数据

怎么【证明】它真的做对了

怎么把公司的知识，喂成它能用的形态

对应：验证 · 评测 · 语义 · 运营

三条主线，都是同一个 Harness 的三个切面

Agent =

Model

+

Harness

最聪明的时候能做到什么（上限）

最差的时候会差成什么样（下限）

下一期先走执行这条线

一家货运平台的真实难题



业务允许的时间



一票好货几分钟就被抢走

模型需要的时间



几十秒 + 还没个准数

x 业务等不起

x 模型快不了

这个问题怎么解?

✗ 业务等不起

一票好货几分钟被抢走

✗ 模型快不了

几十秒，还没个准数

他们给的解法就一句话

让 AI 当编译器，别当司机

这句话什么意思，我们下期继续

版权声明

唐国梁 Tommy 原创出品 | 熵智未来（深圳）科技有限公司版权所有。保留所有权利。

本课程及配套资料，包括但不限于课程视频、PDF 文档、课件、代码、图表、示例项目、知识库文档及其他学习材料，仅供购买本课程的用户本人学习使用。未经版权所有者书面许可，任何个人或机构不得以任何形式对本课程及配套资料进行录制、复制、传播、上传、公开分享、分发、转售、改编、翻译、汇编，或用于商业培训、企业内训、商业化产品、知识库建设、数据集构建、人工智能模型训练、微调及其他商业用途。如发现侵犯本课程及配套资料相关权益的行为，权利人有权依法要求停止侵害、删除下架、赔偿损失，并追究相应责任。

本课程及配套资料中涉及的第三方开源软件、模型、代码、数据集、商标、图片及其他素材，仅在相关许可证允许范围内进行教学展示，其相关权利归原权利人所有。

侵权举报 / 授权合作: legal@tgltommyai.com

微信公众号: 唐国梁TGLTommy | 官方网站: TGLTommy.com (国内访问需科学上网)