

Set Contamination: When Your Test Set Leaks Into the Agent

评测集污染——你的测试集正在偷偷给 Agent 泄题

2026年10月26日 | AI Agent 知识分享 | 第86期

一、一句话概括

评测集是 AI 测试里唯一说真话的地方，而污染是它最隐蔽的失效模式：题目或答案在你不注意的时候流进了模型能看见的地方，指标依然漂亮，结论已经全错。污染不需要攻击者，一次顺手的 few-shot 取材、一次没去重的微调数据准备、一次把题库写进知识库的操作就够了。它不产生异常日志，不触发告警，只产生虚高的分数——而你会拿这个分数去做上线决策、去写汇报、去应付审计。所以评测集必须像生产数据一样被管理：版本化、访问控制、按周期换血，并且在每次重大变更后用一份从不参与任何环节的留出集做盲测。

二、核心原理

>> 泄漏的四条通道

1. Few-shot 示例回流：改 prompt 时从黄金集里挑几个例子贴进 system prompt，评测集当天作废。
2. 微调与偏好数据混入：SFT / RLHF 数据准备没做去重，评测集被当成训练集用了一遍。
3. 检索库回填：把评测题目和标准答案写进 RAG 语料，Agent 检索到答案直接输出，看起来像「真会了」。
4. 缓存与日志复用：prompt cache、trace 日志、会话记忆里存了答案，后续请求命中缓存即「答对」。

>> 污染的三种形态

题目泄漏：见过原题，模型记住的是这道题的解法。答案泄漏：题目变了但答案还在，模型记住的是那个字符串。分布泄漏：没见过具体题，但题型、难度、答案格式被摸清，等于提前划了重点。

>> 为什么污染极难发现

污染不产生异常信号：日志正常、延迟正常、成本正常，只有指标偏高。而且偏高得合理——污染带来的提升通常在 3 到 15 个百分点之间，看起来像一次成功的模型升级。越是认真做的项目越容易中招：评测做得越久，示例被复用的次数越多，污染是累积的。

>> 量化污染的三个抓手

重叠度：评测集与训练数据、语料、日志之间的 n-gram 或 MinHash 相似度，直接测「见过没有」。成员推断：原题与改写题的通过率差值，测「记的是题还是答案」。留出集盲测：一份从不参与任何环节的 holdout，测「真实水平还剩多少」。

>> 银行语境的额外要求

评测集属于测试数据资产，要纳入数据管理：谁能看、谁能导出、用了哪一版，都要有记录。对外汇报和监管报送的结论，必须能说明评测集与训练数据是隔离的，否则结论不可采信。

核心要点

- ◆ 污染不报错，只让指标虚高——这是最危险的失效模式
- ◆ 四条通道：few-shot 回流 / 微调混入 / 检索回填 / 缓存日志复用
- ◆ 三种形态：题目泄漏 / 答案泄漏 / 分布泄漏
- ◆ 黄金集必须版本化 + 访问控制 + 按季度换血
- ◆ 上线前用留出集做一次盲测，才算真实水平
- ◆ 三件套一起上：重叠扫描 + 成员推断 + holdout 盲测

三、测试实战

测试1：n-gram 重叠扫描，把泄漏通道逐个查一遍

- ◆ 工具：text-dedup (github.com/ChenghaoMou/text-dedup) 、 datasketch (github.com/ekzhu/datasketch)
- ◆ 操作：把评测集每条 query 切成 8-gram，对四个库分别建 MinHash 指纹——few-shot 示例文件、SFT/RLHF 的 jsonl、RAG 语料的 chunk、线上 trace 日志库；用 LSH 批量比对，Jaccard 大于 0.6 的标红
- ◆ 指标：泄漏率 = 命中条数 / 评测集总条数；目标小于 0.5%；超过 2% 整批冻结，禁止用于决策

测试2：成员推断攻击，测「答案泄漏」而不是「题目泄漏」

- ◆ 工具：自写脚本 + 模型 API (temperature 固定为 0，每个 query 采样 5 次)
- ◆ 操作：对每条评测 query 采样 5 次，统计输出与标准答案的关键片段重合率；同时构造 50 条「像但不是」的改写题作为对照组跑同样流程；若原题重合率显著高于改写题，说明模型记住的是答案
- ◆ 指标：MIA 记忆度 = 原题通过率 - 改写题通过率；大于 0.15 判为疑似泄漏，需人工复核

测试3：时间切片留出集，用三次切分看分数稳不稳

- ◆ 工具：pandas + 自写划分脚本，评测框架用 OpenAI Evals (github.com/openai/evals)
- ◆ 操作：按数据采集时间切，训练与微调只用 T 之前的数据，评测只用 T 之后的数据；分别在 T-30 天、T-60 天、T-90 天三个切点各跑一轮；若留出越晚分数越高，说明早期数据漏进了训练
- ◆ 指标：三个切片之间的分数方差小于 3% 视为干净；方差超过 5% 需追查数据来源

测试4：检索回填检测，专查 RAG 通道

- ◆ 工具：Elasticsearch 或 OpenSearch 做 BM25，Milvus 或 Qdrant 做向量检索
- ◆ 操作：把评测集每条 query 拿去 RAG 语料做 top-10 检索，检查返回 chunk 是否含标准答案的关键片段；再做反向验证——把标准答案原文丢进检索接口，看是否命中同一个 chunk
- ◆ 指标：答案命中率应小于 1%；任何一条命中即判定该条评测集作废并隔离，不得进入汇报数据

测试5：留出集盲测，上线前最后一道关

- ◆ 工具：pytest 编排 + Braintrust 或 LangSmith 做结果对比 (braintrust.dev)
- ◆ 操作：维护一份 100 到 200 条的 holdout 集，永不进入任何 prompt、微调、语料和缓存目录，单独存放并限制读取权限；每次模型或 prompt 变更后跑一次，与主评测集分数对比
- ◆ 指标：污染指数 = 主集分数 - holdout 分数；持续大于 5 个百分点说明主集已被污染，必须换血

四、常见坑点

坑点1：把评测集当 few-shot 素材库用

改

prompt

时顺手从黄金集里挑几个例子贴进去，评测集当天就废了，而且没有任何报错。正确做法：示例集与评测集物理隔离，示例从独立的训练池取，或固化成单独文件并纳入版本管理，任何人不得从评测集直接复制示例。

坑点2：只查题目重复，不查答案泄漏

n-gram

扫描很容易扫出「题目重复」，但更危险的是「答案泄漏」——题目被改写过，答案却原样留在语料或日志里，只做重叠扫描会漏掉这一类。正确做法：重叠扫描、成员推断、留出集盲测三件套一起上，任何单一手段都不足以排除污染。

坑点3：一套黄金集用一年不换血

评测集会被反复微调、反复缓存、反复被当示例引用，污染是累积的，用得越久越不能代表真实水平。正确做法：按季度换血，旧集转入训练池并保留使用记录，换血时间和原因写进评测报告——银行审计要看这一条。

坑点4：训练和评测共用同一套环境

同一个容器镜像、同一个 prompt cache 目录、同一套 trace 存储，泄漏会在你不知道的情况下发生，而且事后无法归因。正确做法：评测跑在独立命名空间和独立缓存前缀下，训练产物与评测资产的访问权限分开授权。

五、延伸阅读

- text-dedup（去重与重叠检测工具集）：<https://github.com/ChenghaoMou/text-dedup>
- datasketch（MinHash / LSH 大规模相似度检测）：<https://github.com/ekzhu/datasketch>
- 成员推断攻击原始论文（Shokri et al., 2017）：<https://arxiv.org/abs/1610.05820>
- HELM（斯坦福评测框架，含污染与泛化分析）：<https://crfm.stanford.edu/helm/>
- OpenAI Evals（评测集管理与留出集实践）：<https://github.com/openai/evals>
- Braintrust（Agent 评测平台，支持 holdout 与回归对比）：<https://www.braintrust.dev/>