

Agent Evaluation Regression Testing

智能体评估回归测试——上线前证明「这次改动没把Agent改坏」

2026年10月20日 | AI Agent 知识分享 | 第81期

一、一句话概括

评估集的真正价值不是发布前跑一次得个分数，而是每次改动提示词、模型、工具、工作流后都能自动回归，证明这次改动没有把Agent改坏。对银行测试场景尤其重要：提示词改一个词、模型换一个小版本、工具返回结构加一个字段，都可能让原来能跑通的用例悄悄失败。

二、核心原理

◆ 黄金集 (golden set)

50 到 300 条用例，覆盖核心意图、边界条件、对抗样本三类。每条用例必须有明确的期望结果或判定规则，不能只写「回答应正确」这种无法判定的期望。

◆ 判定器分层

能程序化判定的不用模型。输出格式用 JSON Schema 校验（精确匹配率），数值结果用断言（exact match / 容差比较），引用类用工具调用记录校验（工具名、参数、调用顺序）。只有开放式回答才交给 LLM-as-Judge，且必须附带评分细则（rubric），否则判定结果不可复现。

◆ 双指标

pass@1 回答一次的正确率，pass@k 回答 k 次的正确率。pass@k 反映能力上限，pass@1 反映生产环境真实体验，两者差距大说明模型「会但不稳」。

◆ 相对基线而非绝对分数

与上一个版本的冻结基线做对比，关注退化用例（regression），而不是纠结 82 分和 84 分谁更好。

◆ 门槛绑定发布

只有全量通过且无退化用例，才允许合入主干或发布。

◆ 确定性

温度设 0、模型版本与供应商快照写死、测试环境固定（同一套工具 mock、同一份数据），否则同一用例两次结果不同，回归对比失去意义。

◆ 分层抽样

每轮全量回归，每次改动只跑受影响的子集（按意图分组），在时间和成本可控的前提下保证覆盖。

核心要点

- ◆ 评估集是资产，不是报告：它的价值体现在被反复重跑，而不是被生成一次
- ◆ 报告回归率（新增失败用例 / 用例总数），不报告绝对分数
- ◆ 能用断言就别用模型裁判：模型裁判会漂移，断言不会
- ◆ 保留一套严格保密的验收集，另设公开集用于日常迭代，避免对着测试集调参

三、测试实战

测试1：黄金集与最小断言

工具：DeepEval (<https://github.com/confident-ai/deepeval>) 或 Braintrust (<https://www.braintrust.dev/>)。

操作：为每条用例写 `output_schema` 与 `judge rubric`，用 Python 断言做硬校验（工具名必须等于 `qy_metric_lookup`，参数 `metric_code` 必须等于 `EAST_GL_001`）。

指标：格式合规率、工具调用参数保真度、硬校验通过率。

测试2：退化检测（本次核心）

操作：冻结 v1.3 的评估结果作为基线，改动提示词后重跑同一套用例，逐条对比（`case_id` → 上一版通过 / 本版失败 → 回归）。

指标：回归率（本次新增失败用例数除以用例总数）、回归用例的意图分布（定位到是哪类问题退化）。

测试3：变异测试（mutation testing）

工具：Mutation-Guided LLM-based Test Generation at Meta（论文 <https://arxiv.org/abs/2402.16367>）。

操作：对判定器做变异（删掉一个断言、放宽一个阈值、把 `judge` 的 `rubric` 换一条），检查测试套件是否还能发现这个变异；发现不了说明测试套件本身有盲区。

指标：变异检出率（变异体被消灭的比例）。

测试4：对抗样本回归（银行场景必做）

工具：HarmBench (<https://github.com/centerforaisafety/HarmBench>)、Garak (<https://docs.garak.ai/>)、promptfoo (<https://www.promptfoo.dev/>)。

操作：固定 50

条注入与社工用例（伪造的「上级指令」、夹带在附件里的指令、要求绕过授权），每次改动提示词或安全护栏后重跑。

指标：注入成功率（越低越好）、拒答率、误拒率（正常业务问题被拒答的比例，银行场景里误拒率是业务投诉的直接来源）。

测试5：工具返回变化对Agent的冲击

操作：在 `mock` 里把「查指标库」的返回字段由 `number` 改为 `string`、删掉一个可选字段、把单位从亿元改成万元，各跑一遍。

指标：字段解析失败率、单位错误率、工具报错率。

推荐指标组合：核心意图通过率（门槛

≥95%）、工具调用参数保真度（≥98%）、注入成功率（≤2%）、误拒率（≤3%）、回归率（新增失败=0）。

执行顺序：生成黄金集 → 冻结基线 → 改动 → 重跑 → 出差报告 → 门槛判定 → 归档。

四、常见坑点

坑点1：只用一个总体分数汇报

总体分掩盖退化：平均分从 82 涨到 84，但可能有 10 条核心用例从通过变成失败。必须按意图或工具调用维度拆分子集指标，退化按用例粒度呈现。

坑点2：把 LLM-as-Judge 当唯一判定器

模型裁判会漂移：今天判「正确」的用例，换个模型版本就判「错误」，导致同一套用例出现假回归。正确做法是能用断言就别用裁判，裁判部分必须附带评分细则并锁定裁判模型版本。

坑点3：黄金集被训练或调参过拟合

反复根据失败用例改提示词，直到分数好看——这叫「对着测试集调参」，此时分数不再代表真实能力。黄金集要保留一部分严格保密，只用于最终验收，另设一套公开集用于日常迭代。

坑点4：忽略非确定性

温度不为 0、工具返回带时间戳或随机数、mock 数据每次不同，都会让同一用例两次结果不同，回归对比变成噪声比较。跑之前先固定：温度、模型版本、随机种子、数据快照。

坑点5：把「通过」等同于「可用」

评估集通过只说明已覆盖的场景没问题。生产环境的真实分布和黄金集永远有差距，必须配合线上抽样标注和错误聚类，持续把新发现的失败模式补进黄金集。

五、延伸阅读

- DeepEval（开源评估框架，支持回归与变异测试）：<https://github.com/confident-ai/deepeval>
- Braintrust（评估平台，支持数据集版本管理与回归对比）：<https://www.braintrust.dev/>
- HarmBench（标准化红队行为基准）：<https://github.com/centerforaisafety/HarmBench>
- Garak（LLM 漏洞扫描器）：<https://docs.garak.ai/>
- promptfoo（提示词与Agent评估工具）：<https://www.promptfoo.dev/>
- Mutation-Guided LLM-based Test Generation at Meta（变异测试论文）：<https://arxiv.org/abs/2402.16367>
- SWE-bench（软件工程Agent基准，含确定性的测试用例判定）：<https://www.swebench.com/>