软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(8):3180−3204 [doi: 10.13328/j.cnki.jos.007594] [CSTR: 32375.14.jos.007594]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

大语言模型智能体在软件系统根因分析中的应用综述*

康俊驰[1,2], 丁 博[1,2], 冯大为[1,2], 翟远钊[1,2], 张迅晖[1,2], 王怀民[1,2]

[1](国防科技大学 计算机学院, 湖南 长沙 410073)

[2](复杂关键软件环境全国重点实验室, 湖南 长沙 410073)

通信作者: 翟远钊, E-mail: yuanzhaozhai@nudt.edu.cn

摘　要: 在现代软件系统, 尤其是云计算和微服务系统中, 根因分析是保障系统稳定性和高效运行的关键技术. 大语言模型由于其强大的自然语言处理和数据分析能力, 为根因分析提供了新的解决方案. 基于大语言模型的智能体在大语言模型的基础上, 为根因分析带来了更高的自动化程度和更精准的问题定位能力. 然而, 尽管已有相关研究探讨了大语言模型在根因分析中的应用, 但关于大语言模型智能体在根因分析的研究仍然处于早期阶段. 聚焦于大语言模型智能体在云计算和微服务系统根因分析的研究现状进行全面的分析和总结. 主要内容包括: (1) 概述基于大语言模型的智能体框架构成和根因分析中的数据使用; (2) 从信息获取、根因定位和效果评估这 3 个主要流程系统地分析大语言模型智能体在根因分析中的应用方式; (3) 探讨大语言模型智能体技术在根因分析任务中面临的主要挑战和未来的发展方向.

关键词: 根因分析; 大语言模型智能体; 软件系统

中图法分类号: TP311

中文引用格式: 康俊驰, 丁博, 冯大为, 翟远钊, 张迅晖, 王怀民. 大语言模型智能体在软件系统根因分析中的应用综述. 软件学报, 2026, 37(8): 3180–3204. http://www.jos.org.cn/1000-9825/7594.htm

英文引用格式: Kang JC, Ding B, Feng DW, Zhai YZ, Zhang XH, Wang HM. Survey on Application of LLM-based Agents in Root Cause Analysis of Software Systems. Ruan Jian Xue Bao/Journal of Software, 2026, 37(8): 3180–3204 (in Chinese). http://www.jos.org.cn/1000-9825/7594.htm

Survey on Application of LLM-based Agents in Root Cause Analysis of Software Systems

KANG Jun-Chi[1,2], DING Bo[1,2], FENG Da-Wei[1,2], ZHAI Yuan-Zhao[1,2], ZHANG Xun-Hui[1,2], WANG Huai-Min[1,2]

[1](College of Computer Science and Technology, National University of Defense and Technology, Changsha 410073, China)

[2](State Key Laboratory of Complex & Critical Software Environment, Changsha 410073, China)

Abstract: Root cause analysis plays a critical role in ensuring the stability and efficiency of modern software systems, particularly in cloud computing and microservice-based systems. Large language models (LLMs), with their powerful natural language processing and data analysis capabilities, have provided new solutions for root cause analysis. LLM-based agents have further enhanced root cause analysis capabilities, such as higher levels of automation and more precise problem localization. While existing research has explored the application of LLMs in root cause analysis, research on LLM-based agents is still at an early stage. To address this gap, this survey provides a comprehensive analysis and summary of current research on LLM-based agents for root cause analysis in cloud computing and microservices systems. The main contents include (1) an overview of the architecture of LLM-based agents and the types of data involved in root cause analysis; (2) a systematic analysis of how LLM-based agents are applied to root cause analysis through the main stages of information collection, root cause localization, and effectiveness evaluation; (3) an exploration of the main challenges and future directions of LLM-based agent technologies in root cause analysis tasks.

Key words: root cause analysis (RCA); LLM-based agent; software system

---

* 基金项目: 并行与分布计算全国重点实验室开放基金 (2024-KJWPDL-02); 国家重点实验室课题 (231-HF-D04-01)
本文由“大模型与软件工程”专题特约编辑聂长海教授、王璐副教授、王莹教授、姜艳杰副教授、张敏灵教授推荐.
收稿时间: 2025-09-07; 修改时间: 2025-10-28; 采用时间: 2025-12-08; jos 在线出版时间: 2025-12-24
CNKI 网络首发时间: 2026-02-04

1 引 言

随着信息技术的迅猛发展, 软件系统已成为支撑现代社会正常运行的核心基础设施, 从金融交易[1]到医疗保健[2], 从智能制造[3]到智慧城市[4], 无不依赖于稳定可靠的软件系统. 然而, 随着软件系统规模的扩大和复杂性的提升, 系统故障发生的频率逐渐增加, 导致的影响也越来越严重. 软件系统的故障不仅会导致直接的经济损失, 如系统停机带来的业务中断、维修成本增加等, 还可能引发间接的连锁反应, 如用户信任度下降、企业声誉受损等. 在关键领域, 如航空航天和核能管理等, 软件故障甚至可能威胁到人民生命安全和社会稳定[5]. 因此, 如何有效地识别、诊断和解决软件故障, 成为一个亟待解决的问题. 在众多软件系统类型中, 云计算和微服务系统是当前最具代表性的架构形态. 其一, 微服务将复杂应用拆解为大量松耦合的服务单元, 提升了系统的灵活性与可扩展性, 但同时引入了庞大的服务依赖图和频繁的交互链路, 使得故障传播路径更加复杂. 其二, 云计算平台则具备大规模、多租户和动态调度等特性, 导致运行环境高度动态化, 服务实例随负载变化频繁扩缩容, 传统静态假设下的诊断方法往往失效. 因此, 云计算和微服务系统中的根因分析不仅面临规模大、依赖复杂、动态性强等挑战, 也直接关系到大规模生产系统的稳定性与企业服务连续性. 因此, 本文特别聚焦于云计算和微服务系统中的根因分析研究.

根因分析 (root cause analysis, RCA)[6−15]可以为软件系统提供快速故障定位、故障诊断和制定解决方案等能力, 是保障软件系统稳定运行的关键技术. 典型的根因分析流程包括数据收集、根因定位和效果评估等阶段. 图 1 展示了一个根因分析的典型案例, 从前期收集系统错误信息, 到分析后确定根因并最终给出分析效果评估, 完成了软件系统报错到解决的整体流程. 具体来说, 数据收集部分关注如何构建一个高效的数据处理框架, 从海量的软件系统数据中提取出对根因分析任务有重要作用的信息; 根因定位部分关注大语言模型智能体在得到所需信息后, 如何有效利用上述信息准确识别系统故障的根因; 而鉴于根因分析任务的复杂性, 效果评估部分的重点在于如何设计一套合理的评价体系以客观评估根因分析的结果.

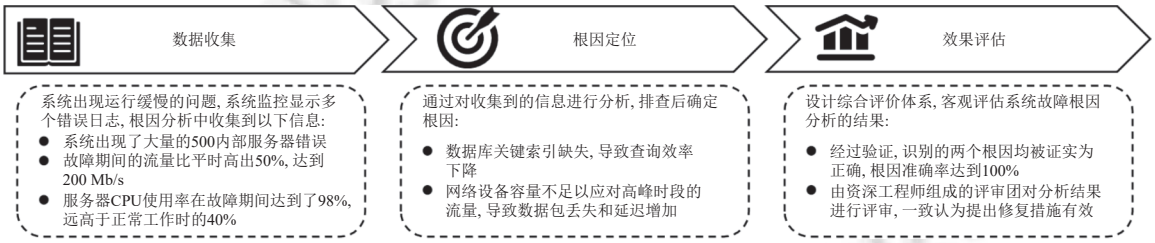

图 1 根因分析主要流程

传统软件故障根因定位的方法包括利用关联规则挖掘、基于启发式搜索、基于机器学习或者基于深度学习等数据驱动方法[16], 主要通过从系统生成的数据中提取特征并建立模型来识别故障的模式和潜在的根因. 然而, 这些方法存在一些局限性, 例如需要复杂的特征工程、缺乏处理不同任务时的灵活性、模型的可迁移性有限以及自动化程度较低[17,18]等. 基于大语言模型 (large language model, LLM) 的根因分析方法则利用大语言模型强大的自然语言处理能力, 能够更有效地处理日志等非结构化数据[19,20]. 例如, 大语言模型可通过自动解析日志中的关键事件 (如错误代码“HTTP 500”与数据库连接超时的关联), 生成包含故障时间、可能导致故障的系统组件集和修复建议的自然语言报告, 帮助运维人员将平均故障修复时间降低 40%[21]. 但基于大语言模型的方法通常仅作为信息提取和汇总的工具, 缺乏面向目标的任务自主规划和执行能力. 为了解决上述问题, 近年来兴起的大语言模型智能体技术[22,23]为根因分析任务提供了新的可能. 大语言模型智能体在继承语言模型的语言理解和生成能力的基础上, 结合根因分析实际场景, 进一步引入了长期记忆机制、外部工具集成能力以及多步规划与推理机制. 与传统方法及基于大语言模型的方法相比, 基于大语言模型智能体的根因分析在以下 3 个关键维度具有独特的优势, 如表 1 所示.

- 根因识别的准确性与可解释性方面: 传统的根因分析方法通常缺乏自主性和规划能力, 它们依赖于预设的

规则或模型, 无法覆盖组件间的因果关系. 基于大语言模型的根因分析虽然通过利用自然语言理解和生成能力, 能够进行基本的故障分析和报告生成, 但推理链条通常不透明. 而基于大语言模型智能体的根因分析则可以根据系统状态和环境变化自主制定和调整故障处理计划, 逐步构建因果链并输出自然语言解释, 增强了根因分析的准确性和可解释性[24−26].

表 1 根因分析能力维度对比

对比维度	传统方法	基于大语言模型方法	基于大语言模型智能体方法
根因识别的准确性与可解释性	规则匹配, 推理链有限	生成式推理, 过程不透明	因果推理, 过程可追溯
多源数据协同与自动化能力	人工脚本依赖, 自动化程度低	支持非结构化, 自动化程度中等	多源融合, 自动化程度高
学习与适应历史故障能力	无学习机制	上下文记忆依赖	长期记忆与经验复用

● 多源数据协同与自动化能力方面: 传统方法中工程师通常会编写脚本来自动化进行一些诊断和修复的过程或者使用日志分析工具来协助工程师分析系统日志, 通过关键词搜索和正则表达式匹配等方式快速定位到可能的问题点. 但这类传统方法通常需要较多的专业知识和相对繁琐的人工操作. 基于大语言模型的根因分析虽然利用其自然语言处理能力可以处理非结构化文本, 降低了人工介入的程度, 但通常仍需要人工干预才能完成故障修复, 自动化程度方面存在上升空间. 而基于大语言模型智能体的根因分析则可以使用 API 等方式集成外部工具[27], 例如自动化执行测试工具、配置管理工具和监控工具, 实现故障修复的自动化[28−31].

● 学习与适应历史故障能力方面: 传统方法往往忽视历史数据的利用, 缺乏记忆能力而无法从历史故障中学习, 从而导致其故障处理的效率较低. 基于大语言模型的根因分析主要依赖于当前的输入数据进行分析, 虽然可以通过少样本学习[32]的方式临时提供给大语言模型类似的历史故障信息, 但是其记忆能力是短暂的, 只在当前场景有效. 而基于大语言模型智能体的根因分析则可以存储和检索历史故障信息、系统配置和运行状态, 从而更好地理解系统历史状态和当前状态. 例如在分析故障原因时, 通过引入历史故障向量数据库, 大语言模型智能体可以在历史记录中找到参考排查路线, 从而有效提高根因分析的准确性.

当前有少量基于大语言模型进行根因分析的综述文献[33−36], 本文与现有综述的比较如表 2 所示. 首先, 现有综述对于根因分析过程中包括固定程序收集与智能体自动收集在内的非人工收集部分的分析尚不充分, 而本文弥补了这一不足. 其次, 目前很少有综述针对大语言模型智能体进行根因定位的技术路线作出总结, 多涉及传统机器学习的方法或基于大语言模型微调的方法, 而本文针对基于大语言模型智能体进行根因分析的工作进行了多角度的充分讨论. 最后, 现有综述对于根因分析效果的评估也尚未进行充分的探索, 当前领域内的工作除了 BLEU[37]和 NUBIA[38]等评估文本生成质量的指标外, 还会使用实际业务相关的评估指标以及利用人类和大语言模型对根因分析效果进行主观评估, 本文针对这一点进行了细致的分析.

表 2 本文与现有综述的比较

文献	数据收集			根因定位			效果评估		
	人工收集	程序收集	智能体收集	传统方法定位	模型微调定位	智能体定位	客观评估	人类评估	大模型评估
本文	√	√	√	√	√	√	√	√	√
文献[33]	√	√	×	√	√	×	√	√	×
文献[34]	√	√	×	√	√	×	√	√	×
文献[35]	√	√	×	√	√	√	√	√	×
文献[36]	√	×	×	×	×	×	√	×	×

为了保证综述的系统性与完整性, 本文在撰写过程中对相关文献进行了系统收集与筛选. 本文所收集的文献主要来自 ACM Digital Library、IEEE Xplore、SpringerLink、ScienceDirect 以及中国知网 (CNKI) 等数据库, 同时参考了部分 arXiv 预印本与开源社区成果. 检索关键词包括“root cause analysis”“large language model”“agent”“multi-agent”“cloud system”和“microservice”等, 通过交叉组合进行搜索. 本文主要聚焦 2022–2025 年间的研究成

果. 此外, 我们也引用少量基于大语言模型微调的研究作为方法演进的参考. 最终, 经过初筛以及全文阅读, 我们纳入了 44 篇相关文献, 并按照数据获取、根因定位和效果评估这 3 个主要流程进行整理与归纳, 得到基于大语言模型的根因分析代表性方法如表 3 所示.

表 3　基于大语言模型的根因分析代表性方法

分析流程	主要方法	代表工作
数据收集	人工收集	Ahmed等人[39]、Zhang等人[40]、Xpert[41]、Hamadanian等人[42]
	固定程序收集	RCACopilot[43]、Flow-of-Action[44]、ProvTalk[45]、Huang等人[46]、MRCA[47]、Kuang等人[48]、ChangeRCA[49]、Eadro[50]、AIOpsLab[51,52]、ARCAS[53]、COMET[54]、MonitorAssistant[55]、Jin等人[56]
	智能体自动收集	RCAgent[57]、Roy等人[58]、mABC[59]、D-Bot[60]、RCA-agent[61]
根因定位	基于大语言模型微调的方法	Ahmed等人[39]、Huang等人[46]、Kuang等人[48]、Jin等人[56]、Ezukwoke等人[62]、Razouk等人[63]、LLM4MST[64]、Shehu等人[65]、Islam等人[66]
	基于单智能体的方法	Xpert[41]、Roy等人[58]、Ojuolape等人[67]、LLM-TSFD[68]、Siddeshwar等人[69]、AIOpsLab[51,52]、ARCAS[53]、COMET[54]、Sarda等人[70]、Sun等人[71]、Goel等人[72]、LLmiRQ[73]、LasRCA[74]、Vidyaratne等人[75]、FaultExplainer[76]、FLASH[77]、Jambigi等人[78]
	基于多智能体的方法	Hamadanian等人[42]、Flow-of-Action[44]、MonitorAssistant[55]、RCAgent[57]、mABC[59]、D-Bot[60]、RCA-agent[61]、Cloud Atlas[79]、Huang等人[80]
效果评估	客观评估指标	Flow-of-Action[44]、ChangeRCA[49]、AIOpsLab[51,52]、mABC[59]、D-Bot[60]、LLM4MST[64]、Siddeshwar等人[69]、LLmiRQ[73]、Vidyaratne等人[75]
	人类评估指标	Ahmed等人[39]、RCACopilot[43]、ProvTalk[45]、Huang等人[46]、MonitorAssistant[55]、Jin等人[56]、RCAgent[57]、Roy等人[58]、mABC[59]、D-Bot[60]、Goel等人[72]、FLASH[77]、Herrmann等人[81]
	大语言模型评估指标	Zhang等人[40]、RCAgent[57]、Herrmann等人[81]、LM-PACE[82]

为了更直观地展示文献的时间分布情况, 图 2 展示了本文所纳入的 2022–2025 年间研究的数量统计. 从图中可以看出, 相关研究在 2022 年仅有少量探索性工作, 2023 年数量开始增加, 而 2024 年出现爆发式增长, 共收录了 31 篇, 是目前该方向的集中研究期. 进入 2025 年以来, 已有 4 篇工作发表, 显示该研究方向仍在持续发展之中. 这一趋势表明, 大语言模型智能体在软件系统根因分析中的应用正迅速成为学术界与工业界关注的热点, 未来仍具有广阔的发展空间.

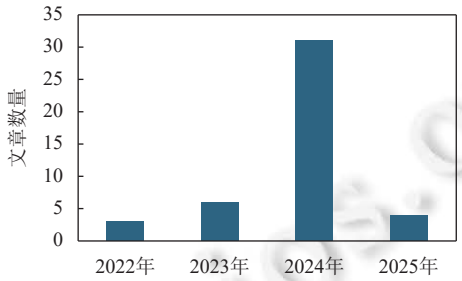

图 2　文章数量分布柱状图

本文第 2 节梳理大语言模型智能体和根因分析的基本知识, 包括大语言模型智能体整体组成和应用以及根因分析中涉及的重要数据来源. 第 3 节分析数据收集部分的应用现状, 从人工收集、固定程序收集到智能体自动收集这 3 个阶段进行梳理. 第 4 节探讨根因定位核心部分的研究方法, 分析传统方法的现状和不足作为背景, 梳理基于大语言模型微调、基于单智能体和基于多智能体的根因定位这 3 条不同技术路线的详细方法. 第 5 节则是针对根因分析具体场景, 从效果评估的角度讨论了如何有效且客观地对根因分析的效果进行全面评估. 第 6 节从根因分析领域本身的挑战和使用大语言模型智能体进行根因分析两部分出发, 提出当前面对的挑战及未来的发展方向. 最后一节总结全文并得出结论.

2 基础知识

为了更好地梳理大语言模型智能体在根因分析场景下的研究现状, 本节首先描述了基于大语言模型的智能体的基本概况、智能体与检索增强生成 (retrieval augmented generation, RAG) 技术[83]的结合以及单智能体系统与多智能体系统在根因分析中的应用. 此外, 本节还说明了根因分析过程中涉及的日志 (log)、调用链 (trace) 和指标 (metric) 这 3 部分可观测数据的概念以及文档信息在大语言模型进行根因分析中的重要作用.

2.1 基于大语言模型的智能体

随着大语言模型的广泛使用, 以大语言模型为核心的智能体系统也得到了广泛的发展. 基于大语言模型的智能体作为一种智能实体, 由 3 部分内容具体实现: 记忆、工具和规划. 检索增强生成技术的融入, 为大语言模型智能体提供了获取和利用外部知识的能力, 提升了其在特定领域任务中的表现, 拓宽了大语言模型智能体在实际应用中的范围. 同时, 从智能体数量的角度来分析, 可以将智能体系统具体分为单智能体系统和多智能体系统两种类型. 单智能体系统简洁高效, 但处理复杂问题能力有限, 而多智能体系统通过协同作业的方式, 展现了更高的灵活性和适应性, 能够有效应对复杂场景[84].

2.1.1 智能体整体框架

如图 3 所示, 基于大语言模型的智能体作为一种具有自主性和反应性的智能实体, 以大语言模型为核心单元, 同时围绕其构建 3 项关键能力组件[85]: 记忆、工具和规划. 记忆使得智能体可以保留与外部交互产生的信息. 参考人类记忆的种类可以将智能体的记忆也分为短期记忆和长期记忆两种. 短期记忆指的是只在当前场景下可以提供给智能体的额外信息. 上下文学习 (in-context learning, ICL)[86]就是一种短期记忆的实现, 在无需调整模型权重的条件下, 只向大模型提供一些输入输出示例, 即使用上下文学习中的少样本学习方法, 就可以使大模型展示出新的能力. 而长期记忆为智能体提供了长时间保留和反复调用信息的能力, 通常通过借助外部向量存储工具和高效的检索机制实现. 工具使得智能体可以在不调整本身权重的情况下获取外界信息, 通过使用 API 等方式调用外部接口实现代码执行、访问特定信息源数据等能力, 是对大模型本身的一种能力扩展. 规划使得智能体可以分解复杂任务并且针对任务进行规划. 最常见的一种方式是使用思维链 (chain of thought, CoT) 技术[87], 模型通过分步思考的方式, 在推理阶段将困难任务分解为众多较小且较为简单的子任务, 并同时解释每一个子任务的功能, 从而更好地控制整体推理流程向前推进.

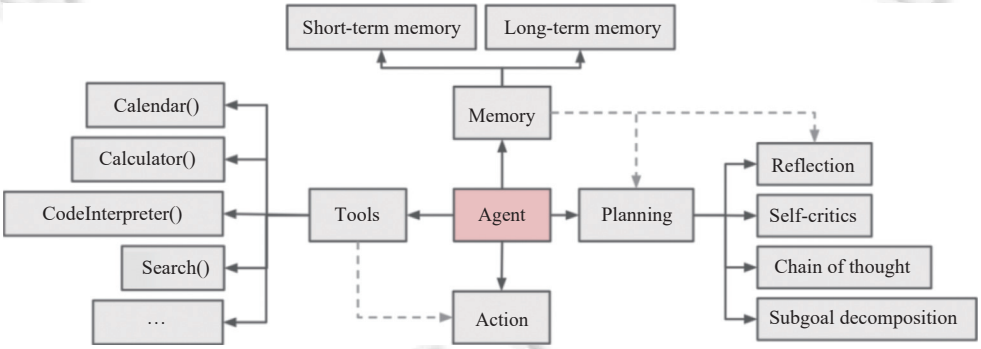

图 3 智能体整体组成[85]

2.1.2 智能体与检索增强生成

检索增强生成是一种结合了检索和生成技术的自然语言处理技术, 其核心思想是利用检索过程在上下文中加入检索到的相关信息以提高大语言模型的输出质量. 检索增强生成框架通常遵循索引、检索和生成的流程: 首先, 索引工作从清洗和提取多样化格式的原始数据开始, 将其转换为统一的纯文本格式并分割成更小的块以适应语言模型的上下文限制. 这些块被编码成向量表示并存储在向量数据库中, 为后续的检索阶段提供高效的相似性搜索基础. 在检索阶段, 当接收到用户查询时, 检索增强生成技术将查询转换为向量表示, 并与索引语料库中的块进行

相似度计算, 优先检索出与查询最相似的前 k 个块. 这些块作为扩展上下文加入生成阶段的提示中. 在生成阶段, 大语言模型负责根据提出的问题和选定的文档合成的连贯提示来制定回答. 这个过程涉及深层次的语义理解和上下文融合, 大语言模型智能体利用其先进的自然语言处理能力, 分析检索到的文档块, 提取关键信息, 并与用户查询意图相对应. 对于大语言模型智能体来说, 检索增强生成技术克服了原有大语言模型缺乏特定领域知识和知识库难以更新的问题[88], 同时一定程度上缓解了大语言模型的幻觉问题, 从而提供更加准确的故障处理建议, 减少了传统大模型在知识更新和准确性方面的局限性.

2.1.3 单智能体系统与多智能体系统在根因分析中的应用

在根因分析的实际应用中, 单智能体系统通过其高度集中的处理能力, 能够深入挖掘和分析数据, 以识别软件系统问题的主要根源. 大语言模型在单智能体系统中扮演着关键角色, 它通过精确理解问题上下文, 运用强大的推理能力, 对潜在的原因进行逐一排查, 可以为用户提供清晰直接的根因分析结果[57]. 然而, 这种分析往往受限于模型自身的知识边界和单一视角, 可能无法全面覆盖复杂问题背后的多重因素. 同时, 在实际故障处理工作中, 单智能体的方法会出现指令遵循方面的不稳定以及在多步骤分析中局限于细节处理而忘记整体规划的问题.

相比之下, 多智能体系统在根因分析中展现出更为全面的视角和协同处理的能力. 多个智能体通过协作、竞争以及两者结合的方法, 从不同的角度对同一个问题进行分析, 通过共享信息来互补优势, 共同揭示问题的深层次原因[44]. 这种多智能体系统在面对具有多个相关因素和相互作用关系的复杂问题时, 能够通过智能体之间的协作, 有效地分配资源和解决冲突, 形成更为全面和深入的根因分析, 从而在解决实际问题时提供更为精准和高效的策略.

2.2 根因分析中的数据

数据在根因分析中具有重要地位, 揭示软件系统故障的根本原因依赖于精确的数据分析. 在根因分析的一般过程中, 日志信息、调用链信息以及指标信息这 3 类可观测数据起着至关重要的作用. 日志信息是分析系统状态和异常行为的重要依据, 通常包含错误信息、用户活动记录和系统操作记录等内容. 日志信息对于诊断原因是一个重要参考, 日志分析工具可以实现对海量日志的快速检索、过滤和关联分析, 从而揭示事件背后的因果关系; 调用链信息则为程序执行过程中的调用关系提供了详细视图, 记录了不同服务之间交互的信息. 借助分布式追踪系统, 可以实现对跨多个服务和组件的事件传播路径的追踪, 为定位根因提供有力支持; 而指标信息是评估系统性能的关键数据, 通常以时序数据的形式进行处理. 利用如 Prometheus、Grafana 等监控工具收集和分析指标数据, 运维人员可以及时发现性能瓶颈、资源瓶颈和潜在的安全风险, 并通过对指标数据进行趋势分析、相关性分析等, 为根因分析提供更加全面的视角.

此外, 在大语言模型时代, 文档信息的重要性日益凸显. 标准作业程序 (standard operating procedure, SOP)、故障排除指南 (troubleshooting guide, TSG) 及其他操作手册等文档蕴含着丰富的领域知识和专家经验. 利用自然语言处理技术, 大语言模型可以从这些文档中有效地提取结构化知识, 例如故障现象、可能原因以及对应的解决方案等, 提供更丰富的语义信息, 从而提升根因分析的准确性和效率.

3 数据收集

数据收集是根因分析的基础, 关键在于从海量系统数据中筛选出高质量的关键信息. 本节围绕 3 种主要方式展开: 人工收集、固定程序收集和智能体自动收集, 并以自动化程度为划分标准. 人工收集依赖人为输入, 虽灵活但受限于格式不一和数据异构; 固定程序收集通过预设策略提升效率, 但缺乏动态适应性. 智能体自动收集则借助函数调用 (function calling) 与模型上下文协议 (MCP), 实现了任务理解与数据获取的闭环, 显著降低信息处理复杂度. 相较于前两种方式, 该方式更能支持大语言模型智能体在复杂系统中的根因分析任务, 提供更精准和自动化的数据支撑.

3.1 人工收集

在基于大语言模型的根因分析技术发展初期, 人工数据收集作为核心数据供给方式, 主要体现为工程师整理

系统事件中的数据并手动输入给大语言模型. 以微软事件管理系统为典型代表, 如图 4 所示, 该系统中存储的事件信息理论上包含标题、摘要、参考根因和参考解决方案这 4 个标准化部分. 但在实际应用中, 工程师记录过程存在显著的非规范性特征. 由于值班工程师 (on-call engineer, OCE) 在应对突发事件时优先以解决问题为导向, 事件解决过程的完整记录往往不是最高优先级, 因此事后整理的数据信息中常夹杂表格数据、历史事件链接以及代码截图等内容. 同时也存在瞬时从故障到自愈的系统事件, 这类事件通常也较难被完整记录. Ahmed 等人[39]和 Zhang 等人[40]采用 GPT-3[89]系列模型进行早期实验时, 受限于模型仅支持纯文本处理能力, 通过人工筛选机制剔除表格和图像等非结构化数据, 仅保留标题与摘要等文本信息, 构建出最终数据.

标题: 虚拟机连接超时错误
摘要: 工作区未与任何虚拟网络相关联。客户已经有一个在虚拟网络中运行的虚拟机。他们希望将该虚拟机附加到 [产品名称]。我们尝试了用户界面和命令行方法, 但仍然因为相同的连接超时错误而失败。错误指向它解析到某个公共IP地址 [...]
参考根因: 不支持直接将私有虚拟机附加到公共工作区。
参考解决方案: 在虚拟机主题下为客户创建一个任务, 以便提供更完善的官方文档指导。

图 4 微软事件管理系统故障信息示例图[40]

Xpert[41]在同属于微软的事件管理系统中展现了更精细的人工处理策略, 重点针对微软云管理系统的 Kusto 查询语言 (Kusto query language, KQL) 场景进行设计. 这是一种微软开发的领域特定语言 (domain specific language, DSL)[90], 用于处理类似 SQL[91]的层次结构排列的模式实体, 如数据库、表格和列等内容. 工程师将原始事件信息拆解为 4 个层级: 包含创建时间和触发服务等基础属性的元数据信息、描述事件核心特征的标题信息、概括事件全貌的摘要信息以及整合系统日志和工程师讨论记录等复杂信息的参考内容信息. 在处理过程中, 工程师会系统性地删除重复出现的冗余信息, 并对超出大语言模型输入长度限制的长文本进行裁剪, 这种人工干预显著提升了模型对复杂事件场景的理解效率. Hamadanian 等人[42]的研究表明, 人工收集模式在微软实践中展现出独特价值, 工程师主动输入事件描述和系统日志等关键信息时, 不仅能基于经验筛选相关性数据, 还能通过解释错误代码及设备状态等专业信息为模型提供有价值的上下文信息, 有效指导大语言模型的根因分析过程.

总结而言, 人工收集的方法在根因分析数据收集环节中承担着双重使命: 既需要解决多源异构数据的结构化处理难题, 又需要实现领域知识到模型认知的有效迁移. 尽管各行业在数据形态和知识密度方面存在差异, 但都可以体现出工程师经验与模型能力的深度融合趋势. 同时, 当前实践中普遍存在的格式非标准化和信息不完整等问题, 也正在推动着人工收集与自动采集的协同机制发展, 为构建更智能的根因分析数据收集系统奠定基础.

3.2 固定程序收集

手动收集信息的过程相对繁琐, 而固定程序收集则以规则驱动为核心方法, 可以显著提升收集效率. 在根因分析的过程中, 固定程序数据收集可归纳为两类主流方法: 基于独立触发的采集机制和基于标准化工具的采集机制. 前者依赖定制化逻辑实现事件驱动的定向采集, 后者则通过预定义工具链实现数据管道的统一管理.

基于独立触发的采集机制的核心思路是通过特定条件激活数据收集流程. 如图 5 所示, RCACopilot[43]采用专家系统式工具, 为不同告警类型配置对应的处理模块, 并以决策树组织可复用操作, 模拟工程师的诊断路径. 模块操作包括: 单元切换 (定位故障相关单元)、查询操作 (检查系统状态) 和修复操作 (如建议重启). 该设计使工程师仅需维护操作序列即可完成信息采集. 为应对未知错误, RCACopilot 还提供如收集指标与调用链信息的通用检查流程, 提升了系统的适应能力. 这样的设计使得其收集信息的过程大多是预定义的, 很好地避免了无关信息的干扰, 可以更有效地解决问题.

类似地, ProvTalk[45]通过在云计算平台的网络和存储等管理服务中部署中间件, 实时拦截所有操作指令, 确保关键数据的完整捕获. 这种触发机制在日志处理中尤为重要. Huang 等人[46]使用 Drain 工具解析日志模板, 结合决策树模型识别异常会话, 仅对可疑日志提取关键特征; MRCA[47]则通过统计日志模板频率生成时间序列数据, 有效降低数据量. 除了日志信息的收集外, 更复杂的触发逻辑体现在 Kuang 等人[48]的项目中, 它同时监控指标异常和

客户反馈, 实现自动化与人工触发的双向联动. 而 ChangeRCA[49]框架进一步优化了触发机制, 当检测到服务指标、调用链路或日志中存在异常时, 其 RCCA Trigger 模块会自动启动根因分析流程, 形成从发现问题到启动诊断的闭环.

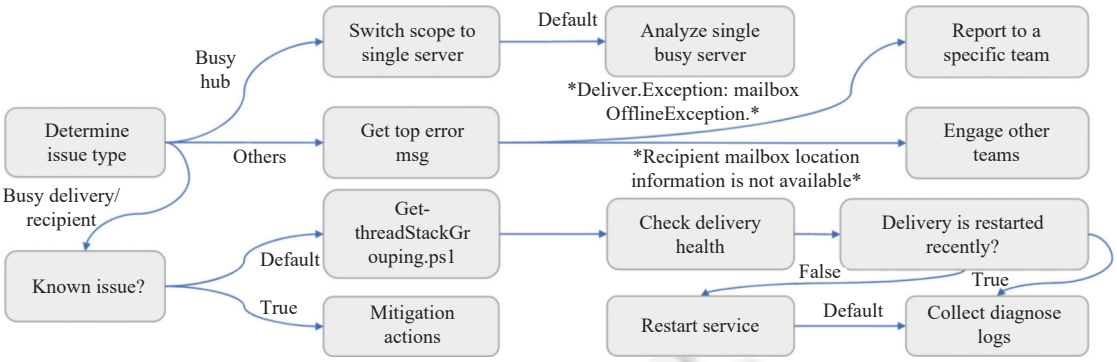

图 5 微软专家系统式数据采集工具[43]

而基于标准化工具的采集机制则侧重于构建统一的数据管道. Flow-of-Action[44]是这一方向的典型代表, 它整合了 Prometheus、Elasticsearch 和 Jaeger 等监控工具链, 以标准化流程自动化采集指标、日志和调用链数据, 通过预处理模块将原始数据转化为适合大语言模型处理的文本格式. 后续研究在此基础上扩展了工具链的灵活性. Eadro[50]组合使用容器监控工具 cAdvisor、Prometheus 和 Jaeger 系统, 构建了覆盖全技术栈的监控体系; AIOpsLab[51,52]则设计了可扩展的监控层, 除了默认采集 CPU 和内存等基础指标, 还允许通过 API 接入自定义数据. ARCAS[53]展示了标准化工具的高级应用, 它通过 Kusto 查询语言对产品遥测数据进行分析, 并结合 Auto-TSGs 故障排查指南决策图动态更新诊断上下文. 而对于需要处理的数据数量过多的情况, COMET[54]提出的 AutoExtractor 技术具有重要意义, 解决了传统方法难以处理海量日志的问题.

固定程序数据收集机制的发展轨迹呈现出两条清晰的脉络: 一方面通过集成 Prometheus 等标准化工具链构建可靠的数据基础设施, 另一方面借助触发机制提升采集过程的特定性. 固定程序数据收集不仅解决了传统手动收集的效率瓶颈, 更重要的是通过程序化质量控制大幅降低了噪声干扰. 当前的研究趋势显示, 两类方法正在相互融合. 例如, MonitorAssistant[55]在标准化监控体系中引入历史事件特征库, 通过比对实时数据与历史异常模式动态调整采集策略; AIOpsLab 通过统一的可观察性层同时支持标准化工具和自定义数据源的接入. 这种融合使得系统既能利用成熟工具链的稳定性, 又能通过如 Jin 等人[56]采用的流式数据处理方法提升数据相关性. 标准化工具提供基础数据支撑, 而事件触发机制则可以按需控制数据采集的粒度与范围, 两者协同工作显著降低了后续根因分析的噪声干扰.

3.3 智能体自动收集

随着大语言模型中函数调用技术以及 MCP 技术的出现, 设计函数实现大语言模型智能体的自动收集成为一种可能. 开发者可以通过定义函数作为大语言模型智能体的工具, 让智能体来选择并输出包含调用函数需要参数的信息, 实现数据收集工具的使用. RCAgent[57]编写了向日志系统和数据库进行查询的函数工具, 这些工具通过隐藏访问云系统中数据的无关细节, 使大语言模型智能体可以轻松地采取有效的行动. 例如, 日志系统只接受简单参数如实体 ID, 而不是复杂的 SQL 语句或日志查询 API. 这种语义最小化的工具设置显著降低了大语言模型采取有效行动的门槛, 并减少了在大型数据仓库中无意义的探索带来的资源浪费. 此外, 为了减少重复数据带来的计算成本, 这些工具会对返回的数据条目进行模糊匹配, 并消除重复项. 同时如图 6 所示, RCAgent 提出了 OBSK (observation snapshot key) 方法, 旨在解决上下文长度问题. OBSK 方法的主要思路是仅向大语言模型智能体展示观察结果的开头部分, 而将完整的观察结果映射到一个哈希 ID, 即快照键 (snapshot key). 然后, RCAgent 会建立一个键值存储, 用于将快照键映射到实际的观察结果. 当解析过程中发现一个快照键时, RCAgent 会通过键值存储来查询并返回相应的观察结果. 通过这种方法, RCAgent 可以在避免信息丢失的情况下有效地压缩上下文长度.

Example of OBSK mechanism

Thought: I will now proceed to examine the runtime logs of the taskmanager and jobmanager for the Flink job using the "runtime_log" function.
Function: {
"function": "runtime_log",
"kwargs": {
"job_id": "**********"
}
}
Observation:
org.apache.flink.runtime.taskexecutor.
TaskManagerRunner <::> -A fatal error occurred while executing the TaskManager. Shutting it down...
stdout F org.apache.flink.util.FlinkRuntimeException: The task did not exit gracefully within 180+seconds.
...53 lines omitted.
[snapshot: 2975241420]
Thought: The "runtime_log" function returned the runtime logs of the taskmanager and jobmanager for the Flink job. The logs indicate that the task did not exit gracefully within 180 seconds. I will now use the "log_agent" function to analyze a long log with another agent and provide cause, solution, and evidence.
Function:{
"function": "log_agent",
"kwargs": {
"snapshot": "2975241420"
}
}
Observation:
…

图 6 RCAgent 中的 OBSK 方法示例[57]

另一项工作 RCA-agent[61]扩展了智能体在数据收集阶段的能力, 引入控制器-执行器的协同机制实现更精细的分工. 控制器基于诊断目标与系统特征, 生成用于数据读取、过滤、异常检测和统计分析的 Python 代码, 由执行器执行并返回结构化结果. 这种方式结合自然语言理解与程序化执行, 动态构建采集流程, 提升数据获取的灵活性与准确性. 例如, 分析服务延迟时, 控制器可生成读取响应时间并计算百分位数的代码, 从而规避了固定查询模板和自然语言不确定性的问题.

与此同时, Roy 等人[58]明确了信息获取在根因分析过程中的核心地位, 并提出了两种主要的信息获取工具: 事件报告工具和历史事故数据工具. 与直接使用事件摘要不同, 事件报告工具使用了问答的方式来减少摘要时可能出现的信息丢失. 历史事故数据工具提供的数据作为重要参考, 为下一步的根因定位提供了丰富的信息. 此外, Roy 等人[58]提出了主动收集信息的方法, 包括基于工具的主动收集方法和人类介入的方法. 其中, 基于工具的主动收集方法如数据库查询工具, 能有效减轻人工负担, 提高根因分析效率; 而人类介入在处理复杂事件时可以为智能体提供额外的信息来源和操作指导. mABC[59]在微服务架构中结合了区块链技术来更好地实现多智能体之间的交互. mABC 专门指定了一个数据检测智能体 (data detective) 负责根据指定时间在指定节点收集数据. 为了更好地分析并最大化受限上下文窗口内的信息量, 数据监测智能体使用了数据收集工具和数据分析工具. 数据收集工具可以排除非必要数据, 并基于节点和时间等关键参数进行模糊匹配, 而数据分析工具可以将数据处理成图表和报告的形式, 便于其他智能体理解. 这种设计策略对外屏蔽了不同数据集的复杂性, 提高了不同智能体之间的沟通效率. D-Bot[60]是一个使用大语言模型来进行数据库诊断的系统, 同样采用了多个智能体来协同进行分析. 而与 RCAgent 和 mABC 的方法不同, D-Bot 并没有单独给一个智能体指定数据收集任务, 相反, 经负责分配任务的智能体分析事件信息并分配任务给各个多数据源智能体后, 各个数据源智能体内部均可以选择使用自己的数据收集工具, 这样的设计充分地利用了每个角色特有的设置和不同数据源信息获取方式不同的特点.

当前智能体在数据收集中的创新主要体现在 3 个方面: 工具抽象化、架构协同化与流程程序化. 工具抽象化通过语义接口简化数据访问流程, 如 RCAgent 的 OBSK 机制利用哈希压缩降低上下文成本, Roy 等人[58]则以事件

问答替代摘要文本, 提升信息可用性. 架构协同化则借助多智能体协作增强系统鲁棒性, 如 mABC 利用区块链与图表可视化优化交互, D-Bot 允许各采集智能体自主决策, 提升适应性. 流程程序化方面, RCA-agent 采用控制器-执行器架构, 实现从语义理解到代码执行的闭环采集流程. MCP 协议作为统一接口标准, 规范了数据格式与访问方式, 为 3 种技术路径的融合提供支撑, 推动根因分析从静态规则走向智能决策.

3.4 小 结

数据收集是根因分析的起点, 其方式正从人工整理向高智能化演进. 人工收集依赖工程师经验, 能弥补模型对语义的理解不足, 但效率低、主观性强, 难以应对大规模告警. 固定程序收集通过预设规则实现流程标准化, 提升一致性和可控性, 适用于常规场景, 但灵活性不足. 智能体收集则通过函数调用与 MCP 协议, 赋予大语言模型任务执行能力, 实现从需求理解到操作决策的闭环, 更具动态性与完整性. 三者在准确性、时效性与自动化方面各有优势, 融合发展成为趋势, 如在固定流程中引入智能体判断, 在人工流程中加入辅助工具, 以提升整体智能化水平. 总体而言, 大语言模型智能体在数据收集环节中的核心贡献体现在任务自动化和信息语义化两个方向. 一方面, 通过函数调用和 MCP[92]等机制, 实现从人工输入到主动收集的演化; 另一方面, 智能体借助对日志、调用链和指标的语义整合能力, 提升了数据质量. 第 4 节将进一步探讨如何在此基础上提炼因果关系并定位根因.

4 根因定位

根因定位是在完成数据收集与预处理后, 识别导致系统故障根本原因的关键环节. 其挑战在于如何高效利用数据并准确找出事件发生的根因. 在大语言模型出现之前, 根因分析主要依赖于统计与机器学习方法. 典型方法包括关联规则挖掘 (如 Apriori 算法[93])、启发式搜索 (如 HotSpot[94])、机器学习与深度学习模型 (如 traceanomaly[95] 和 MSCRED[96]). 近年来的研究也尝试在传统框架下引入多模态融合与图建模, 如 ProvTalk[45]、ChangeRCA[49]、Eadro[50]、OCEAN[97]和 MRCA[47]等. 这些方法在特定场景下提升了故障定位精度, 但普遍依赖人工特征、预设规则和数据分布稳定性, 难以适应动态复杂的云计算和微服务系统. 这些不足促使研究者进一步探索无需显式特征工程且具备自适应推理能力的新思路, 也为大语言模型与智能体方法的引入奠定了基础. 大语言模型凭借强大的语言理解与生成能力, 不仅提升了分析准确性, 还显著减少了对人工分析的依赖. 通过利用预训练过程中学习到的语义理解能力, 模型可自动解析日志与报告, 增强根因定位的效率与自动化水平. 然而, 其泛化能力和稳定性仍是实际部署中的难题. 本节将介绍 3 类大语言模型驱动的方法: 模型微调、单智能体与多智能体方案, 探讨它们各自应对挑战的能力.

4.1 基于模型微调的根因定位

在大语言模型智能体出现之前, 研究者尝试通过模型微调来完成根因定位的任务. 这些方法通过在领域数据上训练模型, 提升了故障分类与因果推理的精度. 然而, 它们往往局限于固定任务和数据集, 缺乏动态环境下的适应性, 难以支撑复杂系统中的持续推理需求. 因此, 虽然微调方法并不属于严格意义上智能体的范畴, 但它们为智能体的引入提供了重要的过渡与启发. 例如, 尽管 GPT 等通用大模型通过海量语料训练具备基础逻辑分析能力, 但其在垂直领域的推理能力仍旧有待提高. 为突破这一瓶颈, 研究者普遍通过参数高效调整实现预训练知识与领域特性的深度融合. 例如, Ahmed 等人[39]采用 LoRA[98]技术冻结原始模型权重, 仅注入可训练的秩分解矩阵, 在保留通用语义理解能力的同时显著提升了故障摘要生成质量. 类似地, Kuang 等人[48]运用 p-tuning 方法在模型各层添加少量可训练参数, 使大模型能够捕捉警报与标准操作流程间的复杂关联, 该策略在真实运维场景中展现出优异的泛化性能.

模型微调的关键在于进行知识的高效融合. Razouk 等人[63]提出的 BERT-ConvE 框架结合文本编码与图嵌入技术, 通过微调使 BERT 能够解析 FMEA 文档中的技术术语, 同时 ConvE 模块补全了半导体制造知识图谱中的因果缺失链路. 在复杂系统分析方面, LLM4MST[64]将微服务的指标、日志等多数据源数据编码为图序列, 通过微调 GPT-2 的位置编码层使其理解服务依赖的时空模式, 这种图神经网络与大模型的对齐策略有效提升了微服务故障的定位精度. 而 Ezukwoke 等人[62]开发的 Big GCVAE 模型则通过自适应超参数动态调整重建损失与 KL 散度的

平衡, 其解耦的潜在空间能更精确表征故障定位步骤间的逻辑关联并生成故障分析图, 展现出超越 GPT-2 的合理性. 而针对结构化数据处理难题, 研究者提出了多种微调方法. Shehu 等人[65]采用拼接池化层融合隐藏状态的时间维度特征, 结合逐步解冻策略缓解模型遗忘问题, 这种多层次特征增强显著提升了日志分析的鲁棒性. 在代码漏洞分析领域, Islam 等人[66]微调 CodeT5 模型并整合图卷积网络, 使模型不仅能解析代码语义, 还能通过语法结构分析定位漏洞根源, 配合 DeepLiftSHAP 可解释技术形成闭环诊断系统. 这些方法共同体现了微调技术在平衡领域适配与知识保留方面的独特优势. Huang 等人[46]和 Jin 等人[56]的研究均表明, 当历史故障数据以工程师摘要格式进行指令微调时, 模型能有效内化云系统领域知识, 其生成的根因分析报告与专家认知高度吻合.

然而, 微调策略仍面临数据质量与模态适配的双重约束. 尽管 p-tuning 等技术降低了对标注数据量的需求, 但 FMEA 文档和运维警报日志等专业数据的获取与清洗成本仍居高不下. 此外, 现有方法更擅长处理事件摘要等结构化文本, 对调用链和时序指标等半结构化数据的特征提取效率仍有待提升, 这导致部分研究需额外引入图神经网络进行辅助表征. 同时, 更根本的问题在于微调后的模型缺乏持续的学习能力, 难以实时吸收新兴故障模式的知识更新.

总体而言, 大语言模型微调已成为提升根因定位领域适应性的关键技术路径. 从 LoRA 到 p-tuning 的参数高效微调, 从 BERT-ConvE 的多模态融合到 LLM4MST 的图序列对齐, 各类方法通过不同的技术切入点实现了预训练知识向垂直领域的定向迁移. 这些实践共同证明, 在保留模型通用推理能力的基础上, 通过领域数据驱动的精细化调整, 能够有效弥合通用大模型与专业根因定位需求之间的语义鸿沟. 但模型微调的技术路线仍存在高质量数据成本高昂和持续学习能力缺失的问题, 未来突破方向可能在于开发动态微调机制, 结合持续学习策略实现模型知识的增量演进, 同时加强多模态特征提取架构设计.

4.2 基于单智能体的根因定位

智能体技术可以实现无需修改模型参数的方式来增强大语言模型的根因定位能力. 面对自动化的挑战, 单智能体需要能够在无需人工干预的情况下自动执行根因定位, 同时保持高度的准确性和效率. 如图 7 所示, 单智能体的根因定位通常遵循信息理解-行动执行-结果反馈的闭环. 智能体在接收输入后先解析故障上下文, 再通过调用外部工具或数据库执行查询与验证操作获取额外信息, 最后利用推理的方法来分析数据. 这种执行方法赋予了单智能体在根因定位中准确而高效的自动化能力.

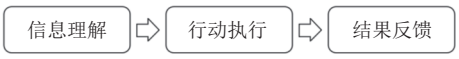

图 7 单智能体根因分析交互流程

例如 RCACopilot[43]将收集到的历史信息以向量化的方式存储到数据库中. 当收集到需要的信息后先使用大模型进行总结, 目的是减少信息量的同时转换为相对统一的自然语言描述, 再使用 k 邻近算法 (k-nearest neighbour algorithm) 在历史数据库中进行相似度匹配. 在相似度计算的过程中, 考虑到时间上相近的事件更有可能重复发生, 将事件发生的时间也作为一个参数参与计算, 最终得到 k 个最为相似的历史事件信息和历史根因. 参考上下文学习的思想, 将历史事件信息和历史根因以及一个特殊的未知事件选项作为全部选项当作智能体进行根因分析的输入, 要求智能体根据现在获得的事件信息在选项中进行选择, 得到最后的根因分析分类结果. Xpert[41]在此基础上, 将收集的历史事件信息和历史生成的 KQL 语句存入向量数据库, 在分析当前事件时在向量数据库中进行检索得到 k 个历史事件信息和对应 KQL 语句. 为了最大化利用大语言模型的输入窗口, Xpert 使用了贪心策略为输入给智能体的提示增加尽可能多的样本, 使智能体返回一个基于提示输入生成的 KQL 语句. 同时, 为了验证生成的 KQL 语句的合法性, 在得到生成的 KQL 语句后, Xpert 继续进行语法和语义检查, 并设计了一个用于量化生成效果的 Xcore 分数. 如果没有通过检查则将错误信息返回给智能体再次生成新的 KQL 语句, 直到生成符合规范的语句. 接着, 智能体将生成的 KQL 语句和 Xcore 分数一起返回给工程师. 工程师进行评测后将符合结果的语句和事件信息再次加入向量数据库作为新的数据, 以便后续遇到问题时可以重复使用. Roy 等人[58]使用了 ReAct[99]框架进行根因分析, 这个框架将原本的大模型推理过程修改为思考-行动-观察的循环, 模型思考并决定行动后, 还会观

察行动给环境带来的变化, 并根据变化进行进一步的思考和行动, 直到模型找到根因为止. 这是一种与向量数据库检索截然不同的方式, 实验结果同时证明了 ReAct 可以与检索增强生成和思维链等其他方式达到水平相当的效果.

随着技术演进, 研究者持续挖掘大语言模型的原生能力, 推动单智能体在故障诊断中的应用扩展. Siddeshwar 等人[69]提出融合文本理解、数据库查询与历史案例回溯的三重机制, 验证了单智能体在多源异构数据下的诊断能力. LLM-TSFD[68]引入时间序列分析, 使模型可识别信号数据中的频率异常, 并结合 API 查询完成跨模态推理. Ojuolape 等人[67]提出双阶段方案, 先利用统计过程控制压缩多维变量空间, 再借助语义关系分析识别关键因果链, 智能体提示策略可通过记忆机制动态优化.

为增强系统间知识映射能力, ARCAS[53]构建结构化输入范式, 将架构描述、子系统诊断结果及遥测数据转化为语义一致的智能体输入, 并整合工具链实现诊断-验证闭环. COMET[54]设计了 TrimmedLogs 处理流程, 通过 AutoExtractor 过滤非关键日志, 结合 FastText 生成事件嵌入进行相似案例匹配. AIOpsLab[51,52]开发 Agent-Cloud-Interface 接口, 使智能体能直接调用云服务 API 获取实时指标, 并通过历史对话记忆学习故障模式特征. Sarda 等人[70]提出的 LMTA 数据集整合了多源监控数据, 利用 LLM 的上下文学习能力建立事故根因关联图谱. Sun 等人[71]则结合命名实体识别与历史状态序列, 实现可解释推理.

为提高推理精度和交互性能, Goel 等人[72]采用 FAISS 库构建高维事件特征空间, 结合服务依赖拓扑信息生成诊断提示, 显著提升 GPT-4 的定位精度. LLmiRQ[73]设计动态查询策略, 针对编程错误报告采用查询缩减技术, 对纯文本故障则实施语义扩展, 并引入交互机制优化结果. LasRCA[74]框架将故障指南转化为自然语言形式的指令流, 通过小型分类器与智能体的协同决策平衡效率与准确性. Vidyaratne 等人[75]提出两阶段提示方案, 先提取设备组件列表, 再递归生成故障树分支, 其情境学习机制确保结构化输出的一致性. FaultExplainer[76]融合变量偏差分析与特征降维模型, 提升因果推理的透明性与物理一致性. FLASH[77]引入状态监督机制, 通过指南解析、动态诊断计划调整及失败案例回溯提升系统鲁棒性. 而 Jambigi 等人[78]则将 SQL 执行日志经智能体抽象为语义特征, 为传统机器学习模型提供增强输入.

基于单智能体的根因定位方法的优势在于其高度的综合性和自主性, 能够快速整合信息并从历史经验中学习, 从而提高故障处理的效率. 然而, 这种方法的缺点在于在复杂任务场景下存在计算瓶颈导致效率较低, 同时在动态环境中单智能体难以感知全面信息, 无法通过协作共享局部观测数据. 因此, 尽管单智能体在根因定位中提供了自动化和效率的提升, 但其面临的挑战仍需进一步解决.

4.3 基于多智能体的根因定位

模型微调和基于单智能体进行根因定位的方式本质上都还是以单个大语言模型为核心, 为大语言模型提供尽量丰富的信息, 使其拥有根因分析的能力. 但是随着研究逐渐深入以及多智能体在其他领域的成功应用, 给每个智能体分配设定的方式成为更好地解决复杂问题的一个方法. 利用智能体系统进行根因定位不仅需要决策过程是准确的, 并且也需要最终结果具有可解释性. 为了应对这一挑战, 多智能体系统通常会进行工作流编排和数据共享, 以便追踪每个智能体的决策路径. 如图 8 所示, 多智能体方法通过角色分工与结果协同实现更复杂的分析, 常见流程包括任务分配、并行分析、结果验证与共识生成. 控制器智能体负责调度任务, 各个执行智能体针对不同数据源开展独立分析, 报告智能体整合多方结果输出最终报告, 从而实现多视角的联合推理. 相较于单智能体的线性执行, 多智能体的体系能显著增强鲁棒性与决策稳定性.

图 8 多智能体根因分析交互流程

例如, RCAgent[57]是一种基于 ReAct 的方法, 通过任务分解和智能体角色分工实现性能突破. 首先 RCAgent 将代码分析和日志分析分别分配给两个智能体, 并由一个控制器智能体控制整体流程. 代码分析智能体以递归的方式工作, 在代码库中递归搜索相关代码并依次分析, 最终总结后返回给控制器智能体, 而日志分析智能体将上下文信息作为检索内容使用检索增强生成等一系列方法得到可能的辅助信息返回给控制器智能体. 同时, RCAgent 引

入了两种稳定化方法, JSON 修复技术和错误处理技术. JSON 修复技术确保了结构化交换数据的正确性, 而错误处理技术减少了无效操作的发生频率, 这两种方法的结合使得 RCAgent 可以在复杂的云系统中可靠地运行. 此外, RCAgent 还引入了文本数据的自一致性聚合和工具使用轨迹的自一致性聚合方法, 对智能体生成的开放性文本结果进行聚合, 进一步提高了推理路径的多样性. 最终的实验结果显示, RCAgent 相比 ReAct 方法有着明显的优势, 同时在 RCA 的效果上也相比前面的工作都有明显的提高, 这也同时说明了多智能体根因定位技术路线的有效性. 这种结构化分工在 D-Bot[60]中得到进一步延伸, D-Bot 在任务分配智能体分配任务给多数据源智能体后, 数据收集工作由各个数据源智能体自行进行. 每个智能体在处理任务时使用结合 ReAct 的树搜索策略, 并且智能体之间会进行异步通信, 共享每一步搜索策略得到的结果并分析其中存在的问题. 由于数据库根因分析的范围相对固定, 可能出现的问题和对应的解决方案可以很大程度上事先确定, 每个智能体分析的过程使用的是近似算法 BM25[100]来寻找最相近的内容. 在各个智能体搜索策略结束后还会进行一次交叉审查, 将每个专家的分析过程进行总结并共享, 针对每一个专家的分析结果, 其他专家可以提供改进建议并令该专家进行重新评估和推理, 直到最终没有其他建议. 最终, 任务分配智能体将包含详细分析信息的结果生成一份包含标题、异常日期、来自警报的详细异常描述、诊断结果、解决方案和详细诊断过程这 6 部分内容的完整报告.

随着多智能体架构的广泛应用, 研究者开始探索更复杂的协作范式. mABC[59]同样是按照角色设定的方式使用多智能体进行分析, 但是与构建知识库的方式不同, mABC 通过采用基于区块链技术的多智能体系统, 包括事件检测智能体 (alert receiver)、进程调度智能体 (process scheduler)、依赖分析智能体 (dependency explorer)、节点故障概率分析智能体 (probability oracle)、故障网智能体 (fault mapper) 和解决方案智能体 (solution engineer) 等, 以增强根因定位的准确性与信赖度. 进程调度智能体在事件检测智能体检测到报警事件时, 负责启动一系列分析任务, 激活依赖分析智能体以解析节点间的依赖关系, 节点故障概率分析智能体以评估各节点的故障概率, 以及故障网智能体以构建故障网络的图形化表示. 这些智能体通过高效的工作流模式, 如 ReAct 工作流或直接答案工作流, 共同推进分析过程, 为解决方案智能体提供深入的数据支持. 区块链技术的应用为 mABC 框架带来了去中心化的通信和投票机制, 这在解决方案智能体的方案评估阶段尤为关键. 故障网智能体结合节点故障概率分析智能体的数据, 形成对故障传播路径的直观展示, 而依赖分析智能体的工作则为解决方案智能体的根源分析提供了必要的信息. 在此基础上, 解决方案智能体制定针对性的解决方案, 并通过区块链投票机制, 由其他智能体根据其专业知识和系统贡献进行评估, 确保了根因分析流程的严密性和方案决策的有效性. 这种设计理念在 Hamadanian 等人[42]的工作中体现为假设形成-测试-解决规划的三阶段架构, 其假设形成器智能体综合事件描述与系统日志生成候选根因, 测试器智能体设计验证步骤并评估假设, 缓解规划器智能体则生成可行应对策略, 整个过程通过多个大语言模型智能体的链式协作完成, 无需微调模型参数.

为提升多智能体系统的领域适应能力, MonitorAssistant[55]提出基于历史知识库的协同框架. 其核心在于构建监控指标配置数据库, 通过 GPT-4-Turbo 智能体提取历史事件中的 shapelets 特征和语义描述, 建立异常模式与根因的关联映射. 当检测到新异常时, 分类智能体基于形状相似性和语义相关性筛选历史案例, 总结智能体则生成包含物理意义、故障类型和修复建议的综合性报告. 这种知识驱动的方法与 Cloud Atlas[79]的语义化系统建模形成互补, 后者将云组件实例化为具有推理能力的智能体网络, 每个组件智能体根据自身功能描述和交互关系参与因果图构建. 通过迭代式推理请求和数据驱动的因果验证, Cloud Atlas 实现了系统级故障传播路径的动态推演, 最终结合 DoWhy 库的分布变化算法确定根因概率.

在提升决策可靠性方面, 研究者提出了多种智能体协同验证机制. RCA-agent[61]采用控制器-执行器双智能体架构, 控制器负责策略规划和代码生成, 执行器专注遥测数据收集与分析, 两者通过程序合成技术实现闭环协作. Flow-of-Action[44]则通过 SOP 流程标准化和动作集机制改进 ReAct 框架. SOP 流程通过标准化的操作步骤约束大语言模型智能体的推理路径, 例如从宏观网络问题逐步细化到具体网络分区, 避免了 ReAct 中因随机性导致的推理错误. 动作集机制则通过生成多个可行动作并附加解释, 结合判决智能体 (JudgeAgent) 的终止判断和观测智能体 (ObAgent) 的信息过滤, 动态平衡探索与确定性, 解决了复杂场景下多可行动作选择的难题. 与 ReAct 的思考-

动作-观察范式相比, Flow-of-Action 进一步提出思考-动作集-动作-观察范式. 通过预生成动作集减少无效尝试, 同时将 SOP 转换为可执行代码, 确保步骤的连贯性. 这些改进不仅减少了幻觉问题, 还使故障定位过程更贴近实际运维场景. Huang 等人[80]提出的 LDRA 框架更将多智能体协作提升到群体决策层面, 通过建立多智能体辩论机制对数据驱动模型输出的候选根因进行排序. 每个大语言模型智能体在辩论轮次中评估他人观点, 经过投票达成共识, 这种设计显著降低了单一模型的认知偏差, 在工业设备故障诊断中展现出独特优势.

多智能体系统的优势在于其分布式和协作的特性, 这使得系统能够处理更复杂的故障场景, 并且在某些智能体失效时仍能保持整体性能. 此外, 多智能体系统可以通过并行处理来提高根因定位的速度, 每个智能体可以专注于问题的不同方面, 从而提供更全面的解决方案. 然而, 多智能体系统仍旧存在系统复杂性和资源消耗过高, 协作机制不足等问题. 此外, 虽然多智能体系统在理论上可以分别分析每个智能体来提高可解释性, 但由于复杂系统不可解释的特性, 在实践中智能体之间的相互作用往往使得整体系统的可解释性降低. 因此, 尽管多智能体系统为根因定位带来了更多的灵活性, 但其可解释性方面的挑战仍然是一个需要通过技术创新和理论发展来解决的问题.

4.4 小　结

故障根因定位技术经历了从传统方法到大语言模型驱动的多阶段演进, 分别对应传统方法、模型微调、单智能体和多智能体这 4 个部分. 传统方法依赖人工设计的特征和静态因果模型, 适用于场景有限、变化较小的系统, 但面对现代软件系统的复杂性和动态性, 表现出适应性差和泛化能力不足的问题. 模型微调阶段通过高效调整模型参数, 实现了领域知识的迁移和适应, 提升了定位的准确性和灵活性, 但仍存在推理能力受限和场景覆盖不全面的短板.

从智能体的视角来看, 根因定位环节的研究体现了智能体技术的逐步深入. 单智能体在任务规划、知识记忆与推理解释上展现了独立分析的能力, 能够动态构建因果链路, 提高了诊断的深度和广度. 而多智能体则进一步引入角色分工和协作机制, 使根因分析具备了更高级的智能特征, 为下一步的自动化运维和智能诊断奠定基础. 然而, 不同阶段的技术在计算效率与实际应用效果之间仍需权衡. 如何建立科学全面的多维度评估体系, 将成为未来研究的重要方向, 第 5 节将对此展开详细讨论.

5 效果评估

效果评估也是根因分析流程中至关重要的一环, 其目的是验证分析结果的准确性和实用性, 确保根因定位的有效性, 从而提升故障排除的效率. 这个过程的关键在于设计一套客观且能够全面反映分析效果的评估体系, 确保评估结果与实际系统性能和用户需求相符合. 本节将介绍现有研究中的评估指标, 涵盖客观评估指标、基于人类评估的主观指标和基于大语言模型评估的主观指标. 客观评估指标通过标准化测试集对文本生成质量和系统性能进行评估, 但可能无法完全反映根因分析的复杂性. 基于人类评估的主观指标更从实际应用效果出发, 提供直观的人类反馈, 但成本较高且难以大规模实施. 基于大语言模型评估的主观指标利用模型自身能力进行综合评估, 但仍存在幻觉和评分标准的问题. 这些指标的综合应用为根因分析模型的优化提供了多维度的评价标准. 从现有研究结果来看, 不同类型的大语言模型智能体在根因分析的应用中展现出不同优势. 基于模型微调的方法在稳定性和领域适配性上表现突出, 这类方法能将领域知识嵌入模型, 从而在固定任务或标准化数据集上获得更稳定的分类和定位性能; 单智能体方案突出在工具调用能力与交互推理上的优势, 它们将检索增强、函数调用和逐步推理结合, 能在单次会话中通过调用外部接口完成流程; 而多智能体体系则在复杂场景分工表现突出: 通过任务分配、并行分析与跨智能体的交叉验证, 多智能体能在复杂故障中提供更全面的视角与更高的鲁棒性.

5.1 客观评估指标

使用大语言模型智能体进行根因分析, 本质上是一类自然语言处理任务, 因此可以借助自然语言处理领域中常用的文本生成质量评估指标来进行效果衡量. 为系统梳理当前研究在该方向上的评估实践, 图 9 展示了各类指标在现有工作中的使用频率统计. 同时, 为清晰展示当前研究在根因分析中采用的评估手段, 我们将指标按任务适

用性进行归类, 并将代表性的研究工作与每类指标对应列出, 如表 4 所示. 具体而言, 分类性能类指标用于评估根因定位与告警分类的准确性; 文本生成类用于衡量根因报告的语言质量; 语义相似度类指标侧重于衡量解释文本的语义一致性; 任务定制化指标则用于评估面向特定任务或模块的过程性与系统级效果.

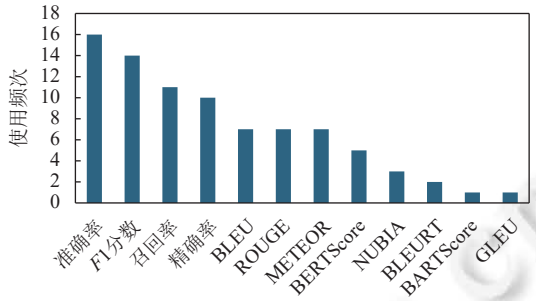

图 9 文本生成质量评估指标使用频率统计图

表 4 文本生成质量评估指标类别、主要使用任务与代表工作

指标类别	典型指标	核心思想	代表文献
分类性能	*F*1分数、准确率、精确率、召回率	根因分类与定位任务	Shehu等人[65]、Sarda等人[70]、LasRCA[74]
文本生成	BLEU、ROUGE、METEOR、GLEU	报告的文本质量	Roy等人[58]、Jin等人[56]、Goel等人[72]
语义相似度	BERTScore、BLEURT、NUBIA	强调上下文与语义一致性的生成任务	Ahmed等人[39]、Zhang等人[40]、RCAgent[57]
任务定制化	Xcore、轨迹长度、路径准确率、覆盖率、IoU、MRR/MAP、命中率等	针对特定任务设计的过程与结果评估	Xpert[41]、mABC[59]、D-Bot[60]

从结果可以看出, *F*1 分数[101]、准确率、精确率和召回率等传统分类指标被广泛使用, 表明研究者普遍重视结果的分类性能; 而在生成文本的质量评估方面, BLEU[37]、METEOR[102]、ROUGE[103]、GLEU[104]等指标用于衡量文本的表面特征, 如词汇匹配和覆盖度. 例如, Ahmed 等人[39]就采用了 BLEU、ROUGE 和 METEOR, 以体现其工作对输出文本结构完整性的重视. 此外, 针对语义层面的评价, 研究者也逐步引入 BERTScore[105]、BLEURT[106]、BARTScore[107]、NUBIA[38]语义相似度指标, 它们可以更好地捕捉上下文与语义一致性. 典型如 RCAgent[57]和 Roy 等人[58]的研究, 采用了这些语义评估方法, 从而强调对模型理解能力的衡量. 这反映出当前评估趋势正逐步从表层匹配走向更深层的语义理解.

特别值得注意的是, 除了图表中的自然语言处理任务的常见指标外, 研究者们也提出了一些可以用于评估特定任务的新指标. 例如, Xpert 为 KQL 语句生成场景, 从包括语法、语义、词汇相似度和输出模式匹配等多个维度设计了 Xcore、表格准确率等指标, 用于评估生成 KQL 查询的整体质量. 而 RCAgent 针对自一致性聚合方法中的轨迹设计了轨迹长度和通过率等指标, 用于描述自一致性聚合的效果. D-Bot 提出了结果准确率用于衡量诊断结果的准确性, 即诊断系统推荐的根因与实际根因的匹配程度; mABC 针对智能体协作过程中的结果准确率和路径准确率进行整体效果的判断标准; 而 Flow-of-Action[44]在 mABC 的基础上采用根因位置准确率、根因类型准确率和平均路径长度作为核心评估指标. 此外, Vidyaratne 等人[75]提出的覆盖率与幻觉率分别量化诊断完整性和冗余信息比例, Siddeshwar 等人[69]的杰卡德相似系数通过集合论评估多结果重叠度, Islam 等人[66]的 IoU 交并比则从空间定位角度增强评估粒度. 同时, LLmiRQ[73]和 LLM4MST[64]等研究引入的 MRR、MAP 以及 ChangeRCA[49]的命中率, 实质是将推荐系统的排名机制迁移至根因候选列表的质量评估. 这些补充指标共同特征在于, 其设计目标并非直接衡量文本生成质量, 而是围绕诊断准确性和过程合理性构建评估体系.

在聚焦文本生成质量评估的同时, 实际业务场景要求将系统性能和运维成本纳入考量. 本研究主要从资源消

耗与性能效率、操作复杂度以及成本效益与实用性这 3 个维度对实际业务进行分析. 资源消耗与性能效率关注模型推理过程中的计算负载与响应时效, 确保系统满足实时诊断的硬件约束; 操作复杂度通过诊断步骤数、决策路径长度等参数量化运维人员理解与执行成本; 成本效益与实用性结合误报率、修复时间等业务指标, 验证系统部署的长期经济效益. 最终得到系统性能评估指标分布表, 如表 5 所示. 分析可知, 资源消耗与性能效率仍是当前主要研究方向, 而操作复杂度仅见于 Flow-of-Action、mABC、AIOpsLab 这 3 项工作, 成本效益与实用性更局限在 ARCAS[53]等研究中. 这种不均衡性表明, 尽管资源效率是系统部署的技术前提, 但实际运维中的人工操作成本和长期维护收益仍需更系统的评估. 研究趋势显示, 仅 AIOpsLab 等极少数团队开始尝试跨维度评估, 大多数研究仍停留在单一性能指标的孤立分析层面.

表 5　系统性能评估指标分布表

研究	资源消耗与性能效率	操作复杂度	成本效益与实用性
RCACopilot[43]	√	—	—
Flow-of-Action[44]	—	√	—
mABC[59]	—	√	—
ARCAS[53]	—	—	√
Shehu等人[65]	√	—	—
ProvTalk[45]	√	—	—
Kuang等人[48]	√	—	—
OCEAN[97]	√	—	—
ChangeRCA[49]	√	—	—
AIOpsLab[51,52]	√	√	—
Hamadanian等人[42]	√	—	√
COMET[54]	√	—	—

综上, 根因分析的效果评估需兼顾文本生成质量与业务场景的特殊需求. 文本生成质量方面的指标虽能验证语言模型的文本生成能力, 但诊断准确性和工程可行性等维度需依赖场景化指标补充. 当前系统性能评估体系已初步形成资源、操作、成本的三维框架, 但对后两者的研究深度仍需加强. 未来的研究应在保留自然语言指标评估机制的基础上, 进一步引入多维性能指标的协同设计, 以提升大语言模型智能体在实际系统中的可落地性和工程适配能力.

5.2　基于人类评估的主观指标

在根因定位的技术评估体系构建过程中, 研究者逐渐意识到如 BLEU、ROUGE 等传统自然语言处理指标存在显著局限性. Ahmed 等人[39]的研究表明, 尽管编码器-解码器架构模型在文本生成指标上表现优异, 但其生成的通用性答案难以辅助工程师的实际分析. 这一发现推动了以人类评估为核心的多维度评价体系的形成, 其核心在于衡量模型输出对实际运维场景的实用价值和技术支持能力. 人工评估通常适用于生成类与解释类任务, 例如根因报告的准确性、可理解性以及建议的可执行性, 并常与客观指标联合使用以补偿单一指标无法捕捉的工程实用性维度.

针对评估体系的构建, 多个研究团队提出了新方法. RCACopilot[43]联合 SRE 设计了正确性与可读性双指标评估框架, 通过专家在 0–5 分区间对诊断结果进行评分, 重点考察结果对故障排查效率的提升效果. RCAgent[57]进一步提出 H-Helpfulness 指标, 采用李克特量表量化模型输出对 SRE 团队的实际支持度, Tukey HSD 检验更验证了其相对于 ReAct 框架的优越性. 类似地, mABC[59]通过随机抽取案例, 邀请 10 位领域专家对 R-Useful 指标进行 1–5 分评价, 发现基于 GPT-4-Turbo 的解决方案与专家预期高度契合, 尤其在复杂微服务架构故障中展现出独特优势.

在评估维度拓展方面, MonitorAssistant[55]将评估范围扩展到模型推荐有效性、异常报告指导性和人机交互便捷性这 3 个层面. 工程师反馈显示, 系统推荐的检测模型能有效降低误报率, 而包含类似事件、可能原因和排障建

议的异常报告可以使平均故障定位时间大幅缩短. D-Bot[60]则通过 HEval 指标体系, 由资深专家对诊断报告的准确性、逻辑合理性和解决方案可行性进行多维度评分, 验证了其在数据库异常诊断中的实用价值. ProvTalk[45]的评估更具特色, 通过分组实验对比静态输出与交互式分析的效果, 发现多层级溯源图结合规则翻译可使事件原因识别效率有效提升.

针对评估方法的优化, Huang 等人[80]在云系统中开展大规模用户研究, 邀请工程师对故障案例进行开放性评估, 结果显示自动提取的故障指示符在信息浓缩度和指向性方面分别获得较高分数. Jin 等人[56]的研究则采用双盲评估法, 将 Oasis 生成的摘要与人工撰写版本混合排序, 最终超过一半的故障负责人将大模型生成摘要列为最优选择. FLASH[77]框架的评估体系尤为全面, 除诊断准确性和排障计划正确性等传统维度外, 更引入缓解时间预测偏差率和工程师整体满意度等新的指标.

值得注意的是, 评估方法论正在向深度细分化发展. Herrmann 等人[81]设计的专家评估体系要求领域专家从帮助性、正确性和偏好度这 3 个维度对大语言模型的输出进行交叉验证, 发现模型预测与专家共识存在较高的相关系数. Goel 等人[72]则提出了依赖关系与历史事件相结合的评估方法, 通过评估者对历史事件的独立评分与共识讨论, 建立了客观指标与人工判断的映射关系.

这些研究表明, 有效的人类评估体系须具备 3 个核心特征: 首先, 评估主体必须包含领域专家以确保结果可信度; 其次, 评估维度需覆盖准确性、可操作性和时效性等实际需求; 最后, 评估过程需要建立标准化流程以控制主观偏差. 当前研究已证明, 人类评估的方法能够更全面地反映大语言模型在根因定位中的实际效能, 为模型优化提供明确方向. 然而, 评估成本高和标准不统一等问题仍然存在, 未来需要建立跨领域的基准测试集和评估协议, 有助于降低主观偏差并提升跨场景的适应能力.

5.3 基于大语言模型评估的主观指标

随着大语言模型智能体在根因分析领域的深度应用, 研究者们逐步探索出基于模型自身能力的效果评估范式, 形成了用模型评估模型的创新路径. 以 LM-PACE[82]框架为例, 该工作不仅构建了置信度评估 (COE) 和根因评估 (RCE) 的双阶段评估体系, 还引入 GPT-4 等大语言模型对评估结果进行二次校准. 具体来说, 在生成 COE 阶段的信息充分性判断和 RCE 阶段的根因合理性评分后, 系统会将历史事件上下文、当前事件描述及评估分数输入大语言模型, 要求其判断这些中间结果的可靠性. 实验表明, 这种递归式评估机制能有效识别并修正原始评分中的偏差, 使最终置信度分数的校准误差大幅降低, 显著提升了工程师对模型输出的信任度.

这种评估方法的优化在 RCAgent[57]中得到进一步拓展. 该团队提出的 G-Correctness 和 G-Helpfulness 评估体系分别针对根因准确性和方案有效性进行评估, 通过 GPT-4-0613 模型对预测结果进行多维量化评估. 在操作层面, 系统将根因诊断报告与解决方案同时输入评估模型, 要求其以 0–10 分制对结果的准确性和工程实用性进行分级判定. 不过研究者也指出, 模型固有偏见和训练数据偏差可能导致评估分数出现系统性偏移, 因此建议结合人工抽检进行结果验证. 同时, Herrmann 等人[81]通过计算预测根因与参考根因的语义相似度生成平均相似度得分. 评估模型会同时接收故障事件上下文、预测根因描述及标注人员提供的标准答案, 从语义完整性、逻辑连贯性和可操作性等维度进行 1–10 分范围内的量化评分. 这种方法的突破性在于建立了自动化评估与人工评估的映射, 特别是在处理历史积累的海量故障案例时, 能快速完成模型表现的横向对比. Zhang 等人[40]也在跨地域事件案例中发现, 基于情境学习的 GPT-4 模型相较微调 GPT-3 在正确性方面存在较大幅度提升, 印证了上下文理解对诊断质量的关键影响.

总体而言, 当前大语言模型在根因分析效果评估中的应用已从单一的准确性评价逐步扩展到实用性方面, 初步展现出贴近工程实践的潜力. 但需要指出的是, 模型评估仍面临幻觉生成和评分标准等挑战, 特别是在处理领域特异性强或缺乏标注数据的场景时, 仍需构建包含领域知识的人类评估闭环. 未来研究可探索评估模型的领域自适应技术, 以及基于持续学习的动态评估框架, 使评估体系能伴随系统迭代优化.

5.4 小　结

本节围绕根因分析的效果评估阶段, 系统梳理并对比了 3 类主流评估方法, 并揭示了它们在评估维度、适用

场景和发展趋势方面的异同与演化方向.

(1) 客观评估指标首先借助自然语言处理中的通用文本生成评价体系衡量模型输出的语言质量. 与此同时, 近年来研究者意识到这些指标难以全面覆盖根因分析中的实际业务需求, 因而逐步引入轨迹长度及根因类型准确率等具体场景下的指标, 拓展了评估的覆盖范围. 总体来看, 该类指标具有高效和可复现的优点, 但在处理复杂业务逻辑时, 表达能力仍显不足.

(2) 基于人类评估的主观指标从专家视角出发, 通过对模型输出的多种维度进行主观评价, 能够更贴合真实运维需求. 如 H-Helpfulness、R-Useful 和 HEval 等指标从诊断结果对实际业务的正确性、可解释性和交互体验等方面构建多维评估框架. 尽管该方法在识别模型幻觉和评估方案可执行性方面表现优越, 但其实施成本高、主观性强并且标准不统一的问题限制了其在大规模评估中的应用广度.

(3) 基于大语言模型评估的主观指标则代表了一种新兴的自动化评估路径, 强调用大模型评估大模型的效果. 相关方法如 LM-PACE 的 COE/RCE 双阶段机制以及 Herrmann 等人[81]的语义相似度打分框架, 均通过引入大语言模型对候选答案的合理性和一致性进行判断. 该类方法在处理历史大规模数据和降低人工成本方面展现出显著优势, 并有助于构建可扩展的评估流水线. 然而, 其结果仍易受评估模型幻觉及领域适应能力的影响, 需结合人工验证与持续优化以确保稳定性.

大语言模型智能体在效果评估阶段的应用主要集中于两方面: 一是利用智能体生成可解释的评估报告, 从而辅助人工验证; 二是通过多模型互评机制实现对根因分析结果的语义一致性验证. 这些研究为自动化评估体系的构建提供了新的思路. 未来效果评估体系的发展应致力于多维融合与动态演化, 在保持自然语言质量评估基础的同时, 强化语义理解与业务价值导向, 实现从静态评分向智能适配、自我演进的转变, 助力根因分析系统迈向工业级可靠性与实用性.

6 当前挑战和未来发展方向

尽管当前大语言模型智能体在根因分析领域已经取得了一定成果, 但在实际应用过程中仍然面临诸多挑战. 这些挑战既来自根因分析任务自身的复杂性, 也源于大语言模型智能体在任务应用中的特有问题. 本节将系统梳理当前根因分析任务面临的主要挑战与机遇, 并探讨未来可能的发展方向, 为后续研究和实践提供参考.

6.1 根因分析任务的复杂性挑战

根因分析领域面临的复杂性挑战首先体现在多源数据的高复杂性和集成分析的高难度. 系统生成的日志和监控数据往往规模庞大, 并且通常包含多种格式, 这使得数据的整合和分析变得异常复杂. 与此同时, 系统组件复杂性与场景复杂性也是一个重大挑战. 一个完整的软件系统涉及多个组件, 熟练掌握这些组件需要长时间的积累. 根据微软的研究, 培养一名具备全面技能的工程师通常需耗时 1.5 年以上. 此外, 数据质量问题也是一个不可忽视的瓶颈, 系统错误或监控配置不当可能导致数据缺失, 这些都直接影响了根因分析的准确性.

在根因定位方面, 系统的复杂性导致故障模式多样, 故障传播路径错综复杂. 现有的分析方法难以捕捉到故障的全貌, 特别是在多组件交互的复杂系统中. 这一挑战不仅体现在事件识别的困难上, 也体现在根因定位流程本身的复杂性. 它通常涉及从收集事件信息开始的多步骤决策、行为调整和反复沟通, 流程冗长且耗时.

此外, 业务快速发展与知识更新速度的挑战也不容忽视. 业务的快速演进导致系统结构频繁变化, 传统的故障排查指南难以及时跟进, 从而造成更新不及时和难以使用的问题.

6.2 大语言模型智能体应用于根因分析的挑战

大语言模型智能体在根因分析中的应用同样面临多重挑战, 可以归纳为领域知识缺失与自主学习能力受限、推理规划能力不足、模型可解释性差和缺乏标准评估机制这 4 个主要部分.

智能体在处理海量软件系统信息时, 领域知识的缺失和自主学习能力的限制尤为突出. 大语言模型通常在通用数据集上训练, 缺乏特定行业的专业知识和经验, 导致智能体难以准确理解特定行业系统的复杂性和独特性, 进而无法准确识别和解释行业特定故障. 同时由于依赖基于少样本的上下文学习, 智能体往往难以从有限的数据中

提取出足够的信息来进行准确的根因分析.

而在推理和规划方面, 尤其是在制定详细、可行的定位计划上, 智能体的应用仍存在明显挑战. 现有大语言模型虽然在语言理解上有显著优势, 但在逻辑推理和决策规划方面仍有不足, 这可能导致智能体在分析故障传播路径、确定故障影响范围以及制定修复方案时, 缺乏必要的逻辑性和系统性, 影响故障处理效率和效果.

可解释性不足的问题也同样不容忽视. 大语言模型作为黑盒系统, 决策过程不透明, 故障分析结果缺乏充分解释, 可能导致工程师对智能体建议持怀疑或盲从态度, 甚至做出错误操作.

此外, 评估标准缺失是智能体应用的另一重大挑战. 根因分析为开放式问题, 现有评价指标难以全面准确衡量模型表现, 影响模型性能的客观评价及实际应用.

6.3 未来发展方向

针对上述挑战, 为应对根因分析任务和大语言模型智能体应用中所面临的挑战, 未来的研究工作可以从以下 3 个方向入手, 推动系统智能化能力的持续演进.

(1) 构建稳健的数据表示与泛化学习机制. 在复杂软件系统中, 日志、指标和调用链数据通常规模庞大且结构复杂, 这对根因分析的数据驱动能力提出了更高要求. 未来的研究应致力于多源异构数据的标准化与结构化表示, 并结合数据增强与跨系统迁移学习技术, 提升智能体在低资源或变动环境下的泛化能力. 通过构建细粒度、多样性的故障语料库, 可以为智能体提供更强的输入基础, 从而提升其对故障现象的理解深度.

(2) 增强领域知识建模与因果推理能力. 根因分析依赖对系统运行机制与组件依赖的深刻理解, 当前大语言模型缺乏显式的系统知识建模与推理支持. 为此, 未来可探索将领域知识以图谱的形式嵌入模型中, 提供任务上下文和系统结构信息的辅助. 与此同时, 可以集成因果图建模等方法提升模型对故障传播路径的建模能力, 从而实现从故障症状到根因的链式推理过程.

(3) 提升模型的可解释性与可靠性保障. 在工程实践中, 智能体的决策可信度直接影响其部署价值. 未来应重点发展面向根因分析任务的解释方法, 如思维链生成推理过程以及参考面向模型内部行为追踪与特征归因调试方法相关研究[108,109]的可解释性, 帮助用户理解智能体的分析逻辑. 同时, 为了支持系统性优化与横向对比, 需要建立覆盖准确率、故障恢复效率和用户反馈等多维度的性能评估体系, 推动形成行业通用的智能体测评标准.

通过在以上方向上的持续探索, 有望推动大语言模型智能体从语言生成工具演进为根因分析助手, 实现对软件系统问题更智能、更准确和更可信的定位.

7 总 结

根因分析作为确保现代软件系统稳定性和高效运行的关键技术, 正经历着由传统人工依赖向智能化转型的重大变革. 通过对现有文献进行系统性回顾, 本文从数据收集、根因定位和效果评估这 3 个关键环节, 详细剖析了大语言模型智能体在根因分析领域的研究现状与发展趋势, 发现关于大语言模型智能体在此领域的应用研究仍存在诸多不足. 特别是在数据预处理、根因精准定位以及评价体系的完善等方面, 现有的研究尚需进一步深入和拓展. 此外, 本文还揭示了根因分析领域自身的挑战以及大语言模型智能体进行根因分析时在自主学习、推理能力、领域知识融入和可解释性方面的局限性, 并提供了可以扩展的对应研究方向. 本综述不仅为未来的研究提供了有价值的参考, 也为大语言模型智能体在软件系统根因分析的进一步发展奠定了理论基础. 我们相信, 随着相关技术的不断进步和创新, 大语言模型智能体将在根因分析乃至更广泛的复杂任务处理中发挥更加重要的作用, 为软件系统的稳定性和可靠性提供更加坚实的保障.

References

[1] Todorović V, Pešterac A, Tomić N. The impact of automated trading systems on financial market stability. Facta Universitatis, Series: Economics and Organization, 2019, 16(3): 255–268. [doi: 10.22190/FUEO1903255T]

[2] Richardson I, Reid L, O'Leary P. Healthcare systems quality: Development and use. In: Proc. of the 2016 Int'l Workshop on Software Engineering in Healthcare Systems. Austin: ACM, 2016. 50–53. [doi: 10.1145/2897683.2897686]

[3] Meziane F, Vadera S, Kobbacy K, Proudlove N. Intelligent systems in manufacturing: Current developments and future prospects. Integrated Manufacturing Systems, 2000, 11(4): 218–238. [doi: 10.1108/09576060010326221]

[4] Mugarza I, Amurrio A, Azketa E, Jacob E. Dynamic software updates to enhance security and privacy in high availability energy management applications in smart cities. IEEE Access, 2019, 7: 42269–42279. [doi: 10.1109/ACCESS.2019.2905925]

[5] Wong WE, Li XL, Laplante PA. Be more familiar with our enemies and pave the way forward: A review of the roles bugs played in software failures. Journal of Systems and Software, 2017, 133: 68–94. [doi: 10.1016/j.jss.2017.06.069]

[6] Alquraan A, Takruri H, Alfatafta M, Al-Kiswany S. An analysis of network-partitioning failures in cloud systems. In: Proc. of the 13th USENIX Symp. on Operating Systems Design and Implementation. Carlsbad: USENIX Association, 2018. 51–68.

[7] Chen HC, Dou WS, Jiang YY, Qin F. Understanding exception-related bugs in large-scale cloud systems. In: Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). San Diego: IEEE, 2019. 339–351. [doi: 10.1109/ASE.2019.00040]

[8] Gao Y, Dou WS, Qin F, Gao CS, Wang D, Wei J, Huang RR, Zhou L, Wu YM. An empirical study on crash recovery bugs in large-scale distributed systems. In: Proc. of the 26th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Lake Buena Vista: ACM, 2018. 539–550. [doi: 10.1145/3236024.3236030]

[9] Ghosh S, Shetty M, Bansal C, Nath S. How to fight production incidents? An empirical study on a large-scale cloud service. In: Proc. of the 13th Symp. on Cloud Computing. San Francisco: ACM, 2022. 126–141. [doi: 10.1145/3542929.3563482]

[10] Leesatapornwongsa T, Stuardo CA, Suminto RO, Ke H, Lukman JF, Gunawi HS. Scalability bugs: When 100-node testing is not enough. In: Proc. of the 16th Workshop on Hot Topics in Operating Systems. Whistler: ACM, 2017. 24–29. [doi: 10.1145/3102980.3102985]

[11] Liu HP, Lu S, Musuvathi M, Nath S. What bugs cause production cloud incidents? In: Proc. of the 2019 Workshop on Hot Topics in Operating Systems. Bertinoro: ACM, 2019. 155–162. [doi: 10.1145/3317550.3321438]

[12] Lou C, Huang P, Smith S. Understanding, detecting and localizing partial failures in large system software. In: Proc. of the 17th USENIX Symp. on Networked Systems Design and Implementation (NSDI 2020). Santa Clara: USENIX Association, 2020. 559–574.

[13] Ma MH, Yin Z, Zhang SL, Wang S, Zheng C, Jiang XH, Hu HW, Luo C, Li YL, Qiu NJ, Li FF, Chen CC, Pei D. Diagnosing root causes of intermittent slow queries in cloud databases. Proc. of the VLDB Endowment, 2020, 13(8): 1176–1189. [doi: 10.14778/3389133.3389136]

[14] Yuan D, Luo Y, Zhuang X, Rodrigues GR, Zhao X, Zhang YL, Jain PU, Stumm M. Simple testing can prevent most critical failures: An analysis of production failures in distributed data-intensive systems. In: Proc. of the 11th USENIX Symp. on Operating Systems Design and Implementation (OSDI 2014). Broomfield: USENIX Association, 2014. 249–265.

[15] Zhang YL, Yang JW, Jin ZQ, Sethi U, Rodrigues K, Lu S, Yuan D. Understanding and detecting software upgrade failures in distributed systems. In: Proc. of the 28th ACM SIGOPS Symp. on Operating Systems Principles. ACM, 2021. 116–131. [doi: 10.1145/3477132.3483577]

[16] Cheng Y, Wang L, Zhao XY. Review of root cause analysis research. Application Research of Computers, 2023, 40(4): 961–966 (in Chinese with English abstract). [doi: 10.19734/j.issn.1001-3695.2022.07.0450]

[17] Yu EY, Dong H, Ren YX, Yan MZ, Zhang XC, Yang Y, Yue L, Huang ZB. HRCA: A heterogeneous graph-based adaptive root cause analysis framework. In: Proc. of the 34th IEEE Int'l Symp. on Software Reliability Engineering Workshops (ISSREW). Florence: IEEE, 2023. 63–68. [doi: 10.1109/ISSREW60843.2023.00048]

[18] Ma QP, Li HY, Thorstenson A. A big data-driven root cause analysis system: Application of machine learning in quality problem solving. Computers & Industrial Engineering, 2021, 160: 107580. [doi: 10.1016/j.cie.2021.107580]

[19] Guo HX, Yuan SH, Wu XT. LogBERT: Log anomaly detection via BERT. In: Proc. of the 2021 Int'l Joint Conf. on Neural Networks (IJCNN). Shenzhen: IEEE, 2021. 1–8. [doi: 10.1109/IJCNN52387.2021.9534113]

[20] Le VH, Zhang HY. Log-based anomaly detection with deep learning: How far are we? In: Proc. of the 44th Int'l Conf. on Software Engineering. Pittsburgh: ACM, 2022. 1356–1367. [doi: 10.1145/3510003.3510155]

[21] Locke S, Li H, Chen THP, Shang WY, Liu W. LogAssist: Assisting log analysis through log summarization. IEEE Trans. on Software Engineering, 2022, 48(9): 3227–3241. [doi: 10.1109/TSE.2021.3083715]

[22] Sumers TR, Yao SY, Narasimhan K, Griffiths TL. Cognitive architectures for language agents. arXiv:2309.02427, 2023.

[23] Xi ZH, Chen WX, Guo X, *et al*. The rise and potential of large language model based agents: A survey. Science China Information Sciences, 2025, 68(2): 121101. [doi: 10.1007/s11432-024-4222-0]

[24] Park JS, O'Brien J, Cai CJ, Morris MR, Liang P, Bernstein MS. Generative agents: Interactive simulacra of human behavior. In: Proc. of the 36th Annual ACM Symp. on User Interface Software and Technology. San Francisco: ACM, 2023. 2. [doi: 10.1145/3586183.3606763]

[25] Wang L, Ma C, Feng XY, Zhang ZY, Yang H, Zhang JS, Chen ZY, Tang JK, Chen X, Lin YK, Zhao WX, Wei ZW, Wen JR. A survey on large language model based autonomous agents. Frontiers of Computer Science, 2024, 18(6): 186345. [doi: 10.1007/s11704-024-40231-1]

[26] Wang ZK, Zhang G, Yang KX, Shi N, Zhou WCS, Hao SC, Xiong GZ, Li YZ, Sim MY, Chen XY, Zhu QQ, Yang ZZ, Nik A, Liu Q, Lin CH, Wang S, Liu RB, Chen WH, Xu K, Liu DYH, Guo YK, Fu J. Interactive natural language processing. arXiv:2305.13246, 2023.

[27] Li MH, Zhao YX, Yu BW, Song FF, Li HY, Yu HY, Li ZJ, Huang F, Li YB. API-bank: A comprehensive benchmark for tool-augmented LLMs. In: Proc. of the 2023 Conf. on Empirical Methods in Natural Language Processing. Singapore: ACL, 2023. 3102–3116. [doi: 10.18653/v1/2023.emnlp-main.187]

[28] Ling Y, Wu FY, Dong SJ, Feng YR, Karypis G, Reddy CK. International workshop on multimodal learning—2023 theme: Multimodal learning with foundation models. In: Proc. of the 29th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining. Long Beach: ACM, 2023. 5868–5869. [doi: 10.1145/3580305.3599208]

[29] Qin YJ, Liang SH, Ye YN, Zhu KL, Yan L, Lu YX, Lin YK, Cong X, Tang XR, Qian B, Zhao SH, Hong L, Tian RC, Xie RB, Zhou J, Gerstein M, Li DH, Liu ZY, Sun MS. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In: Proc. of the 12th Int'l Conf. on Learning Representations. Vienna: OpenReview.net, 2024. 1–23.

[30] Schick T, Dwivedi-Yu J, Dessì R, Raileanu R, Lomeli M, Hambro E, Zettlemoyer L, Cancedda N, Scialom T. Toolformer: Language models can teach themselves to use tools. In: Proc. of the 37th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2023. 2997.

[31] Vemprala SH, Bonatti R, Bucker A, Kapoor A. ChatGPT for robotics: Design principles and model abilities. IEEE Access, 2024, 12: 55682–55696. [doi: 10.1109/ACCESS.2024.3387941]

[32] Wang YQ, Yao QM, Kwok JT, Ni LM. Generalizing from a few examples: A survey on few-shot learning. ACM Computing Surveys, 2021, 53(3): 63. [doi: 10.1145/3386252]

[33] Zhang LZ, Jia T, Jia MX, Wu YF, Liu AW, Yang Y, Wu ZH, Hu XM, Yu P, Li Y. A survey of AIOps in the era of large language models. ACM Computing Surveys, 2025, 58(2): 44. [doi: 10.1145/3746635]

[34] Zhou DZ, Fokaefs M. AI assistants for incident lifecycle in a microservice environment: A systematic literature review. arXiv:2410.04334, 2024.

[35] Wang TT, Qi GL. A comprehensive survey on root cause analysis in (micro) services: Methodologies, challenges, and trends. arXiv:2408.00803, 2024.

[36] Yu GB, Tan G, Huang HJ, Zhang ZY, Chen PF, Natella R, Zheng ZB, Lyu MR. A survey on failure analysis and fault injection in AI systems. ACM Trans. on Software Engineering and Methodology, 2026, 35(1): 28. [doi: 10.1145/3732777]

[37] Papineni K, Roukos S, Ward T, Zhu WJ. BLEU: A method for automatic evaluation of machine translation. In: Proc. of the 40th Annual Meeting of the Association for Computational Linguistics. Philadelphia: ACL, 2002. 311–318. [doi: 10.3115/1073083.1073135]

[38] Kane H, Kocyigit MY, Abdalla A, Ajanoh P, Coulibali M. NUBIA: Neural based interchangeability assessor for text generation. In: Proc. of the 1st Workshop on Evaluating NLG Evaluation. Dublin: ACL, 2020. 28–37.

[39] Ahmed T, Ghosh S, Bansal C, Zimmermann T, Zhang XC, Rajmohan S. Recommending root-cause and mitigation steps for cloud incidents using large language models. In: Proc. of the 45th IEEE/ACM Int'l Conf. on Software Engineering (ICSE). Melbourne: IEEE, 2023. 1737–1749. [doi: 10.1109/ICSE48619.2023.00149]

[40] Zhang XC, Ghosh S, Bansal C, Wang RJ, Ma MH, Kang Y, Rajmohan S. Automated root causing of cloud incidents using in-context learning with GPT-4. In: Proc. of the 32nd ACM Int'l Conf. on the Foundations of Software Engineering. Porto de Galinhas: ACM, 2024. 266–277. [doi: 10.1145/3663529.3663846]

[41] Jiang YX, Zhang CY, He SL, Yang ZH, Ma MH, Qin S, Kang Y, Dang YN, Rajmohan S, Lin QW, Zhang DM. Xpert: Empowering incident management with query recommendations via large language models. In: Proc. of the 46th IEEE/ACM Int'l Conf. on Software Engineering. Lisbon: ACM, 2024. 92. [doi: 10.1145/3597503.3639081]

[42] Hamadanian P, Arzani B, Fouladi S, Kakarla SKR, Fonseca R, Billor D, Cheema A, Nkposong E, Chandra R. A holistic view of AI-driven network incident management. In: Proc. of the 22nd ACM Workshop on Hot Topics in Networks. Cambridge: ACM, 2023. 180–188. [doi: 10.1145/3626111.3628176]

[43] Chen YF, Xie HB, Ma MH, Kang Y, Gao X, Shi L, Cao YJ, Gao XD, Fan H, Wen M, Zeng J, Ghosh S, Zhang XC, Zhang CY, Lin QW, Rajmohan S, Zhang DM, Xu TY. Automatic root cause analysis via large language models for cloud incidents. In: Proc. of the 19th European Conf. on Computer Systems. Athens: ACM, 2024. 674–688. [doi: 10.1145/3627703.3629553]

[44] Pei CH, Wang ZX, Liu FR, Li ZY, Liu Y, He X, Kang R, Zhang TY, Chen JJ, Li JH, Xie GG, Pei D. Flow-of-Action: SOP enhanced LLM-based multi-agent system for root cause analysis. In: Proc. of the 2025 ACM Web Conf. Sydney: ACM, 2025. 422–431. [doi: 10.

1145/3701716.3715225]

[45] Tabiban A, Zhao HY, Jarraya Y, Pourzandi M, Zhang MY, Wang LY. ProvTalk: Towards interpretable multi-level provenance analysis in networking functions virtualization (NFV). In: Proc. of the 29th Network and Distributed System Security Symp. San Diego: The Internet Society, 2022. 1–18. [doi: 10.14722/ndss.2022.23103]

[46] Huang JJ, Jiang ZH, Liu JY, Huo YT, Gu JZ, Chen ZB, Feng C, Dong H, Yang ZY, Lyu MR. Demystifying and extracting fault-indicating information from logs for failure diagnosis. In: Proc. of the 35th IEEE Int'l Symp. on Software Reliability Engineering (ISSRE). Tsukuba: IEEE, 2024. 511–522. [doi: 10.1109/ISSRE62328.2024.00055]

[47] Wang YD, Zhu ZRX, Fu QA, Ma YC, He PJ. MRCA: Metric-level root cause analysis for microservices via multi-modal data. In: Proc. of the 39th IEEE/ACM Int'l Conf. on Automated Software Engineering. Sacramento: ACM, 2024. 1057–1068. [doi: 10.1145/3691620.3695485]

[48] Kuang JX, Liu JY, Huang JJ, Zhong RY, Gu JZ, Yu L, Tan R, Yang ZY, Lyu MR. Knowledge-aware alert aggregation in large-scale cloud systems: A hybrid approach. In: Proc. of the 46th Int'l Conf. on Software Engineering: Software Engineering in Practice. Lisbon: ACM, 2024. 369–380. [doi: 10.1145/3639477.3639745]

[49] Yu GB, Chen PF, He ZL, Yan QY, Luo Y, Li FY, Zheng ZB. ChangeRCA: Finding root causes from software changes in large online systems. Proc. of the ACM on Software Engineering, 2024, 1(FSE): 2. [doi: 10.1145/3643728]

[50] Lee C, Yang TY, Chen ZB, Su YX, Lyu MR. Eadro: An end-to-end troubleshooting framework for microservices on multi-source data. In: Proc. of the 45th IEEE/ACM Int'l Conf. on Software Engineering (ICSE). Melbourne: IEEE, 2023. 1750–1762. [doi: 10.1109/ICSE48619.2023.00150]

[51] Shetty M, Chen YF, Somashekar G, Ma MH, Simmhan Y, Zhang XC, Mace J, Vandevoorde D, Las-Casas P, Gupta SM, Nath S, Bansal C, Rajmohan S. Building AI agents for autonomous clouds: Challenges and design principles. In: Proc. of the 2024 ACM Symp. on Cloud Computing. Redmond: ACM, 2024. 99–110. [doi: 10.1145/3698038.3698525]

[52] Chen YF, Shetty M, Somashekar G, Ma MH, Simmhan Y, Mace J, Bansal C, Wang RJ, Rajmohan S. AIOpsLab: A holistic framework to evaluate AI agents for enabling autonomous clouds. arXiv:2501.06706, 2025.

[53] Demarne M, Cilimdzic M, Falkowski T, Johnson T, Gramling J, Kuang W, Hou H, Aryan A, Subramaniam G, Lee K, Mejia M, Liu LS, Vermareddy D. Automated root cause analysis system for complex data products. arXiv:2412.15374, 2024.

[54] Wang ZX, Li JH, Ma MH, Li Z, Kang Y, Zhang CY, Bansal C, Chintalapati M, Rajmohan S, Lin QW, Zhang DM, Pei CH, Xie GG. Large language models can provide accurate and interpretable incident triage. In: Proc. of the 35th IEEE Int'l Symp. on Software Reliability Engineering (ISSRE). Tsukuba: IEEE, 2024. 523–534. [doi: 10.1109/ISSRE62328.2024.00056]

[55] Yu ZY, Ma MH, Zhang CY, Qin S, Kang Y, Bansal C, Rajmohan S, Dang YN, Pei CH, Pei D, Lin QW, Zhang DM. MonitorAssistant: Simplifying cloud service monitoring via large language models. In: Proc. of the 32nd ACM Int'l Conf. on the Foundations of Software Engineering. Porto de Galinhas: ACM, 2024. 38–49. [doi: 10.1145/3663529.3663826]

[56] Jin PX, Zhang SL, Ma MH, Li HZ, Kang Y, Li LQ, Liu YD, Qiao B, Zhang CY, Zhao P, He SL, Sarro F, Dang YN, Rajmohan S, Lin QW, Zhang DM. Assess and summarize: Improve outage understanding with large language models. In: Proc. of the 31st ACM Joint European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. San Francisco: ACM, 2023. 1657–1668. [doi: 10.1145/3611643.3613891]

[57] Wang ZF, Liu ZC, Zhang YY, Zhong AX, Wang JH, Yin FB, Fan LT, Wu LF, Wen QS. RCAgent: Cloud root cause analysis by autonomous agents with tool-augmented large language models. In: Proc. of the 33rd ACM Int'l Conf. on Information and Knowledge Management. Boise: ACM, 2024. 4966–4974. [doi: 10.1145/3627673.3680016]

[58] Roy D, Zhang XC, Bhave R, Bansal C, Las-Casas P, Fonseca R, Rajmohan S. Exploring LLM-based agents for root cause analysis. In: Proc. of the 32nd ACM Int'l Conf. on the Foundations of Software Engineering. Porto de Galinhas: ACM, 2024. 208–219. [doi: 10.1145/3663529.3663841]

[59] Zhang W, Guo HC, Yang J, Tian ZJ, Zhang Y, Chaoran Y, Li ZJ, Li TL, Shi X, Zheng LF, Zhang B. mABC: Multi-agent blockchain-inspired collaboration for root cause analysis in micro-services architecture. In: Proc. of the 2024 Findings of the Association for Computational Linguistics: EMNLP 2024. Miami: ACL, 2024. 4017–4033. [doi: 10.18653/v1/2024.findings-emnlp.232]

[60] Zhou XH, Li GL, Sun ZY, Liu ZY, Chen WZ, Wu JM, Liu JS, Feng RH, Zeng GY. D-Bot: Database diagnosis system using large language models. Proc. of the VLDB Endowment, 2024, 17(10): 2514–2527. [doi: 10.14778/3675034.3675043]

[61] Xu JJL, Zhang QN, Zhong ZQ, He SL, Zhang CY, Lin QW, Pei D, He PJ, Zhang DM, Zhang Q. OpenRCA: Can large language models locate the root cause of software failures? In: Proc. of the 13th Int'l Conf. on Learning Representations. Singapore: OpenReview.net, 2025. 1–29.

[62] Ezukwoke K, Hoayek A, Batton-Hubert M, Boucher X, Gounet P, Adrian J. Big GCVAE: Decision-making with adaptive Transformer

model for failure root cause analysis in semiconductor industry. Journal of Intelligent Manufacturing, 2025, 36(4): 2423–2438. [doi: 10.1007/s10845-024-02346-x]

[63] Razouk H, Kern R, Mischitz M, Moser J, Memic M, Liu L, Burmer C, Safont-Andreu A. AI-based knowledge management system for risk assessment and root cause analysis in semiconductor industry. In: Vermesan O, ed. Artificial Intelligence for Digitising Industry—Applications. New York: River Publishers, 2022. 113–129. [doi: 10.1201/9781003337232-11]

[64] Ding S, Xu YF, Lu Z, Tang F, Li T, Ge JG. Power microservices troubleshooting by pretrained language model with multi-source data. In: Proc. of the 2024 IEEE Int'l Symp. on Parallel and Distributed Processing with Applications (ISPA). Kaifeng: IEEE, 2024. 1768–1775. [doi: 10.1109/ISPA63168.2024.00241]

[65] Shehu Y, Harper R. Enhancements to language modeling techniques for adaptable log message classification. IEEE Trans. on Network and Service Management, 2022, 19(4): 4662–4675. [doi: 10.1109/TNSM.2022.3192756]

[66] Islam NT, Bethany M, Manuel D, Jadliwala M, Najafirad P. Unintentional security flaws in code: Automated defense via root cause analysis. arXiv:2409.00199, 2024.

[67] Ojuolape AE, Hu SF. Explainable fault diagnosis of control systems using large language models. In: Proc. of the 2024 IEEE Conf. on Control Technology and Applications (CCTA). Newcastle upon Tyne: IEEE, 2024. 491–498. [doi: 10.1109/CCTA60707.2024.10666634]

[68] Zhang Q, Xu C, Li J, Sun YC, Bao JS, Zhang D. LLM-TSFD: An industrial time series human-in-the-loop fault diagnosis method based on a large language model. Expert Systems with Applications, 2025, 264: 125861. [doi: 10.1016/j.eswa.2024.125861]

[69] Siddeshwar V, Azim A, Alwidian S, Makrehchi M. Towards enhancing aviation safety through advanced incident analysis using large language models. In: Proc. of the 34th Int'l Conf. on Collaborative Advances in Software and COmputiNg (CASCON). Toronto: IEEE, 2024. 1–7. [doi: 10.1109/CASCON62161.2024.10837802]

[70] Sarda K, Namrud Z, Watts I, Shwartz L, Nagar S, Mohapatra P, Litoiu M. Augmenting automatic root-cause identification with incident alerts using LLM. In: Proc. of the 34th Int'l Conf. on Collaborative Advances in Software and Computing (CASCON). Toronto: IEEE, 2024. 1–10. [doi: 10.1109/CASCON62161.2024.10838171]

[71] Sun Q, Li YH, Zhou CJ, Tian YC. Root cause analysis for industrial process anomalies through the integration of knowledge graph and large language model. In: Proc. of the 43rd Chinese Control Conf. (CCC). Kunming: IEEE, 2024. 6855–6860. [doi: 10.23919/CCC63176.2024.10662704]

[72] Goel D, Husain F, Singh A, Ghosh S, Parayil A, Bansal C, Zhang XC, Rajmohan S. X-lifecycle learning for cloud incident management using LLMs. In: Proc. of the 32nd ACM Int'l Conf. on the Foundations of Software Engineering. Porto de Galinhas: ACM, 2024. 417–428. [doi: 10.1145/3663529.3663861]

[73] Shao S, Yu TT. Enhancing IR-based fault localization using large language models. arXiv:2412.03754, 2024.

[74] Han YQ, Du QF, Huang Y, Wu JQ, Tian FL, He C. The potential of one-shot failure root cause analysis: Collaboration of the large language model and small classifier. In: Proc. of the 39th IEEE/ACM Int'l Conf. on Automated Software Engineering. Sacramento: ACM, 2024. 931–943. [doi: 10.1145/3691620.3695475]

[75] Vidyaratne L, Lee XY, Kumar A, Watanabe T, Farahat A, Gupta C. Generating troubleshooting trees for industrial equipment using large language models (LLM). In: Proc. of the 2024 IEEE Int'l Conf. on Prognostics and Health Management (ICPHM). Spokane: IEEE, 2024. 116–125. [doi: 10.1109/ICPHM61352.2024.10626823]

[76] Khan A, Nahar R, Chen H, Constante Flores GE, Li C. FaultExplainer: Leveraging large language models for interpretable fault detection and diagnosis. Computers & Chemical Engineering, 2025, 199: 109152. [doi: 10.1016/j.compchemeng.2025.109152]

[77] Zhang XC, Mittal T, Bansal C, Wang RJ, Ma MH, Ren ZX, Huang H, Rajmohan S. FLASH: A workflow automation agent for diagnosing recurring incidents. 2024. https://www.microsoft.com/en-us/research/publication/flash-a-workflow-automation-agent-for-diagnosing-recurring-incidents

[78] Jambigi N, Hammesfahr J, Mueller M, Bach T, Felderer M. On enhancing root cause analysis with SQL summaries for failures in database workload replays at SAP HANA. In: Proc. of the 35th IEEE Int'l Symp. on Software Reliability Engineering Workshops (ISSREW). Tsukuba: IEEE, 2024. 85–90. [doi: 10.1109/ISSREW63542.2024.00052]

[79] Xie ZQ, Zheng YJ, Ottens L, Zhang K, Kozyrakis C, Mace J. Cloud atlas: Efficient fault localization for cloud systems using language models and causal insight. arXiv:2407.08694, 2024.

[80] Huang H, Shah T, Karigiannis J, Evans S. Physics and data collaborative root cause analysis: Integrating pretrained large language models and data-driven AI for trustworthy asset health management. Annual Conf. of the PHM Society, 2024, 16(1): 1–12. [doi: 10.36001/phmconf.2024.v16i1.3881]

[81] Herrmann JE, Gopinath AM, Norrlof M, Müller MN. Diagnosing robotics systems issues with large language models.

arXiv:2410.09084, 2024.

[82] Zhang D, Zhang XC, Bansal C, Las-Casas P, Fonseca R, Rajmohan S. LM-PACE: Confidence estimation by large language models for effective root causing of cloud incidents. In: Proc. of the 32nd ACM Int'l Conf. on the Foundations of Software Engineering. Porto de Galinhas: ACM, 2024. 388–398. [doi: 10.1145/3663529.3663858]

[83] Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih WT, Rocktäschel T, Riedel S, Kiela D. Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 793.

[84] Dorri A, Kanhere SS, Jurdak R. Multi-agent systems: A survey. IEEE Access, 2018, 6: 28573–28593. [doi: 10.1109/ACCESS.2018.2831228]

[85] Weng LL. LLM powered autonomous agents | Lil'Log. 2023. https://lilianweng.github.io/posts/2023-06-23-agent

[86] Dong QX, Li L, Dai DM, Zheng C, Ma JY, Li R, Xia HM, Xu JJ, Wu ZY, Chang BB, Sun X, Li L, Sui ZF. A survey on in-context learning. In: Proc. of the 2024 Conf. on Empirical Methods in Natural Language Processing. Miami: ACL, 2024. 1107–1128. [doi: 10.18653/v1/2024.emnlp-main.64]

[87] Wei J, Wang XZ, Schuurmans D, Bosma M, Ichter B, Xia F, Chi EH, Le QV, Zhou D. Chain-of-thought prompting elicits reasoning in large language models. In: Proc. of the 36th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2022. 1800.

[88] Gao YF, Xiong Y, Gao XY, Jia KX, Pan JL, Bi YX, Dai Y, Sun JW, Wang M, Wang HF. Retrieval-augmented generation for large language models: A survey. arXiv:2312.10997, 2023.

[89] Brown TB, Mann B, Ryder N, *et al*. Language models are few-shot learners. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 159.

[90] Mernik M, Heering J, Sloane AM. When and how to develop domain-specific languages. ACM Computing Surveys, 2005, 37(4): 316–344. [doi: 10.1145/1118890.1118892]

[91] Beaulieu A. Learning SQL: Master SQL Fundamentals. 2nd ed., Sebastopol: O'Reilly Media, 2009.

[92] Hou XY, Zhao YJ, Wang SN, Wang HY. Model context protocol (MCP): Landscape, security threats, and future research directions. arXiv:2503.23278, 2025.

[93] Hegland M. The apriori algorithm—A tutorial. In: Goh SS, Ron A, Shen ZW, eds. Mathematics and Computation in Imaging Science and Information Processing. Singapore: World Scientific Publishing, 2007. 209–262. [doi: 10.1142/9789812709066_0006]

[94] Sun YQ, Zhao YJ, Su Y, Liu DP, Nie XH, Meng Y, Cheng SW, Pei D, Zhang SL, Qu XP, Guo XY. HotSpot: Anomaly localization for additive KPIs with multi-dimensional attributes. IEEE Access, 2018, 6: 10909–10923. [doi: 10.1109/ACCESS.2018.2804764]

[95] Liu P, Xu HW, Ouyang QY, Jiao R, Chen ZK, Zhang SL, Yang JH, Mo LL, Zeng JC, Xue WM, Pei D. Unsupervised detection of microservice trace anomalies through service-level deep Bayesian networks. In: Proc. of the 31st IEEE Int'l Symp. on Software Reliability Engineering (ISSRE). Coimbra: IEEE, 2020. 48–58. [doi: 10.1109/ISSRE5003.2020.00014]

[96] Mampaka MM, Sumbwanyambe M. Poor data throughput root cause analysis in mobile networks using deep neural network. In: Proc. of the 2nd IEEE Wireless Africa Conf. (WAC). Pretoria: IEEE, 2019. 1–6. [doi: 10.1109/AFRICA.2019.8843409]

[97] Zheng LC, Chen ZZ, Chen HF, He JR. Online multi-modal root cause analysis. arXiv:2410.10021, 2024.

[98] Hu EJ, Shen YL, Wallis P, Allen-Zhu Z, Li YZ, Wang SA, Wang L, Chen WZ. LoRA: Low-rank adaptation of large language models. In: Proc. of the 10th Int'l Conf. on Learning Representations. OpenReview.net, 2022. 1–13.

[99] Yao SY, Zhao J, Yu D, Du N, Shafran I, Narasimhan KR, Cao Y. ReAct: Synergizing reasoning and acting in language models. In: Proc. of the 11th Int'l Conf. on Learning Representations. Kigali: OpenReview.net, 2023. 1–33.

[100] Robertson S, Zaragoza H. The probabilistic relevance framework: BM25 and beyond. Foundations and Trends® in Information Retrieval, 2009, 3(4): 333–389. [doi: 10.1561/1500000019]

[101] Sokolova M, Lapalme G. A systematic analysis of performance measures for classification tasks. Information Processing & Management, 2009, 45(4): 427–437. [doi: 10.1016/j.ipm.2009.03.002]

[102] Banerjee S, Lavie A. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In: Proc. of the 2005 ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization. Ann Arbor: ACL, 2005. 65–72.

[103] Lin CY. ROUGE: A package for automatic evaluation of summaries. In: Text Summarization Branches Out. Barcelona: ACL, 2004. 74–81.

[104] Johnson M, Schuster M, Le QV, Krikun M, Wu YH, Chen ZF, Thorat N, Viégas F, Wattenberg M, Corrado G, Hughes M, Dean J. Google's multilingual neural machine translation system: Enabling zero-shot translation. Trans. of the Association for Computational

Linguistics, 2017, 5: 339–351. [doi: 10.1162/tacl_a_00065]

[105] Zhang TY, Kishore V, Wu F, Weinberger KQ, Artzi Y. BERTScore: Evaluating text generation with BERT. In: Proc. of the 8th Int'l Conf. on Learning Representations. Addis Ababa: OpenReview.net, 2020. 1–43.

[106] Sellam T, Das D, Parikh A. BLEURT: Learning robust metrics for text generation. In: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics. ACL, 2020. 7881–7892. [doi: 10.18653/v1/2020.acl-main.704]

[107] Yuan WZ, Neubig G, Liu PF. BARTScore: Evaluating generated text as text generation. In: Proc. of the 35th Int'l Conf. on Neural Information Processing Systems. Curran Associates Inc., 2021. 2088.

[108] Yang XL, Lin Y, Zhang YF, Huang LP, Dong JS, Mei H. DeepDebugger: An interactive time-travelling debugging approach for deep classifiers. In: Proc. of the 31st ACM Joint European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. San Francisco: ACM, 2023. 973–985. [doi: 10.1145/3611643.3616252]

[109] Liu RF, Lin Y, Yang XL, Dong JS. Debugging and explaining metric learning approaches: An influence function based perspective. In: Proc. of the 36th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2022. 568.

附中文参考文献

[16] 程燕, 王磊, 赵晓永. 根因分析研究综述. 计算机应用研究, 2023, 40(4): 961–966. [doi: 10.19734/j.issn.1001-3695.2022.07.0450]

作者简介

康俊驰, 硕士生, 主要研究领域为大语言模型智能体, 故障根因分析.

丁博, 博士, 研究员, CCF 专业会员, 主要研究领域为分布式计算, 软件维护与演化.

冯大为, 博士, 研究员, CCF 专业会员, 主要研究领域为分布式计算, 智能软件系统, 智能计算技术.

翟远钊, 博士, 助理研究员, 主要研究领域为大模型智能体, 大模型对齐, 强化学习.

张迅晖, 博士, 助理研究员, 主要研究领域为软件仓库挖掘, 开源知识产权保护, 代码克隆检测.

王怀民, 博士, 教授, CCF 会士, 主要研究领域为分布式系统, 云际计算, 群体智能.