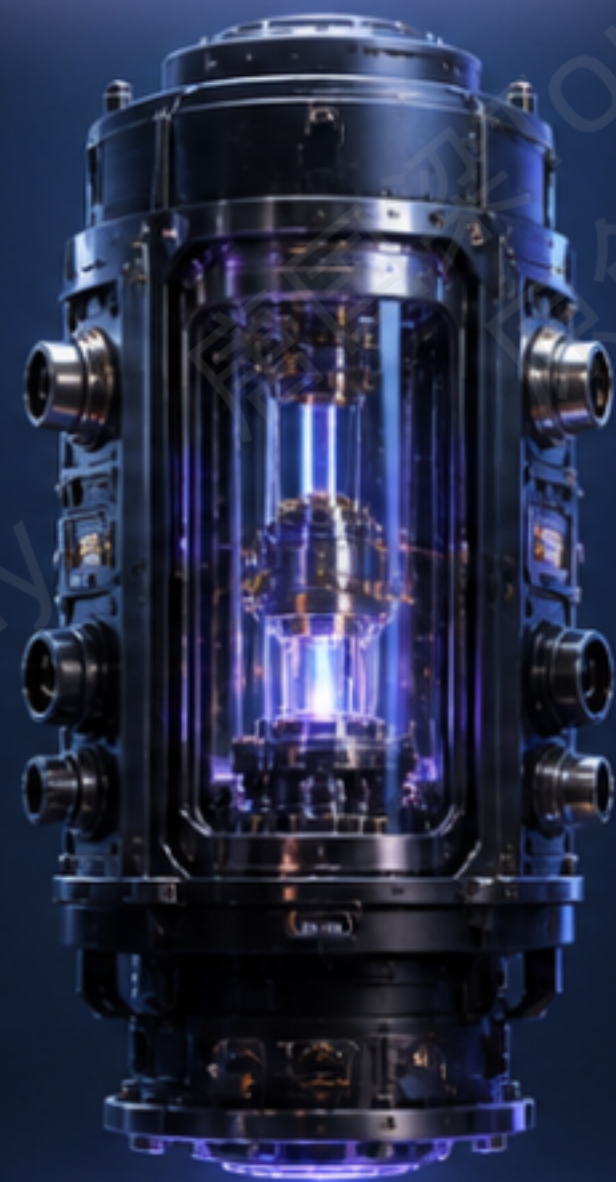


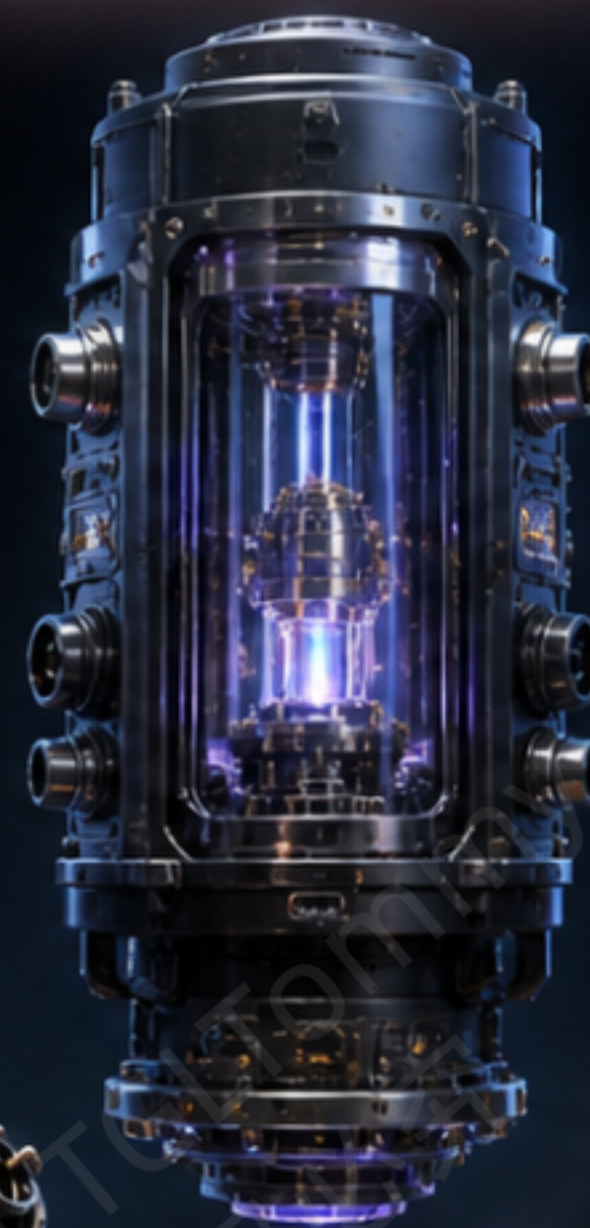
Demo ≠ 生产系统

B站/公众号/YouTube : 唐国梁Tommy
官网 TGLTommy.com

• Demo · 单机完美



• 生产 · 接口全对不上



Demo 里模型能拆任务、能调工具，看着已经很接近一个能干活的同事

接进企业系统

问题立刻换成另一批



① 它用的业务口径对不对？

② 会不会调到不该调的工具？

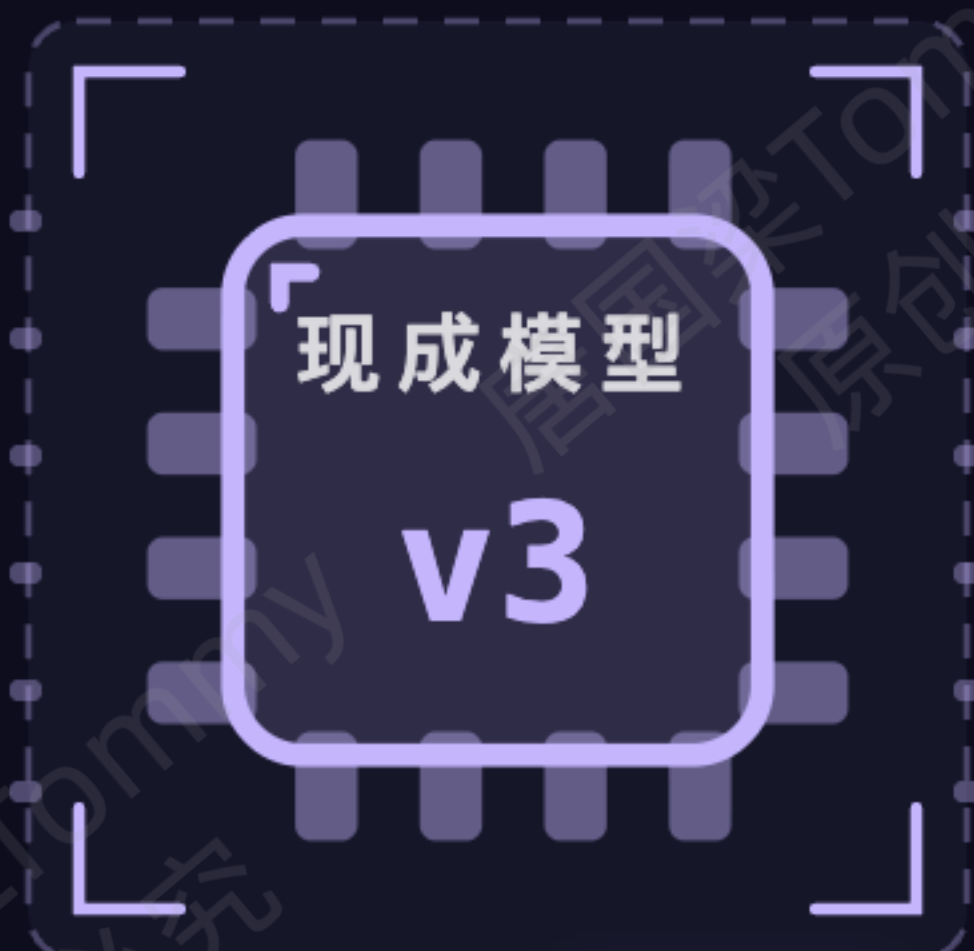
③ 算错了谁来拦？

④ 一次任务到底花多少钱？

⑤ 量大了还跑不跑得动？

这几个问题有个共同点

同一个插槽 · 换上更强的模型



v1

v2

v3 · 更强

接进企业系统之后要回答的五个问题

- ❌ 业务口径对不对
- ❌ 会不会调到不该调的工具
- ❌ 算错了谁来拦
- ❌ 一次任务花多少钱
- ❌ 量大了跑不跑得动

芯片越换越大 · 五个红叉一个都没消失

换一个更强的现成模型，基本都解决不了

公开的企业级实践 · 六家团队

腾讯

游戏数据分析 Agent

阿里

XiYan-SQL

百度

线上异常洞察

eBay

On-call 支持系统

火山引擎

Data Agent

Elastic

检索 MCP Server

六家的技术路线差得很远

提炼出五道边界



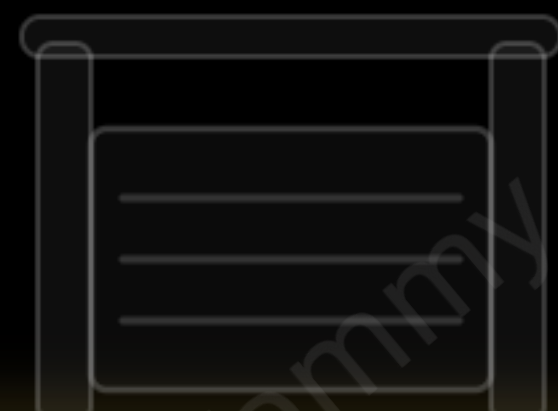
一份能直接拿去评审自己项目的清单

听完这一期，你会拿到它

Agent 真正落地

六家技术路线差得很远，但结论是同一个

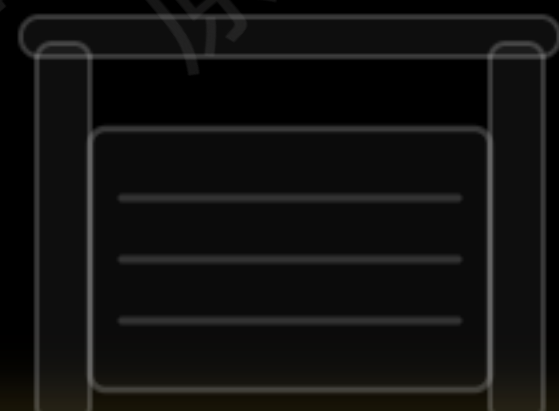
①



语义

懂不懂业务

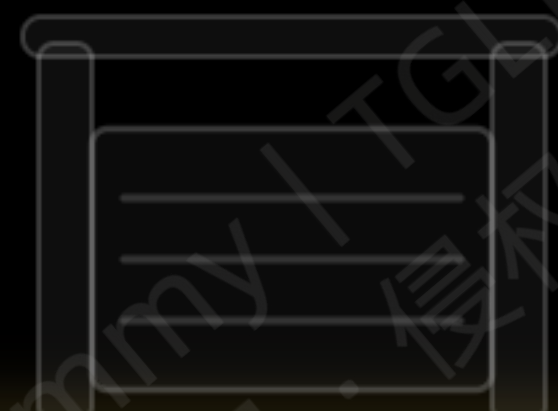
②



执行

该怎么行动

③



权限

能调到什么程度

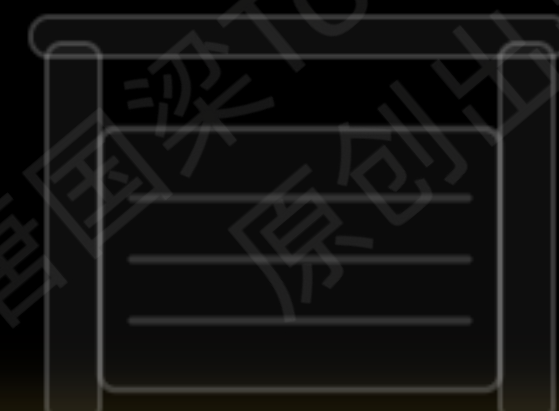
④



验证

结果可不可信

⑤



成本

规模化跑不跑得动

x 不是让模型更自由 | → 而是把模型放进一套受控的行动系统

受控系统的主干

一条线，就能说清楚



后面每讲一道边界，都会回到这张图

三条预算

权限 · 成本 · 时延



从头到尾压在整条链路上

五道边界

这条链路上，最容易出事的五个位置



下面一道一道讲

每道边界挑一个做得最透的系统展开

※ 再带上几种不同的做法



① 语义边界

能听懂，不等于懂业务

用户原话

「这个月销售额为什么跌了」

模型每个字都认识 · 字面全懂 ✓

?

销售额
用哪个指标

?

统计窗口
怎么定义

?

退款
算不算扣除

?

数据
分布在哪儿几张表

① 语义边界

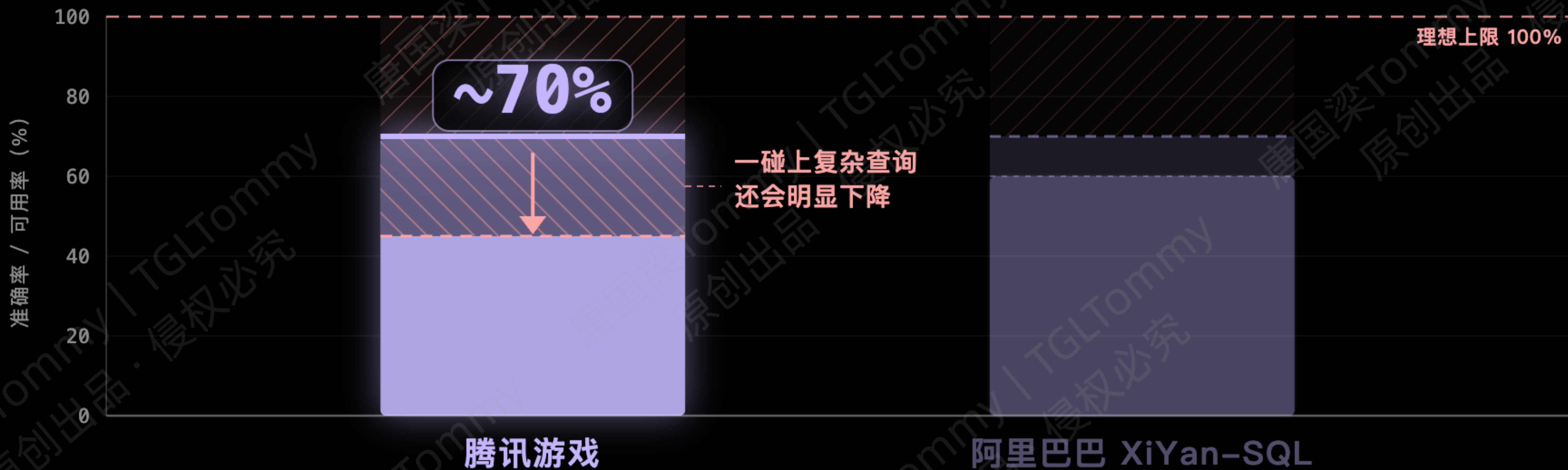
有两组数字可以参照

口径

Spider 2.0 / BIRD 公开评测集
理想条件下的 SQL 生成准确率

口径

客户反馈的直接生成 SQL
开箱可用率



① 语义边界

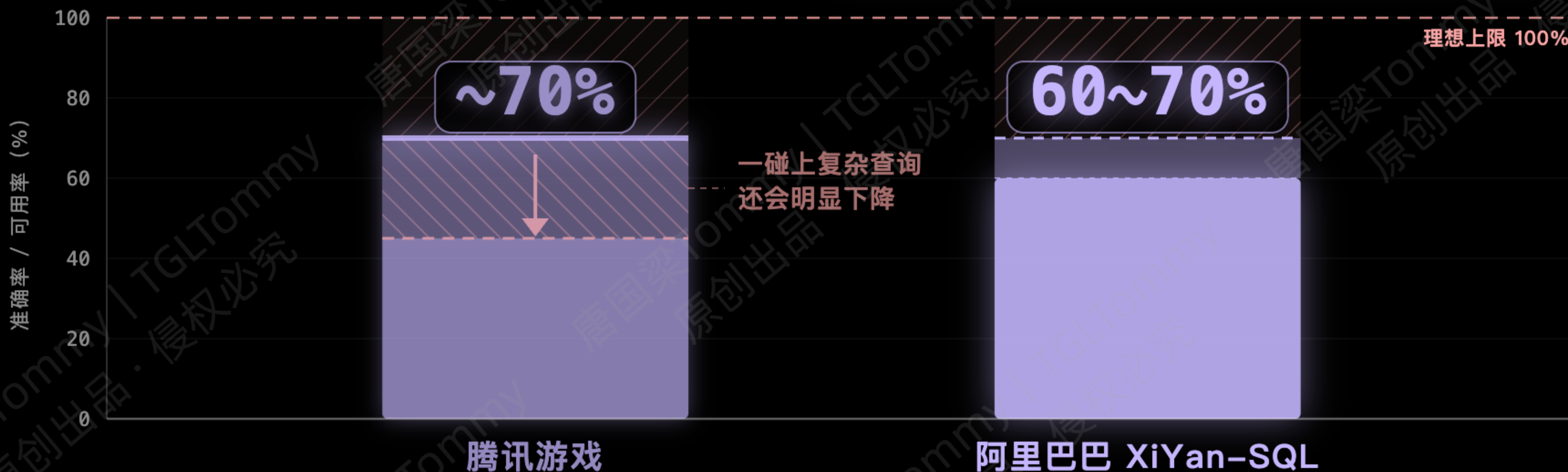
有两组数字可以参照

口径

Spider 2.0 / BIRD 公开评测集
理想条件下的 SQL 生成准确率

口径

客户反馈的直接生成 SQL
开箱可用率

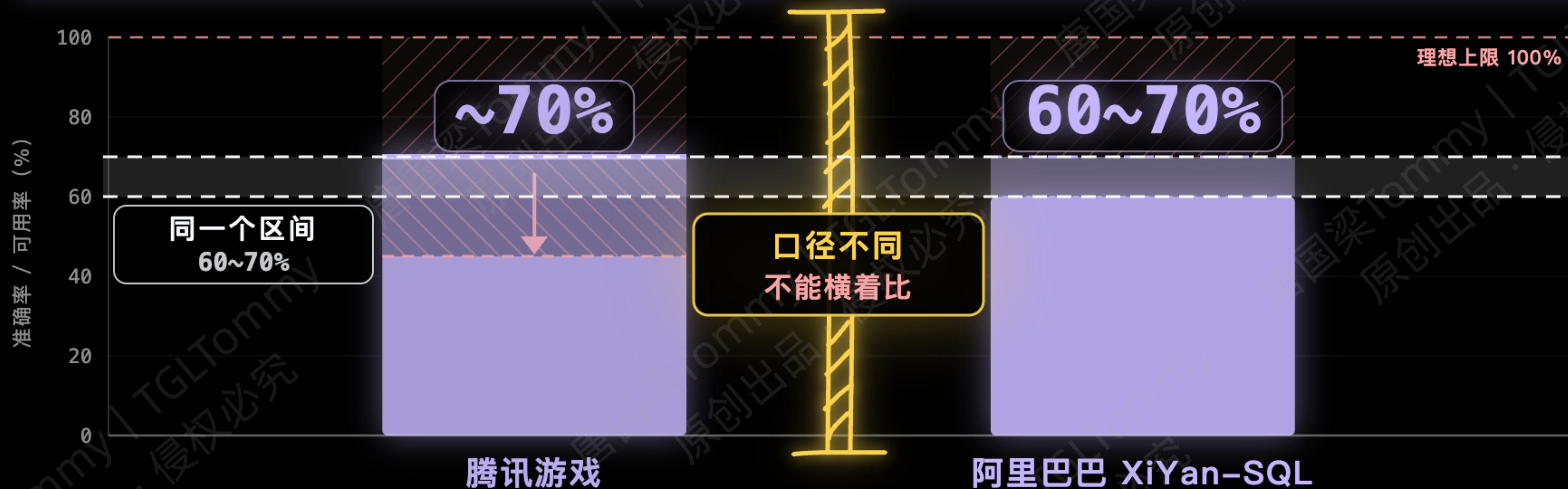


口径

Spider 2.0 / BIRD 公开评测集
理想条件下的 SQL 生成准确率

口径

客户反馈的直接生成 SQL
开箱可用率



① 语义边界

SQL 能生成出来，不代表能可靠承接复杂的企业业务



腾讯 · 阿里 两边都点明

主要瓶颈不在 SQL 语法
而在语义映射

选错表

语法 ✓

业务 ✗

选错指标

语法 ✓

业务 ✗

选错时间口径

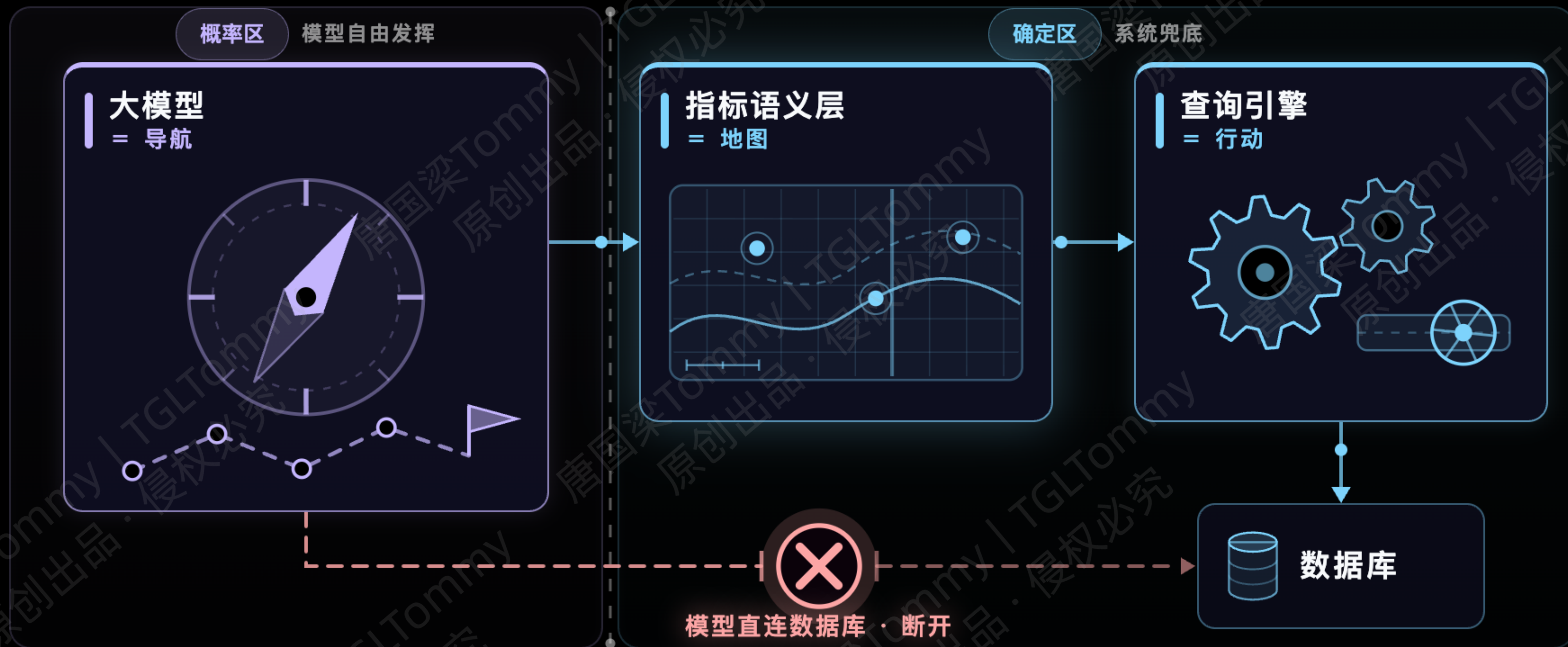
语法 ✓

业务 ✗

① 语义边界

语法完全通过 · 业务完全错 最典型的失败方式

不让模型直接碰数据库，中间插一层指标语义层



语义层画地图 · 大模型只导航 · 查询引擎才动手



自然语言

用户怎么问，就怎么写

模型负责理解



MQL

指标查询语言

Metrics Query Language

一句 只讲业务、~~不讲存储~~ 的查询

SQL 讲存储 · MQL 讲业务

SQL

得写明白这三件事

01

查哪张表

FROM dw.fact_sales

02

取哪个字段

SELECT SUM(pay_amount)

03

表和表怎么连

JOIN dim_region ON ...

全是存储细节

MQL

只声明这几件事

◆ 要哪个指标

◆ 按什么维度拆

◆ 加什么过滤条件

◆ 看哪段时间

全是业务口径

一句话拆成三枚槽位

用户原话

“今年华南大区的销售金额”

指标 =

销售金额

维度 =

区域

过滤条件 =

华南 + 今年

语义层

预先定义好的计算逻辑

落在哪张表

对哪个字段求和

× 不用模型操心

权限前置 · 100% 指的是哪一段



①

MQL → SQL

引擎确定性编译

100% ✓

②

自然语言 → MQL

的准确率

不是 100% ?

两个可以分别测试的阶段

③

最终结论

一定对?

也不是 x

独立测试

独立测试

阶段 1

阶段 2

MQL

问题理解得对不对

自然语言 → MQL · 概率性

SQL 生成得对不对

MQL → SQL · 确定性

它把这两件事变成两个可以分别测试的阶段

另一条路：把语义补齐

四个工位 · 往语义里补进来的东西



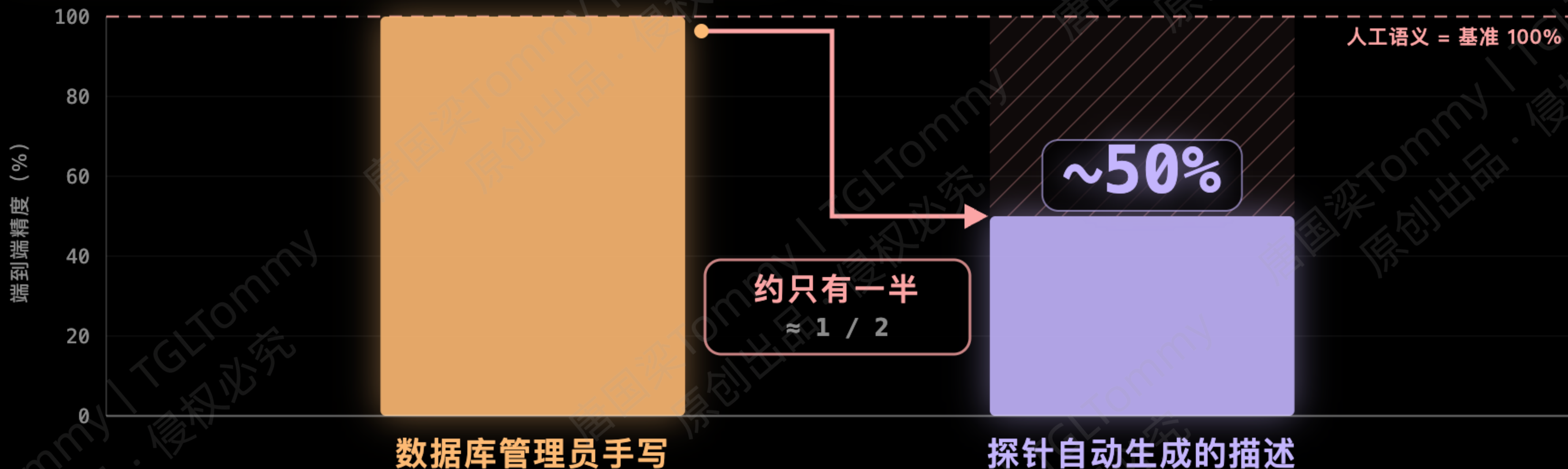
四个工位做的是同一件事：往语义里补料

口径

人工写的业务语义 · 端到端精度基准

口径

自动反推的表 / 列含义 · 同一套评测



✓ 自动化能省下标注成本

✗ 替代不了高质量的人工语义

Apache Gravitino

站的位置更靠前

Agent 用数据

取数

算指标

写回

地基层 Agent 用数据之前的前置条件



统一的元数据坐标

同一份数据，同一个地址



统一的血缘

从哪来、动了影响谁



统一的鉴权

谁能看、谁能动，一处判定

先有这层地基，Agent 才谈得上用数据

① 语义边界

没有语义与元数据这一层



只能靠表名文件名猜



没法判断数据新不新鲜、动了会影响谁



新鲜度



影响范围





第一步

✗ 生成 SQL

✓ 确认业务语义

Agent 的第一步不该是生成 SQL，而是先确认这个问题对应的业务语义是什么



② 执行边界

能调用工具，不等于可以自由行动

这一层最有说服力的样本

eBay · On-call 支持系统

线上故障值班 · 工单要在分钟级给结论

一张线上工单

INC-20482

支付回调超时
下单链路受影响

值班窗口 · 分钟级要结论

概率性的那一半

支持 Agent

工具全在手边
自己决定先做哪一步

可调用的工具

查日志

只读 · 安全

查配置变更

只读 · 安全

重启服务

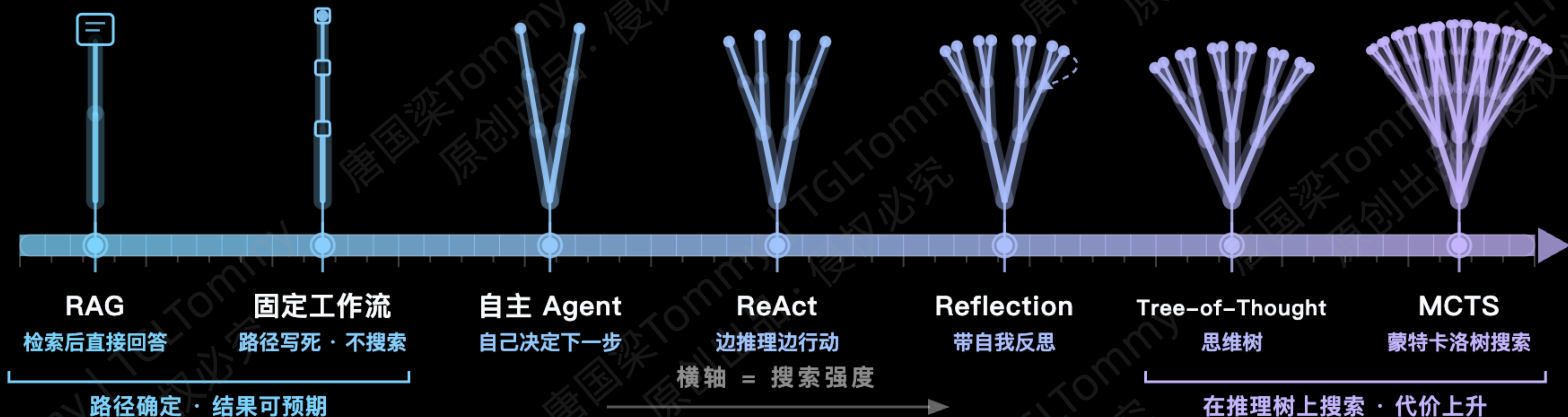
有后果 · 不可回退

回滚配置

有后果 · 不可回退

② 执行边界

他们把主流路线调研了一圈



不是七个并列选项 —— 是同一个维度上的七个位置

② 执行边界

结论有点反直觉

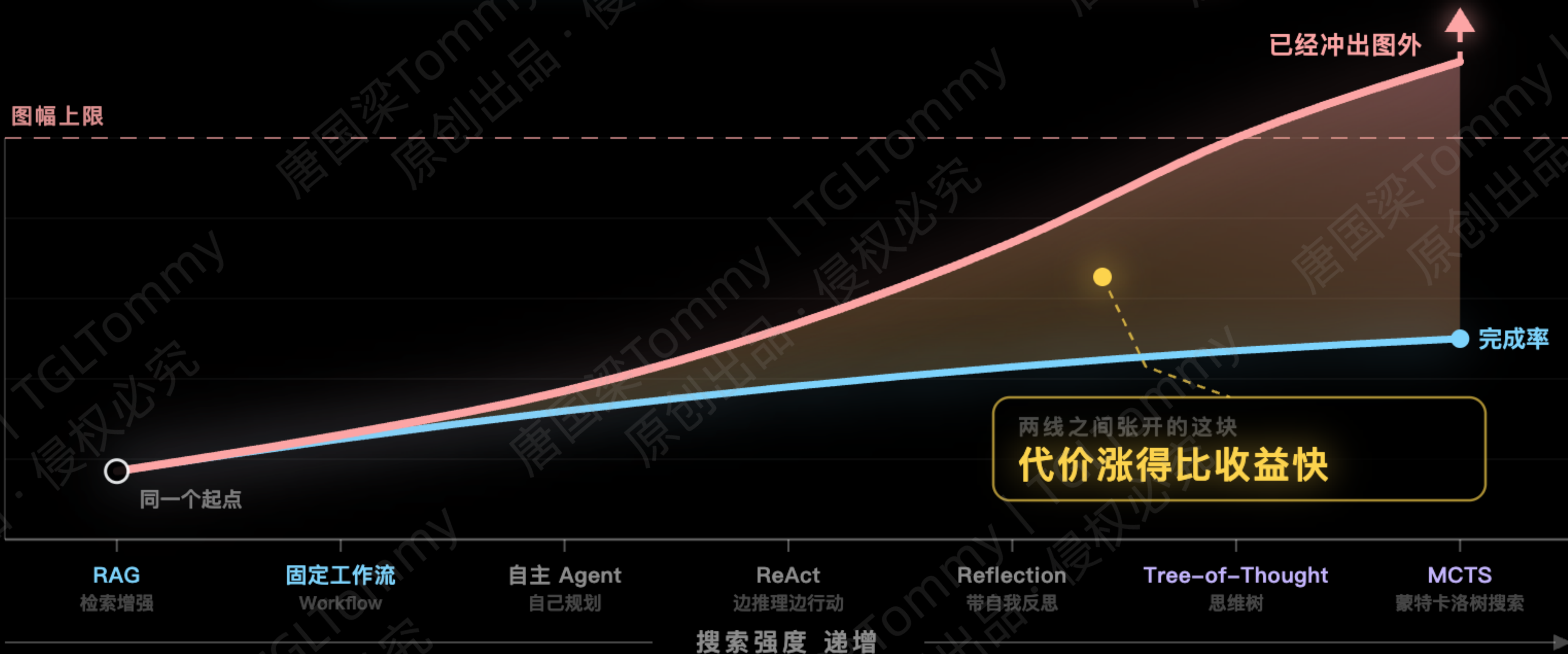
离线评测完成率

缓慢上扬

模型调用次数 · 路径长度 · 时延

陡峭上扬 · 冲出画面顶部

相对水平（同一起点归一化）



② 执行边界

他们援引了一组社区实测案例



eBay 调研援引 · 一组社区实测案例

跑完一次完整的 Agent 路径评估

30 ~ 300 美元 几十到几百 · 单题

\$1 \$10 \$100 \$300

对中小团队已经是不小的负担

靠蒙特卡洛树搜索找答案

2000+ 步 **5** 小时 才返回答案

⌚ ⌚ ⌚ ⌚ ⌚ 计时器指针转满五圈

在实时支持里基本没有可用性

对一个等着解决线上故障的人来说，评测分再高，也受不了五小时的等待

② 执行边界

他们援引了一组社区实测案例



eBay 调研援引 · 一组社区实测案例

跑完一次完整的 Agent 路径评估

30 ~ 300 美元 几十到几百 · 单题

\$1 \$10 \$100 \$300

对中小团队已经是不小的负担

靠蒙特卡洛树搜索找答案

2000+ 步 **5** 小时 才返回答案

⌚ ⌚ ⌚ ⌚ ⌚ 计时器指针转满五圈

在实时支持里基本没有可用性

对一个等着解决线上故障的人来说，评测分再高，也受不了五小时的等待

所以 eBay 最后是往回退的



改成由专家写好的标准作业流程库

专家写好

SOP · 标准作业流程库

Agent 的职责收缩成三件

① 选哪条 SOP

② 补参数

③ 按步骤执行

例外路径 · 才回到概率侧

实在没有匹配的 SOP，才让模型动态补全

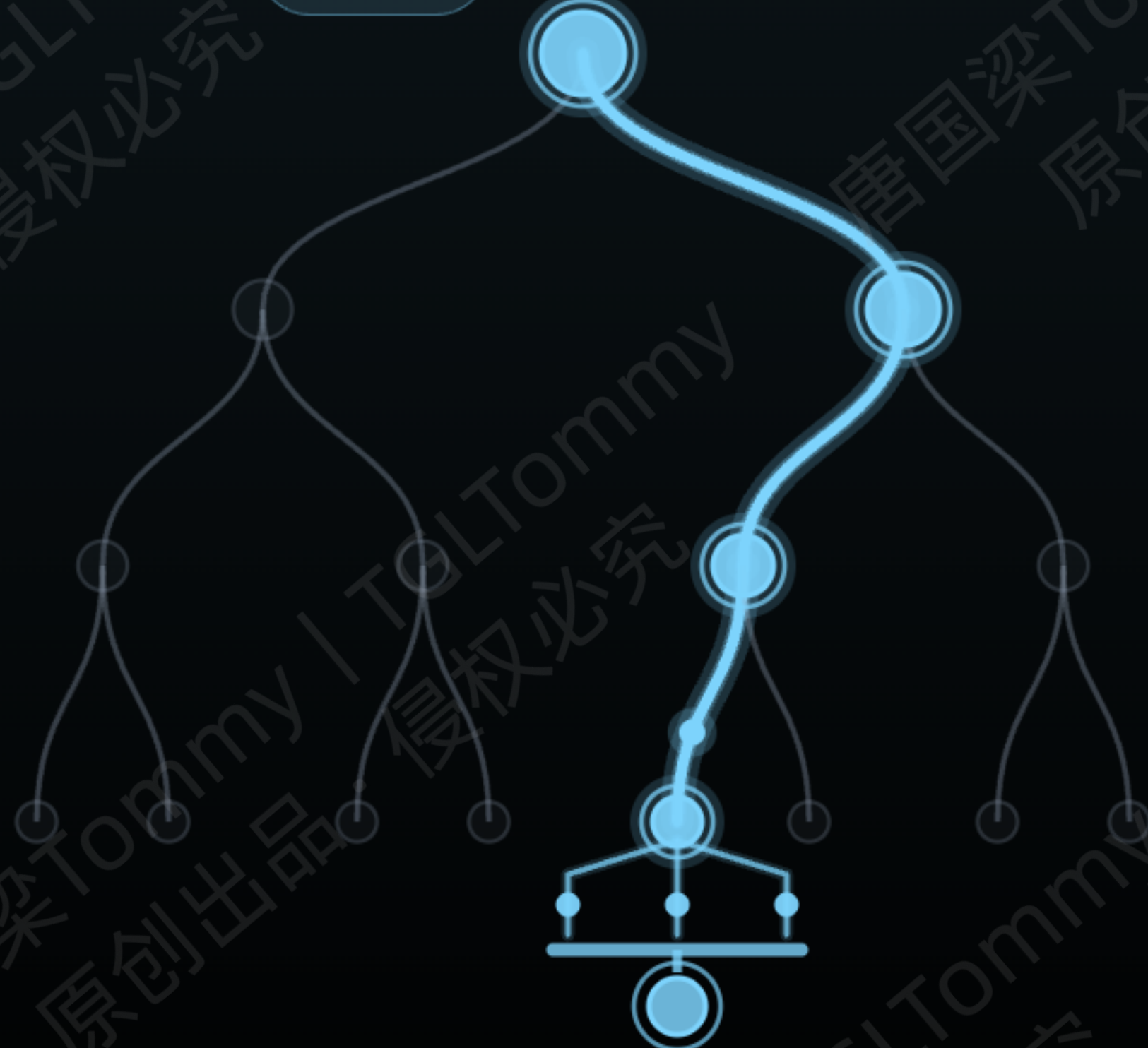
概率区 树搜索



遍历所有分支 · 反复回溯

成本与时延不可预测

确定区 单路径最优优先 每一步只选当前最好的工具往下走



多次采样投票 · 提高稳定性

第 1 步 选当前最好的工具

第 2 步 选当前最好的工具

第 3 步 选当前最好的工具

✓ 不回溯

✓ 不遍历其他分支

主动放弃全局最优，换可预测的成本和时延

整条链路 · 八个环节

从意图识别，到可扩展性



一条链路，八个可以单独设计、单独兜底的环节



路由 = 决定这个问题交给哪个 Agent 处理

而一次分派，完全可能是错的

重新路由 = 允许中途承认分派错了

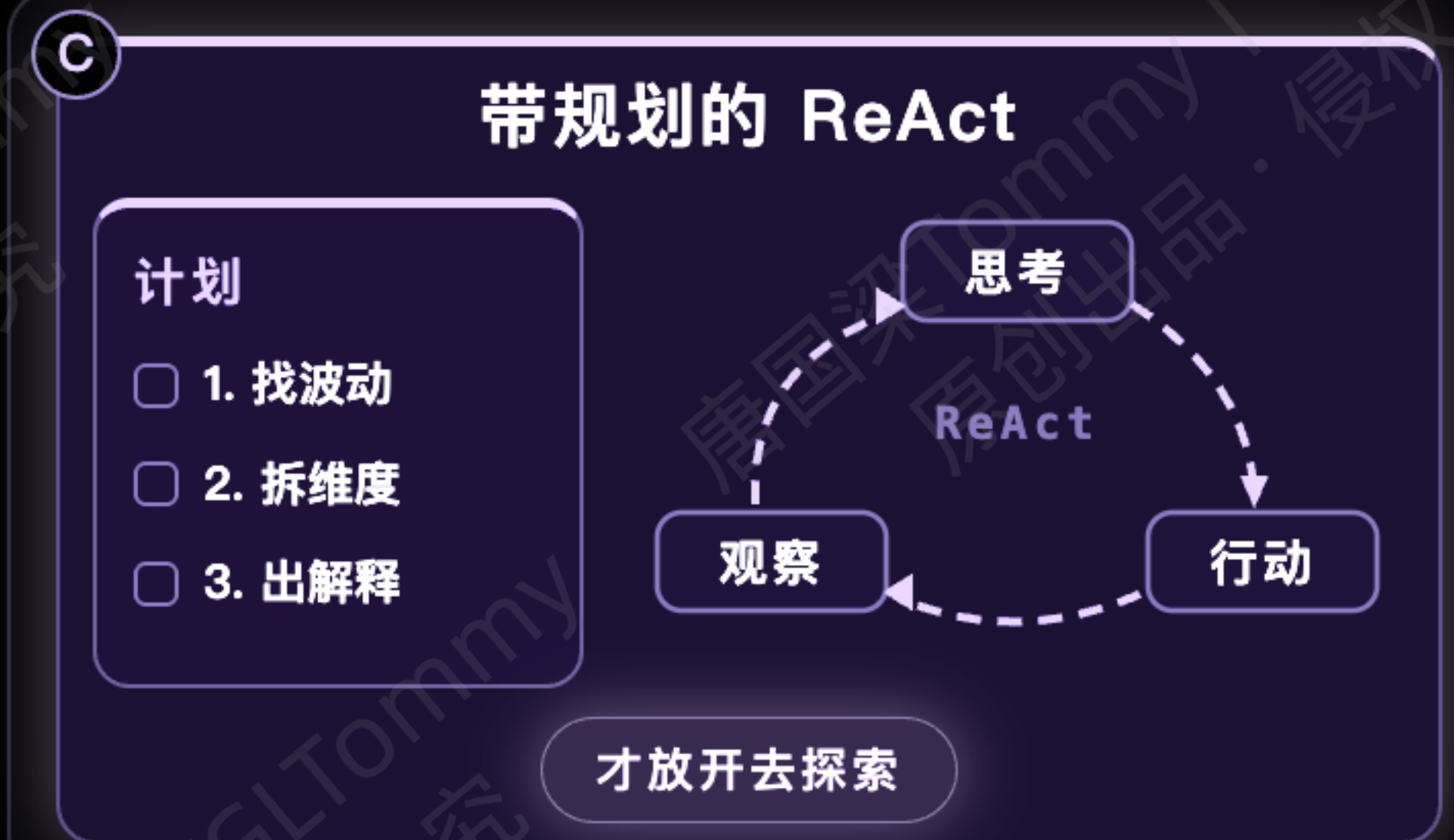


腾讯游戏：按阶段分治

① 数据准备 确定性 · 结构先钉死



② 分析解释 概率性 · 才放开探索



同一件事：该确定的先钉死，该探索的才放开

场景 · 线上异常洞察

百度：编排成三阶段的显式状态图



流程写成状态图 · 中间判断留一个人工入口

编排方式：他们对比过三种

判据 · 状态、节点和边能不能显式化

①

对比过

低代码 workflow

在画布上拖节点连线

②

对比过

角色扮演式的多 Agent

让多个角色互相商量

③

✓

代码编排

把图直接写在代码里

```
1 state = {"anomaly", "window", ...}
2 g.add_node("影响判断")
3 g.add_edge("影响判断", "特征定位")
```

状态 ✓

节点 ✓

边 ✓

选中理由

状态、节点和边能显式化

再往前推一步：做成代码里的状态对象

提示词里 边界模糊 · 正在往外漏

记得带上当前异常

异常

每个 Agent 都要给证据

证据

时间窗口最多是一小时

窗

口

时间窗口最多是一小时

回

不是留在提示词里

搬进结构

`class RunState:`

代码里的状态对象

`current_anomaly`

当前异常

Anomaly

`time_window`

时间窗口

TimeRange

`agent_evidence`

每个 Agent 的证据和置信度

Evidence[]

`retry_policy`

分支重试、超时和回退

RetryPolicy

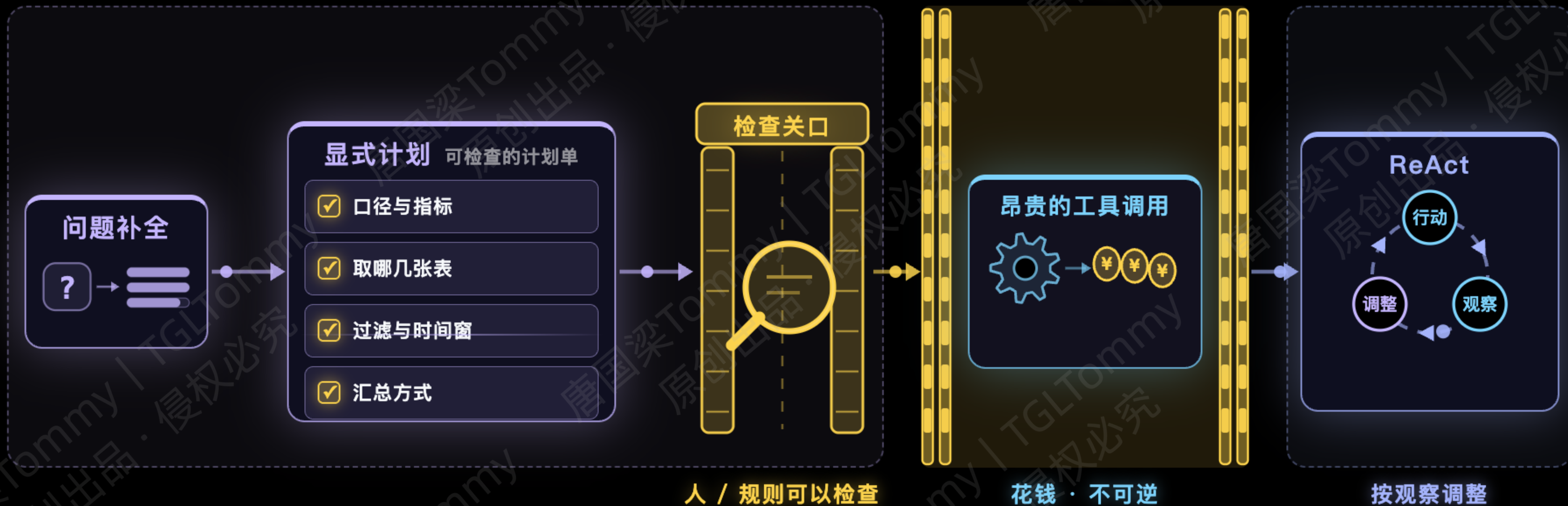
※ 这是我的引申，不是他们的原话

关口，落在昂贵调用之前

Plan · 计划阶段

成本墙

ReAct · 执行阶段



一根连续的滑轨，不是二选一

越接近真实的业务动作，流程越应该确定；
越接近意图理解，模型才越可以灵活。

← 意图理解

真实的业务动作 →

意图识别

规划下一步

受控编排

SOP 步骤

权限判定

概率区

模型越可以灵活

确定区

流程越应该确定



③ 权限边界

能连接工具，不等于能治理工具

动作已封装成工具

查数据

?

改配置

?

建工单

?

发通知

?

常见说法

~~MCP 只解决连接~~

? 谁能调这些工具

? 能调到什么程度

这个说法已经不准确了

③ 权限边界

协议能力，不等于治理落地

协议已经标准化 2 项

☒ 基于 OAuth 的授权框架

☒ 接企业的统一身份管理

还得企业系统自己实现 5 项

☐ 最终用户身份 → 数据权限映射

☐ 行列级控制

☐ 审批审计

☐ 运行沙箱

☐ 资源配额

MCP 划走的那块地 —— 比大家以为的小

③ 权限边界

腾讯太极的内部 MCP 架构

前提

~~公共 MCP 生态~~

~~未直接开放~~

内部业务

进料口 · 只接舱内

内部市场

内部镜像

① 只允许使用企业内部的 MCP 市场

② 依赖包从内部镜像取

③ Server 跑在沙箱里

④ 限制 网络 文件系统 资源

⑤ 注册与运行时保留软件包、访问 token 这类元数据

协议适配只是里面很小一块

协议适配

水面

①



供应链安全

②



运行隔离

③



凭证代理

④



权限映射

⑤



版本兼容

⑥



审计

这道边界上还有几个具体的风险点



✓ 治理价值

Schema 不可见 $\neq$ 安全

✗ 掩盖越权

✗ 一个统一的高权限账号

✗ 每扇门 · 同一种锁

✗ 看不见 $\rightarrow$ 掩盖了越权

遮住门牌 $\neq$ 换锁

帘后 · 一把万能主钥匙

Elastic · 检索能力做成 MCP Server

标准化

复用

真放到生产上，至少还要补齐这四样



限制允许访问的索引和字段

索引白名单



把最终用户身份传到查询层

身份透传



设置 Top-K、超时和结果大小

资源上限



对查询和返回做审计脱敏和注入防护

留痕 · 过滤

如果 Agent 能任意指定索引和查询语句，标准协议只会把攻击面放大

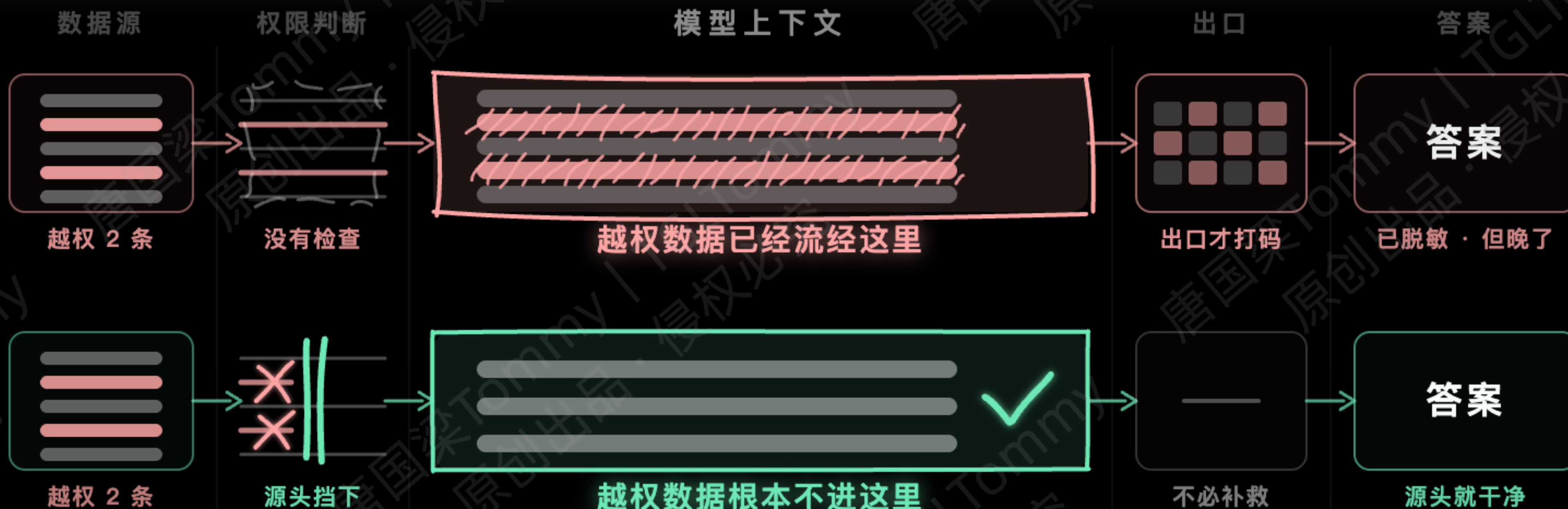
攻击面 ↑

Agent 能任意指定索引和查询语句 —— 标准协议只会把攻击面放大

两种做法

后置

生成答案之后再脱敏



权限判断放在执行之前 —— 这比生成答案之后再脱敏，可靠得多



权限五问

你可以照着对



身份透传

最终用户的身份传到数据层，不是一个公用服务账号



行列与租户隔离

行级 · 列级 · 租户级，三层都要卡住



调用预算

单次与累计的调用配额，超了就停



审批留痕

谁、在什么时候、调了什么，全程可回溯



注入防护

提示注入与越权指令，拦在执行之前

协议已经标准化的

接入与授权流程

还得企业自己建的

业务权限策略

审计

资源预算



④ 验证边界

能完成任务，不等于结果可信

任务这一侧



工具调用完全合规

参数、权限、审批全部通过

① 语义



② 执行



③ 权限



前三道边界，全过

≠

结果这一侧



结果是否可信

还没有任何一步验过

④ 验证边界 · 待判定

第四道，从这里开始

④ 验证边界

Agent 最危险的状态 —— 不是报错



同一份结果 · 三项判定



结构完整

章节、表格、图注一个不缺



语言流畅

读起来像专业分析师写的



口径完全错

源头那一根，接在空接口上

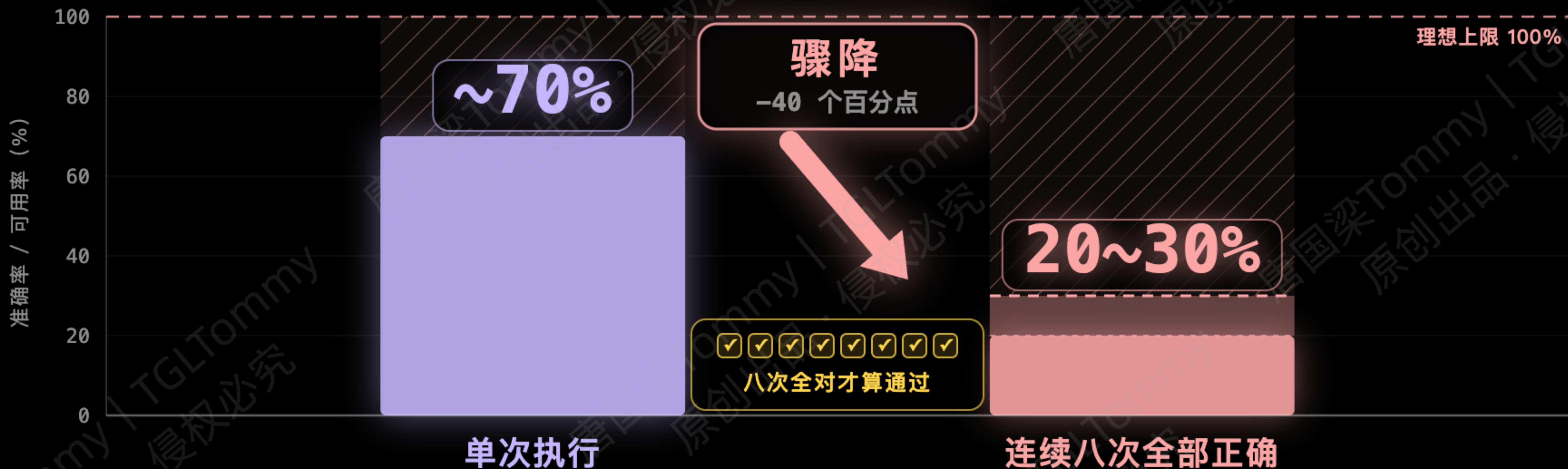
最危险的错误 —— 长得像对的

口径

BFCL / Tau-bench 这类基准
单次执行任务的准确率

口径

同一个任务连续跑八次
八次结果都正确



④ 验证边界

衡量的不是 **多步链路的误差累积**，而是 **一致性**
同一道题，它这次做对了，下次未必还对

④ 验证边界

同一道题 · 连跑八次

八次全对的题

4 / 16

≈ 25% 也就是两三成

一致性：绿 = 这次对 · 红 = 这次错

题目	1	2	3	4	5	6	7	8	八次全对
Q01	●	●	●	●	●	●	●	●	✓全对
Q02	●	●	●	●	●	●	●	●	错 3
Q03	●	●	●	●	●	●	●	●	错 4
Q04	●	●	●	●	●	●	●	●	错 4
Q05	●	●	●	●	●	●	●	●	错 1
Q06	●	●	●	●	●	●	●	●	✓全对
Q07	●	●	●	●	●	●	●	●	错 3
Q08	●	●	●	●	●	●	●	●	错 3

题目	1	2	3	4	5	6	7	8	八次全对
Q09	●	●	●	●	●	●	●	●	错 2
Q10	●	●	●	●	●	●	●	●	错 2
Q11	●	●	●	●	●	●	●	●	✓全对
Q12	●	●	●	●	●	●	●	●	错 1
Q13	●	●	●	●	●	●	●	●	错 4
Q14	●	●	●	●	●	●	●	●	错 2
Q15	●	●	●	●	●	●	●	●	✓全对
Q16	●	●	●	●	●	●	●	●	错 3

衡量的不是多步链路的误差累积， 而是一致性 同一道题这次做对了，下次未必还对

单次跑通 ， 替代不了 稳定跑通

他们的应对

分层验证

上三层 · 模型侧自检

下两层 · 确定性校验与兜底

① 按步骤选不同模型

不让一个模型一条道走到黑

② 用验证 Agent 检查结果

执行之外再挂一层复核

③ 大结果先切片，再做细粒度验证

整块看不出的错，切开才看得见

④ 工具调用前后都保留校验

参数校验

权限校验

结果校验

⑤ 长尾高风险任务用固定 workflow 兜底

确定性地基 · 不交给模型自由发挥



他们自己也承认的局限

两台仪器坏在同一个地方

执行 Agent

出结果的那一台

验证 Agent

检查结果的那一台

相近的模型 + 相同的上下文

= 共享同一批盲点

同一副镜头上的同一处污点 —— 缺陷正好落在污点后面，
两台仪器同时读作「合格」

④ 验证边界

自检替代不了确定性验证

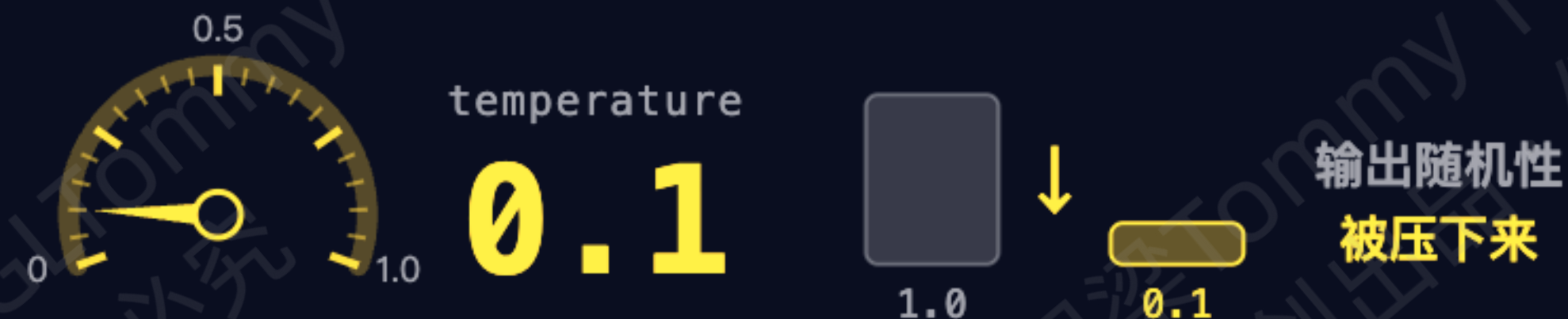
针对幻觉，他们给了四条机制

① 既定知识优先



企业标准文件与制度文档先进知识库

② 生成参数压低



他们举的例子：温度设到 0.1

③ 每步输入输出明确定义



消除 workflow 黑箱

④ 模型只出初稿



最终由领域专家审核确认

四条里，最容易被高估的一条

温度压低：能做什么，不能做什么

温度 · 生成参数

0.5

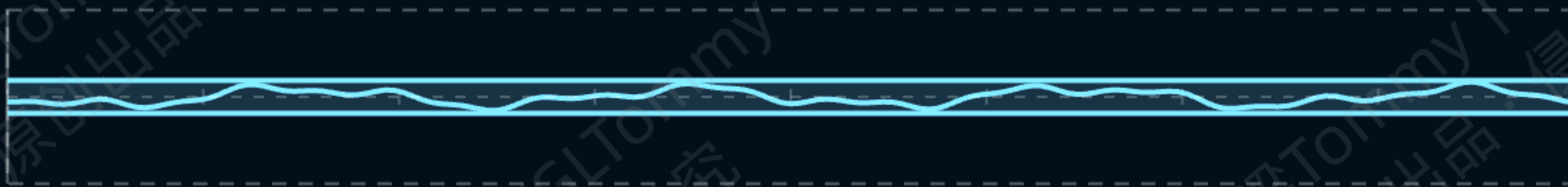


0.1

temperature

输出波动带

原波动范围 (温度 1.0)



波动明显收窄

事实正确性指示灯



事实正确性

x 不保证 · 红灯不灭

温度从 1.0 拧到 0.1 · 这盏灯不变色

只能降低输出的波动，不能保证事实正确

替代不了规则校验和人工审核

「输入输出明确定义」的真正价值

出错的时候，分得清责任

分责定位

数据

源数据读取

检索

知识库命中

模型

生成初稿

规则

校验兜底

本次问题定位

到底是数据的问题、检索的问题、模型的问题，还是规则的问题

④ 验证边界

他们给的人机分工 —— Agent 八成，人工两成

Agent

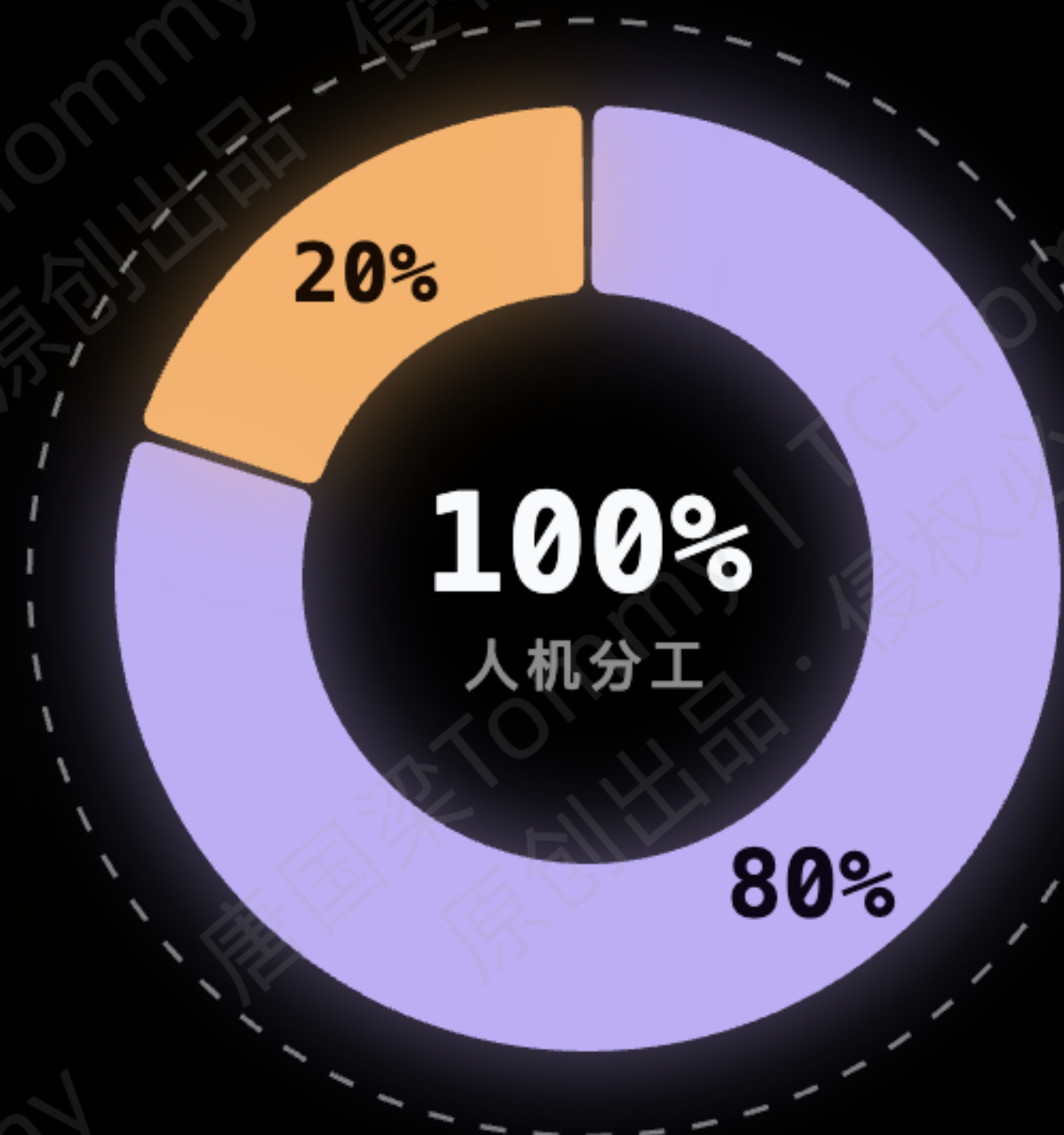
自动完成

80%

人工

审核与调整

20%



这条虚线以内，才是那个 100%

排除项

不含前期的
Agent 开发阶段

不在这个 100% 里面

不计入

④ 验证边界

Aloudata: 把最容易算错数的那部分, 从模型手里整个拿走

用户原话

还是开头那个问题

「这个月销售额为什么跌了」

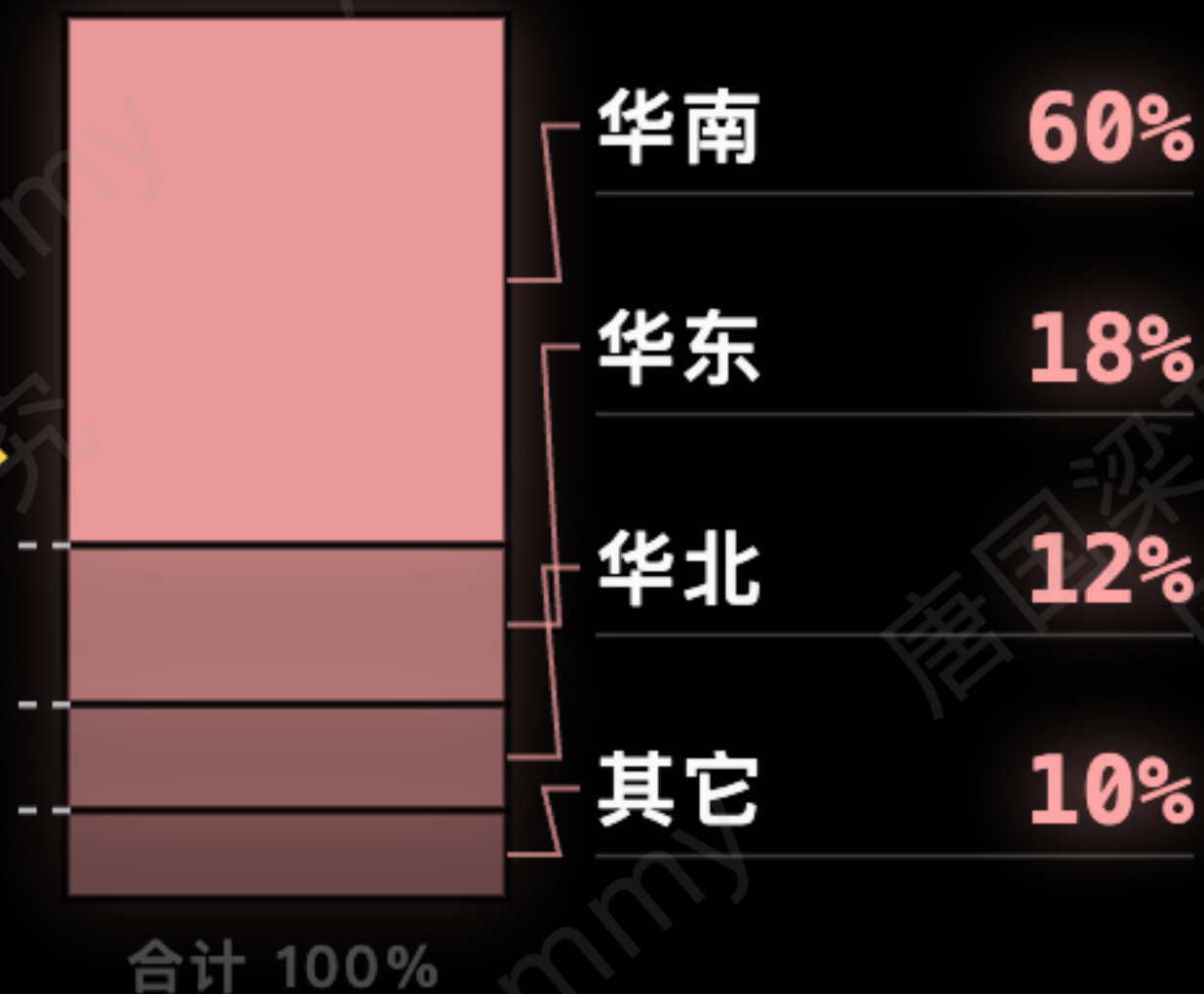
本月总跌幅

100%

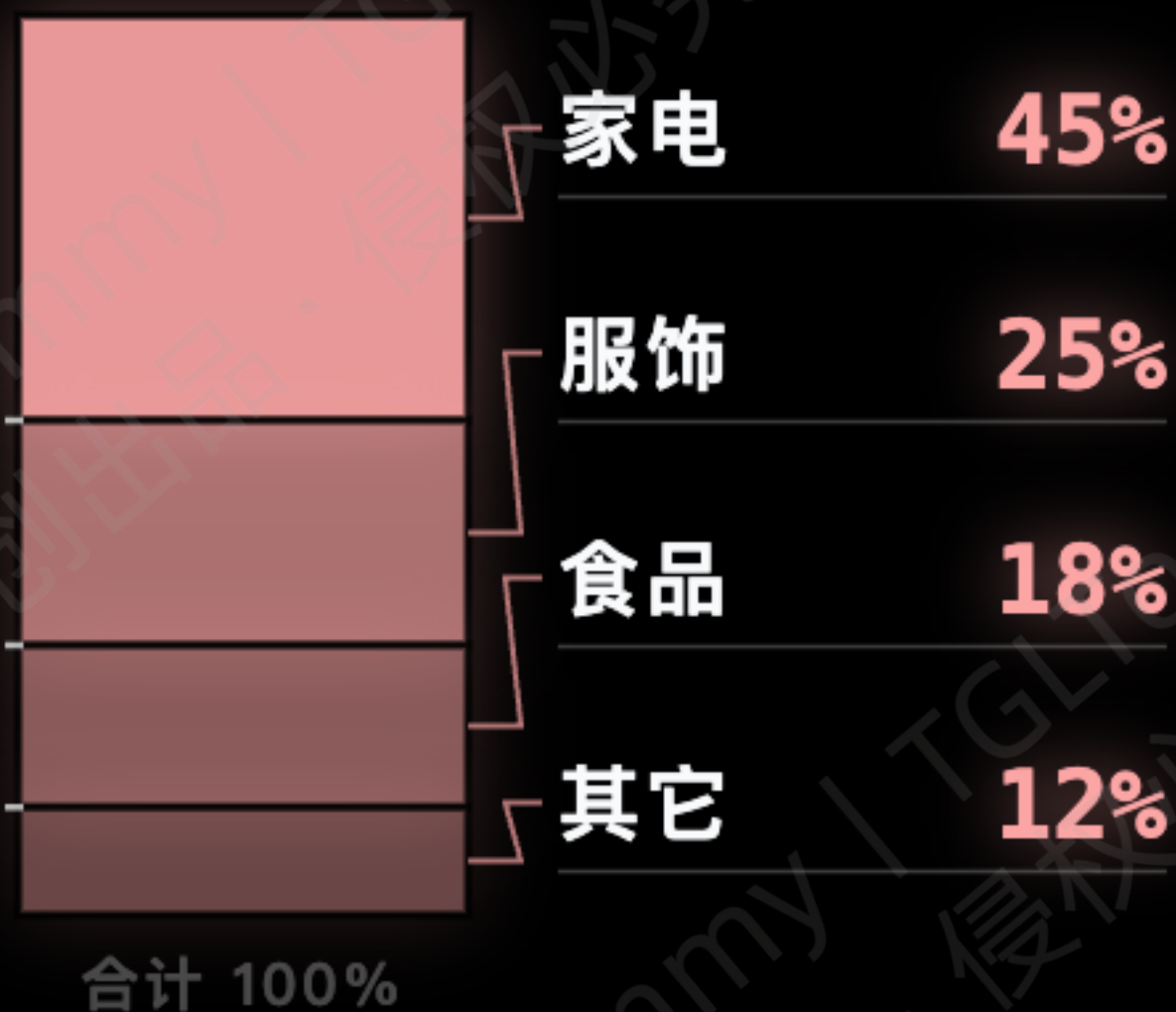
一整块

拆开

按地区拆



按品类拆



术语

归因

把这次下滑拆开, 看哪些地区、哪些品类各自贡献了多少跌幅

术语

贡献度

每一块占多大比例

算的归引擎，说的归模型

概率区

模型自由发挥

大模型

= 只做三件事

① 判断用户问的是哪一类归因
识别问题类型，不算数

② 组织分析流程
决定引擎该算哪几步

③ 把结果写成报告
把算好的数字翻译成成人话

全程不碰一个数字的计算

确定区

系统兜底

算法引擎

= 贡献度全部精确计算

全量

逐项

可复算

维度

取值

贡献度

占跌幅

地区

华南

-6.24pp

品类

家电

-3.10pp

渠道

直播

-1.82pp

商品编号

SKU-0038412

-0.77pp

商品编号

其余 386 万个取值

-0.47pp

Σ

合计跌幅

-12.40pp

✓ 逐项精确 · 加总闭合

贡献度计算 · 整块搬走

最容易算错数的那部分，从模型手里整个拿走

几百万个取值，全丢给模型根本装不下

高基数维度

全丢给模型
根本装不下

SKU-0038412

SKU-2947106

SKU-0915773

SKU-3860000

商品编号 一个高基数维度

进漏斗 · 候选取值

3,860,000

≈ 386 万
个商品编号

引擎算完 · 排好序



按贡献度从大到小

漏斗拦下
3,859,782 项

218 项

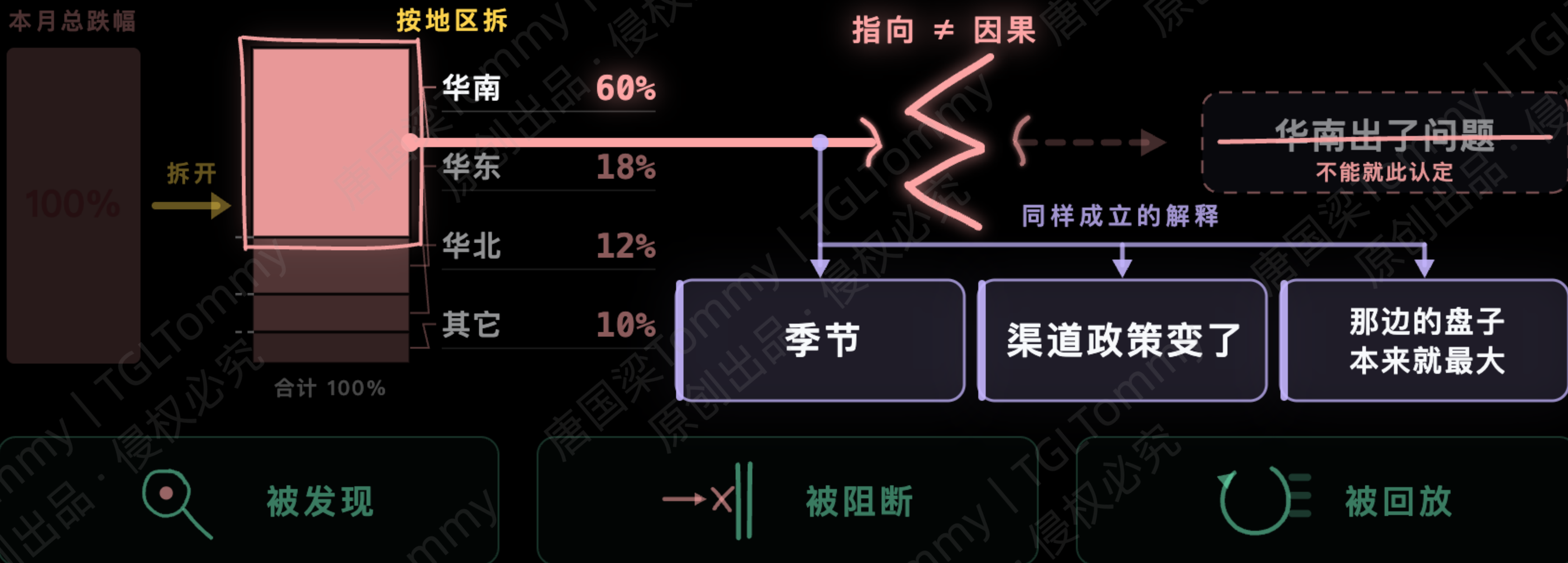
交给模型去解释

贡献最大的那一批 其余的连进都不进模型

两百来项

④ 验证边界

贡献度拆解回答的是「集中在哪儿」，不是「为什么」

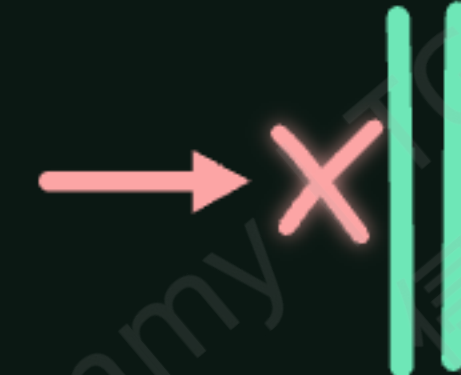




旧标准 ~~有没有回答~~ → 新标准 **错误能不能被处理**



被发现
出错看得见



被阻断
错的走不下去



被回放
过程可复现

④ 验证边界 **三格都过，这道边界才算过**



⑤ 成本边界

能运行一次，不等于能规模化

看起来 · 只有一次

一次用户请求

实际 23 次调用



同一款硬币 · 一枚 vs 一整摞

四种有效的做法 · 下面逐个讲

高出十几倍

单价低 ≠ 任务成本低

模型单价

完成一个真实任务的成本

① 按问题深浅分级

② 漏斗

③ 能力下沉到服务端

④ 别让 Agent 自己扛计算



⑤ 成本边界

能运行一次，不等于能规模化

看起来 · 只有一次

一次用户请求

实际 23 次调用



同一款硬币 · 一枚 vs 一整摞

四种有效的做法 · 下面逐个讲

高出十几倍

单价低 ≠ 任务成本低

模型单价

完成一个真实任务的成本

① 按问题深浅分级

② 漏斗

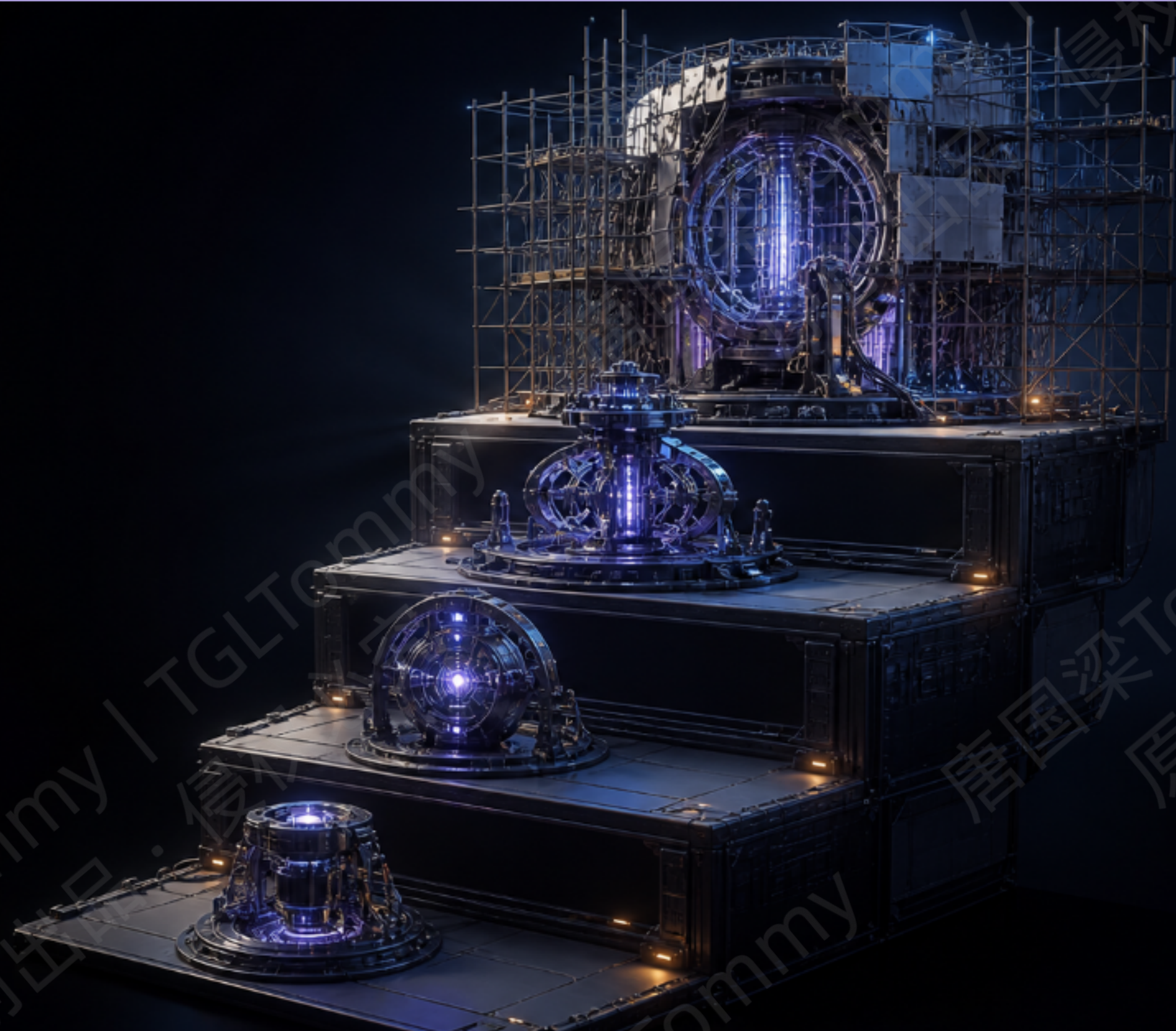
③ 能力下沉到服务端

④ 别让 Agent 自己扛计算

做法一

按问题深浅分级

火山引擎 · 深度研究能力分四级



自主性 ↑ 越往上一格越自主

顶层还罩着脚手架 · 未完工

四级

数据一有变化就自动识别异常、
直接给出解决方案，全程不用人插手

... 完全自主洞察还没实现

三级

从多个角度探索，提出假设并逐一验证

✓ 已经在不少场景里跑出了效果

二级

不光给答案，还能解释原因

一级

你问它答，直接返回结果

做法一

按问题深浅分级

火山引擎 · 深度研究能力分四级

自主性 ↑ 越往上一格越自主

顶层还罩着脚手架 · 未完工

四级

数据一有变化就自动识别异常、
直接给出解决方案，全程不用人插手

... 完全自主洞察还没实现

三级

从多个角度探索，提出假设并逐一验证

✓ 已经在不少场景里跑出了效果

二级

不光给答案，还能解释原因

一级

你问它答，直接返回结果



级别每往上一格，要验证的东西也跟着往外扩

这是我的引申，不是他们的原话

纵 · 自主性往上一格 ↑

- ④ 四级
- ③ 三级
- ② 二级
- ① 一级



横 · 验证成本往外一圈 →

- 验这个问题值不值得系统主动提出来
验证半径
- 验假设和证据够不够
验证半径
- 验解释站不站得住
验证半径
- 验答案对不对
验证半径

自主性涨一格，验证成本也跟着涨一格 —— 不该一味往高级别冲

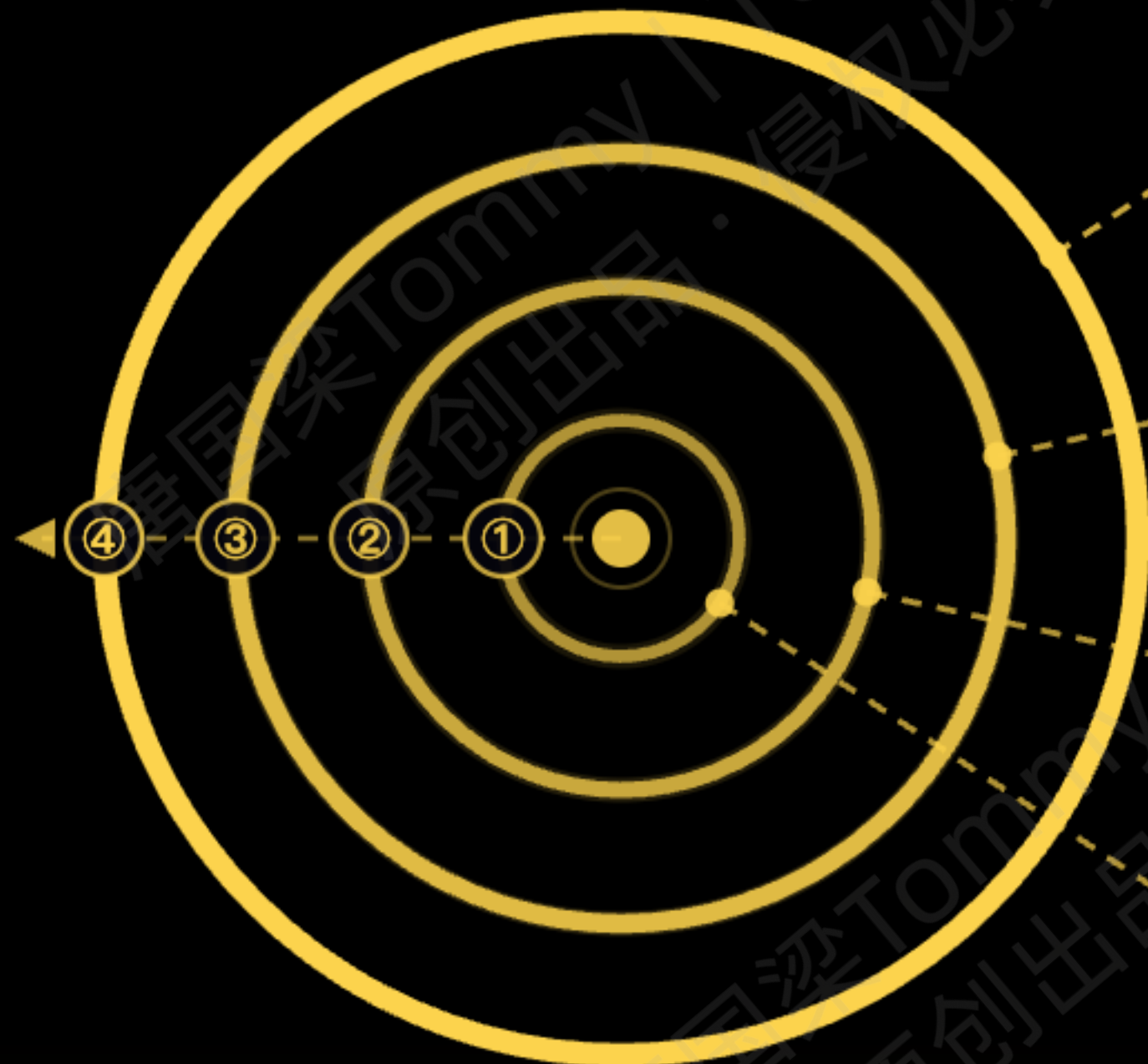
级别每往上一格，要验证的东西也跟着往外扩

这是我的引申，不是他们的原话

纵 · 自主性往上一格 ↑



横 · 验证成本往外一圈 →



验这个问题值不值得系统主动提出来

验证半径

验假设和证据够不够

验证半径

验解释站不站得住

验证半径

验答案对不对

验证半径

自主性涨一格，验证成本也跟着涨一格 —— 不该一味往高级别冲

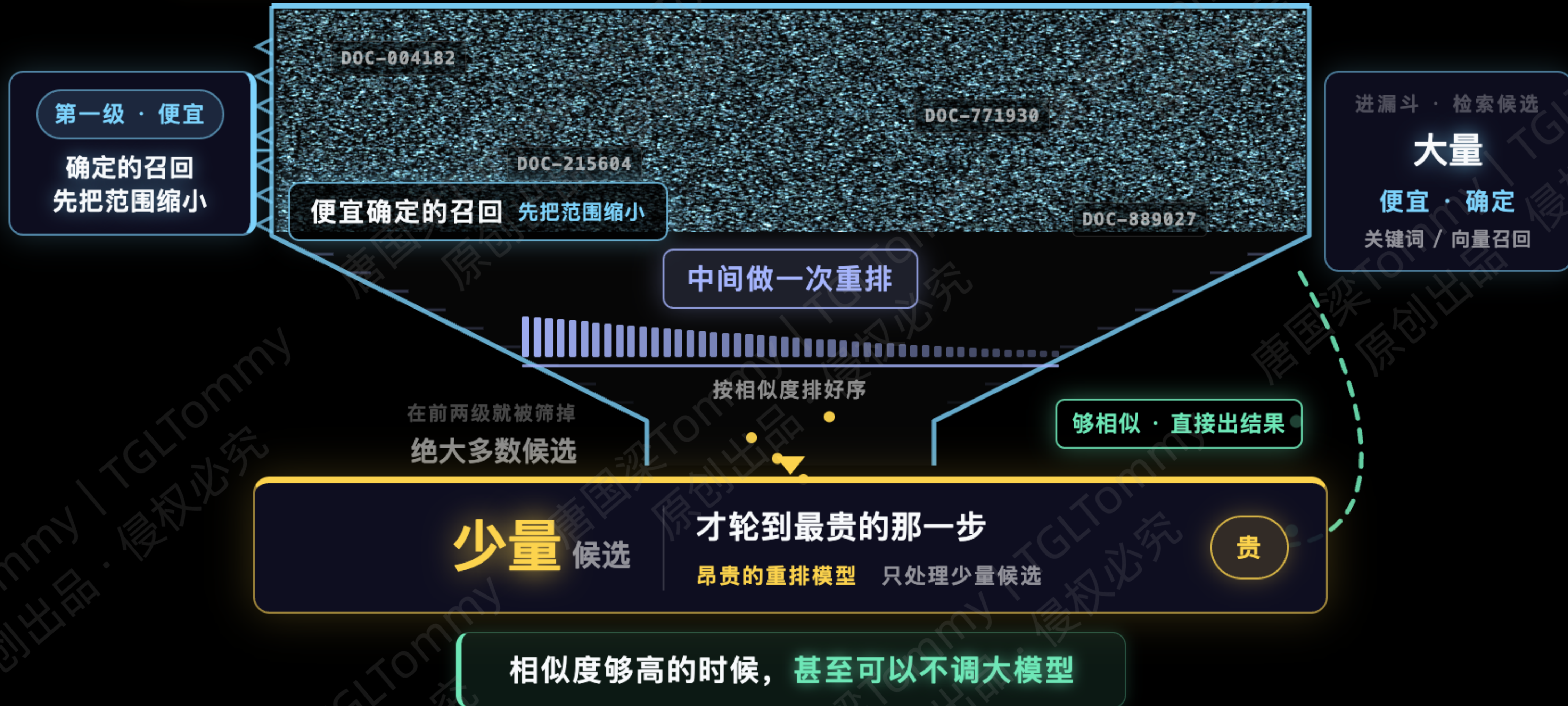
分级的实际作用

按复杂度分配研究深度

不让所有任务都走最重的流程

⑤ 成本边界

第二种做法 · 漏斗：便宜的先筛，昂贵的最后才动



⑤ 成本边界

XiYan-SQL 的声明式 MCP · 一组 PostgreSQL 评测

口径

↑ +21.58pp

准确率

一组 PostgreSQL 评测

口径

↓ 约十分之一

客户端 Token 消耗

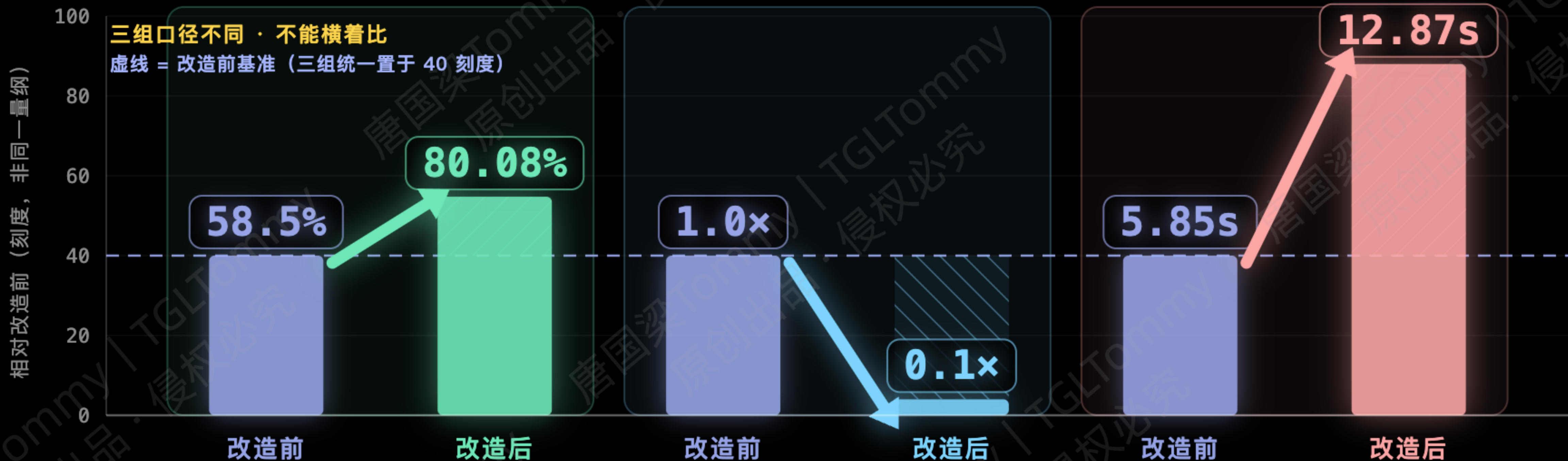
同一批任务

口径

↑ 约 2.2 倍

端到端响应时间

代价就在这一栏

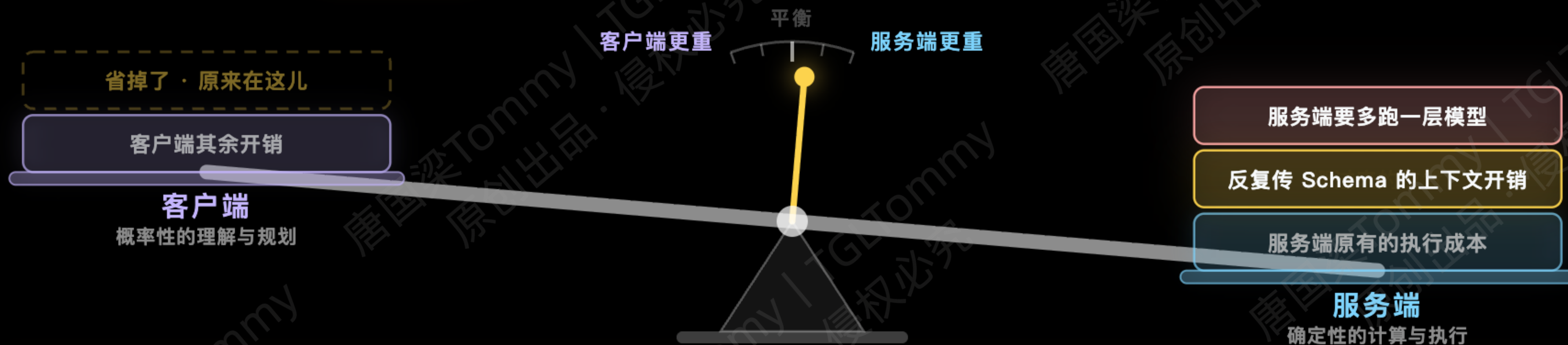


这几个数字得一起看

同样的改造在 MySQL 上增益很小

⑤ 成本边界

成本天平：省下来的那一部分，去哪了



改造前	客户端其余	反复传 Schema 的上下文开销	服务端原有执行	总成本没有变轻 反而多了一层
改造后	客户端其余	服务端原有执行	同一块 · 挪到服务端	

Token 降十倍 $\neq$ 总成本降十倍

省下来的东西没有消失，只是挪了个位置 —— 这正是响应时间涨上去的原因

成本边界 · 第四种做法

别让 Agent 自己扛计算



精确的大规模计算

整个交给



MaxCompute 执行底座

模型 负责理解和编排

平台 负责规模化执行

 别让 Agent 把大表往上下文里搬

x 不是这个口径

单次模型调用的价格

只看模型单价

✓ 要统计的是这个

每个成功任务的端到端成本

重试 · 失败 · 编排开销全算进去

✓ 语义

✓ 执行

✓ 权限

✓ 验证

⑤ 成本

— 五道边界讲完

成本边界 · 第四种做法

别让 Agent 自己扛计算

精确的大规模计算

整个交给

MaxCompute 执行底座

模型 负责理解和编排

平台 负责规模化执行

 别让 Agent 把大表往上下文里搬

x 不是这个口径

~~单次模型调用的价格~~ x

只看模型单价

✓ 要统计的是这个

每个成功任务的端到端成本

重试 · 失败 · 编排开销全算进去

✓ 语义

✓ 执行

✓ 权限

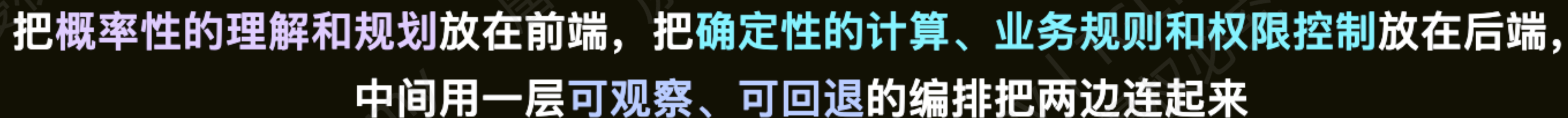
✓ 验证

✓ 成本

— 五道边界讲完

放回同一张图

✓ 成本

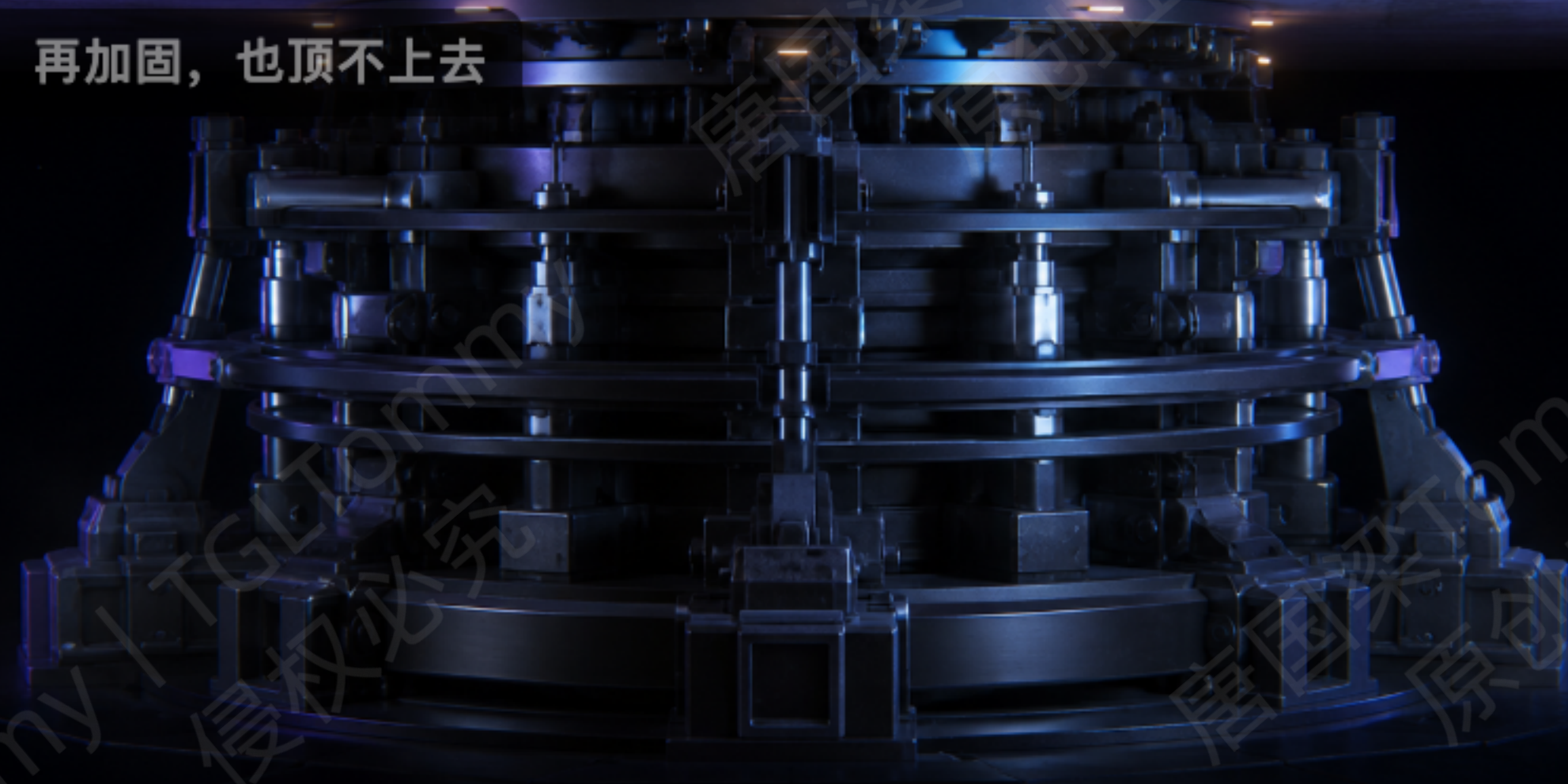


五道边界 · 共同前提

外面能加的，都加上了

天花板

↑ 再加固，也顶不上去



外加约束

模型本体 · 0 改动

模型不稳



加校验

模型乱走



用 SOP 把路径框住

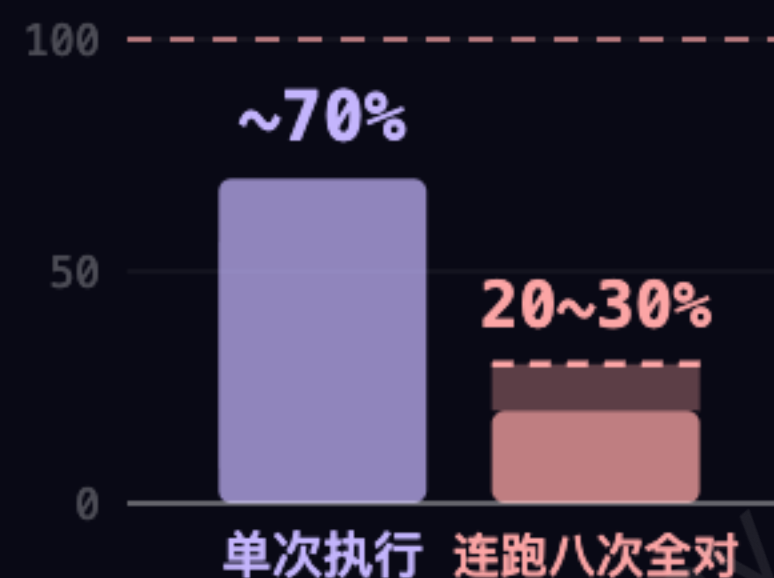
全都是在模型不动的情况下，从外面加的约束

✅ 这条路，能把系统送上生产

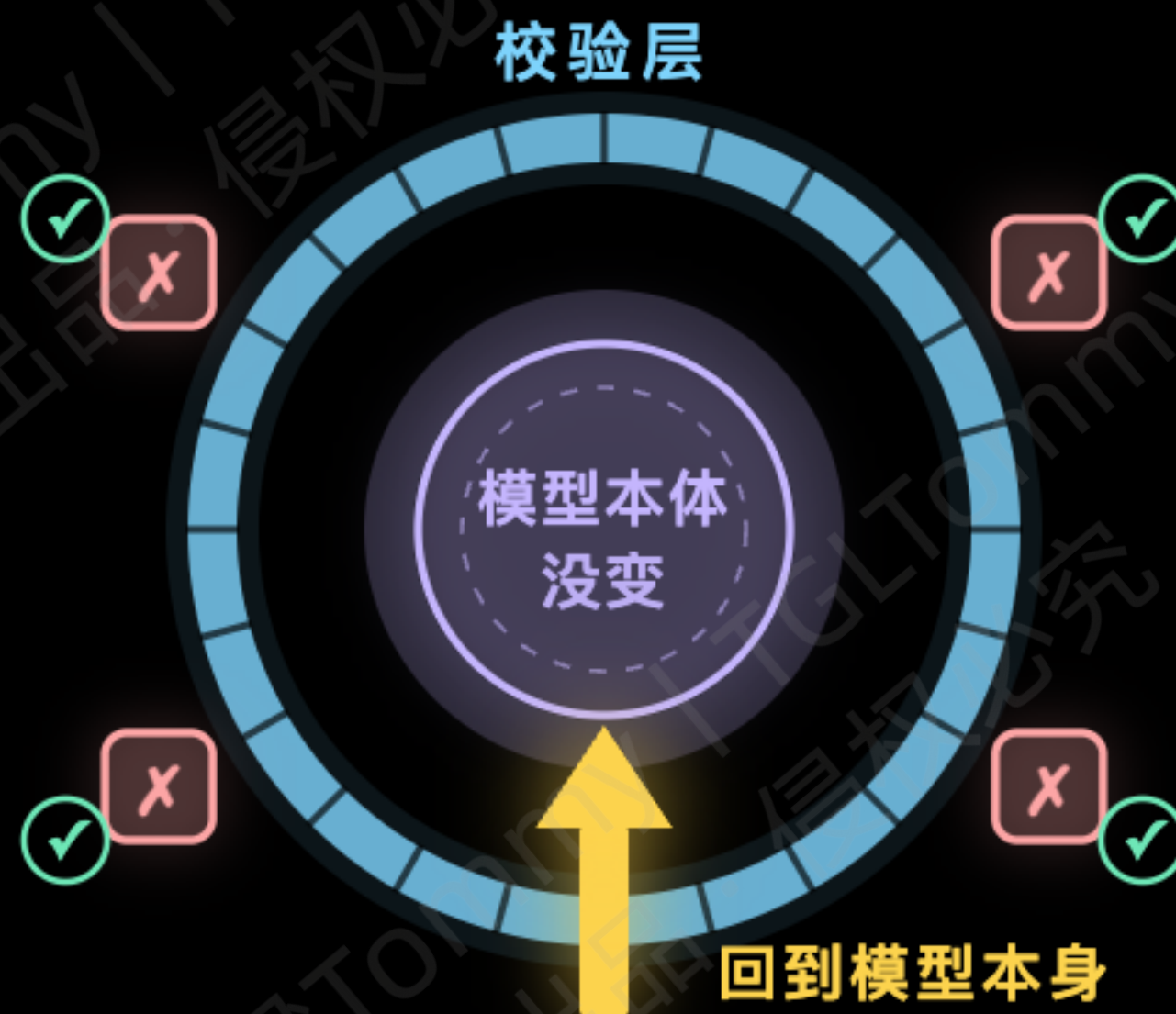
绕不过去的上限

拦下来 $\neq$ 下次做对

回到前面那个数字



连跑八次
只有两三成全对



外面那层校验

☒ 能把错的拦下来

☐ 没法让它下次做对

已拦截
32

模型改动
0

一轮接一轮：每一轮都全部拦下 —— 每一轮的错，一个没少

第 1 轮 ☐ ☐ ☐ ☐ ☐ 全拦下 ✓

第 2 轮 ☐ ☐ ☐ ☐ ☐ 全拦下 ✓

第 3 轮 ☐ ☐ ☐ ☐ ☐ 全拦下 ✓

第 4 轮 ☐ ☐ ☐ ☐ ☐ 全拦下 ✓

第 5 轮 ☐ ☐ ☐ ☐ ☐ 全拦下 ✓

第 6 轮 ☐ ☐ ☐ ☐ ☐ 全拦下 ✓

要真正动这个数字，只能 回到模型本身

在 多步工具调用里学会

哪一步该走

错了又错在哪儿

镜头进到内核里面

Agentic RL 在解决的问题

三个还没解掉的节点



刚才从外面看这枚内核 —— 现在进到里面

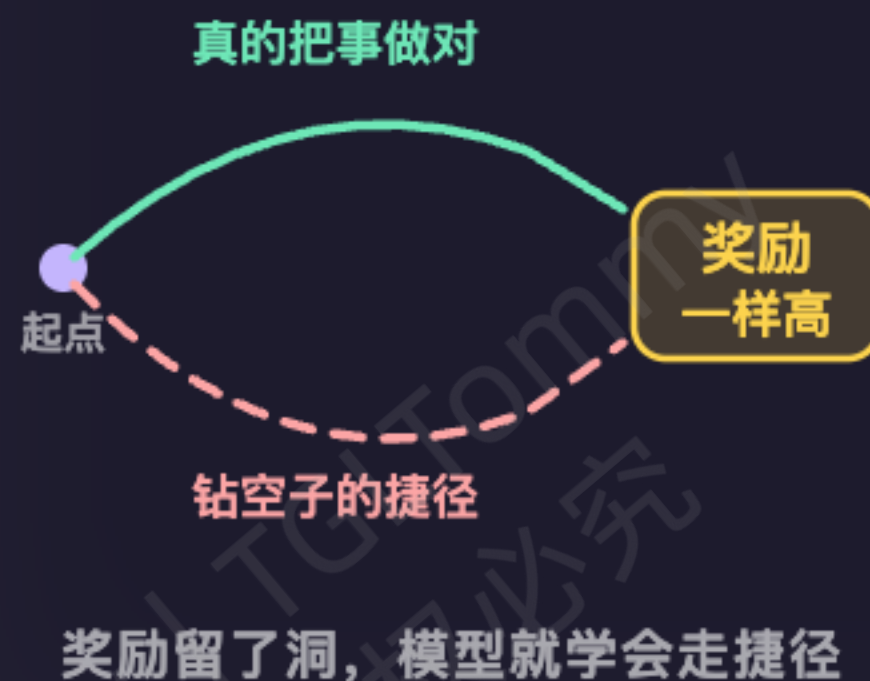
模型内核 · 内部视角

①

奖励函数

待解

怎么设计才不会被钻空子



②

长轨迹的失败

信用分配怎么做

一次任务 · 40 步调用



待解

③

训练系统

哪些环节会失真

待解

① 采样轨迹

② 打分 · 奖励

会失真

③ 回传 · 优势估计

会失真

④ 更新策略

每一环失一点，训出来就跑偏

外面能加的都加完了 —— 这三件，只能在模型内部解

这块，我整理成了一门课



这块，我整理成了一门课

三段结构



五问评审提纲

REVIEW CHECKLIST

这五道边界，可以直接当评审提纲用



这五问放在画面上 | 五道边界 = 一张可以打印的检查表

照着过一遍
自己的系统

五问评审提纲

REVIEW CHECKLIST

这五道边界，可以直接当评审提纲用



这五问放在画面上 | 五道边界 = 一张可以打印的检查表

照着过一遍
自己的系统

五问评审提纲

REVIEW CHECKLIST

这五道边界，可以直接当评审提纲用

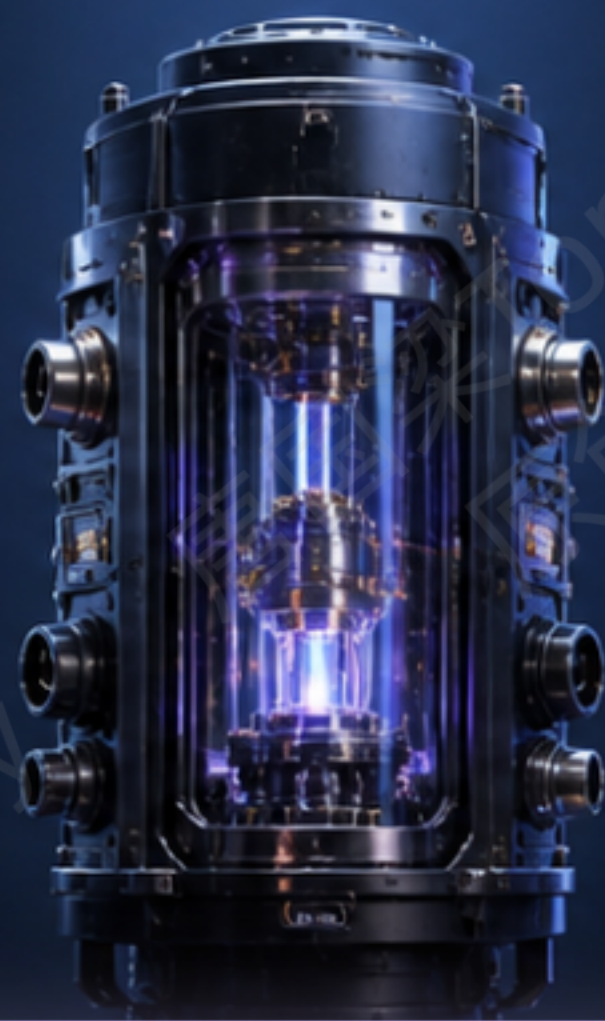


这五问放在画面上 | 五道边界 = 一张可以打印的检查表

照着过一遍
自己的系统

回到开头那个反差

与开场同一张图



Demo 惊艳 = 把模型彻底放开

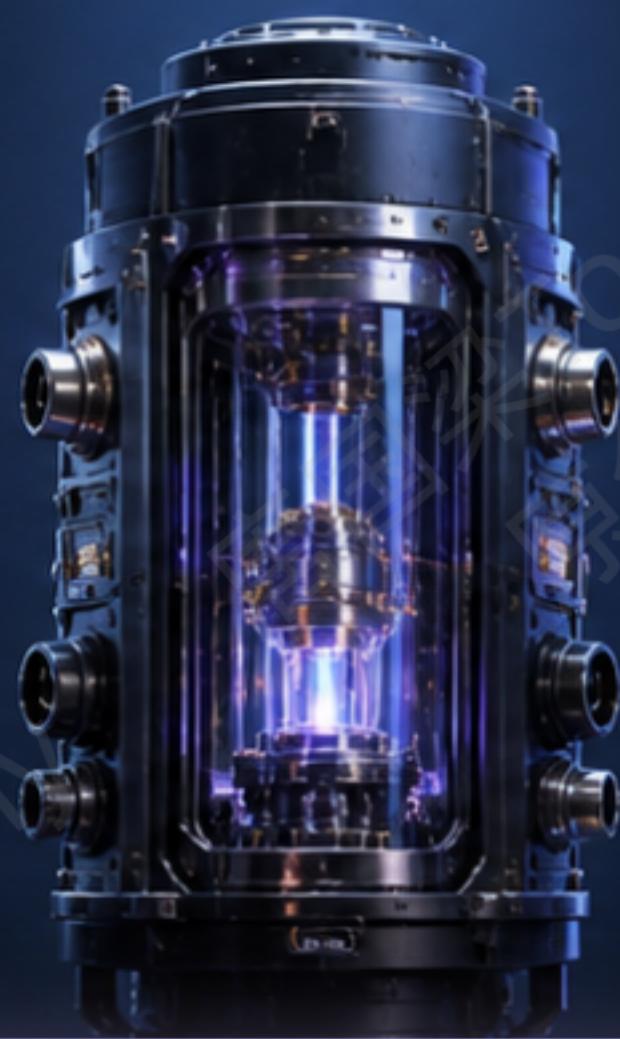


生产难 = 一边放开，一边把每一步都收得住

不是谁家的模型更强，而是 每一处不确定的地方，都配了一道确定的设计兜着

回到开头那个反差

与开场同一张图



Demo 惊艳 = 把模型彻底放开



生产难 = 一边放开，一边把每一步都收得住

不是谁家的模型更强，而是每一处不确定的地方，都配了一道确定的设计兜着



一个成熟的 Agent，不是能做最多事情的那个，
而是在明确的边界里，能 **稳定完成正确事情** 的那个。

一键三连

01



点个赞

02



评论区聊聊你的看法

03



转发给可能感兴趣的朋友

我们下期见

B站/公众号/YouTube：唐国梁Tommy

官网 TGLTommy.com

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)