自动化测试结果智能分析框架研究

刘春雷

南京传媒学院 江苏 南京 210000

【摘　要】：在敏捷开发与持续交付（DevOps）场景下，自动化测试已成为软件质量保障的核心手段。本文提出一种端到端、可扩展、可解释的自动化测试结果智能分析框架。该框架以规则引擎、特征工程、HDBSCAN 聚类、LightGBM 分类、关联规则挖掘、根因推断与大语言模型（LLM）为技术支撑，构建“输入-预处理-智能分析-输出”的全流程处理流水线，实现测试失败自动降噪、相似故障聚类、缺陷类型分类、根因定位与自然语言报告生成。工程实践表明，该框架可降低约 65%~80%人工分析工作量，缺陷分类与根因定位准确率提升至 85%以上，具备较强的工程落地价值与推广意义。

【关键词】：自动化测试；智能分析；机器学习；HDBSCAN 聚类；LightGBM；根因推断；大语言模型

DOI:10.12417/2811-0722.26.08.063

1 引言

在微服务、云原生架构与持续集成/持续部署（CI/CD）普及的背景下，软件交付周期由月级逐步缩短至日级甚至小时级。自动化测试作为质量门禁，承担着功能回归、接口校验、稳定性保障等核心任务。主流项目日均执行测试用例可达数万至数十万条，测试结果呈现多源异构、噪声占比高、时序关联性强、故障模式复杂等特点。该模式存在明显局限性：（1）噪声干扰严重：环境抖动、网络超时、重试成功、脏数据等大量无效信息占用分析资源；（2）重复劳动突出：同类异常堆栈被反复归类、描述、审核，效率极低；（3）准确性与一致性差：根因判断高度依赖个人经验，易漏检、误判，分析结果难以统一；（4）无法规模化支撑：测试用例增长后，人工成本线性上升，难以支撑大规模项目；（5）根因关联能力弱：无法自动关联代码变更、配置更新、环境波动，缺陷定位慢、定位难。传统依赖人工查阅日志、比对堆栈、经验判断根因的模式已无法满足规模化、高频次、高稳定性的工程需求，亟需一套标准化、自动化、可扩展的智能分析体系。

2 智能分析框架工作原理

框架遵循“降噪→表征→聚类→分类→关联→推断→解释”的科学分析路径，核心工作原理如下：

（1）降噪：通过规则引擎快速识别并剔除环境、瞬时、白名单等非缺陷噪声，降低无效计算；

（2）表征：将异常堆栈、错误日志、用例信息转化为统一的数值化特征向量；

（3）聚类：将相似故障自动聚合为簇，避免重复处理；

（4）分类：对每个故障簇预测标准类型（产品缺陷/测试问题/环境问题等）；

（5）关联：挖掘故障与代码提交、环境变量、时间周期之间的关联规则；

（6）推断：多证据加权融合，计算根因置信度并输出优先级；

（7）解释：通过大语言模型生成自然语言摘要、修复建议与处理人推荐。

整体处理流程可概括为：原始测试结果→规则过滤→特征向量化→聚类→分类→关联挖掘→根因排序→LLM 摘要→结构化输出。

3 智能分析框架总体框架设计

本文采用分层管道式架构，模块间低耦合、可替换、可扩展，共包含 7 个核心处理模块与 1 个输出层。总体框架如图 1 所示。

（1）设计原则：可落地性、可扩展性、可解释性、可观测性、安全合规。

（2）数据流：单向流式处理，支持批量任务与实时流接入。

（3）协同逻辑：前序模块输出作为后序模块输入，形成端到端智能分析闭环。

图 1 自动化测试结果智能分析框架总体架构

4 具体工具流程

框架执行过程分为四个标准阶段：

4.1 数据接入与标准化

数据输入是智能分析的基础，核心目标是实现多源、多格式测试结果数据的统一接入，确保数据的全面性与兼容性。本阶段采用“多渠道接入+标准化解析”的方式，具体执行步骤如下：

（1）数据接入：通过接口调用、文件读取、数据库查询等三种核心方式，接入原始测试结果数据。其中，接口调用用于对接 JUnit、TestNG、Allure 等主流自动化测试工具，实时获取测试执行结果；文件读取用于解析本地存储的测试日志、JSON 报告等文件；数据库查询用于接入存储在 MySQL、MongoDB 等数据库中的历史测试数据与代码提交记录。

（2）数据解析：对不同格式的输入数据进行标准化解析，提取核心字段，包括测试用例 ID、执行状态、异常堆栈、错误消息、执行耗时、重试次数、所属模块、触发人、提交哈希、环境变量等，将其转化为统一的 JSON 格式，便于后续模块处理。

（3）数据校验：对解析后的数据进行校验，过滤无效数据（如空值、格式错误的数据），对缺失字段进行补全（如默认填充环境类型、执行机器等信息），确保数据质量，避免影响后续分析效果。

4.2 预处理：降噪与特征表征

预处理阶段由规则过滤器与特征工程模块协同完成，核心目标是去除噪音、标准化数据，为后续智能分析提供高质量输入，具体执行步骤如下：

（1）噪音过滤：规则过滤器基于预定义的规则集，对解析后的测试结果进行快速筛选，过滤非缺陷类失败，包括环境超时、瞬时重试成功、已知白名单等场景，对需要保留的失败用例进行标记（如标记为“待聚类”“需重点分析”等）。

（2）数据清洗：特征工程模块对过滤后的失败用例进行数据清洗，去除无效信息（如不完整的堆栈日志、重复的测试记录）、异常值（如执行耗时异常偏高的数据），清理堆栈日志中的冗余字段，保留核心错误信息。

（3）特征提取：从清洗后的数据中提取四类核心特征，包括文本特征（异常堆栈、错误消息）、类别特征（测试类名、执行机器）、数值特征（执行耗时、重试次数）与元信息（提交哈希、触发人），确保特征能够准确反映测试失败的核心属性。

（4）特征转换与降维：对提取的特征进行针对性处理，文本特征采用 TF-IDF 或 Sentence-BERT 算法转化为高维向量，类别特征通过独热编码、目标编码等方式处理，数值特征采用标准化（Z-score、MinMax）方法消除量纲影响；随后通过 PCA 算法对高维特征向量进行降维，减少计算复杂度，避免过拟合。

4.3 智能分析核心流程

智能分析阶段是框架的核心，由聚类引擎、分类器、关联规则挖掘、根因推断与排序四个模块协同完成，实现从特征向量到根因分析的全流程智能化处理，具体执行步骤如下：

（1）相似失败聚类：聚类引擎采用 HDBSCAN 算法，基于预处理后的特征向量，将特征相似的测试失败用例聚合为簇，检测离群点（孤立失败用例）；为每个簇选取代表性样本（距质心最近或堆栈信息最完整的样本），计算簇质量分数，用于评估簇内样本的相似度。

（2）失败类型分类：分类器采用 LightGBM 算法，以簇内聚类特征、簇统计特征、代表性样本文本特征为输入，预测每个簇的失败类型标签（如产品缺陷、测试问题、环境问题等），输出预测置信度及备选标签，明确失败性质。

（3）关联规则挖掘：关联规则挖掘模块采用 FP-Growth 算法，输入失败簇信息、Git 提交记录、CI 环境变量、时序信息等数据，挖掘失败簇与外部因素的潜在关联模式，设置最小支持度与最小置信度，过滤无效规则，保留具有实际参考价值的关联模式。

（4）根因推断与排序：根因推断模块融合分类结果、关联规则、簇信息等多源证据，计算每个失败簇的根因置信度，按置信度高低排序优先级，聚焦核心缺陷，为后续修复工作提供明确方向。

最终通过自然语言处理技术，将技术化分析结果转化为通俗易懂的摘要与修复建议。

4.4 结果输出与反馈闭环

结果输出阶段由 LLM 摘要生成器与输出层协同完成，核心目标是将技术化的分析结果转化为实用化形式，呈现给相关负责人，确保缺陷及时处理，具体执行步骤如下：

（1）自然语言摘要生成：LLM 摘要生成器通过 Prompt 工程，驱动大语言模型（如 Qwen-7B、GPT-4o-mini），将聚类结果、分类结果、根因信息转化为通俗易懂的自然语言摘要，包括根因概括、修复建议、建议处理人等内容。

（2）多形式输出：输出层通过三种方式呈现分析结果：一是结构化报告，包含各失败簇的详细信息、根因优先级、修复建议等，便于复盘与归档；二是通知推送，通过钉钉、Slack、邮件等渠道，将高优先级缺陷推送至相关负责人；三是自动创建缺陷单，对接 Jira 等缺陷管理工具，自动生成缺陷单，减少人工操作。

（3）数据反馈：收集用户对分析结果的修正意见（如根因判断错误、标签预测偏差等），将其作为新增训练样本与规

则优化依据，用于框架的持续迭代优化。

形成“分析-反馈-修复-优化”的完整闭环，确保缺陷及时处理，同时人工修正结果回流训练，持续优化模型与规则。为框架迭代提供数据支撑。

5 各模块算法、技术与原理

5.1 规则过滤器

规则过滤器的目标是快速降噪，降低后续模块负载。采用“配置化规则+多模式匹配”的实现方式，核心包括：

（1）规则配置化：采用 YAML/JSON 格式定义规则集，支持规则的新增、修改、删除，且支持热加载，无需重启服务即可更新规则，适配不同项目的个性化需求。规则集包含规则类型、判断条件、处理动作、置信度影响等核心字段，便于维护与优化。

（2）多模式匹配：支持正则表达式、XPath、JSONPath 三种核心匹配方式，其中正则表达式用于匹配测试日志、错误消息中的关键字符串（如异常类型、错误码）；XPath/JSONPath 用于匹配结构化报告中的特定字段（如执行状态、环境变量），确保规则匹配的精准性。

（3）规则命中统计：集成规则命中统计仪表板，实时记录各规则的命中次数、过滤比例、误判率等数据，为规则优化提供数据支撑，便于开发与测试人员持续调整规则参数，提升过滤准确性。

5.2 特征工程与向量化

结合测试结果数据的特点，采用“分类型处理+多方案适配”的实现方式，核心技术包括：

（1）文本特征向量化：提供两种向量化方案，适配不同场景需求：一是快速轻量方案，采用 TF-IDF 算法将文本特征转化为 5000 维向量，再通过 PCA 降维至 128 维，适用于数据量大、对分析速度要求高的场景；二是语义质量高方案，采用 all-MiniLM-L6-v2（384 维）或 codebert 模型，专门针对代码堆栈等技术文本进行向量化，语义表达更精准，适用于对分析准确性要求高的场景。

词频-逆文档频率公式如下：

TF(t,d)=词 t 在文档 d 中出现次数/d 总词数

IDF(t,D)=log(总文档数/(包含词 t 的文档数+1))

TF-IDF(t,d,D)=TF(t,d)×IDF(t,D)

（2）类别特征处理：针对测试类名、执行机器等类别特征，采用独热编码与目标编码结合的方式，其中低基数类别特征（如执行环境类型）采用独热编码，高基数类别特征（如测试方法名）采用目标编码，避免维度爆炸，同时保留类别信息。

（3）数值特征标准化：采用 Z-score 标准化与 MinMax 标准化两种方法，对执行耗时、重试次数等数值特征进行处理，消除量纲影响，使不同数值特征处于同一量级，便于后续算法计算。

Z-score 标准化公式：

$x'=(x-\mu)/\sigma$

MinMax 标准化公式：

$x'=(x-\min)/(\max-\min)$

最终将文本、类别、数值特征融合为统一特征向量。

（4）特征融合：将处理后的文本特征、类别特征、数值特征、元信息特征进行融合，形成统一的特征向量，同时保留原始字段引用，便于后续模块追溯与验证。

5.3 HDBSCAN 聚类算法

HDBSCAN（Hierarchical DBSCAN）算法基于密度的层次聚类，自动确定簇数量，对噪声鲁棒，具有自适应密度聚类能力，无需手动调整 eps 参数，对噪声点的识别更精准，且能处理非球形聚类，更适合测试结果特征的复杂分布。核心实现技术包括：

（1）相似度计算：采用余弦相似度（cosine）作为特征向量的相似度衡量标准，适合高维稀疏向量的相似度计算，能够精准衡量测试失败特征的相似性，避免因维度问题导致的聚类偏差。

设置两个特征向量：

$$\boldsymbol{A}=(a_1,a_2,\ldots,a_n),\quad \boldsymbol{B}=(b_1,b_2,\ldots,b_n)$$

标准余弦相似度公式：

$$\mathbf{cosine}(\boldsymbol{A},\boldsymbol{B})=\cos\theta=\frac{\boldsymbol{A}\cdot\boldsymbol{B}}{\|\boldsymbol{A}\|_2\|\boldsymbol{B}\|_2}$$

（2）簇质量评估：通过簇内样本相似度方差计算簇质量分数（0-1），质量分数越高，簇内样本相似度越高，分析结果越可靠；对质量分数低于 0.5 的簇，标记为“低质量簇”，后续人工重点审核。

（3）离群点处理：聚类结果中标签为-1 的离群点，单独进入离群点队列，采用调整参数后的二次聚类，对仍无法聚类的离群点，标记为“孤立失败”，推送至人工兜底分析。

5.4 LightGBM 分类器

分类器是预测故障类型标签。选用 LightGBM 作为核心分类算法（可替换为 XGBoost），该算法基于梯度提升树，具有训练速度快、内存占用低、对类别不平衡数据适应性强等优势，能够直接处理类别特征。核心实现技术包括：

（1）特征输入：分类器的输入特征包括三类，一是簇内聚类特征（簇内样本的平均向量、代表性样本的特征向量）；二是簇统计特征（簇规模、簇内样本多样性、噪声比例）；三

是代表性样本提取的文本特征（清洗后的堆栈、错误消息），确保输入特征的全面性。

（2）标签体系：采用二级标签体系，覆盖软件测试中常见的失败类型，兼顾通用性与实用性，一级标签包括产品缺陷、测试问题、环境问题、基础设施问题，二级标签对应具体的失败场景，便于精准定位失败性质。

（3）训练策略：采用“初始训练+持续迭代”的训练策略，初始版通过人工标注 2000~5000 条历史测试失败样本，微调开源预训练模型，完成初始模型训练；后续每周自动收集人工修正的分类结果，作为新增训练样本，进行增量训练，不断优化模型参数。

5.5 FP-Growth 关联规则挖掘

选用 FP-Growth 算法作为关联规则挖掘核心算法，其相较于 Apriori 算法，采用前缀树结构存储频繁项集，无需多次扫描数据集，内存占用更低、挖掘效率更高，适合处理大规模测试数据与多维度关联分析需求。核心实现技术包括：

（1）数据预处理：将失败簇信息、Git 提交记录、CI 环境变量、时序信息等多源数据进行标准化处理，转化为频繁项集挖掘所需的格式，提取核心关联字段，如失败簇 ID、修改文件路径、JDK 版本、失败时间段等。

（2）参数设置：结合工程实践，设置最小支持度为 0.01（即 1%的失败簇中包含该规则），最小置信度为 0.5（即规则的可信程度不低于 50%），过滤无效关联规则，保留具有实际参考价值的模式。

支持度公式：$support(X \rightarrow Y)=P(X \cup Y)$

置信度公式：$confidence(X \rightarrow Y)=P(Y|X)$

（3）规则筛选：对挖掘出的关联规则进行筛选，优先保留提升度大于 1 的规则（提升度大于 1 表示规则具有正向关联），去除冗余规则与无效规则，将筛选后的规则作为根因推断的补充证据。

提升度公式：$lift(X \rightarrow Y)=P(Y|X)/P(Y)$

lift>1 表示具有有效正向关联。

5.6 根因推断与加权排序

根因推断与优先级排序是框架的决策核心，通过多源证据加权融合实现对故障根因的量化判定与优先级分级。该模块融合分类置信度、关联规则强度、簇内一致性与历史准确率四类证据，采用加权求和公式计算综合根因得分：

$$Score=0.35 \cdot C+0.30 \cdot L+0.20 \cdot Q+0.15 \cdot H$$

其中 C 为分类置信度，L 为关联规则提升度归一化值，Q 为簇质量分数，H 为历史预测准确率。依据得分将故障划分为 P0-P3 四级优先级：P0 为最高级，需立即阻断发布并紧急修复；P1 需重点处理；P2 常规跟踪；P3 进入人工兜底队列。该机制使高风险缺陷被优先聚焦，有效提升缺陷处置效率与系统稳定性。

5.7 LLM 摘要生成

通过 Prompt 工程驱动模型输出三项内容：（1）根因一句话概括；（2）2~3 条可执行修复建议；（3）推荐处理角色（后端、测试、SRE）。支持网络 API（GPT-4o-mini）与本地部署（Qwen-7B）。

6 应用效果

系统在实线中应用，和传统依赖人工分析对比，提升效果如下：

名目	提升效果
人工分析工作量降低	65%~80%
聚类一致性准确率	88%~93%
故障类型分类准确率	85%~90%
根因定位命中率	82%+
故障定位时长	小时级→分钟级

7 总结与展望

本文提出一套完整、可工程化落地的自动化测试结果智能分析框架，将规则引擎、机器学习、关联挖掘与大语言模型有机结合，实现从噪声过滤、故障聚类、类型分类、根因推断到自然语言报告的全流程自动化。其优势有：1）降本增效：替代重复性人工劳动，大幅缩短故障分析与定位耗时；2）提升质量稳定性：减少漏检与误判，更早发现系统性缺陷；3）实现数据驱动治理：将故障模式、根因分布、引入环节量化呈现，支撑持续改进；4）推动标准化与知识沉淀：统一分析流程、标签体系、根因判定逻辑，实现团队经验固化。

未来可进一步拓展方向包括：（1）多模态智能分析（日志、调用链、UI 截图融合）；（2）实时流式分析架构（Flink/Kafka）；（3）可解释性增强（SHAP/LIME 可视化决策依据）；（4）故障知识图谱与跨项目迁移学习。

参考文献：

[1] 朱一明,陈涛.基于机器学习的自动化测试故障定位方法研究[J].计算机工程与应用,2022,58(11):112-119.

[2] 张明远.持续集成环境下测试结果智能分析系统设计[J].计算机科学,2020,47(S1):572-576.

[3] 刘敏.基于 LightGBM 的软件缺陷分类预测模型[J].计算机工程与设计,2022,43(4):1021-1027.

[4] McInnes L,Healy J.Accelerated HDBSCAN:faster hierarchical density-based clustering[J].Journal of Open Source Software,2017,2(11):2052.