基于LLM的日志故障诊断

许 婷[1],肖 桐[2],张圣林[1],孙一丹[1],孙永谦[1],裴 丹[2*]
(1. 南开大学软件学院,天津 300457;2. 清华大学计算机科学与技术系,北京 100084)

摘 要: 随着软件服务系统日益庞大、复杂,基于日志的故障诊断对保证软件服务的可靠性至关重要. 已有的日志故障诊断方法虽然可以确定故障类型,但无法为其推理过程提供解释让运维人员信服,从而导致它们难以在实际生产环境中进行部署. 为此,本文提出了一种全新的通过自动构建思维链指令提示(log Chain of Thought-Prompting,CoT-Prompting)来进行日志故障诊断的框架——LogCoT(Log Chain of Thought),它利用基于两阶段思维链提示工程(Auto-Few-Shot-CoT,Auto-FSC)算法,通过大语言模型(Large Language Model,LLM)提取日志的语义信息,从而生成可解释的根因分析报告. 此外,LogCoT结合无类别标注的指令优化(prompt-tuning)工程和有类别标注的参数微调(preference-tuning)技术优化微调Mistral基座模型. 然后通过大模型反馈身份偏好优化(Large-Language Model feedback Identity Preference Optimisation,LLMf-IPO)算法纠正Mistral生成的错误诊断结果,以更好对齐用户意图. 最后,本文基于从一家互联网服务提供商和一家云服务提供商的生产环境中收集的两个日志数据集对LogCoT的性能进行了全面综合的实验评估. 实验结果表明,LogCoT在Accuracy、Macro-F1、Weighted-F1等三个性能指标上均优于当前典型的基线模型,在两个数据集上比现有最佳模型的Accuracy分别高出31.88个百分点和10.51个百分点.

关键词: 日志故障诊断;可解释性;大语言模型;提示工程;偏好对齐;思维链
基金项目: 国家自然科学基金(No.62272249,No.62302244)
中图分类号: TP391 **文献标识码:** A **文章编号:** 0372-2112(2025)04-1123-19
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20240801

Log Fault Diagnosis Based on Large Language Models

XU Ting[1], XIAO Tong[2], ZHANG Sheng-lin[1], SUN Yi-dan[1], SUN Yong-qian[1], PEI Dan[2*]
(1. *College of Software*, *Nankai University*, *Tianjin* 300457, *China*;
2. *Deptment of Computer Science and Technology*, *Tsinghua University*, *Beijing* 100084, *China*)

Abstract: As the software service systems become increasingly large and complex, log-based fault diagnosis is critical to ensure the reliability of software services. Although existing research in log fault diagnosis methods can identify the type of the fault, they often fails to explain the reasoning process to convince the operation and maintenance personnel, which makes the above method challenging to apply in the production environment. The LogCoT (Log Chain of Thought) is proposed in this paper as a new framework for fault diagnosis based on automatically constructing chain of thought prompting (CoT-Prompting) to address the above issues. The auto-few-shot-CoT (Auto-FSC) algorithm of the two-stage CoT-Prompting engineering extracts semantic information from the large language mode (LLM) table root cause analysis reports. In addition, the combination of prompt-tuning with category-unlabelled and preference-tuning with category-labelled is used to optimally align the base model Mistral. Then, the large language model feedback identity preference optimisation (LLMf-IPO) algorithm is used to correct the wrong diagnosis results generated by the base model Mistral to better align the user's intention. Finally, we provide a comprehensive experimental evaluation of LogCoT's performance based on two log datasets collected from the production environment of the top-tier global Internet service provider and a cloud service provider. The experimental results show that LogCoT outperforms the three baseline models in three performance metrics, including Accuracy, Macro-F1, and Weighted-F1 on two datasets, and outperforms the Accuracy of the best existing model by 31.88 percentage points, 10.51 percentage points, respectively.

Key words: fault diagnosis of log; interpretability; large language model; prompt engineering; preference alignment; chain of thought
Foundation Item(s): National Natural Science Foundation of China (No.62272249, No.62302244)

收稿日期:2024-09-03;修回日期:2025-04-18;责任编辑:宋甲英
*通讯作者:裴丹

1 引言

随着现代软件系统的规模性和复杂性不断提升，故障的发生频率也随之不断升高[1,2]. 为了及时诊断故障并将系统恢复到正常运行状态，往往需要有经验的工程师们查看系统产生的海量日志并进行多轮沟通. 这种人工排查根因并做出运维决策的方式不仅耗时耗力、容易出错，而且随着日志数据量的急剧增加逐渐变得不再可行. 因此，基于日志的自动化故障诊断受到了学术界和工业界的极大关注[3,4]. 然而，尽管现有的很多方法[5~11]在公开数据集上已经取得了显著的效果，但它们往往只给出故障的类型[12~15]，而不解释推理的过程，导致其结果难以让运维人员信服. 因此，这些方法没有在实际生产环境中受到运维人员的青睐.

近几年，大语言模型(Large Language Model，LLM)迅猛发展，在很多自然语言处理任务上表现突出. 由于日志是一种自然语言的半结构化文本，可以很容易地用LLM进行分析处理. 鉴于LLM强大的理解能力和逻辑推理能力，基于LLM的日志故障诊断可为诊断结果给出易于理解、可解释性的推理过程，方便运维人员理解从而相信故障诊断的结果. 然而，将LLM应用于日志故障诊断面临如下挑战：

(1)基于中等规模开源LLM的日志故障诊断准确性欠佳. 由于超大规模闭源LLM(参数量在700亿以上，如GPT-4o、Claude3.5 Sonnet、GLM-4-Plus等)使用成本过高、定制化程度低且严重依赖外部服务的稳定性. 另外，企业通常为运维团队分配的算力资源有限. 因此采用中等规模开源LLM(参数量一般不超过100亿，如Llama-3.1-8B-Instruct、Gemma-2-9B-it、Qianwen-2-7B-Instrauct)诊断故障根因更符合实际需求. 但由于中等规模开源LLM的参数量相对较少，因此其进行日志故障诊断容易出现幻觉导致错误诊断结果.

(2)人工生成推理过程不仅耗时耗力且难以应对新类型的故障. 使用中等规模LLM时，一般基于提示策略(prompting)生成推理过程，如链式思维提示[16](Chain-of-Thought-Prompting，CoT-Prompting)，但在零样本[17](zero-shot)场景下，中等规模LLM生成的推理过程往往准确性较低，从而需要人工生成大量的推理过程样本对LLM进行微调. 然而，人工撰写推理过程案例通常会耗费大量的人力资源，且随着软件系统不断演进会持续发生新类型故障，这使得运维人员需持续生成新的推理过程以应对新类型故障.

为了应对上述挑战，本文提出了一个自动构建CoT-Prompting的日志故障诊断框架——LogCoT(Log Chain of Thought). 具体来说，本文首先设计了一种新型自动少样本思维链(Auto-Few-Shot-CoT，Auto-FSC)算法. 该算法利用超大规模闭源LLM构造思维链以实现自动推理，并借助其生成的高质量示例逐步引导中等规模开源LLM[18~21]进行自动推理并分析未见过的故障，从而提高生成推理过程的准确性. 其次，本文设计了一种新型大模型反馈身份偏好优化(Large Language Model feedback Identity Preference Optimisation，LLMf-IPO)算法. 该算法通过结合多个中等规模LLM的诊断结果纠正所选取的基座模型(Mistral)生成的错误诊断结果. 最后，本文结合指令优化(prompt-tuning)工程[22]和偏好微调(preference-tuning)技术[23]，以混合调优方式增强LogCoT的推理效果.

总结来说，本文的主要贡献如下：

(1)为了应对第一个挑战，本文设计了LLMf-IPO算法. 该算法不仅可以解决语义理解错误和思考步骤遗漏的问题，还可以有效利用中等规模开源LLM的反馈信号[24,25]构造偏好数据集，提升纠错数据集的质量. 另外，该算法实现了模型优化偏好对齐，无须人工介入便能提升模型的准确性，并解决了上下文记忆遗忘难题.

(2)为了应对第二个挑战，本文提出Auto-FSC算法. 该算法使用零样本思维链指令提示(Zero-Shot-CoT-Prompting)工程[26,27]自动生成、获取有关日志的推理样本，并通过选用少量示例对中等规模LLM进行少样本思维链指令提示[28~30](Few-Shot-CoT-Prompting)，以将超大规模闭源LLM的强大推理能力迁移至中等规模开源LLM，实现了少样本场景下高效且准确的推理，提高了输出结果的准确性和可解释性.

(3)基于从一家互联网服务提供商和一家云服务提供商的生产环境中收集的两个日志数据集对LogCoT的性能进行了全面综合的实验评估. 实验结果表明，LogCoT在Accuracy、Macro-F1、Weighted-F1这三个性能指标上均优于当前典型的三种基线模型，在两个数据集上比现有最佳模型的Accuracy分别高出31.88个百分点和10.51个百分点.

2 相关工作和背景知识

日志故障诊断是IT运维管理中一个关键的组成部分，对于确保系统稳定性和性能至关重要. 故障诊断，其目的是确定故障的根本原因，通常被形式化为一个分类任务. 近年来，国内外研究人员提出了一些故障诊断方法，下面将从基于现有的深度学习、机器学习等领域以及目前比较热门的LLM领域分别介绍在故障诊断方面的相关工作.

2.1 故障诊断相关工作

由于采用手动故障诊断方式，容易出错且耗费大量人力. 因此本文将从以下三个领域分析有关日志故障诊断工作.

2.1.1 机器学习

LogCluster[31]用向量表示每个日志序列,采用词频-逆向文件频率(Term Frequency-Inverse Document Frequency,TF-IDF)[32]方法将日志事件序列转化为加权的事件计数向量,然后通过层次聚类,一个日志序列被分类为与最近的聚类相同的故障类型. Cloud19[33]通过Word2Vector[34]算法进行语义特征提取,用向量表示每个日志项,向量化后获得异常的特征表示,并使用分类器来识别每个向量并附加标签,以指示故障类型. 它们[31,33]在预处理含有大量噪声和冗余信息的日志数据时存在不足,会降低模式识别的准确性.

2.1.2 深度学习

MoniLog[35]是一种在大规模环境中实时进行异常检测的分布式方法,通过构建日志流并执行异常序列的监控,从而检测结构化日志流中的Sequential、定量异常、异常的组件. 目标输出是具有指定临界度的分类异常流. 这种方法没有明确给出故障失败的根本原因,例如虚拟机关闭、软件漏洞、进程重启,因此它们并不适用于我们的场景. SwissLog[36]将语义嵌入和时间嵌入方法相结合用于诊断故障. 主要针对导致日志序列顺序更改和日志时间间隔更改的故障. 这三者在面对模棱两可的错误信息时,或解析复杂或模糊的日志数据时会产生误报或漏报的情况.

2.1.3 数据挖掘

LogBASA[37]使用一个预先训练的BERT模型来感知日志事件序列的语义信息. 当异常检测模型检测到异常时,系统功能路径图可以提高故障诊断效率. LogKG[38]从日志中充分提取实体和关系,通过知识图谱挖掘多字段信息及其关系,然后使用面向故障的日志表示模块FOLR来表示故障日志序列,以此提取与故障相关的模式. 最后,使用OPTICS[39]聚类的历史故障日志序列的在线故障诊断. 这两种方法都依赖于图,在特定类型的日志数据上表现良好,但在未知或新类型的日志格式上泛化能力不足.

综上所述,以上现有的三大类方法均有弊端,在预处理含有大量噪声和冗余信息的日志时,模式识别能力低,解析复杂或模糊的日志易产生误报、漏报,难以使用自定义格式的日志导致泛化能力差. 最大缺点是它们无法为故障分类结果提供可解释性. 最近随着LLM在各大领域迅猛发展,已有研究将LLM应用于日志分析[40~42]方面,这些LLM方法都集中在日志解析[43,44]或异常检测[45]方面,而在日志的故障诊断方面仍然处于空白状态,需要弥补这一短板.

2.2 LLM的基础知识

2.2.1 思维链提示工程

思维链提示工程是一种在自然语言处理领域中用于改进大语言模型推理能力的技术. 通过设计特定的提示,可以使模型更好地理解任务并生成更合理的输出. 本文通过指令提示工程,引导LLM输出故障类别的结果. 对于标准提示工程,系统输入脱敏日志,并给定推理问题Q、提示工程ξ和参数化概率超大规模闭源LLM,记作P_c,ξ由二元组(Q,L)组成. 目标是最大化故障分类答案L的可能性,如式(1)所示:

$$P\left(L|\xi,Q\right)=\prod_{t=1}^{|L|}P_c\left(l_t|\xi,Q,l<t\right) \tag{1}$$

其中,l_t表示第t个令牌,$|L|$表示最终答案的长度. 虽然模型在生成回答简单问题方面表现出色,但在处理需要多步骤推理或复杂逻辑推理的日志诊断信息时,仍然可能出现困难. 传统的零样本或少样本提示方法往往只引导模型输出单一答案,忽略了问题背后的复杂推理过程,因此需要引入思维链技术.

CoT关键在于提示的设计,通常在提示中使用类似“让我们逐步思考(let's think step by step)”这样的语言,引导模型根据输入的日志内容进行分步推理. 通过识别和连接因果关系,CoT逐步分析故障工单中的每条日志的语义,并生成对应的解释. 通过链式思维可以提供分析相关清晰的日志路径. 这样对分析故障发生前后的日志记录,可以更清晰地理解故障的传播路径和影响范围. 我们将从以下两个方面介绍思维链提示工程.

(1)零样本思维提示(Zero-Shot-CoT-Prompting)

零样本思维指令提示是一种在没有给定具体日志示例的情况下,通过提示工程利用大型语言模型(LLM)的推理能力来直接引导模型生成解决分类问题的推理过程步骤的技术. 过去的研究[46~51]已表明CoT关键点在于可解释性的中间步骤[52](rationales),它可以用作额外的输入来指导模型对无标注数据进行预测. 通常这种提示会包含明确的指示,告诉模型需要输出连贯的推理链或解答过程,尽管模型从未在类似的提示上被训练过.

对于零样本思维指令提示工程,CoT将日志推理过程R添加到标准提示(prompting)中. 在每次推理时将附加到二元组(Q,L)的日志记录上下文中,如式(2)所示,并在R中植入超大规模闭源LLM,记作C_g,让其快速接入标准化的API接口并进行推理. 如式(3)所示,在植入日志诊断信息的生成过程中,超大规模闭源LLM具有优化日志语义理解和推理复杂任务的能力. 于是我们便得到新的提示工程$\xi=\{(D_t,R_t,L_t)\}_{t=1}^{v*w}$,其中$w$是故障类别数,$v$是每个类别下的示例数. 基于这些情况,本文Zero-Shot-CoT-Prompting被定义为如式(4)所示.

$$P\left(R|\xi,D\right)=\prod_{s=1}^{|R|}P_c\left(R_s^z|\xi,D,r<s\right) \tag{2}$$

$$P\left(L|\xi,R,D\right)=\prod_{t=1}^{|L|}P_c\left(L_t|\xi,R,D,l<t\right) \tag{3}$$

$$P\left(L\left|\xi,Q\right.\right)=\sum_{R}\prod_{t=1}^{|L|}P_c\left(L_t|\xi,R,D,l<t\right)\prod_{s=1}^{|R|}P_c\left(R_s^z|\xi,D,r<s\right) \tag{4}$$

其中，R^Z是总推理步骤的第Z步，L_t表示第t个令牌，$|L|$表示最终答案的长度，R_s表示第s个令牌，$|R|$表示最终答案的长度.

(2)少样本思维提示(Few-Shot-CoT-Promting)

少样本思维链提示其定义是将少样本学习(few-shot-learning)与推理链提示(CoT)相结合，通过少量日志示例和逐步推理的提示，引导中等规模开源LLM进行复杂的推理和分析任务，以此自动构建复杂的输入-理由-输出的映射. 与目前需要大量标注数据的方法相比，少样本学习更加高效且灵活，特别适用于那些难以获取大量标注数据的场景.

由零样本思维提示得到的推理样本，可选用其中少量示例，记作$E=(D,R,L)$. 假设基于ICL[53]日志测试样本输入表示为<input,demonstrations>，那么加入少样本思维提示的日志测试样本，其输入可表示为<input,CoT>. 在ICL的范式中，即使仅提供中等规模开源LLM的几个演示作为少样本提示工程中的示例，但表现仍高出标准完全监督方法. 具体来说，ICL无需训练参数优化，直接提示Mistral，这表现在新测试日志样本中加入示例E，来重构少样本思维链提示工程. CoT对每个日志示例E，均会使用中间推理过程来重构示例，使Mistral在对新日志样本预测时，先生成中间推理的思维链，再生成分类结果，从而实现规模超大LLM引导规模中等模型Mistral的推理. 其目的是提升Mistral在新日志样本中的表现. 换句话说，将模型置于一些特定任务的示例中有助于提高模型的推理性能. 由于需要更新日志推理过程$R'^w_{v_i}$，因此本文的Few-Shot-CoT-Prompting被定义为式(5). 总而言之，整个提示过程涉及指令微调可以有效地逐步推理每个故障案例的原因，识别故障并缩小故障范围.

$$P\left(L^w_{v_i}|\xi,D_{\text{test}},E^w_{v_i}\right)=\sum_{R}\prod_{t=1}^{|L|}P_m\left(R'^w_{v_i}|\xi,D_{\text{test}},E^w_{v_i},l<t\right)\prod_{s=1}^{|R|}P_m\left(L'^w_{v_i}|\xi,R'^w_{v_i},D_{\text{test}},E^w_{v_i},r<s\right) \tag{5}$$

2.2.2 偏好学习(preference learning)

在模型与人类意图对齐方面，人类反馈的强化学习(Reinforcement Learning with Human Feedback，RLHF)已经成为一大流行范式. 然而RLHF受到数据的限制，它需要大量的人工反馈和偏好数据. 这样一来，不仅会消耗大量的人力资源，还可能会引入人为的偏见. 为寻求更简单有效的优化策略，出现了三种有前景的RLHF对齐的算法：直接偏好优化(Direct Preference Optimization，DPO)[54]、身份偏好优化(Identity Preference Optimisation，IPO)[55]和卡尼曼-特沃斯基优化(Kahneman-Tversky Optimization，KTO)[56]. 通常来说，对齐算法的工作过程分为两个阶段：使用人类偏好学习一个奖励函数；采用强化学习优化所学习的奖励来对齐模型.

偏好学习旨在根据一组训练样例来学习一个模型，日志样例表示了一组对象的相对偏好. 目标是预测在新的、未见过的日志数据对上的偏好关系. 在成对偏好日志数据对中，通常会构建一个函数Ξ，它能够对每对对象(D_t,L_t)计算一个偏好分数，模型的目标是最小化损失函数，如式(6)所示：

$$\min_{\Xi}\sum_{(D_t,L_t)\sim\mathcal{D}}P\left(\Xi(D_t,L_t),\lambda_t\right) \tag{6}$$

其中，D是成对的日志训练数据集，λ_t表示对象D_t相对于L_t的偏好标签. P是一个损失函数，用于衡量$\Xi(D_t,L_t)$预测与实际偏好λ_t之间的误差. 此外，为了解决过拟合问题，本文采用IPO对齐训练，主要是对偏好概率(preference-probability)进行非线性变化的非衰减函数，如式(7)所示：

$$\Psi:[0,1]\to\mathbb{R} \tag{7}$$

其中，Ψ的偏好优化目标定义为式(8)：

$$\max_{\Xi_*}\mathbb{E}_{D\sim\rho}\left[\Psi\left(\rho^*\left(L\succ L'|D\right)\right)\right]-\tau\mathbb{D}_{\text{KL}}\left(\Xi_*||\Xi_{m'}\right) \tag{8}$$

如果给Ψ具体定义，即为式(9)：

$$\Psi(q)=\log\left(q/(1-q)\right) \tag{9}$$

通过这种方式，偏好学习模型能够从日志数据中捕捉到复杂的偏好模式，学习到不同故障类别的日志症状的偏好表征. 对新日志数据中的故障症状进行识别并根据严重程度进行排序，辨别同一故障案例出现最严重的故障根因优先级.

3 LogCoT框架

本节将展开介绍LogCoT框架，如图1和图2所示. 该框架主要由三个部分组成：日志预处理、自动构造推理链示例Auto-FSC算法、模型反馈的偏好对齐LLMf-IPO算法，详细介绍如下.

3.1 LogCoT结构

LogCoT框架通过利用商业化的超大规模闭源LLM推理能力(本文对商业性超大规模闭源LLM不作限制，这里仅指通用框架)，以高效指令优化中等规模开源LLM，并通过微调将中等规模开源LLM对齐用户意图. 本研究采用了混合调优的技术，即对基座模型Mistral进行Prompt-Tuning指令优化和Preference-Tuning偏好微调训练以达到最大化模型推理能力，如图1所示. LogCoT主要包括三阶段，其中第二阶段又可进一步细化为两部分，具体如下.

(1)第一阶段. 原始日志经过对关键字段进行脱敏处理后，再利用TF-IDF算法的特征工程筛选出故障相关日志.

(2)第二阶段. CoT-Prompting指令提示工程,Zero-Shot-CoT-Prompting 和 Few-Shot-CoT-Prompting 两个部分. 首先,给予超大规模闭源LLM(比如Claude、GPT-4o均担任推理执行器角色)的零样本思维链指令提示工程,自动构建小样本推理链,将其生成的日志解释作为有推理链的示例样本. 然后,从超大规模闭源LLM生成的日志示例池中选取少量示例,并将其输入到少样本思维链指令提示工程中以引导基座模型Mistral推理.

(3)第三阶段. 采用辅助的三大中等规模开源LLMs,得到有关诊断日志信息的不同推理回应. 紧接着,再次借助超大规模闭源LLM(充当评估器角色)对不同中等规模开源LLM的推理回复进行排名以最大化偏好模型中优选回复,从而构造优质偏好数据集进行IPO训练以纠错相关日志的诊断结果.

通过以上三个阶段,本文进行两次微调策略,在基座模型Mistral中实现推理复杂任务,可得到更具准确性且合理性的解释,也即作为故障发生后的诊断信息,从而提炼合理的推理过程,给运维工程师提供相关的故障修复建议.

本文LogCoT框架可分为模型训练和推理诊断两个部分. 图1显示了一个高级用户与LLM的问答响应交互过程,其中训练流程如图1所示的左中右三部分:左部分显示了两阶段链式思维链的提示工程(CoT-Prompting),即从LLM中提取日志诊断信息的Auto-FSC算法,包含超大规模闭源LLM构造推理链和生成示例池这两个步骤(更多详情可见3.3.1节);右部分展示了模型反馈偏好对齐LLMf-IPO算法,主要是通过三个中等规模开源LLM生成推理回应并由超大规模闭源LLM评分方式收集二元反馈信号,以构造偏好数据集对齐用户意图;中间部分是基座模型Mistral进行混合调优策略的训练,其训练过程把指令优化提示工程和偏好微调纠错技术两者相结合以达到最优模型性能. 此外,图2展示了在业务场景中实际部署规模中等基座模型Mistral的诊断过程. 主要是通过预处理日志筛选故障相关的日志,再使用优化微调后的Mistral对新日志任务进行推理诊断,实现故障分类的目的.

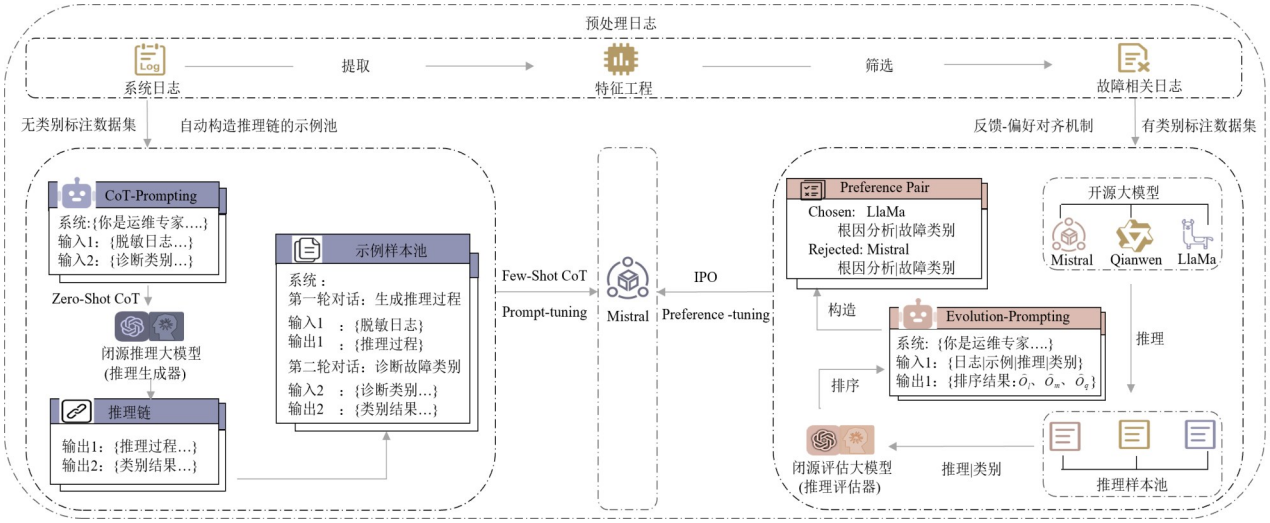


图1　模型训练阶段

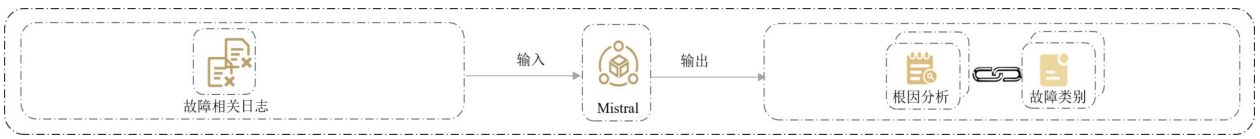


图2　模型推理诊断过程

3.2　预处理日志

根据实际业务场景,故障发生之前可能会有预警日志产生,故障发生之后可能会产生日志风暴. 首先,本研究把收集到的原始日志通过Time字段按照距离故障发生的时间间隔,切分成不同的时间区间,由此得到新日志. 然后,结构化提取新日志信息,将敏感信息的关键字段进行脱敏处理,从而得到脱敏后的新日志,如表1所示. 最后,再利用特征工程构造统计特征,筛选出故障相关日志.

预处理日志时采用TF-IDF的特征工程,将Content字段经过TF-IDF编码,再使用eli5得出每个类别下关键错误单词的贡献程度,权重越高表示区分该类别的贡献越大. 例如,表1展示了在真实的业务场景中,发生故障时运维服务器所产生的日志实例,即每一条故障工单的日志数据. SM代表服务器型号,SN是服务器序列号,EventID是故障案例的标识ID号.

表1 脱敏后的日志样本

SM	SN	Time	EventID	Content
SM70	Server_173	2020-10-09 18:05:03	aac0c39b	Memory CPU0F0_DIMM_Stat | Uncorrectable ECC | Asserted
SM70	Server_173	2020-10-09 18:05:03	aac0c39b	Processor CPU0_Status | Configuration Error | Deasserted
SM70	Server_175	2020-10-09 19:05:03	aac0c39b	System Boot Initiated BIOS_Boot_Up | Initiated by warm reset | Asserted
SM70	Server_175	2020-10-19 19:05:03	aac0c39b	Memory CPU0F0_DIMM_Stat | Correct ECC | Asserted

通过关键词权重和词频占比可得,第二条日志中标记红色部分“Processor”权重分最高,以及第四条日志中的“Correct ECC”也较高,这样便可找到了故障相关的日志,但是如何进一步准确地分类故障,还需借助大模型相关的技术. 也即需将第二条日志归类为“CPU”这一故障类别,第四条归类“Memory”这一故障类别. 但有些是次要或者误报的故障信息,需分辨主次、真伪以找到故障发生时最关键的日志信息,分析后给出故障类别,以便修复故障,这就需要借助大模型的相关技术.

3.3 两阶段思维链提示工程Auto-FSC策略

本研究采用两个阶段思维链指令提示工程CoT-Prompting,主要利用超大规模闭源LLM的推理能力,把诊断日志信息的知识提炼到中等规模的基座模型Mistral中. Mistral通过学习这些相关经验来预测未见过的故障类别. 具体而言,该提示工程通过以下步骤进行:第一步,本文利用超大规模闭源LLM及一个无标签的日志数据集D_t,采用自我指导式的零样本思维链指令提示工程引导超大规模闭源LLM生成相应合理性解释,再得预测类别标签. 第二步,从超大规模闭源LLM生成推理过程样本中选取少量示例,输入到少样本思维链指令提示工程中以优化基座模型.

3.3.1 闭源LLM零样本思维链指令提示工程

(1)自动构造思维链

为了解决专家依据故障预测结果手动人工编写根因分析的问题,本文借助具有强大文本理解能力的超大规模闭源LLM,如Claude,记作C,生成推理分析过程R_t. 这里,本研究使用超大规模闭源LLM作为外部推理引擎,相比于自己从头开始训练一个类似的模型更加经济高效. 首先,本文假设随机采样一部分故障相关的日志,作为无类别标注训练数据集$D_t=\{d_1,d_2,\cdots,d_m\}$. 然后,创建零样本思维链提示工程的分类任务提示词,系统只需要再加入一个特定的指令“Let's think step by step”,指示C通过零样本思维链指令提示方式,驱动助手生成根因分析的推理过程R_t,最后用户接收“Therefore, the answer is”提示,得到最终故障分类答案L_t. 整个优化过程是对零样本思维链指令提示生成三元组(D_t,R_t,L_t)的模型P_c进行指令提示迭代. 此方法无需额外训练数据便可直接应用于推理诊断故障任务中.

对于Zero-Shot-CoT-Prompting提示工程,CoT进一步将推理过程R_t添加到标准提示中(更多详情见2.2.3节). 在R_t中植入C,让其快速接入标准化的API接口并进行推理,每次推理后,将R_t附加到二元组(ξ,D_t)的上下文中. 对于零样本思维链提示工程,主要试图在第一轮对话过程中提高提示工程ξ的质量,这里一轮对话指的是一个输入-输出过程.

(2)生成示例样本池

C遵循Zero-Shot-CoT-Prompt指令,经过两轮任务的对话过程,形成了一个完整的推理链条,其中C在两轮对话中,分别担任推理执行器C_r、提示生成器C_g的角色.

第一轮:C_r经过输入的脱敏日志D_t,推理执行并输出具有解释过程的思维链$R_{v_i}^w$,如式(10)所示.

第二轮:C_g分析日志的推理过程$R_{v_i}^w$,提示生成预测标签$L_{v_i}^w$,并输出故障类别,如式(11)所示.

第一轮用“Let's think step by step”,模型给出日志推理过程,第二轮把推理过程和分类问题一起输入模型,给予“Therefore, the answer is”提示,模型给出结果. Claude通过以上两轮对话预测到无类别标注数据分类结果,如式(12)所示.

$$R_{v_i}^w=C_r(D_t) \tag{10}$$

$$L_{v_i}^w=C_g(R_{v_i}^w) \tag{11}$$

$$S_t\leftarrow S_t\cup\left\{(D_t,R_{v_i}^w,L_{v_i}^w)\right\} \tag{12}$$

其中,在第一轮中逐步推理故障诊断分析的过程,并生成第一轮中的合理性解释. 通过上下文学习第一轮可以为后续第二轮的类别结果提供诊断依据. 这是由于第一轮的输出(推理过程,也即故障诊断信息)和第二轮的输入(评判日志故障类别)有先后因果关系. 此外,零样本思维链指令提示在Prompt中不需要人工构造示例的方式,便可直接生成推理步骤,这是借用生成的CoT来导出分类答案. 由此一来,本研究即可以得到一个高质量的示例池.

3.3.2 中等开源LLM少样本思维链指令微调工程

虽然商业化的超大规模闭源LLM展示了强大的对零样本推理能力,但在使用零样本思维链指令提示设置时,它们在更复杂的任务上仍然表现不佳. 作为ICL特例的少样本学习可以辅助零样本思维链提示来引导模型实现更好的性能. 本文在少样本思维链指令提示工程中添加零样本思维链指令提示生成的示例,可作为后续示例的条件,促使模型可以生成高质量的响应.

由于超大规模闭源LLM在推理时会遇到网络延迟、调用API接口受限、预算成本高及受限于大模型支持的最大长度token等问题. 本文在处理较大日志数据集时并未选用所有示例池样本,而是根据C预测类别结果$L_{v_i}^w$,从超大规模闭源LLM生成示例池S_t中筛选出k个最具有代表性的典型示例作为零样本思维链指令提示工程中的输入示例样本,记作$E_{v_i}^w=\{E_{v_1}^w,E_{v_2}^w,\cdots,E_{v_n}^w\}$,具体见式(13). 由此一来,不需人为构造日志示例,而使用Claude生成示例E. 它囊括了不同数据集w类别下任务v的示例数.

$$E_{vi}^w \leftarrow (S_t,k) \tag{13}$$

由此,借助C得到推理样本示例$E=(D_t,R_{v_i}^w,L_{v_i}^w)$,并通过第一阶段预测分类标签$L$与提取的日志推理过程$R$作为额外诊断信息的任务,视为样本$E$的一部分. 这样一来,本研究将样本示例$E$添加到系统指令少样本思维链指令提示工程(定义见2.2.1节)中,并根据给定的不同任务$(D_s,E_{v_i}^w)$输入,启用ICL引导Mistral迭代优化推理. 这意味着将先验知识引入一个规模中等的基座模型Mistral. 目标是生成测试集D_s对应的推理过程$R_{v_i}^w$以及分类结果$L'^w_{v_i}$,并保存所有生成的测试过程,见式(14). 整个过程通过指令微调来参数化概率基座模型P_m.

$$L_s \leftarrow L_s \cup \left\{(D_s,R'^w_{v_i},L'^w_{v_i})\right\} \tag{14}$$

第一轮:基座模型O_m输入D_s以及C生成的少样本$E_{v_i}^w$进行推理分析后,输出根因分析的推理过程$R_{v_i}^w$,见式(15).

$$R'^w_{v_i}=O_m(D_s,E_{v_i}^w) \tag{15}$$

第二轮:基座模型O_m启用ICL,分析第一轮D_s对应推理过程$R_{v_i}^w$,提示输出故障类别,见式(16).

$$L'^w_{v_i}=O_m(R'^w_{v_i}) \tag{16}$$

通过少样本思维链指令提示工程来优化基座模型Mistral,进行高质推理过程,执行高效分类任务. 综上,自动构造思维链示例的Auto-FSC算法,可提升Mistral推理性能,整体实现流程如算法1所示.

通过随机采样自动构建推理链示例池方法,选择预测不同故障类别的示例进行推理新日志任务. 相比于传统手工设计诊断日志信息,这种自动构建方式不仅提高效率,还避免人工设计带来主观偏差和误差,具有更高效率和可扩展性.

算法1　自动构造推理链的少样本示例的Auto-FSC算法

输入:无类别标注的训练集日志数据$D_t=\{d_1,d_2,\cdots,d_m\}$,测试集日志数据$D_s=\{d_{m+1},d_{m+2},\cdots,d_n\}$,选取的少样本示例数量$k$

输出:测试数据的最终分类结果$L=\{l_{m+1},l_{m+2},\cdots,l_n\}$… …

1. PROCEDUER 两阶段分类(D_t,D_s,k)
2. $S_t \leftarrow \phi$
   /*训练数据的第一阶段结果
3. $E \leftarrow \phi$
   /*少样本示例
4. $L \leftarrow \phi$
   /*测试数据的最终结果
   /*第一阶段:适用闭源模型对训练数据进行分析预测
5. FOR D_t中的每一个d_i DO
6. $r_i \leftarrow$闭源模型分析(d_i)
   /*闭源模型的思维链分析
7. $l_i \leftarrow$闭源模型预测(r_i)
   /*闭源模型的标签生成
8. $S_t \leftarrow S_t \cup \left\{(d_i,r_i,l_i)\right\}$
   /*保存闭源模型对训练数据结果
9. END FOR
   /*从训练数据中提取少样本示例
10. $E \leftarrow$提取示例(S_t,k)
    /*选择k个最具有代表性的示例
    /*第二阶段:使用Mistral对测试数据进行分析预测
11. FOR D_s中的每一个d_i DO
12. $r_j \leftarrow$MISTRAL分析(d_j,E)/*Mistral少样本分析
13. $l'_j \leftarrow$MISTRAL预测(r_i)　/*Mistral标签生成
14. $L \leftarrow L \cup \left\{\left(d_i,r'_j,l'_j\right)\right\}$/*保存Mistral的测试结果
15. END FOR
16. RETURN L
17. END PEOCEDURE

3.4　反馈偏好对齐LLMf-IPO算法

通过CoT-Prompt提示工程进行指令微调(prompt-tuning)基座模型O_m,其性能提升后的模型O'_m并不能完全准确地响应提示. 为了更好生成符合用户意图的模型,本文采用强化学习反馈的直接偏好对齐IPO技术. 首先,本研究选用三个中等规模开源LLM生成推理分析的回复R. 然后再次借助商业超大规模闭源LLM,比如Claude(评估器,记C_e)使用评估指令提示(evolution-prompting)并依据基准真实值(ground-truth)对这些推理回复进行评分排序. 最后,输出三个中等规模开源LLM排序结果,并转化为模型回复的偏好,由此得到了构造的高质量偏好数据集D_t^*,被作为奖励信号用于IPO训练. 整个偏好微调(preference-tuning)过程,D_t^*可与生成三元组(D,R,L)的概率参数模型P'_m进行优化迭代,即可得到一个与用户意图对齐模型O''_m.

3.4.1　推理三大开源模型

假设有类别标注的日志训练集为D_t^*作为偏好数据集,测试集为D_s,候选模型数d^*个,采样次数n,Mistral为目标模型O''_m. 随后,选取上一阶段少样本思维链提示工程中生成的少量k个日志示例$E_{v_i}^w$. 此外,还需选定

辅助三大中等规模开源LLM的开源模型Llama-3.1-8B-Instruct、Gemma-2-9B-it、Qianwen-2-7B-Instrauct分别用O_l、O_g、O_q表示. 紧接着,依据输入的日志内容,添加少量示例到思维链提示工程中,利用示例诱导推理法和标签诱导法这两种方式分别对三个模型进行推理分析,如式(17)和式(20)所示,再依据推理过程得故障类别,如式(18)所示. 由此,可为每个原始日志内容生成多个不同的推理诊断信息,多样化的推理可以构造丰富的偏好数据集. 为了得到准确的分析推理过程,使用了两种方法,一种是使用示例诱导推理法得故障分类标签,具体来说已知示例,未知推理和标签,见式(19). 以Llama3.1为例,通过推理过程$R_{v_i}^w|l$的式(17)和预测标签$L'^w_{v_i}|l$的式(18),示例诱导推理得标签式(19).

$$\begin{cases} R_{v_i}^w|l = O_l\left(D_t^*, E^w\right) \\ R_{v_i}^w|g = O_g\left(D_t^*, E_{v_i}^w\right) \\ R_{v_i}^w|q = O_q\left(D_t^*, E_{v_i}^w\right) \end{cases} \tag{17}$$

$$\begin{cases} L'^w_{v_i}|l = O_l\left(R'^w_{v_i}|l\right) \\ L'^w_{v_i}|g = O_g\left(R'^w_{v_i}|g\right) \\ L'^w_{v_i}|q = O_q\left(R'^w_{v_i}|q\right) \end{cases} \tag{18}$$

$$\begin{cases} L'^w_{v_i}|l = O_l\left(O_l(D_t^*, E_{v_i}^w)\right) \\ L'^w_{v_i}|g = O_g\left(O_g(D_t^*, E_{v_i}^w)\right) \\ L'^w_{v_i}|q = O_q\left(O_q(D_t^*, E_{v_i}^w)\right) \end{cases} \tag{19}$$

另外一种是使用标签诱导法得推理过程,即已知示例和标签,未知推理,见式(20). 仍以Llama3.1为例,通过提示Llama3.1仅基于给定的少量标签$L_{v_i}^w|l$生成合适的推理过程$R_{v_i}^w|l$.

$$\begin{cases} R_{v_i}^w|l = O_l\left(D_t^*, E_{v_i}^w, L_{v_i}^w|l\right) \\ R_{v_i}^w|g = O_g\left(D_t^*, E_{v_i}^w, L_{v_i}^w|g\right) \\ R_{v_i}^w|q = O_q\left(D_t^*, E_{v_i}^w, L_{v_i}^w|q\right) \end{cases} \tag{20}$$

3.4.2 收集二元反馈信号

通过输入评估指令提示工程中的一组指令提示和少量示例,并根据以上两种诱导法,得到现有的三个LLM输出的多样化推理回复. 这样一来,一个Prompt就对应了不同模型的多个回复. 然后Claude基于基准真实值,对这些推理回复打分,从而获得偏好数据的反馈信号. 其次,根据排序结果,得到反馈数据集,其格式为<Prompt,Response>. 目标是通过Claude对三个中等模型的推理回复进行评分,这样超大规模闭源LLM进行评分并二值化为偏好,提供给基座模型一个额外的偏好信号,模拟人类的偏好构造数据集,实现对齐目标模型.

(1)模型回复及采样

假定$D_1^*, D_2^*, \cdots, D_n^*$是一系列Prompt偏好数据$D_t^*$,代表一组不同的日志数据域. 已知第一阶段少样本思维链指令微调(prompt-tuning)目标模型O'_m最大化,应表述为式(21).

$$\Xi_{m'} = \max_{\Xi} \mathop{\mathbb{E}}_{(D,L)\sim D_t^*} \log \Xi\left(L'^w_{v_i}|D_t^*\right) \tag{21}$$

与Few-Shot-Cot类似,通过收集同一个Prompt来自三个不同模型O_l、O_g、O_q不同概率下的输出回复L(详情见3.4.1节),统一表述如式(22).

$$\begin{aligned} L^l &\sim \Xi_l\left(\cdot|D^*\right) \\ L^g &\sim \Xi_g\left(\cdot|D^*\right) \\ L^q &\sim \Xi_q\left(\cdot|D^*\right) \end{aligned} \tag{22}$$

L遍历所有可能的输出,其中$(\cdot|D^*)$是指在数据D^*下未知的回复. 针对同一个Prompt,每个模型多次采集针对该Prompt的各自答案n次. 这是由于对于每个日志内容,同一条Prompt的指令下,回答10次,模型可能生成不同的推理和分类回答,这样能更好覆盖到输出并真正满足用户偏好. 总之,对于每个D^*,先对三大中等规模开源LLM回复进行多次采样,新指令细化为近似不等式$L^* \sim \Xi_*\left(\cdot|D^*, L^*\right)$,最终采样的日志数据集为$D^* = \left\{(D_1.L_1), (D_2.L_2), \cdots, (D_n.L_n)\right\}$.

(2)模型结果排序

本文依靠能力强大的Claude根据不同故障分类答案的好坏,对指令微调后的三大辅助模型的回复进行多次采样,再对其采样结果进行评估排序. 评估衡量标准是少样本思维链提示生成三元组中的内容相关性(context relevance),即对比基准真实值和召回的上下文(contexts)之间相关性.

由于每个Prompt都有三个中等规模开源LLM生成的响应,三大辅助模型会把这些响应给到担任二元分类评估器Claude,记作Ξ_c. 它作为一个裁判对输入的这一对元组(D^*, L^*)打分,判断它们的相似度再进行排序. 计算"奖励"分数如式(23)所示:

$$\Xi_{m''} = \prod_{t=1}^{s} P_{m''}\left(L_t|D, L<t\right) \tag{23}$$

其中,s是序列的长度;$P_{m''}(L_t|D)$是在给定前文D和之前的输出$L<t$的条件下,目标模型O''_m生成第t个标记L_t的概率. Claude遵循评估指令提示,并对这些响应的分数进行评级排序ς^*,如式(24)所示:

$$\begin{aligned} \varsigma^l &\sim \Xi_c\left(\cdot|D, L^l\right) \\ \varsigma^g &\sim \Xi_c\left(\cdot|D, L^g\right) \\ \varsigma^q &\sim \Xi_c\left(\cdot|D, L^q\right) \end{aligned} \tag{24}$$

由此,就得到一些列Prompt对应三个模型不同回复的排序次序.

(3)偏好数据对构建

假设用户输入日志后,不同模型针对同一日志内容两种回复倾向性为Chosen或Rejected. 通过偏好对齐输出偏好数据集形如<Chosen,Rejected>. 其中,Chosen和Rejected分别用L_g、L_b表示,即得分且排序最高作为"Chosen"响应并保存为L_g,随机选择其余两个中的一个作为"Rejected"响应保存到L_b.

由于偏好数据集需要分布一致,不能只纠正错的日志诊断过程,也需保留原本对的推理,只是为了避免模型遗忘正确推理趋势. 因此,针对模型的输出采样n次,对积极响应L_g和消极响应L_b做笛卡尔积D:$\left\{(D,L_g)\times(D,L_b)\right\}$,以此增强偏好数据样本,使输出真正满足用户偏好. 最终得到形如三元组(D,L_g,L_b)的反馈数据集,随即加入到标注的数据集D_t^*,最终构建二元偏好数据集.

3.4.3　身份偏好优化(IPO)对齐训练

假设经指令微调后的基座模型为O_m'和一组对模型生成结果的偏好数据集D_t^*,偏好对即偏好指令提示的训练集D生成答案对$(L_1,L_2)\sim\Xi_*(L|D)$,Claude标注出L_g相对L_b是更好的答案. 通过生成的偏好数据集D_t^*被用来训练一个奖励模型γ,再进行IPO对齐训练,目标是先训练一个偏好模型$\Xi_{m'}$,从中找到训练模型的偏好策略$\Xi_{m'}(\cdot|D)$来训练一个新的最优策略模型$\Xi_*(L|D)$,如式(25)所示,使其生成的输出更符合人类的意图.

$$\Xi_*\left(L|D\right)=\frac{1}{Z(D)}\Xi_{m'}\left(L|D\right)\exp\left(\frac{\gamma(D,L)}{\tau}\right)\tag{25}$$

其中,配分函数$Z(D)$见式(26),确保$\Xi_{m'}\left(L|D\right)$为一个有效的概率分布.

$$Z(D)=\sum_L\Xi_{m'}\left(D|L\right)\exp\left(\frac{\gamma(D,L)}{\tau}\right)\tag{26}$$

此外,$\gamma(D,L)$是对输入D和输出L的奖励函数,反映了模型预测人类对于给定数据对(D,L),输入输出偏好程度或质量. 正则化参数τ是一个权衡因子,控制策略偏离偏好策略程度参数,见式(27).

$$\gamma(D,L)=\tau\log Z(D)+\tau\frac{\Xi_*(D|L)}{\Xi_{m'}(D|L)}\tag{27}$$

假定从以上分布中采样出来一个数据集为$\mathcal{D}=\{D^i,L_g^i,L_b^i\}_{i=1}^N$. 在偏好模型中,首选$L_g$而非$L_b$的概率再进一步参数微调模型$O_m'$. 而这个偏好模型是通过利用迭代的策略模型的奖励函数$\gamma(D,L)$来确定,即根据模型的最优策略推导出最优的奖励函数. IPO直接依据静态数据去优化偏好模型,接着把奖励函数插入到偏好模型中,这样就得到式(27)的隐式化奖励函数. 本文利用从隐式奖励函数到最优策略的解析映射. 但是高度非线性化且极值无限大的$\Psi(q)$会导致过拟合情况(详情见2.2.4节). 此处把$\Psi(q)$替换成有界函数,即通过符合要求的恒等变换就得到IPO优化的目标函数,如式(28)所示.

$$\Xi_{m''}=\max_{\Xi_*}\mathbb{E}_{D\sim\rho}\left[\rho^*(L\succ L'|D)\right]-\tau\mathbb{D}_{\mathrm{KL}}\left(\Xi_*\|\Xi_{m'}\right)\tag{28}$$

其中,$\mathbb{D}_{\mathrm{KL}}$为散度约束函数,$\Xi_{m'}$是原始Mistral模型经过指令微调后的偏好策略. 答案整合阶段最终$\Xi_{m''}$会被提示问题和更正的理由,以生成一个整合答案,这有望形成一个更准确答案.

综上,基于IPO这个目标函数,从模型的指令微调开始,IPO对齐方法训练时需要迭代每个三元组(D,L_g,L_b). 根据式(28),我们可推导出IPO损失函数,如式(29)所示:

$$\mathcal{L}_{\mathrm{IPO}}(\Xi_*;\Xi_{m'})=-\mathbb{E}_{(D^*,L_g,L_b)\sim D_t^*}\left[\left(\log\frac{\Xi_*(L_g|D^*)\Xi_{m'}(L_b|D^*)}{\Xi_*(L_b|D^*)\Xi_{m'}(L_g|D^*)}-\frac{\tau^{-1}}{2}\right)^2\right]\tag{29}$$

IPO是在DPO损失函数上添加一个正则项,总是通过控制对数似然比之间的差距来将其解正则化$\Xi_{m'}$,从而避免对偏好数据集的过拟合,可使模型加快收敛,增强泛化能力. 也即首先根据CoT-Prompting指令提示计算(D,L_g)和(D,L_b)的概率;然后,根据参数微调的IPO模型计算(D,L_g)和(D,L_b)的概率;最后,优化目标函数做反向传播以更新最终的目标模型.

模型反馈的偏好对齐LLMf-IPO实现过程,如算法2所示. 该算法利用充当评估器Claude,对三个辅助模型O_l、O_g、O_q推理结果进行排序,由此生成高质量的偏好数据集(D,L_g,L_b). 随即进行IPO训练,隐式的奖励会被一起分配给Prompt和响应,以纠正为中心的进化策略有效地覆盖了问题集,识别错误步骤、解释错误原因、纠正错误并生成最终答案. 采用三个中等规模开源LLM进行推理任务,LLMf-IPO算法生成纠正数据,提高了CoT单轮微调性能. 此外,对偏好数据集进行微调以对齐Mistral,这样有助于纠正任何事实上错误并防止错误广泛性传播.

3.5　模型训练、推理诊断

故障诊断其目的是确定故障的根本原因,通常被形式化为一个分类任务. 本文的任务主要分为两个阶段,分别是部署前的训练优化阶段和部署后的推理诊断阶段. 其中,训练优化阶段流程是LogCoT经过日志预处理后,提取关键故障特征,并通过大模型混合调优训练,不断迭代学习不同故障模式下日志案例的类别特征,识别日志中的故障模式. 此外,推理诊断阶段是部署训练优化后的基座模型Mistral,通过大模型上下文学习积累的历史数据进行推理分析未标注的日志,能快速识别并响应故障案例,并直接输出多分类结果,最终可确定故障根因.

算法2 模型反馈的偏好对齐LLMf-IPO机制

输入: 无类别标注训练集D_t,有类别标注偏好训练集D_t^*,候选模型数d^*,采样次数n,目标模型O_m'',选定少量k个示例E_{vi}^w

输出: 优化后的对齐模型O_m'

1. 初始化一个空集合S用于存储采样的响应
2. FOR i=1 to k DO
   /*遍历每个候选模型进行
3. FOR j=1 to n DO
   /*遍历每个采样
4. 样本(D,L) ←在数据集D_t使用k个示例对模型i推理
5. 将样本(D,L)添加到空集合S
6. END FOR
7. END FOR
8. 初始化一个空集合L用于存储标签数据
9. FOR ALL $(D,L)\in S$DO
10. 根据L的分类指派标签$l\in${good, bad}
11. 将(D,L,l)添加到L'
12. END FOR
13. 初始化一个空的偏好数据集D_t^*
14. FOR ALLDin unique queries in L'DO
   /*对于L'中的每一个唯一查询D
15. 提取数据对(D,L_g) where l= good and (D,L_b) where l=bad
16. From Cartesian product of good and bad responses for D:$\{(D,L_g)\times(D,L_b)\}$
   /*形成积极响应和消极响应的笛卡尔积D
17. 将所有结果对(D,L_g,L_b)添加到D_t^*
18. END FOR
19. 从预训练的检查点初始化目标对齐模型O_m''
20. REPEAT
21. Optimize the target model O_m'' on the preference dataset D_t^* using the IPO training objective in Eq.29
   /*使用式(29) IPO训练目标,优化偏好数据集D_t^*上的目标对齐模型O_m''
22. UNTIL convergence
23. RETURN Optimized target model O_m''
   /*返回优化后的目标对齐模型O_m''

4 实验结果与分析

4.1 实验设置

4.1.1 数据集

本研究在真实业务场景中收集的两个数据集上进行实验,分别来自阿里巴巴运营商提供的公开可用的服务器数据集①(用D_1表示),以及世界顶级互联网服务某提供商未公开私有的交换机数据集(用D_2表示),均为生产环境中半个月内发生故障时收集的真实日志数据集.

表2列出了这两大数据集的详细信息,分别包含日志序列、故障类别、故障案例、数据集的分布情况等详情. 本研究使用日志序列作为样本,将其输入模型. 在以下实验中,本研究选取全部数据集故障案例的2%作为无类别标注的训练数据,全部数据集故障案例的5%作为有类别标注的训练集,也即偏好训练集,其余剩下93%为测试数据. 其中,训练集包括无类别的伪标签和有类别的真实标签,故障案例数量是训练集与测试集之和.

4.1.2 实验环境设置

本研究在Ubuntu 24.04 LTS服务器上进行实验,该服务器配备2个16C/32T Intel(R) Xeon(R) Gold 5416s CPUs @4 400 MHz,8个NVIDIA(R) RTX(R) A6 000-48 GB GPUs. 实现LogCoT所用的Python 3.10.14,Cuda 12.1,Openai 1.40.6,vllm 0.5.0.post1. 在LogCoT通用框架下,4.3节、4.4节、4.5节中均统一选用Claude 3.5 Sonnet超大规模闭源LLM实验.

4.1.3 参数设置

实验设置默认情况如下:每个设备训练批量大小Batch_size=1,学习率Learning_rate=5.0e-6,Epochs=3,预热率为Warmup_ratio=0.1,梯度累积步长为Gradient_accumulation_steps=8,以及推理框架vLLM(virtual large language model)默认参数repetition penalty:float=1.0,temperature:float=1.0,top_p:float=1.0.

4.1.4 评价指标

为了公平对比实验,针对超大规模闭源LLM产生的实例,本实验均选择3ε-Shot-CoT进行少样本思维链提示工程的实验. 故障诊断可看作一个多分类的问题,为衡量LogCoT的有效性,选择对每个类别的F1取平均值的宏平均F1(Macro-F1)、加权平均F1(Weighted-F1)、多分类微观平均F1(Micro-F1/Accuracy)作为评估指标.

4.2 超大规模闭源LLM性能对比

4.2.1 Zero-Shot-CoT推理性能

本实验采用零样本链式思维提示工程,预先设计Prompt提示模板的提示结构,利用大语言模型对自然语言的理解让模型进行自主思维链分析,来提升模型输出故障分类的结果. 实验通过对故障场景分类任务描述、输入日志格式,逐步引导模型进行多步骤的推理,使其能够像人类一样逐步思考问题. 为了评估超大

表2 日志数据集和故障案例数据集的统计情况

数据集	日志条目	故障类别	故障案例	无类别标注训练集的案例	有类别标注训练集的案例	训练集的案例	测试集的案例
*D*1	35 781	3	1 717	9	42	51	1 666
*D*2	59 592	9	1 239	27	41	68	1 171

①https://tianchi.aliyun.com/competition/entrance/531947/information

规模闭源LLM在零样本链式思维工程下推理任务的性能,本实验选用Claude 3.5 Sonnet、GPT-4o两大超大闭源模型. 它利用零样本思维链自我生成推理理由,并从LLM中提炼知识迭代优化.

经提示词指示模型执行故障分类任务后,输出预期分类结果. 从表3可看出,在文本分类任务中,输入不同的故障案例后,使用超大规模闭源LLM进行思维链推理,预测故障分类最高准确率高达97%. 具体来说,在两个数据集上,发现数据集D_2在Claude 3.5 Sonnet、GPT-4o下的性能可平稳达到95%以上,然而在数据集D_1的性能差距较大,两个超大闭源LLM相差10个百分点左右. 通过实验对比分析,同等样本量下,Claude模型更擅长在零样本下进行思维链分析. 具体表现在D_1、D_2数据集上,Claude 3.5 Sonnet推理性能的Accuracy分别为88.24%、97.78%,均比GPT-4o高10.25个百分点和2.66个百分点. 实验结果可得出闭源超大语言模型Claude 3.5 Sonnet在零样本链式思维推理任务中表现出色,且推理每个故障案例平均耗时最少,特别是在使用链式思维提示时,模型的推理能力较强. 这表明多步骤的推理链可提高模型性能.

表3　超大规模闭源LLM的Zero-Shot-CoT推理性能

闭源模型	数据集	样本量	Accuracy/%	Macro-F1/%	Weighted-F1/%	每案例平均时间/s
Claude 3.5 Sonnet	D_1	51	88.24	86.41	87.92	9.91
	D_2	68	97.78	97.76	97.76	18.33
GPT-4o	D_1	51	77.99	74.34	77.57	26.15
	D_2	68	95.12	94.63	95.46	30.00

4.2.2　Few-Shot-CoT推理性能

在逻辑推理任务中,我们通过利用商业化的超大规模闭源LLM推理能力,在提示工程指示模型执行故障分类任务时,结合少样本链式思维日志样本的示例,并以高效指令优化开源中等的模型Mistral-7b. Mistral-7b通过ICL学习超大规模闭源LLM预测的样本示例,涉及逐步推理的两轮对话,生成更具解释性的输出,即对日志分类结果的诊断信息过程,可视为故障的根因分析,反映模型对问题逐步思考过程.

为了评估中等规模开源LLM的上下文学习能力,即Mistral借助超大规模闭源LLM预测的样本示例(含有推理理由和伪标签),在少样本思维链推理任务的性能,本节实验仍选用Claude 3.5 Sonnet、GPT-4o闭源模型. 它们具备丰富的知识和上下文理解能力,能够捕捉更复杂的语言模式和结构,用于评估生成的文本或对话的质量. 具体来说,将超大规模闭源LLM推理无标注的数据作为训练集D_t,按其预测故障类别L进行筛选示例. 一方面,我们使用Zero-Shot-CoT预测后生成的范例,假定选取k=2ε-Shot-CoT作为每个测试样本的少量示例,用于评估所选示例数量对基础模型Mistral稳定性能的影响. 实验过程中,本研究分别依次逐渐增加示例数量,如ε-Shot-CoT、2ε-Shot-CoT、3ε-Shot-CoT等来验证能否提高推理准确率. 如表4所示,经测试发现使用最少2ε-Shot-CoT示例,已使Mistral模型性能达到平稳. 此外根据数据集分布,为更准确对比两个闭源模型引导Mistral推理性能,实现模型性能最大化,选取k=3ε-Shot-CoT.

经过超大规模闭源LLM少样本思维链,得到预测故障类别的范例样本,以Few-Shot-CoT引导基座模型Mistral进行推理. 本实验发现,Mistral原始推理性能的无思维链的零样本性能较一般,初始性能的Accuracy为70%左右. 然而Mistral通过借助Claude 3.5 Sonnet、GPT-4o的少样本链式思维提示,其性能明显优于零样本提示. 这表明有范例样本的多步骤推理,能够显著提高模型的推理能力. 在文本分类任务数据集D_1中,采用Claude 3.5 Sonnet、GPT-4o少样本链式思维提示将Accuracy分别从68.01%、67.77%提高到86.73%、86.19%,Macro-F1分别从63.01%、63.14%提高到82.82%、81.72%,以及Weighted-F1分别从69.40%、69.19%提升到86.58%、86.06%,即三大性能指标皆提升了20个百分点左右. 尤其是在D_2数据集上,Mistral在文本逻辑推理分类任务中表现更出色,特别是在使用含有链式思维提示的少量示例时,三个性能指标上最终均为90%左右. 这是因为少样本示例为模型提供了先验知识,减少了不确定性,增强了泛化能力,从而显著提高了模型的推理性能. 此外,经实验统计故障案例耗时后,Claude 3.5 Sonnet在数据集上每个故障案例平均推理时间为6.88 s、6.49 s,然而在GPT-4o在D_1、D_2数据集上分别为6.09 s、5.62 s. 通过实验发现,Claude 3.5 Sonnet与GPT-4o性能几乎持平,但GPT-4o的性能稍微略高出1~2个百分点左右,并且GPT-4o响应速度更快,在少样本下更擅长思维链分析,在保证模型性能的基础上,还可以迅速响应给予回复并输出故障的分类结果.

4.2.3　偏好对齐

由于指令微调后的基座模型Mistral存在推理错误的分类结果,需进一步对Mistral进行纠错对齐. 为了更好使生成的回应符合用户意图,本实验针对模型推理的回应进行采样,作为评估器的超大规模闭源LLM对推理质量排序,形成偏好数据对<Chosen, Rejected>,再使用强化学习的反馈身份偏好IPO训练对齐以微调Mistral,实现偏好微调中等规模开源LLM对齐用户意

表4 不同超大规模闭源LLM使用不同shot的Mistral推理性能效果对比

闭源模型	数据集	方法	Accuracy/%	Macro-F1/%	Weighted-F1/%	每案例平均时间/s
Claude 3.5 Sonnet	D_1	0-Shot	68.01	63.01	69.40	0.40
		1*3-Shot-CoT	85.89	82.22	85.90	7.19
		2*3-Shot-CoT	86.67	82.47	86.58	6.86
		3*3-Shot-CoT	85.53	80.80	85.35	6.79
		4*3-Shot-CoT	86.73	82.82	86.47	6.77
		5*3-Shot-CoT	86.43	82.64	86.29	6.77
	D_2	0-Shot	71.22	66.57	65.83	0.41
		1*9-Shot-CoT	84.37	83.52	82.74	4.25
		2*9-Shot-CoT	90.18	89.94	89.87	7.22
		3*9-Shot-CoT	90.44	91.25	90.19	7.06
		4*9-Shot-CoT	91.37	86.34	90.96	6.96
		5*9-Shot-CoT	91.72	86.96	91.57	6.94
GPT-4o	D_1	0-Shot	67.77	63.14	69.19	0.51
		1*3-Shot-CoT	85.19	79.91	85.01	5.89
		2*3-Shot-CoT	85.45	81.72	85.33	6.09
		3*3-Shot-CoT	85.59	80.99	85.39	6.13
		4*3-Shot-CoT	85.07	81.38	84.88	6.11
		5*-Shot-CoT	86.19	81.23	86.06	6.23
	D_2	0-Shot	73.36	68.27	67.82	0.53
		1*9-Shot-CoT	91.33	91.06	91.34	5.53
		2*9-Shot-CoT	92.39	91.85	92.41	5.61
		3*9-Shot-CoT	92.74	92.35	92.78	5.62
		4*9-Shot-CoT	92.99	92.51	93.03	5.72
		5*9-Shot-CoT	92.41	92.12	92.45	5.63

图. 为了衡量作为评估器的超大规模闭源LLM对输出推理回应质量排序,评估生成模型在文本对齐任务中的性能效果. 实验中仍选用经过大规模语料库预训练的Claude 3.5 Sonnet、GPT-4o的超大规模闭源LLM.

经过实验分析发现,通过构建高质量的偏好数据对后的IPO对齐训练,其模型的性能显著提升. 从表5可以看出Claude 3.5 Sonnet与GPT-4o性能几乎持平,但GPT-4o稍微略高一点. 其中,作为评估器的Claude 3.5 Sonnet提供了更可靠的评分和排序,确保了偏好数据集的高质量. 它在D_1、D_2数据集上经偏好微调的IPO训练后,Accuracy分别为88.06%、98.04%,另外,GPT-4o在两个数据集上Accuracy分别达到了87.09%、98.46%. 此外,对比不同中等规模闭源LLM实验耗时,在D_1数据集上,Claude 3.5 Sonnet、GPT-4o的每个故障案例平均推理时间分别为1.21 s、1.20 s,而在D_2数据集上,分别为1.35 s、1.19 s,可以看出超大规模闭源LLM作为评估器,GPT-4o除了可以提高模型的准确率,还可以及时获取故障分类的回复,响应速度更快.

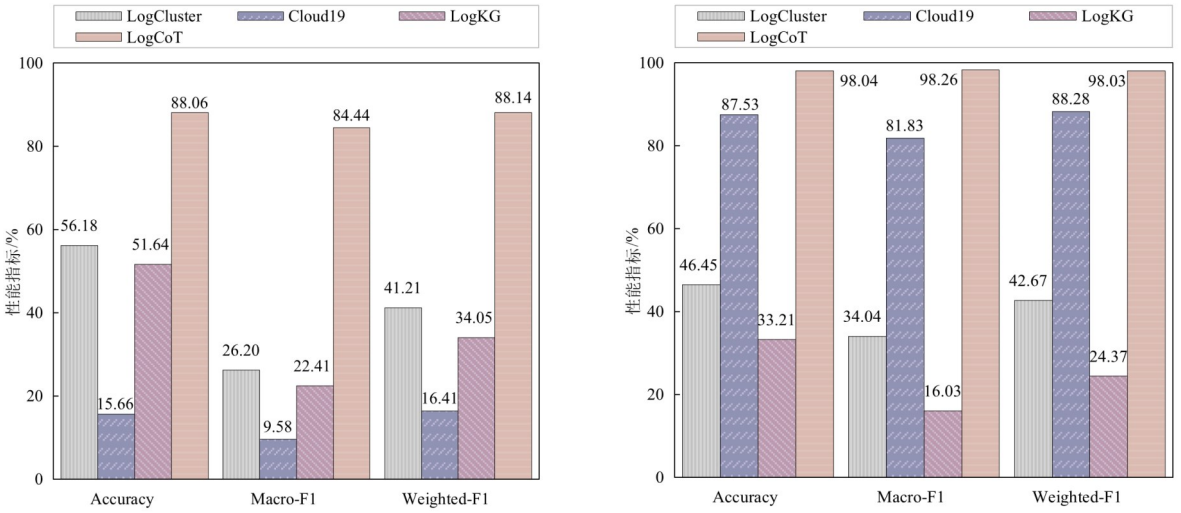
表5 借助闭源模型的Mistral偏好对齐性能对比

闭源模型	数据集	有标注训练集的案例	方法	Accuracy/%	Macro-F1/%	Weighted-F1/%	每案例平均时间/s
Claude 3.5 Sonnet	D_1	42	IPO	88.06	84.44	88.14	1.21
	D_2	41	IPO	98.04	98.26	98.03	1.35
GPT-4o	D_1	42	IPO	87.09	84.06	87.47	1.20
	D_2	41	IPO	98.46	98.07	98.47	1.19

4.3 基线模型对比

为有效比较LogCoT与现有方法,本研究选取了目前深度学习领域内最先进的三种故障诊断方法作为基准方法,分别是LogCluster、Cloud19、LogKG. 具体来说,LogCoT结合了指令优化和偏好微调,选用的超大闭源LLM均为性能平稳的Claude 3.5 Sonnet. LogCluster使用层次聚类模型来实现,将其距离阈值设置为0.5. 对于Cloud19,本实验选择随机森林作为分类模型,并将森林中树的数量设置为100. LogKG中的FOLR模型阈值设置为0.4,OPTICS聚类算法中每个聚类的最小样本数设置为3.

本节实验中,在D_1、D_2两个数据集上将LogCoT与三种基准方法进行对比,以评估方法的有效性. 其中,本实验选择OPTICS[39]作为聚类算法. 如图3所示,LogCoT在两大数据集上的性能指标均优于其他基准方法. LogCoT在D_2数据集上的Accuracy、Macro-F1、Weighted-F1三个性能均表现突出,其结果分别为98.04%、98.26%、98.03%;然而,在D_1数据集上的三个性能指标分别为88.06%、84.44%、88.14%. 这是由于LogCoT使用Prompt-Tuning和Preference-Tuning混合调优的策略,对日志数据集分类错误进行了再次优化,所以在一定程度上提升了准确性. 此外,在D_2数据集上的表现性能超越D_1,是因为D_1存在数据分布不均衡,尤其第三类占比少且存在多样化的故障表征,加剧诊断难度.



(a) D_1　　(b) D_2

图3　四种方法在两大数据集上整体性能比较

更深入对比这四种方法,经实验发现LogCoT、LogCluster、LogKG在数据集D_1表现比Cloud19更优. 这是因为此三种方法均利用特征提取从日志中提取关键特征(如事件类型、时间戳等)进行聚类,来更深层次理解日志语义. 然而Cloud19不能对日志中包含的多个字段进行彻底地理解. 此外LogCoT、Cloud19在D_2较D_1数据集上表现效果更好. 这是由于数据集D_2日志数量多且故障种类多,表现出日志模式简单,每类表征单一化. 由于LogCoT、Cloud19均适用于处理海量易理解分布式系统日志,很容易将故障案例分类到正确的类别中. 从整体看,LogCoT在三个指标表现均稳定,适合处理海量且复杂分布不均衡分布式系统日志.

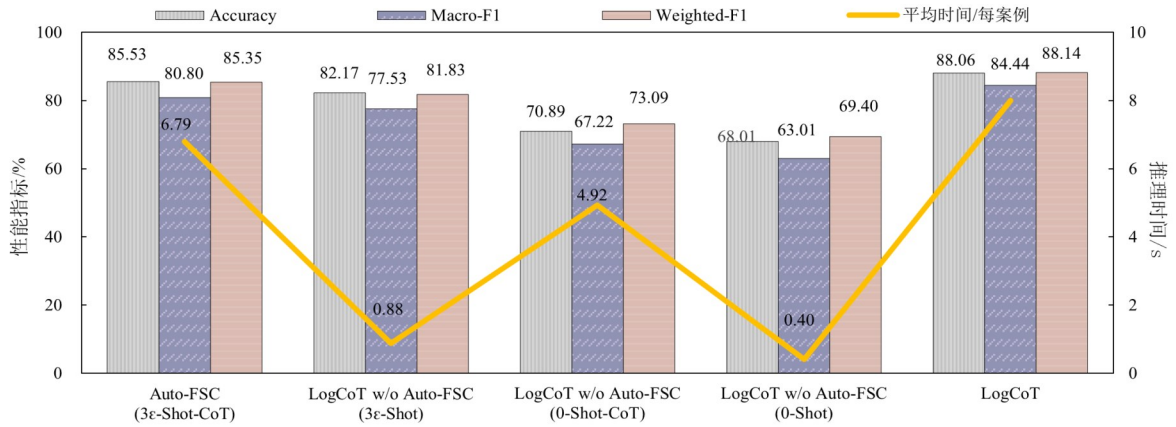
4.4　消融实验

本研究在两个数据集上进行消融实验,以评估LogCoT中两个必备模型:Auto-FSC和LLMf-IPO(以下简称IPO)的有效性,分别如图4(*a*)和图4(*b*)所示.

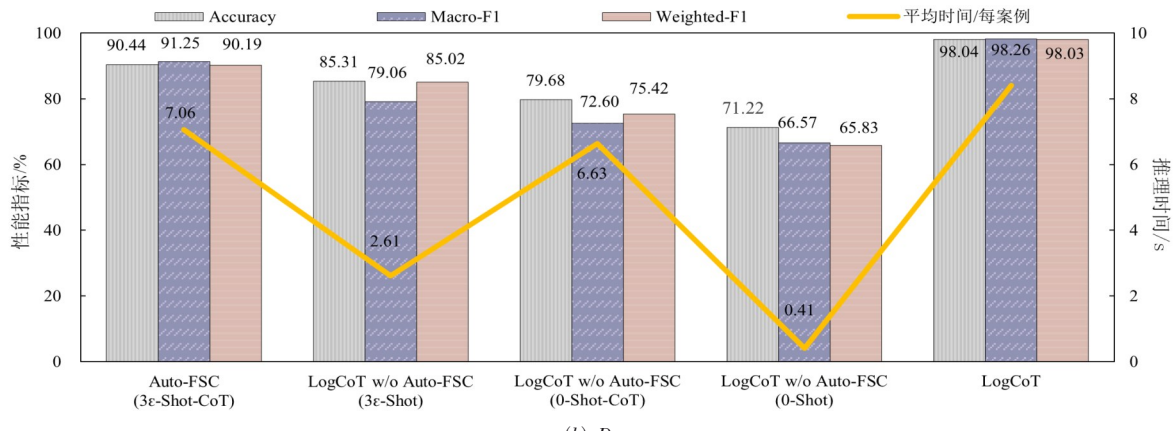
4.4.1　Auto-FSC思维链示例的有效性

Auto-FSC利用超大规模闭源LLM得到推理链,进而利用含有根因分析的示例样本池来推理其余案例,该策略得到更多、更准确、更有效的日志数据诊断信息及分类结果. 为评估该策略有效性,本实验设计以下消融实验:去除了Auto-FSC. 从LogCoT中移除Auto-FSC并分别使用0-Shot-CoT、0-Shot、3ε-Shot-CoT来代替Auto-FSC. 具体来说,假设每个数据集的故障类别为ε,对D_1、D_2分别取3ε样本数. 此处选3ε是通过表4推理对比实验发现3ε-Shot示例已达到基线模型性能平稳且实现最大指标结果. 现有3种方法推理策略,分别为未含有推理思维链的样本实例(3ε-Shot),含有推理思维链的零样本实例(0-Shot-CoT),未含有推理思维链的零样本实例(0-Shot). 这些现有方法的性能在图4依次表示为"LogCoT w/o Auto-FSC"的3ε-Shot、0-Shot-CoT、0-Shot均替代Auto-FSC. 从图4(*a*)和图4(*b*)中可以观察到Auto-FSC确实提高了LogCoT的性能. 如未含Auto-FSC的LogCoT在D_1、D_2两个数据集上的Accuracy、Macro-F1、Weighted-F1得分均低于Auto-FSC. 主要是由于Auto-FSC是有少样本(Few-Shot)和思维链(CoT)两部分组成的,因此为评估这一策略有效性,从两个方面分析并设计了以下消融实验,分别移除少样本和思维链.

一方面,当使用没有参考示例的0-Shot、0-Shot-CoT策略时,也即消融Auto-FSC中的少样本模块,零示例设置下与其他基线的性能比较,均低于提供示例的3ε-Shot、3ε-Shot-CoT的指令优化策略. 证实了基座模型Mistral使用Few-Shot-CoT的示例有效性. 这是由于LogCoT从有关故障案例的相关日志中,通过Auto-FSC筛选出多样性的示例来进行推理分析并通过其他类型的提示来赋予相同的性能改进. 添加



(a) D_1



(b) D_2

图4 两大数据集消融 Auto-FSC 时 LogCoT 的有效性

的这几个示例作为少样本提示中的示例,其另一个潜在好处是在训练期间提供一些参考示例,可以指导模型更好地访问获得的相关知识. 这样一来,它不仅保留了日志完整的语义信息,而且在较小的粒度下挖掘不同故障类别的日志间的关系. 图4可看出,在两大数据集消融Auto-FSC时LogCoT的有效性,及使用不同方法替换Auto-FSC的效果.

另外一方面,当使用无推理链的0-Shot、3ε-Shot策略时,也即消融Auto-FSC中的CoT模块,性能指标得分均低于含推理链分析0-Shot-CoT、3ε-Shot-CoT指令优化策略. 这证实超大规模闭源LLM的Zero-Shot-CoT推理链的有效性.

通过实验观察发现,0-Shot-CoT相对0-Shot方法而言,效果会稍微好一点. 这是由于0-Shot使用标准的提示工程,模型在给出答案之前被提示只输出故障类别结果,在没有自然语言推理步骤的情况下诊断故障类别,这样以来直接让闭源大模型生成答案的准确率比较低. 然而,0-Shot-CoT不需要人工构造并分析推理阶段,而是通过CoT-Prompting指令提示方式,更有可能按照逻辑操作推导答案. 通过使用CoT(0-Shot-CoT、3ε-Shot-CoT)策略,由零样本继续加大到3ε个示例,在D_1、D_2两个数据集上模型的准确率分别由68.01%、71.22%进一步提高到88.06%、98.04%,说明了添加推理过程的思维链可更有效地提升模型性能.

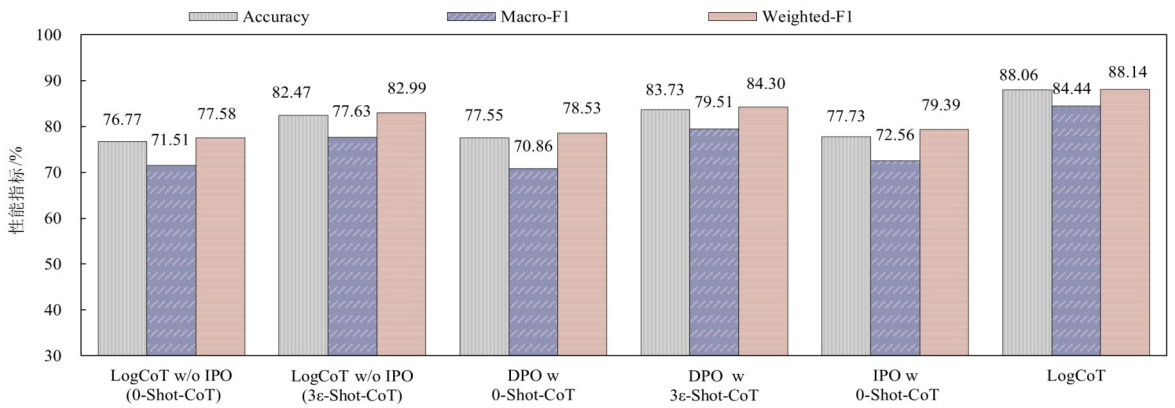
此外,本研究使用大语言模型高速推理vLLM框架,基于3ε-Shot-CoT在两个数据集D_1、D_2情况下,观察右侧纵坐标发现每个故障案例平均推理时间分别为6.79 s、7.06 s. 虽然Few-Shot-CoT的平均推理过程会更久些,但已满足实际业务中诊断故障平均用时需求,还能更好关联日志上下文的内容提示LLMs生成推理链的分析过程. 由此一来,针对故障的日志,LLMs会更好地进行根因分析得出该故障类别,从而使得LogCoT在复杂任务推理上能带来巨大的效果提升. 整体来说,超大规模闭源LLM使用CoT示例和适当少量Few-Shot提示,在复杂推理任务上能提升模型能力.

4.4.2 LLMf-IPO偏好对齐的有效性

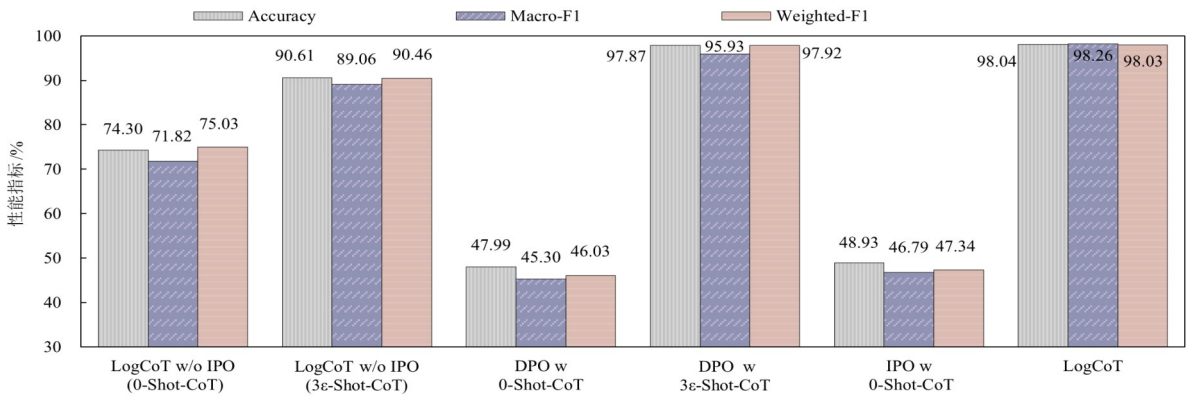
LLMf-IPO通过收集多个LLMs二元反馈信号,然后利用超大规模闭源LLM进行评分并二值化为偏好数据对进行IPO训练. 该策略可以及时提供反馈,并实现纠正说服力极强错误及“创造性”无关响应的推理过程,无须人工即可符合用户意图偏好对齐基座模型. 模型

在进行IPO训练之前,需得到三个辅助开源中等模型Llama、Gemma、Qianwen的反馈信号. 为公平起见,实验统一设置参数β=0.1、learning_rate=5.0×10^{-6}. 以下实验均默认在CoT策略下,这是由于IPO需要微调推理链这一单轮对话. 为了评估该策略的有效性,本研究基于LogCoT使用3ε-Shot-CoT情况下设计以下消融实验:去除了LLMf-IPO,具体表现为以下两个方面.

一方面,LogCoT在D_2数据集上消融IPO后,依次为图5横坐标显示的"LogCoT w/o IPO (3ε-Shot-CoT)、LogCoT w/o IPO (0-Shot-CoT)". 正如图5(b)所示,LogCoT准确性由98.04%、97.87%分别降低为90.61%、74.30%. 同理,在数据集D_1上依次消融IPO后,LogCoT准确性由88.06%、77.73%分别降低到82.47%、76.77%. 这是由于在实际生成环境中,诊断故障过程中都存在大量的噪声日志,容易导致分类不准确,LLMf-IPO通过开源模型反馈的信号以微调并纠正被噪声干扰日志的推理过程,快速实现了LLMs的IPO对齐微调,进一步提升对日志根因分析的准确性.



(a) D_1



(b) D_2

图5　在两大数据集消融LLMf-IPO时LogCoT的有效性及替换IPO的效果对比

另一方面,为了更进一步证实LLMf-IPO的有效性,将IPO与DPO两者进行对比,本实验从LogCoT中移除IPO,并使用DPO来代替. 如图5(b)所示,经观察发现在D_2数据集上,IPO和DPO需要在适量示例前提下,其性能才会有所提升. 模型在3ε-Shot-CoT示例数量降低到不提供任何示例情况下,依次训练IPO、DPO,模型的准确率分别由98.04%、97.87%降低到48.93%、47.99%. 即便如此,IPO的性能仍稍微高于DPO. 最后观察在D_1数据集上,其性能亦是这种趋势. 说明在没有示例下,模型疯狂的胡乱输出导致IPO、DPO性能持续降低. 将IPO与DPO对比,无论是3ε-Shot-CoT还是0-Shot-CoT下,IPO三个性能指标均高于DPO. 一是由于DPO容易存在过拟合问题,然而IPO在DPO损失基础上添加了一个正则,能够让模型快速收敛,总体上泛化性更好;二是由于LLMf-IPO通过采用笛卡尔积方式进行多次随机采样,而不仅仅选择得分最低的响应,这样就可完全覆盖多样性的推理过程,从而得到高质量偏好集且以更高效方式训练IPO. 此外,相比于DPO,IPO可通过简单的分类目标直接优化满足偏好对齐策略. 具体而言,IPO的本质在于增加了被首选的响应相对不被首选响应的对数概率,包含了每个动态的示例的重要性权

重,可防止其概率比让模型的能力退化,并快速实现LLMs的对齐微调. 这些实验结果表明,将超大规模闭源LLM评估能力与IPO训练结合,其偏好微调后,模型能更准确地生成符合用户意图的内容,即LLM通过IPO训练,从错误中学习来提高纠错潜力,偏好微调可更好将中等规模开源LLM对齐用户意图.

基于以上两方面消融实验分析,从图4和图5可看出,在两个数据集上移除Auto-FSC、LLMf-IPO以及使用不同方法替换这两模块时降低了LogCoT的有效性. 实验结果表明,LogCoT策略能有效地改善性能并具备更高的稳健性.

4.5 泛化能力

针对不提供训练示例的0-Shot策略,日志内容对应的类别是未知的,可认为0-Shot的每个类别都相当于是新类别,将识别不了的类别作为新类别,因此0-Shot策略处理新故障的能力很低. 本实验采用了Few-Shot-CoT策略,借助推理器的Claude 3.5 Sonnet闭源超大模型生成的预测的范例示例,这里仍选取使模型性能最大化的示例数量3ε-Shot-CoT,并在两个数据集上的Accuracy分别达到了85.53%、90.44%.

为评估模型在未见过的新数据上的表现及应对新故障的推理能力,在D_1上共3种故障类别,实验随机选取第1类“Processor CPU Caterr”或者第3类“Other”作为新类别. 此外,在D_2上共9种故障类别,选取第5类“CRC Error”或第7类“BFD Down”为新类别. 随后,评估模型在未见过的新数据上的表现,将模型处理新类别故障的性能分别于Zero-Shot与包含所有类别示例的Few-Shot-CoT策略进行对比分析. 如表6所示,采用剔除某类故障类别后的Few-Shot-CoT策略,在数据集D_1上剔除第1类“Processor CPU Caterr”和第3类“Other”故障类别,也即在D_1训练数据集上剔除第1类和第3类故障类别,此时提供示例中只含有2类,而不是所有类别的3类,测试集包含所有3类故障类别. 这样相比于Zero-Shot策略的68.01%,准确率分别提升到82.23%、85.11%,但相较于提供了所有故障类别范例的Few-Shot-CoT策略,性能分别降低了3.30个百分点和0.42个百分点. 为更有效说明模型泛化性,接着在D_2数据集分别剔除第5类“CRC Error”,第7类“BFD Down”故障,其中第5类性能表现显著,Accuracy由原始0-Shot策略的71.22%提升到了90.01%,与包含所有类别范例Few-Shot-Cot策略的性能几乎持平. 这是因为模型通过上下文学习分析D_1剩余的2个类别或D_2余下8个类别示例的思维链模式,进而学到不同的故障分类模式,能识别出新类别的故障模式和其他2个或8个类别的不同模式,经思维链并快速判断故障类别. 此外,新类型故障处理每个故障案例平均耗时与所有故障类别的平均耗时几乎持平,说明模型在处理未见过数据集时响应速度快. 基座模型Mistral不仅在训练数据上表现良好,而且还能有效地应用于未知数据,这是衡量模型有效性、泛化性的一个比较重要的方面.

表6 Claude引导Mistral应对新类型故障的推理性能

数据集	方法	剔除类别	Accuracy/%	Macro-F1/%	Weighted-F1/%	每案例平均时间/s
D_1	0-Shot	1	68.01	63.01	69.40	0.40
		3				
	3*2-Shot-CoT	1	82.23	78.89	82.67	7.24
		3	85.11	80.40	84.86	6.13
	3*3-Shot-CoT	1	85.53	80.80	85.35	6.79
		3				
D_2	0-Shot	5	71.22	66.57	65.83	0.41
		7				
	3*8-Shot-CoT	5	90.01	90.47	89.98	9.19
		7	85.31	81.79	85.83	6.90
	3*9-Shot-CoT	5	90.44	91.25	90.19	7.06
		7				

以上实验说明,增加链式思维提示后,模型能够持续学习新的故障模式,从而提高对新类型故障的检测和诊断能力. 这样以来不仅提高了模型的准确率,还增强了模型对新类型故障的识别能力,从而验证了新类型故障在通用框架LogCoT上表现性能也较为显著,即便应对新类型故障持续发生,在保证模型在新类型故障上高性能的同时,显著减少了训练时间和资源需求,而且应对新类型的故障也具备很强的泛化能力.

5 结论

为诊断真实生产环境中的日志故障情况,本文利用零样本提示对超大规模闭源LLM生成推理链的样本池,实现了在中等规模开源LLM中激发推理性能的目标,并展示了模型推理能力的提取和传递的潜力. 通过

收集两大互联网运营商的日志数据集,证实了LogCoT策略的有效性,并直观地给出故障类别对应的诊断信息的合理性解释. 目前研究主要使用思维链提示工程(CoT-Prompting)进行IPO偏好对齐,但仍有改进空间.

未来工作重点将集中生成更高质量的推理链以及如何更好地利用推理链来对齐中等规模开源LLM,以提高模型对话质量的解释能力,模型在未见过数据上的泛化能力及多样化用户意图的适应能力等方面.

参考文献

[1] DU X Z, YU Y, WANG P, et al. Unstructured log oriented fault diagnosis for operation and maintenance management[C]// Proceedings of the 3rd International Conference on Computer Science and Application Engineering. New York: ACM, 2019: 1-5.

[2] RUAN H X, LIU Z J, DING Y. Large-scale log-based failure diagnosis of server groups: A two-stage mining approach based on drain 3 and weight-based optimization algorithm[C]//2023 Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC). Piscataway: IEEE, 2023: 302-306.

[3] JIA T, LI Y, ZHANG C B, et al. Machine deserves better logging: A log enhancement approach for automatic fault diagnosis[C]//2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). Piscataway: IEEE, 2018: 106-111.

[4] ZHANG L Y, FAN L, GUO N W. Log-based OpenStack fault diagnosis by machine learning[J]. Journal of Physics: Conference Series, 2018, 1069: 012111.

[5] ZOU D Q, QIN H, JIN H. UiLog: Improving log-based fault diagnosis by log analysis[J]. Journal of Computer Science and Technology, 2016, 31(5): 1038-1052.

[6] JIA T, LI Y, WU Z H. Brief announcement: Automatic log enhancement for fault diagnosis[C]//Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing. New York: ACM, 2018: 415-417.

[7] HANKA S. A grammar based approach to distributed systems fault diagnosis using log files[C]//Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE). New York: ACM, 2019: 1-81.

[8] YANG Z, YING S, WANG B M, et al. A system fault diagnosis method with a reclustering algorithm[J]. Scientific Programming, 2021, 2021: 6617882.

[9] ZHANG X, XU Y, QIN S, et al. Onion: Identifying incident-indicating logs for cloud systems[C]//Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. New York: ACM, 2021: 1253-1263.

[10] CHUAH E, JHUMKA A, BROWNE J C, et al. Insights into the diagnosis of system failures from cluster message logs[C]//2015 11th European Dependable Computing Conference (EDCC). Piscataway: IEEE, 2015: 225-232.

[11] TAK B C, TAO S, YANG L, et al. LOGAN: Problem diagnosis in the cloud using log-based reference models[C]// 2016 IEEE International Conference on Cloud Engineering (IC2E). Piscataway: IEEE, 2016: 62-67.

[12] XIE Y X, YANG K, LUO P. LogM: Log analysis for multiple components of hadoop platform[J]. IEEE Access, 2021, 9: 73522-73532.

[13] NAGARAJ K, KILLIAN C, NEVILLE J. Structured comparative analysis of systems logs to diagnose performance problems[C]//Proceedings of NSDI 2012: 9th USENIX Symposium on Networked Systems Design and Implementation. San Jose: NSDI, 2012: 353-366.

[14] IKEUCHI H, WATANABE A, KAWATA T, et al. Root-cause diagnosis using logs generated by user actions[C]// 2018 IEEE Global Communications Conference (GLOBECOM). Piscataway: IEEE, 2018: 1-7.

[15] HE S L, LIN Q W, LOU J G, et al. Identifying impactful service system problems *via* log analysis[C]//Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. New York: ACM, 2018: 60-70.

[16] CHU Z, CHEN J C, CHEN Q L, et al. Navigate through enigmatic labyrinth A survey of chain of thought reasoning: Advances, frontiers and future[C]//Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Stroudsburg: USAACL, 2024: 1173-1203.

[17] KOJIMA T, GU S S, REID M, et al. Large language models are zero-shot reasoners[C]//Proceedings of the 36th International Conference on Neural Information Processing Systems. Virtual Event: NeurIPS, 2022: 22199-22213.

[18] HSIEH C Y, LI C L, YEH C K, et al. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes[C]//Findings of the Association for Computational Linguistics: ACL 2023. Stroudsburg: USAACL, 2023: 8003-8017.

[19] MAGISTER L C, MALLINSON J, ADAMEK J, et al. Teaching small language models to reason[C]//Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Stroudsburg: USAACL, 2023: 1773-1781.

[20] HO N, SCHMID L, YUN S Y. Large language models are reasoning teachers[C]//Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Stroudsburg: USAACL, 2023: 14852-14882.

[21] LI L H, HESSEL J, YU Y, et al. Symbolic chain-of-thought

distillation: Small models can also "think" step-by-step[C]// Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Stroudsburg: USAACL, 2023: 2665-2679.

[22] YANG S, SHANG Z R, WANG Y Q, et al. Data-free multi-label image recognition via LLM-powered prompt tuning[EB/OL]. (2024-03-02)[2025-04-22]. https://arxiv.org/abs/2403.01209v1.

[23] OYMAK S, RAWAT A S, SOLTANOLKOTABI M, et al. On the role of attention in prompt-Tuning[C]//International Conference on Machine Learning. New York: PMLR, 2023: 26724-26768.

[24] OUYANG L, WU J, JIANG X, et al. Training language models to follow instructions with human feedback[C]//Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS). New Orleans: Curran Associates, 2022: 27730-27744.

[25] KAUFMANN T, WENG P, BENGS V, et al. A survey of reinforcement learning from human feedback[EB/OL]. (2024-04-30)[2025-04-22]. https://arxiv.org/abs/2312.14925v2.

[26] QIN L B, CHEN Q G, WEI F X, et al. Cross-lingual prompting: Improving zero-shot chain-of-thought reasoning across languages[C]//Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Stroudsburg: USAACL, 2023: 2695-2709.

[27] KONG A B, ZHAO S W, CHEN H, et al. Better zero-shot reasoning with role-play prompting[C]//Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). Stroudsburg: USAACL, 2024: 4099-4113.

[28] LE V H, ZHANG H Y. Log parsing: How far can ChatGPT go? [C]//2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE). Piscataway: IEEE, 2023: 1699-1704.

[29] LIANG Y Y, WANG J N, ZHU H L, et al. Prompting large language models with chain-of-thought for few-shot knowledge base question generation[C]//Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Stroudsburg: USAACL, 2023: 4329-4343.

[30] WANG L, XU W Y, LAN Y H, et al. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models[C]//Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Stroudsburg: USAACL, 2023: 2609-2634.

[31] LIN Q W, ZHANG H Y, LOU J G, et al. Log clustering based problem identification for online service systems[C]//2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C). Piscataway: IEEE, 2016: 102-111.

[32] QAISER S, ALI R. Text mining: Use of TF-IDF to examine the relevance of words to documents[J]. International Journal of Computer Applications, 2018, 181(1): 25-29.

[33] YUAN Y, SHI W C, LIANG B, et al. An approach to cloud execution failure diagnosis based on exception logs in OpenStack[C]//2019 IEEE 12th International Conference on Cloud Computing (CLOUD). Piscataway: IEEE, 2019: 124-131.

[34] CHURCH K W. Word2Vec[J]. Natural Language Engineering, 2017, 23(1): 155-162.

[35] VERVAET A. MoniLog: An automated log-based anomaly detection system for cloud computing infrastructures[C]// 2021 IEEE 37th International Conference on Data Engineering (ICDE). Piscataway: IEEE, 2021: 2739-2743.

[36] LI X Y, CHEN P F, JING L X, et al. SwissLog: Robust and unified deep learning based log anomaly detection for diverse faults[C]//2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE). Piscataway: IEEE, 2020: 92-103.

[37] LIAO L P, ZHU K, LUO J Z, et al. LogBASA: Log anomaly detection based on system behavior analysis and global semantic awareness[J]. International Journal of Intelligent Systems, 2023, 2023(1): 3777826.

[38] SUI Y C, ZHANG Y Z, SUN J J, et al. LogKG: Log failure diagnosis through knowledge graph[J]. IEEE Transactions on Services Computing, 2023, 16(5): 3493-3507.

[39] ANKERST M, BREUNIG M M, KRIEGEL H P, et al. OPTICS: Ordering points to identify the clustering structure[J]. ACM Sigmod record, 1999, 28(2): 49-60.

[40] LIU Y L, TAO S M, MENG W B, et al. LogPrompt: Prompt engineering towards zero-shot and interpretable log analysis[C]// Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings. New York: ACM, 2024: 364-365.

[41] XU J, CUI Z A, ZHAO Y, et al. UniLog: Automatic logging via LLM and in-context learning[C]//2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE). Piscataway: IEEE, 2024: 1-12.

[42] AN L, MLOUKI O, KHOMH F, et al. Stack Overflow: A code laundering platform? [C]//2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER). Piscataway: IEEE, 2017: 283-293.

[43] XU J, YANG R C, HUO Y T, et al. DivLog: Log parsing with prompt enhanced in-context learning[C]//2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE). Piscataway: IEEE, 2024: 2457-2468.

[44] WANG J B, CHU G J, WANG J Y, et al. LogExpert: Log-based recommended resolutions generation using large language model[C]//Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results. New York: ACM, 2024: 42-46.

[45] JIAO W X, WANG W X, HUANG J T, et al. Is ChatGPT A good translator yes with GPT-4 As the engine[EB/OL]. (2023-11-02)[2025-04-22]. https://arxiv.org/abs/2301.08745v4.
[46] ZHOU H, NOVA A, LAROCHELLE H, et al. Teaching algorithmic reasoning via in-context learning[EB/OL]. (2022-11-15)[2025-04-22]. https://arxiv.org/abs/2211.09066v1.
[47] CHEN W H. Large language models are few(1)-shot table reasoners[EB/OL]. (2023-01-23)[2025-04-22]. https://arxiv.org/abs/2210.06710v2.
[48] CHENG S T, ZHUANG Z Y, XU Y, et al. Call me when necessary: LLMs can efficiently and faithfully reason over structured environments[C]//Findings of the Association for Computational Linguistics ACL 2024. Stroudsburg: USAACL, 2024: 4275-4295.
[49] ZHANG Y F, YANG J Q, YUAN Y, et al. Cumulative reasoning with large language models[EB/OL]. (2025-03-12)[2025-04-22]. https://arxiv.org/abs/2308.04371v7.
[50] AN S N, MA Z X, LIN Z Q, et al. Learning from mistakes makes LLM better reasoner[EB/OL]. (2024-03-29)[2025-04-22]. https://arxiv.org/abs/2310.20689v4.
[51] QIAO S F, OU Y X, ZHANG N Y, et al. Reasoning with language model prompting: A survey[C]//Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Stroudsburg: USAACL, 2023: 5368-5393.
[52] WANG Z H, LIU A J, LIN H W, et al. RAT: Retrieval augmented thoughts elicit context-aware reasoning in long-horizon generation[EB/OL]. (2024-03-08)[2025-04-22]. https://arxiv.org/abs/2403.05313v1.
[53] HU Y S, LEE C H, XIE T B, et al. In-context learning for few-shot dialogue state tracking[C]//Findings of the Association for Computational Linguistics: EMNLP 2022. Stroudsburg: USAACL, 2022: 2627-2643.
[54] RAFAILOV R, SHARMA A, MITCHELL E, et al. Direct preference optimization: Your language model is secretly a reward model[C]//Proceedings of the 37th International Conference on Neural Information Processing Systems. New Orleans: Curran Associates, 2024: 53728-53741.
[55] AZAR M G, ROWLAND M, PIOT B, et al. A general theoretical paradigm to understand learning from human preferences[C]//International Conference on Artificial Intelligence and Statistics. New York: PMLR, 2024: 4447-4455.
[56] ETHAYARAJH K, XU W, MUENNIGHOFF N, et al. KTO: Model alignment as prospect theoretic optimization[EB/OL]. (2024-11-19)[2025-04-22]. https://arxiv.org/abs/2402.01306v4.

作者简介

许　婷　女,1991年10月出生于河南驻马店市. 现为南开大学软件学院软件工程专业博士研究生. 主要研究方向为异常检测、故障定位、根因分析和故障预测等.
E-mail: xuting@mail.nankai.edu.cn

肖　桐　男,1990年出生于湖南邵阳市. 现为清华大学博士后. 主要研究方向为基于日志的异常检测、根因定位、故障预测等.
E-mail: xiaotong@tsinghua.edu.cn

张圣林　男,1989年7月出生于山东滨州市. 现为南开大学副教授,副院长,博士生、硕士生导师. 主要研究方向为基于机器学习的智能运维,包括异常检测、故障定位、根因分析和故障预测等.
E-mail: zhangsl@nankai.edu.cn

孙一丹　男,2002年11月出生于天津市. 本科就读于南开大学软件学院,硕士就读于浙江大学软件学院. 主要研究方向为大语言模型、表征学习等.
E-mail: syd20021134@163.com

孙永谦　男,1988年出生于河北石家庄市. 现为南开大学副教授,博士生、硕士生导师. 主要研究方向为智能运维、人工智能、网络智能管理等.
E-mail: sunyongqian@nankai.edu.cn

裴　丹　男,出生于河北省. 现为清华大学计算机科学与技术系长聘副教授、博士生导师. 主要研究方向为智能运维、时间序列智能等.
E-mail: peidan@tsinghua.edu.cn