# 使用大语言模型为云事件推荐根本原因及缓解措施

Toufique Ahmed*§、Supriyo Ghosh†、Chetan Bansal†
Thomas Zimmermann‡、Xuchao Zhang†、Saravan Rajmohan†
*加州大学戴维斯分校

†微软
‡微软研究院

arXiv:2301.03797v2 [cs.SE] 9 Feb 2023

摘要—**云服务的事件管理是一个涉及多个步骤的复杂过程，对服务健康状况和开发人员生产力都有重大影响。值班工程师需要具备大量领域知识，并付出大量人工努力，才能查明生产环境事件的根本原因并采取缓解措施。人工智能领域的最新进展催生了 GPT-3.x（包括 GPT-3.0 和 GPT-3.5）等最先进的大语言模型，它们已被用于解决从问答到文本摘要等各种问题。在本研究中，我们首次开展大规模研究，评估这些模型在帮助工程师查明生产环境事件根本原因并采取缓解措施方面的效能。我们在微软对超过 40,000 起事件进行了严格研究，并使用语义和词汇指标，对多种大语言模型在零样本、微调和多任务设置下的表现进行比较。最后，我们对实际事件负责人进行人工评估，结果显示，人工智能在解决云事件方面具有良好效能和未来潜力。**

索引词—**事件管理、服务质量、GPT-3.x、大语言模型**

## I. 引言

亚马逊、谷歌、微软和 Salesforce 等大型 IT企业已取代传统的盒装软件，转而在云平台上部署应用程序和服务。在当今的云系统中，生产环境事件（例如服务中断或性能下降、计划外中断）会对客户造成不利影响；此外，违反服务级别协议所带来的罚款，以及缓解事件所需的工程工作，也可能造成高昂成本。例如，据估计，在购物旺季，停机一小时会给亚马逊造成 1 亿美元的损失 [1]。尽管多年来一直致力于提升可靠性，云服务仍不可避免地会发生严重事件。

人工智能在 IT 运维（AIOps）领域日益普及。数据驱动方法和人工智能技术已用于自动化事件生命周期中的部分环节，例如事件优先级排序 [2],检索具有相似症状的事件 [3], 以及缩短事件缓解时间 [4], [5]。然而，值班工程师（OCE）仍需花费大量时间进行手工繁杂工作，通过多轮沟通来确定根本原因和缓解措施。受近期利用 GPT-3 模型完成非平凡任务 [6], [7] 和代码生成 [8],所取得的成功启发，我们将此类模型应用于事件管理。我们确定了以下两种场景：

1) **找出事件的根本原因**。事件诊断通常需要耗费大量时间并进行充分沟通，工程师才能确定事件的根本原因。我们研究大语言模型在提出事件根本原因方面的有效性（RQ1）。
2) **为事件提出缓解措施**。找到根本原因后，工程师会采取行动缓解问题。我们研究大语言模型在为事件推荐缓解措施方面的效果（RQ2）。

应用大语言模型时，需要考虑并作出若干决策。由于模型并非使用事件管理数据训练，是否有必要对模型进行微调 （RQ3）？ 为每项任务分别构建一个模型（单任务），还是构建一个同时支持根本原因和事件的组合模型（多任务），哪种方式更有效（RQ4）？ 根本原因是否有助于语言模型找到更好的缓解措施（RQ5）？ 模型在某些类型的事件上表现是否更好（研究问题6）？ 我们通过对微软 1,759 项服务中的 44,340 起事件进行严格的大规模评估来回答这些问题。除了这类实验通常报告的词汇和语义评估指标外，我们还展示了人工验证的结果：我们请事件负责人评估所建议的根本原因和缓解措施的正确性与可读性。原事件负责人最有资格评估模型在处理事件时的表现。本文的贡献如下：

1) 本文首次证明，在真实场景中，GPT-3.x（包括 GPT-3.0 和 GPT-3.5）等最先进的大语言模型对于解决生产环境事件具有实用价值。（第三节）2) 我们在微软针对来自 1000+云服务的 40,000 多起事件开展了一项严谨的大规模研究，并采用了六种语义和词汇指标。（第四节）

- 微调显著提升了大语言模型处理事件数据的效果。
- 在我们的实验中，GPT-3 和 GPT-3.5 模型的表现显著优于编码器-解码器模型。
- BLEU-4 等指标有助于衡量模型在不同设置下的相对

§这项工作是在作者于微软研究院实习期间完成的。

性能。不过，评估实际性能还需要人工检查并由专家进行验证。

3) 我们对生产环境事件的实际事件负责人开展的人工研究，有助于证明所提方法的效能。（第五节）

## II. 概述

### *A.* 事件管理

大规模云服务中生产环境事件不可避免，而且往往会严重影响客户体验。此外，查明这些事件的根本原因并采取缓解措施，需要投入工程资源，成本可能极高。事件生命周期通常包括以下四个阶段：**(1) 检测：** 事件生命周期的第一步是检测：当给定服务的内部或外部客户发现异常行为后，会报告事件。此外，服务负责人配置的自动化监控器也可能报告事件。**(2) 分诊：** 事件报告后，一支 OCE 团队会分析问题，并将事件工单转交给合适的工程团队。这个过程通常称为事件分诊。**(3) 诊断：** 故障诊断和根因识别过程需要工程师通过多轮沟通，检查事件的不同方面，以了解事件的整体情况并确定根本原因。**(4) 缓解措施：** 根据已确定的根本原因采取行动，以缓解问题、恢复服务健康状况，并尽量减少对服务用户的影响。

近年来，智能运维（面向 IT 运维的人工智能）通过将数据驱动方法和人工智能技术与应用程序日志、时间序列性能指标及服务追踪等数据源相结合，在自动化事件生命周期的各个环节方面日益受到关注 [2]，[4]，[5]，[9]。尽管付出了巨大努力，大型云系统中的事件管理仍需要投入大量工程资源和成本。更具体地说，即便拥有海量历史事件数据，根本原因识别和缓解措施的制定仍是出了名的困难且耗时的任务。在这项工作中，我们提出利用 GPT-3.x 等大语言模型，通过借助历史事件数据，自动为新事件推荐根本原因和缓解措施。

### *B.* 大语言模型/*GPT-3.x* 模型的前景

近年来，GPT-3.x 等大型语言模型（LLM） [7] 已成为自然语言处理领域最热门的趋势之一。GPT-3.x 语言模型拥有 1750 亿个参数，曾创下已开发的最大神经网络的纪录，其规模比此前的语言模型大一个数量级。GPT-3.x 利用这种庞大的模型架构，并使用互联网上几乎所有可访问的数据进行训练，包括 CommonCrawl[10]、WebText [11]，、维基百科 [12],和图书语料库。

**标题：**连接超时导致虚拟机连接失败**摘要：** 该工作区未关联任何虚拟网络。客户有一台虚拟机，已在某个虚拟网络中运行。他们希望将该虚拟机连接到 [产品已省略]。我们尝试了用户界面和命令行界面两种方式，但仍然出现相同的连接超时错误。错误提示解析结果指向某个公共 IP 地址 [...]**根本原因参考：** 不支持将专用虚拟机直接连接到公共工作区。**缓解措施参考：** 创建一项任务，为客户提供有关虚拟机的更完善的官方文档。

图1：生产环境事件示例。

GPT-3.x 模型在多种自然语言处理任务中超越了最先进的模型，包括机器翻译、问答和完形填空。此外，GPT-3.x 模型还实现了一项重要突破：经过充分数据训练的无监督语言模型，只需少量新任务示例，便能以与微调模型相当的水平处理多项任务。凭借在新任务中强大的文本生成能力，GPT-3.x 被广泛应用于从生产力、教育到创造力和游戏等诸多领域和行业。例如，GPT-3.x 可用于创作创意写作内容，包括模仿莎士比亚等知名作家文学风格的博客文章、广告和诗歌。

### *C.* 事件根因分析与缓解

事件的根因分析与缓解是一个复杂的过程，需要投入大量人工工作，还需要掌握有关服务的领域知识。事件可能由各种问题引起，例如代码缺陷、依赖故障、基础设施问题、配置缺陷等。由于可能性繁多，OCE 很难确定事件的根本原因。同样，在确定根本原因后，也可以采取多种缓解措施，例如代码回滚、热修复、基础设施变更、配置更新等。确定正确的缓解措施同样并非易事，需要领域知识和经验。在事件根因分析或缓解过程中出现人为错误，不仅会增加工作量和人工负担，还会影响客户和收入。图1展示了某项服务中的一个真实事件，其中包含客户提供的标题和摘要，以及实际的根本原因和缓解措施。

在本研究中，我们评估了 GPT-3.x 和 Codex 等大型语言模型在分析根本原因并缓解生产环境事件方面的有效性。创建事件时，作者会为事件指定标题，并描述任何相关细节，例如错误消息、异常行为以及其他可能有助于解决问题的信息。OCE 开始调查事件后，可能会通过与事件作者沟通，或查看遥测数据和日志来获取更多详情。在调查过程中，OCE 可能会经常更新事件详情。在我们的评估中，我们使用给定事件创建时的标题和摘要作为输入

并生成根本原因和缓解措施。这样可以确保我们仅使用 OCE 开始调查事件时可获取的信息。

### *D.* 研究问题

我们研究了多个 OpenAI GPT-3.x 模型（即，Curie、Codex-cushman、Davinci、Code-davinci-002），以生成该事件的根本原因和缓解计划。这引出了若干研究问题。

*RQ1* 微调后的 *GPT-3.x* 模型能否有效找出该事件的根本原因？

由于事件管理数据包含敏感的隐私信息，且微软遵循标准协议以确保数据安全，因此 OpenAI 模型并未使用此类数据进行训练。因此，GPT-3.x 模型在零样本／少样本设置下的表现预计不会很好。本文使用不同容量的四种 GPT-3.x 模型进行微调，并观察这些模型在提出该事件根本原因方面的表现。

*RQ2* 微调后的 *GPT-3.x* 模型能否为该事件提出缓解计划？

我们还希望使用 GPT-3.x 模型为该事件生成缓解计划。与根本原因生成一样，我们使用 RQ1 中采用的输入和标准对模型进行微调和评估。

*RQ3* 微调在多大程度上提升了 *GPT-3.x* 模型的零样本学习性能？

尽管我们主要关注微调，但据报道，GPT-3.x 模型在零样本和少样本训练 [7], [8]下执行各种下游任务时表现出色。在少样本学习中，我们将少量示例作为输入放入提示词中，模型随后生成预期的输出。零样本与少样本训练类似，但不提供任何示例。这两种设置在经济和环境方面都有益处（可减少碳足迹），因为我们不会更新模型的任何参数。本文将研究模型在零样本设置下的表现。请注意，少样本学习不适用于我们的项目，因为数据集中包含较长的序列，而且我们发现，微调后仅推理一个序列时，序列会被截断。

*RQ4* 多任务学习能否提升 *GPT-3.x* 模型查找根本原因和缓解计划的性能？多任务学习对某些预训练模型有效 [13]。到目前为止，我们讨论了分别训练模型，并独立使用输入来生成事件的根本原因和缓解计划。我们想了解在特定设置下，GPT-3.x 模型对多任务学习有何反应。在这项实验中，我们将两个任务的所有训练数据合并。不过，在评估期间，我们使用了 RQ1 和 RQ2 中相同的测试集。

研究问题*5*：如果给出根本原因，*GPT-3.x* 模型在提出缓解计划方面是否会表现得更好？

事件的缓解计划取决于具体的根本原因。不同的根本原因可能导致不同的缓解计划。此外，可以通过扩大输入规模或提供更多信息来改进 GPT-3.x 模型。我们还将研究在输入中提供根本原因是否有助于模型找到缓解计划。

研究问题*6*：与人工检测到的事件相比，这些模型是否更擅长为机器检测到的事件提出缓解计划？

事件可以由机器（通过某些监控器）检测，也可以由人工检测。这两类事件各有特定特征。机器检测到的事件通常由监控器观察到的系统变化触发，例如构建失败、资源可用性、请求数量等。相反，人工检测到的事件各不相同，可能仅适用于某个特定客户（例如，网页无法加载）。在该研究问题中，我们将考察模型是否能妥善处理属于特定类别的事件。

### *E.* 人工验证

根本原因和缓解计划可以有不同的表述形式。与自然语言翻译或代码摘要不同，根本原因和缓解措施的开放性要高得多。根据作者的不同，根本原因和缓解计划可以从笼统到具体不等。自动化指标可能无法理想地反映模型的整体表现，因为这些指标会将模型的建议与一个参考答案进行比较，而该答案可能与模型给出的正确且相关的输出完全不同。为了更好地了解模型的表现，我们联系了相关事件的负责人／解决者，并向他们展示了模型和基线给出的解决方案。他们为模型输出的正确性和可读性打分。我们将在第五节讨论人工验证的方法和发现。

## III. 方法

### *A.* 数据集准备

微软每天都会检测到数千起严重程度各异的事件（由机器和人工共同检测）。值班工程师（OCE）持续不懈地工作，为客户提供无缝服务。为管理如此大规模的事件，微软建立了一个设计完善的网站，用于报告和管理事件。数据库还会跟踪该网站从事件报告到缓解措施期间的数据插入、修改和删除。模型的一个输入是在报告或创建事件时撰写的摘要，以防止数据从输入泄漏到输出。

在大多数情况下，OCE 并不遵循任何特定格式来撰写事件摘要、根本原因和缓解计划。这些字段，尤其是摘要，包含多种形式的信息，包括表格、指向先前事件的链接，以及单独的监控器输出或代码片段的图片。这是因为各个事件彼此差异很大，而 OCE 的首要任务是解决事件，而不是记录症状。此外，有些事件是短暂的，会自动缓解。如果严重程度较低，则不会进行事后分析。由于 GPT-3.x 是文本模型，我们从摘要中删除了表格和图片。因此，删除这些信息时，我们可能会丢失一些关键内容。

我们从数据库中收集了创建日期介于2018年1月1日至2022年7月15日之间的事件数据。最初，我们从“已解决”或“已缓解”、严重程度级别为0–3的事件中，收集了123,953条根本原因记录和23,544条缓解措施记录（最严重的事件属于级别0）。缓解措施样本较少，因为这类信息可以在事件的事后分析中找到，而并非每起事件都会进行事后分析。收集数据后，我们发现许多事件的根本原因和缓解措施重复。一些严重事件／拒绝服务会针对同一事件触发数百份事件报告，其中的根本原因和缓解措施完全相同。为了公平评估模型，我们删除了完全重复的根本原因和缓解计划，最终保留57,520条根本原因记录和8,300条缓解计划。根本原因和缓解措施的平均长度分别为87和12个词元。有些根本原因非常长，模型和人工评估者难以生成和评估模型的输出。我们将根本原因截取至最多100个词元，从而保留了73%的根本原因记录。我们还删除了少于三个词元的根本原因和缓解计划，因为这些内容缺乏信息量。

去重和筛选后，我们按照创建日期对数据进行排序，仅使用历史数据来训练模型。我们分别选取了 35820、3000 和 2000 个根本原因用于训练、测试和验证。缓解措施的实例较少，因此，缓解措施的训练集、测试集和验证集分别包含 5455、2000 和 500 条数据。即使经过严格的数据筛选和去重，部分根本原因和缓解措施仍不包含任何有用信息（例如，根本原因位于不同链接中、瞬时事件以及自动缓解的事件）。我们人工审查了测试集中的 3000 个根本原因和 2000 个缓解计划，并选出了 2,621 个根本原因和 1,780 个缓解计划。 [1]

### *B. OpenAI* 模型和基线

深度神经网络模型的近期发展深受 Transformer 模型的影响 [14]。以往的方法（即，LSTM [15] 和 GRU [16]）使用循环架构对生成文本的序列依赖关系进行建模。这些循环模型使用“随时间反向传播”（BPTT），在同一循环单元内通过梯度递归传播损失值，因此在捕捉序列中词元的长距离依赖关系时无法进行并行计算。Bahdanau 等人 提出了一种建立在循环架构之上的注意力机制，通过提供一个注意力向量来指示输入中与目标输出相关的部分，从而提升循环神经网络模型的性能 [17]。Transformer 模型完全移除了循环单元，完全依赖注意力机制。它采用多层、多头自注意力架构，其中的注意力机制能够关联单个序列中的不同位置，以计算序列表示。

[1]我们无法分享该数据集，因为事件数据可能包含机密信息和隐私数据，分享此类数据会违反服务条款。

预训练模型目前在各种自然语言和代码任务中都达到了最先进的性能。这些预训练模型分两个阶段运行（即，预训练和微调）。在预训练阶段，我们以自监督的方式，利用大规模语料库训练模型，使其学习语言（或代码）的统计规律。之后，我们使用规模较小的标注数据集对模型进行微调，以适应特定任务。要拥有足够的标注数据来训练此类高容量深度学习模型几乎是不可能的。预训练模型使我们能够在预训练阶段以自监督的方式，利用未标注数据训练此类大型模型。近期所有的预训练模型（仅编码器和编码器-解码器模型）（例如，BERT [18], RoBERTA [19], BART [20], T5 [21]）以及仅解码器生成模型（例如，GPT [22], GPT-2 [23], GPT-3 [7],OPT [24]），本质上都是采用不同预训练目标训练而成、容量各异的 Transformer模型。以下小节将简要介绍我们在实验中使用的基线和 OpenAI 模型。

*1)* 基线编码器-解码器模型：我们可以将编码器-解码器模型同时应用于根本原因分析和缓解措施生成。编码器将对输入进行编码，解码器则使用编码器提供的编码表示生成根本原因或缓解措施。

预训练的自然语言处理模型（例如，BERT [18], RoBERTa [19],BART [20], T5 [21]）使用不同的自监督预训练目标来学习稳健的语言表示。自然语言处理模型有对应的编程语言模型（例如，CodeBERT [25], GraphCodeBERT [26], PLBART [27],CodeT5 [13], NatGen [28]），在大多数情况下，这些模型使用自然语言处理模型的权重进行初始化，并继续使用代码及其相关的自然语言注释进行预训练。尽管根本原因和缓解措施都是自然语言描述，但其词汇表（例如，标识符）与代码模型所使用的注释重叠更多。因此，我们从 OpenAI 和基线标准中选取了自然语言处理模型和代码模型，以观察模型性能是否会因预训练所使用的领域不同而有所差异。作为基线，我们选择了 RoBERTa [19] 和 CodeBERT [25] 模型，原因有二：i）这两个模型在架构上完全相同，均包含 1.25 亿个参数；ii）这两个模型都被广泛用作基线（事实上，CodeBERT 是 CodeXGLUE [29] 数据集的主要基线模型，而该数据集是一个广受欢迎的基准测试，涵盖 10 个软件工程任务，包括编码器-解码器任务，如代码摘要和代码翻译）。需要注意的是，许多基于 Transformer 的编码器-解码器模型都可以应用于这一问题。然而，与所有模型进行比较超出了本文的范围。

***RoBERTa***：BERT 是首个引入预训练策略并超越传统 Transformer模型的模型。它采用了两种预训练策略：掩码语言模型（MLM）和 NSP（下一句预测）。在 MLM 预训练中，我们会随机遮蔽 15% 的词元，并要求模型还原这些词元；而在 NSP 中，我们训练模型学习预测输入句子之后的下一句。Liu等 [19] 提出了 RoBERTa（一种稳健优化的BERT预训练方法），通过一些改进（例如

动态掩码和移除 NSP），实现了更好的性能。我们将 RoBERTa 用作自然语言处理基线模型。

***CodeBERT***：CodeBERT 在架构上与 RoBERTa 模型完全相同，采用两种预训练目标：MLM 和替换词元检测（RTD）[30]。我们可以将 RTD 定义为一个二元分类问题，其中两个数据生成器（即自然语言和编程语言）会为一组随机掩码位置生成合理的候选词。训练判别器来判断一个词是否为原词。CodeBERT 在 CodeSerachNet [31]数据集上进行预训练。

*2*）*OpenAI* 生成式模型：Radford等人 提出了通用且与任务无关的语言模型生成式预训练（GPT），其表现优于 12 个采用针对特定任务设计的架构进行判别式训练的模型中的 9 个 [22]。在生成式预训练中，我们从左向右自回归地预测给定前序词元时某个词元的概率。这种从左向右的自回归训练会阻止模型检索未来词元中的信息。所有后续生成式模型（例如， GPT- 2、GPT-3）采用非常相似的预训练目标，但容量高于先前模型，且使用大得多的数据集进行预训练。像 GPT-3.x 这样的大型语言模型（大语言模型）拥有 1750 亿个参数；事实证明，少样本学习能有效替代针对特定任务进行微调的需求。然而，对 GPT-3.x 模型进行微调仍有利于某些任务 [7]。本文使用四种 OpenAI [32] GPT- 3.x 模型评估我们的方法：Curie、Codex、Davinci 和 Code-davinci-002。

***Curie***：Curie 是速度最快的 GPT-3 模型，拥有 67 亿个参数。该模型使用自然语言数据进行训练，在语言翻译、复杂分类、文本情感分析和摘要任务上表现出色。这是我们实验中使用的最小模型。

***Codex***：Codex 模型也是 GPT-3 模型，经过训练后能够理解和生成代码。训练数据既包含自然语言，也包含来自 GitHub 的数十亿行公开代码。我们使用了一个模型，即 Codex 系列中的 Codex-cushman，拥有 120 亿个参数。尽管这些模型经过预训练，可用于代码相关任务，但它们与事件管理也有一定关联。根本原因和缓解措施中包含许多术语（例如，文件名、标识符），这些内容与软件开发项目中使用的注释更加相关。

***Davinci***：Davinci 是我们实验中使用的最大 GPT-3 模型（拥有 1750 亿个参数）。与其他 GPT-3 模型相比，它只需较少的指令就能执行任务。Davinci 通常在理解内容或创意内容生成任务方面表现更好，在解决逻辑问题方面也非常出色。然而，训练拥有 1750 亿个参数的模型成本高昂，所需时间也长得多（使用相同数据集时，所需时间几乎是 Curie 的四倍），并且需要更多资源。Davinci 并未经过理解或生成代码的训练。

***Code-davinci-002***：Code-davinci-002 是我们在实验中使用的 1750 亿参数 GPT-3.5 模型。Code-davinci-002 是 Codex 模型的升级版，能力更强，并且是在更新的文本和代码语料数据集上训练的。

### *C.* 模型配置

预训练编码器-解码器模型的局限之一是只能编码 512 个词元。我们观察到，测试集中的若干样本在 GPT-3 模型中被截断，尽管 GPT-3 模型支持的词元数从 2048 个（例如，Curie、Codex）到 4000 个（例如，Code-davinci-002）不等。因此，我们可以推断，传统的编码器-编码器模型没有足够的容量来编码我们的序列。

编码器-解码器模型已成功应用于代码摘要 [13], [25], [27],代码翻译 [29],和自然语言翻译 [14], [20], [21]等问题。对于每个输入，我们通常使用束搜索生成一个样本，并将结果与参考答案进行比较。对于这些问题，生成一个样本已足够，因为目标文本的开放性较低。此外，对于这类问题，成功生成所需的大部分信息都可以在输入中找到。代码翻译模型需要学习两种编程语言之间的句法对齐。学习将条件语句和循环从一种编程语言转换为另一种，或许就足以实现成功的翻译；而这一能力可以通过几千个样本学会。对于自然语言翻译，学习不同自然语言中词语之间的映射对于生成高质量翻译至关重要。代码摘要与这两类任务略有不同，其输入远长于输出。然而，Ahmed 和 Devanbu 发现，代码摘要所需的全部信息都从标识符中提取，而混淆标识符会损害模型的表现 [33]。生成根本原因和缓解计划比这些问题复杂得多，因为输入中可能不包含现成的信息。模型需要能够生成更多样、更具创造性的解决方案来回答问题。我们的问题更接近代码生成问题，因为输入并未包含大部分信息。对于这类问题，研究发现，相较于使用编码器-解码器模型，仅解码器模型（例如，GPT-3.x）表现更为出色；此类模型只需根据先前生成的词元，专注于后续词元。众所周知，在代码生成任务中，编码器-解码器模型的表现不如仅解码器模型。不过，我们仍将编码器-解码器模型应用于我们的问题，并在后续章节中讨论研究发现。对于 RoBERTa [19] 和 CodeBERT [25]，我们采用代码摘要任务中使用的完全相同的设置 [31], [34]。我们将长度调整为 512 个词元，批量大小设为 8，以便向模型提供尽可能多的信息。

全量微调需要重新训练所有参数，对于拥有数十亿参数的 OpenAI 模型而言，成本高昂且颇具挑战。我们采用 LoRA （低秩适应），这是一种新颖的方法：通过冻结预训练模型权重，并在 Transformer 架构的每一层中注入可训练的秩分解矩阵，显著减少可训练参数的数量 [35]。尽管 LoRA 减少了可训练参数的数量，但在 RoBERTa、DeBERTa、GPT-

2 和 GPT-3 上，其模型质量与微调相当或更优。我们对 OpenAI 的 GPT-3（即 Curie、Codex、Davinci）和 GPT-3.5（Code-davinci-002）模型进行了微调，用于生成根本原因和缓解计划。我们按照 OpenAI 的建议，对两个模型都训练 2000 步（4 个周期）。对于较小的模型（即 Curie 和 Codex），我们使用一块 NVIDIA V100 GPU；对于 Davinci，我们使用四块 NVIDIA V100 GPU。对于 Code-davinci-002 模型的微调，我们使用四块 NVIDIA A100 GPU。我们每训练 100 步就在验证集上评估模型，并选择在验证集上训练损失最低的模型。

如前所述，模型需要生成更多样、更具创意的建议，以解决预测根本原因和制定缓解计划等问题。控制生成结果质量的两个关键参数是 温度 和 核采样参数，建议每次只更新一个参数。根据以往研究 [8], [36], ，我们决定调整温度值。较高的温度会鼓励模型承担更多风险，这对于创意应用是必要的 [32]。较低的取值会执行 argmax 采样，这与我们在 CodeBERT 等编码器-解码器模型中采用的方法非常相似。对于创意任务，通常最常用的温度范围是 0.50–0.90。不过，温度过高会带来负面影响（导致输出过于发散） [36]。我们进行了网格搜索，并为 Curie、Codex 和 Davinci 模型选择了 0.7，为 Code-davinci-002 实验选择了 0.5，以尽量减少生成五个样本时出现的发散问题。

### *D.* 评估指标

我们简要介绍用于两个下游任务——根本原因分析和缓解生成的评估指标。

*1)* 词汇指标：对于词汇指标，我们采用平滑句子级 **BLEU-4** （双语评估替补）[37] 指标，计算参考文本与生成文本之间从 1 元到 4 元语法的重叠情况。此外，我们使用 ROUGE 指标（面向摘要评估的召回率替补） [38]来比较候选文档与一组参考文本。具体而言，我们选择 **ROUGE-L** [38], ，该指标考虑句子级结构相似度，并基于最长公共子序列（LCS） [39]统计，识别序列中共同出现的最长 n 元语法。**METEOR** （显式排序翻译评估指标） [40]是我们选用的最后一项词汇指标，它基于一元词精确率和召回率的调和均值，并将词干提取和同义词匹配作为额外特征。

*2*）语义指标：由于词汇指标通常进行精确的词语匹配，而忽略词语的含义，我们选择了三种语义指标，根据语义含义评估结果。我们使用**BERTScore** [41], ，它利用 BERT [18] 模型中的预训练上下文嵌入，并基于余弦相似度匹配候选句和参考句中的词语。接着，我们选择 **BLEURT** 分数 [42] ，以体现候选文本的流畅程度以及其传达参考文本含义的程度。最后，我们选择**NUBIA** （基于神经网络的可互换性评估器） [43], ，这是一种近期提出的基于神经网络的度量方法它通过预训练语言模型的各层表示，综合考量语义相似度、逻辑推理和句子可读性，包括 RoBERTa STS [19], RoBERTa MNLI 和 GPT-2 [23]。

语义指标的计算耗时较长，且需要昂贵的图形处理器资源（表 I 和表 II 在单个图形处理器上耗时两天）。因此，我们报告了前两个研究问题的语义指标；对于其余研究问题，我们仅采用计算成本较低的词汇指标。

## IV. 结果

### *A.* 微调后的 *GPT-3.x* 模型在生成事件根因推荐方面有多有效？（*RQ1*）

表 I 展示了基线编码器-解码器模型和微调后的 GPT-3.x 模型在根因推荐方面的效果。我们使用 2621 个测试样本来评估模型。我们为 OpenAI 模型生成了十个样本，原因有二：i）通过使用温度，我们可以从 GPT-3.x 模型生成多样且富有创意的样本；ii）我们发现，即使排名较低，GPT-3.x 模型也能生成有价值的建议。我们观察了特定排名下所有样本的平均 BLEU-4，发现所有 OpenAI GPT-3.x 模型即使在排名第八或更低时，也能生成 BLEU-4 更高的示例。不过，对于人工 OCE 来说，十个示例太多了，因此我们只选取模型给出的前五条建议。在表 I 中，每项指标均列出了前 1 和前 5。前 1 表示所有测试样本中首个候选项的均值；计算前 5 时，我们从前五个候选项中取最大值，再计算所有样本的平均值。前 5 能够总体反映模型的表现。对于基线编码器-解码器模型，每个模型只有一个样本。

令人惊讶的是，在全部六项自动化指标上，编码器-解码器模型的表现都远胜 GPT-3 模型。事实上，这六项指标都无法区分 OpenAI 模型之间的显著差异。编码器-解码器模型在自动化指标上表现出色的原因是，这些模型的探索性较弱，并且在解码过程中会根据最大值概率尽可能提高成功率。如今，“代码中存在一个漏洞”是一句非常常见且笼统的话，可以出现在任何根本原因中。模型只需复制这一特定片段，就能提高成功率，而自动化指标对此也无能为力。我们尝试了三种语义指标来解决这个问题，但编码器-解码器模型仍然受益于自动化指标。表 III 展示了模型生成的唯一样本数量。为进行公平比较，对于 OpenAI 模型，我们只考虑第一个候选结果。我们发现，RoBERTa 和 CodeBERT 的唯一样本数量分别占总数的 6.10% 和 16.67%；而所有 OpenAI GPT-3.x 模型的这一比例都超过 97%。请注意，我们已经对数据集进行了去重，因此反复生成相同样本在这里并无助益。在第五节中，我们采访了事件负责人，其中大多数人抱怨编码器-解码器模型的建议过于笼统，而且这些模型

T表 I：微调后的 GPT-3.x 模型查找事件**根本原因**的效果

| 模型 | BLEU-4 | | ROUGE-L | | METEOR | | BERTScore | | BLEURT | | NUBIA | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Top1 | Top5 | Top1 | Top5 | Top1 | Top5 | Top1 | Top5 | Top1 | Top5 | Top1 | Top5 |
| RoBERTa | 4.21 | NA | 12.83 | NA | 9.89 | NA | 85.38 | NA | 35.66 | NA | 33.94 | NA |
| CodeBERT | 3.38 | NA | 10.17 | NA | 6.58 | NA | 84.88 | NA | 33.19 | NA | 39.05 | NA |
| Curie | 3.40 | 6.29 | 9.04 | 15.44 | 7.21 | 13.65 | 84.90 | 86.36 | 32.62 | 40.08 | 33.52 | 49.76 |
| Codex | 3.44 | 6.25 | 8.98 | 15.51 | 7.33 | 13.82 | 84.85 | 86.33 | 32.50 | 40.11 | 33.64 | 49.77 |
| Davinci | 3.34 | 5.94 | 8.53 | 15.10 | 6.67 | 12.95 | 83.13 | 84.41 | 31.06 | 38.61 | **35.28** | 50.79 |
| Davinci-002 | **4.24** | **7.15** | **11.43** | **17.2** | **10.42** | **16.8** | **85.42** | **86.78** | **36.77** | **42.87** | 32.3 | **51.34** |
| %gain for Davinci-002 | 23.26 | 13.67 | 26.44 | 10.90 | 42.16 | 21.56 | 0.61 | 0.49 | 12.72 | 6.88 | -8.45 | 1.08 |

在正确性标准方面表现不佳。在 OpenAI 模型中，GPT-3.5（即 Code-davinci-002）模型在全部六项自动化指标上的表现都显著优于所有 GPT-3 模型以及其他基线。

尽管自动化指标无法检测出编码器-解码器模型的不足，但这些指标仍被广泛使用。人工评估难以在每种场景下开展，而这些指标有助于比较模型的相对表现。因此，即使我们在这些指标上的分数较低，在尝试衡量模型在不同设置下的相对表现时，这些指标仍然有用。此外，词汇指标得分较低并不令人意外，因为词汇指标只考虑词元重叠，而根本原因和缓解措施都是开放式内容，同一个根本原因或缓解措施可以有不同的表达方式。在第五节中，通过对 OCE 的访谈，我们发现，即使 BLEU-4 或其他指标得分较低，相关建议仍然有帮助。

*B.* 微调后的 *GPT-3.x* 模型在为事件推荐缓解计划方面效果如何？（*RQ2*）

表二显示，我们取得的缓解得分（4.44–6.76 BLEU-4）略高于根本原因建议的得分（3.38–4.24 BLEU-4）。我们观察到，与根本原因的情况相同，输出呈现出相似且一致的模式（表 III）。编码器-解码器模型会生成泛泛的评论（例如，“问题已自行缓解”“修复已部署到所有区域”），与之前一样，这些建议对 OCE 几乎毫无用处。对于 RQ1 和 RQ2，根据自动化指标，微调后的 Davinci 模型（即使拥有 1750 亿个参数）的表现也显著逊于其他基线方法。然而，根据事件负责人的评价，Davinci 和 Code-davinci-002 模型的表现最佳（见第五节）

*C.* 微调使 *GPT-3.x* 模型的表现比零样本学习提高了多少？（*RQ3*）

如第二节 D 所述，我们将考察 OpenAI 模型在零样本设置下的性能。表四展示了 OpenAI 模型在根本原因和缓解措施方面的表现。正如预期，由于这些模型并未使用事件管理领域的机密数据进行训练，因此在此设置下表现不佳。对于最佳候选项，模型的 BLEU-4 为 0.80–2.18；在推荐缓解措施时，这远低于我们通过微调模型取得的结果（5.47–6.76，低 210%）。尽管在微调期间，缓解措施的分数高于根本原因，但在零样本设置下，根本原因的数值略高（最佳候选项为 1.18–2.83）。模型会根据给定的输入尝试补全序列。从输入中复制几个词元可能会对模型有所帮助，因为根本原因通常比缓解措施更长，而且往往与输入共享更多词元。由于存在一元词重叠，METEOR 的表现优于其他指标（BLEU-4 和 ROUGE-L），因为它考察一元词精确率和召回率，因此比 BLEU-4 和 ROUGE-L 更宽松。我们还观察到另一个有趣的现象：尽管 Davinci 模型在 RQ1 和 RQ2 中表现不佳，但在根本原因和缓解措施两方面的零样本设置下，其表现都显著优于其他 OpenAI 模型。这是因为该模型的参数更多，并且接受了更多数据的训练，因此即使没有明确训练，也能进行更好的推断。

*D.* 多任务学习能否提升 *GPT-3.x* 模型查找根本原因和制定缓解计划的表现？（*RQ4*）

为评估多任务训练在根因推荐和缓解规划任务中的效果，我们将这两项任务的训练集合并，用于训练 GPT-3.x 模型。随后，分别使用相应的测试集对模型进行测试。表 V 展示了多任务训练在根本原因和缓解措施方面的结果。总体而言，我们发现，多任务训练的表现并未显著优于单任务训练。Curie 和 Codex 模型的 BLEU-4、ROUGE-L 和 METEOR 分数平均分别下降了 2.8%、2.0% 和 7.2%。只有 Davinci 模型在 BLEU-4 指标上略优于单任务训练，提升了 6.2%。在多任务设置下，Code-davinci-002 在几乎所有词汇指标上的表现都较低。类似地，缓解生成的结果显示，四种模型的平均表现下降了 4.1%。根本原因与缓解措施之间缺乏关联，是导致性能下降的主要因素。由于两项任务的答案空间分布不同，例如根本原因和缓解措施在长度和具体性方面存在差异，因此难以将一项任务中的知识迁移到另一项任务中。

*E.* 如果给出根本原因，*GPT-3.x* 模型提出缓解计划的效果会更好吗？（*RQ5*）

我们评估了根本原因逐步揭示时的缓解生成表现。由于事件中缺少根本原因信息，缓解措施训练集规模从 5,455 条减少到 2,973 条，因此我们有 166 个测试

TABLE II：微调后的 GPT-3.x 模型寻找事件缓解计划的有效性 ts

| 模型 | BLEU-4 | | ROUGE-L | | METEOR | | BERTScore | | BLEURT | | NUBIA | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Top1 | Top5 | Top1 | Top5 | Top1 | Top5 | Top1 | Top5 | Top1 | Top5 | Top1 | Top5 |
| RoBERTa | 4.44 | NA | 7.10 | NA | 4.52 | NA | 86.33 | NA | 26.80 | NA | 14.90 | NA |
| CodeBERT | 6.02 | NA | 4.40 | NA | 3.37 | NA | 86.83 | NA | 28.44 | NA | 27.89 | NA |
| Curie | 5.47 | 10.62 | 8.03 | 16.31 | 6.22 | 12.75 | 85.65 | 87.13 | 27.20 | 37.23 | 15.30 | 25.46 |
| Codex | 5.53 | 10.62 | 8.15 | 16.23 | 6.19 | 13.15 | 85.68 | 87.35 | 28.43 | 37.92 | 15.77 | 26.33 |
| Davinci | 5.54 | 10.66 | 8.10 | 15.96 | 6.08 | 12.49 | 85.72 | 87.19 | 27.15 | 37.00 | 15.71 | 25.61 |
| Davinci-002 | **6.76** | **11.66** | **10.22** | **18.14** | **8.23** | **15.13** | **86.17** | **87.65** | **30.19** | **38.96** | **17.58** | **28.81** |
| Davinci-002 的 %gain | 22.02 | 9.38 | 25.40 | 11.22 | 32.32 | 15.06 | 0.52 | 0.34 | 6.19 | 2.74 | 11.48 | 9.42 |

表 III：模型建议的独特性

| 模型 | 根本原因 | | 占总数的百分比 | 缓解措施 | | 占 |
|---|---|---|---|---|---|---|
| | 唯一项数量 | | | 唯一项数量 | | |
| | 建议 | 总数 | | 建议 | 总数 | |
| RoBERTa | 160 | 6.10 | | 4 | 0.22 | |
| CodeBERT | 437 | 16.67 | | 2 | 0.1 | |
| Curie | 2612 | 99.65 | | 1669 | 93.76 | |
| Codex | **2614** | **99.73** | | **1743** | **97.92** | |
| Davinci | 2587 | 98.70 | | 1731 | 97.24 | |
| Davinci-002 | **2614** | **99.73** | | 1696 | 95.28 | |

表四：零样本设置下 OpenAI 模型推荐根本原因和缓解措施的有效性

| 目标 | 模型 | BLEU-4 | | ROUGE-L | | METEOR | |
|---|---|---|---|---|---|---|---|
| | | Top1 | Top5 | Top1 | Top5 | Top1 | Top5 |
| 根本原因 | Curie | 1.26 | 2.01 | 4.75 | 7.80 | 7.94 | 13.30 |
| | Codex | 1.18 | 1.94 | 3.80 | 7.07 | 6.58 | 12.20 |
| | Davinci | 2.83 | 4.37 | 6.11 | 11.55 | 6.04 | 11.87 |
| | 达芬奇-002 | 1.35 | 2.5 | 4.89 | 8.58 | 7.65 | 13.55 |
| | 微调-达芬奇-002 | **4.24** | **7.15** | **11.43** | **17.2** | **10.42** | **16.8** |
| | 为……带来的增益百分比微调 | 49.82 | 63.62 | 87.07 | 48.92 | 31.23 | 23.99 |
| 缓解措施 | Curie | 0.81 | 1.50 | 2.45 | 4.59 | 5.33 | 9.40 |
| | Codex | 0.80 | 1.57 | 1.97 | 4.05 | 4.56 | 8.55 |
| | Davinci | 2.18 | 3.67 | 3.84 | 7.84 | 4.99 | 10.44 |
| | 达芬奇-002 | 0.92 | 1.89 | 2.31 | 4.52 | 4.92 | 9.2 |
| | 微调-达芬奇-002 | **6.76** | **11.66** | **10.22** | **18.14** | **8.23** | **15.13** |
| | % 增益（针对微调 | 210.1 | 217.7 | 166.2 | 131.4 | 54.4 | 44.9 |

样本用于评估模型。尽管训练集样本有所减少，表 VI 显示，加入额外的根因信息后，模型性能显著提升：Curie 模型的三项指标平均提升 9.8%，Codex 模型提升 8.3%，Davinci 模型提升 5.4%，Code-davinci-002 模型提升 26%。不过，我们发现，与第一项结果的提升相比，Code-davinci-002 模型的前五项建议提升幅度较小。尽管如此，整体上令人鼓舞的结果凸显了根因信息对于生成缓解计划的重要性。

*F.* 对于机器检测到的事件，模型提出的缓解计划是否比针对人工检测事件提出的更好？（研究问题*6*）

我们在表 VII 中分析 GPT-3.x 模型对机器检测事件和人工检测事件的缓解生成性能。我们采用相同的训练集，但按照人工检测事件和机器检测事件的类别划分测试样本。测试样本包括 592 起机器识别的事件和 1188 起人工检测的事件，

表 V：多任务学习的有效性

| 目标 | 模型 | 多任务？ | BLEU-4 | | ROUGE-L | | METEOR | |
|---|---|---|---|---|---|---|---|---|
| | | | Top1 | Top5 | Top1 | Top5 | Top1 | Top5 |
| Root 原因 | Curie | No | 3.40 | 6.29 | 9.04 | 15.44 | 7.21 | 13.65 |
| | | Yes | 3.30 | 6.13 | 8.66 | 15.51 | 6.60 | 12.97 |
| | Codex | No | 3.44 | 6.25 | 8.98 | 15.51 | 7.33 | 13.82 |
| | | Yes | 3.42 | 6.11 | 8.64 | 15.24 | 6.53 | 12.81 |
| | Davinci | No | 3.34 | 5.94 | 8.53 | 15.10 | 6.67 | 12.95 |
| | | Yes | 3.60 | 6.27 | 9.11 | 15.66 | 7.31 | 13.64 |
| | Davinci-002 | No | **4.24** | **7.15** | **11.43** | **17.2** | **10.42** | **16.8** |
| | | Yes | **4.24** | 7.09 | 11.32 | 17.14 | 10.32 | 16.34 |
| 缓解措施 | Curie | No | 5.47 | 10.62 | 8.03 | 16.31 | 6.22 | 12.75 |
| | | Yes | 5.49 | 10.89 | 7.98 | 16.14 | 5.92 | 12.54 |
| | Codex | No | 5.53 | 10.62 | 8.15 | 16.23 | 6.19 | 13.15 |
| | | Yes | 5.15 | 10.88 | 7.49 | 15.87 | 5.55 | 11.85 |
| | Davinci | No | 5.54 | 10.66 | 8.10 | 15.96 | 6.18 | 12.49 |
| | | Yes | 5.64 | 10.74 | 7.88 | 15.97 | 6.13 | 12.99 |
| | Davinci-002 | No | **6.76** | **11.66** | **10.22** | **18.14** | **8.23** | **15.13** |
| | | Yes | 6.58 | 11.36 | 10.04 | 17.76 | 7.91 | 14.36 |

表 VI：GPT-3 模型根据根本原因提出缓解计划的有效性

| 模型 | 根本原因给定? | BLEU-4 | | ROUGE-L | | METEOR | |
|---|---|---|---|---|---|---|---|
| | | Top1 | Top5 | Top1 | Top5 | Top1 | Top5 |
| Curie | No | 5.92 | 11.29 | 9.46 | 17.76 | 7.34 | 13.35 |
| | Yes | 6.59 | 12.40 | 10.25 | 18.61 | 8.24 | 16.00 |
| Codex | No | 6.25 | 11.23 | 8.94 | 17.62 | 6.46 | 13.00 |
| | Yes | 6.23 | 12.03 | 9.32 | 18.48 | 7.73 | 15.96 |
| Davinci | No | 6.35 | 12.05 | 8.75 | 18.21 | 7.28 | 15.07 |
| | Yes | 7.02 | 11.47 | 9.49 | 18.20 | 8.40 | 16.17 |
| Davinci-002 | No | 6.8 | 12 | 9.48 | 17.37 | 8.15 | 15.53 |
| | Yes | **8.6** | **13.28** | **11.56** | **19.46** | **10.9** | **18.08** |
| | %gain | 26.47 | 10.21 | 21.94 | 12.03 | 33.74 | 16.42 |

在 Code-davinci-002 模型的 Top-1 推荐方面，表 VII 显示，机器识别的事件表现优于人工检测的事件：BLEU-4 高出 9.5%，ROUGE-L 高出 20%，METEOR 高出 23%。这是因为机器检测事件通常遵循某些模式，机器学习模型更容易识别这些模式。

## V. 站在事件负责人的视角审视

### *A.* 方法

从用于评估根本原因和缓解计划的测试集中，我们选取了同时包含根本原因和缓解措施的事件，以便每位事件负责人都能在同一次访谈中评估这两个模型。事件解决是一项复杂的任务，需要充分了解服务的背景和领域知识，也需要了解具体事件。因此，我们邀请了实际分析出事件根本原因并采取缓解措施的负责人参与此次人工评估。我们选取了最近两个月发生的 50 起事件，

表 VII：模型在机器检测事件与人工检测事件上的表现

| 模型 | 机器检测? | BLEU-4 | | ROUGE-L | | METEOR | |
|---|---|---|---|---|---|---|---|
| | | Top1 | Top5 | Top1 | Top5 | Top1 | Top5 |
| Curie | Yes | 5.49 | 10.54 | 8.54 | 16.63 | 6.45 | 13.13 |
| | No | 5.45 | 10.65 | 7.78 | 16.15 | 6.10 | 12.56 |
| Codex | Yes | 5.76 | 10.54 | 9.10 | 16.84 | 6.80 | 13.88 |
| | No | 5.41 | 10.67 | 7.68 | 15.93 | 5.88 | 12.78 |
| Davinci | Yes | 5.56 | 10.51 | 8.49 | 16.17 | 6.34 | 12.59 |
| | No | 5.52 | 10.74 | 7.91 | 15.86 | 5.95 | 12.44 |
| Davinci-002 | Yes | **7.18** | **11.83** | **11.5** | **18.59** | **9.41** | **15.66** |
| | No | 6.56 | 11.57 | 9.58 | 17.92 | 7.65 | 14.87 |
| | %gain | 9.45 | 2.25 | 20.04 | 3.74 | 23.01 | 5.31 |

以评估模型的表现，使事件负责人能够准确回忆起处理特定事件时发生的情况。我们联系了所有事件负责人，其中 25 位作出了回应。每次访谈约持续 20 至 30 分钟。

我们展示了所有待评估模型的输出。对于根本原因和缓解计划，我们分别设置了六组候选项。前四组面向 OpenAI 模型，每组有六个选项（包括“无”）；最后两组分别面向 RoBERTa 和 CodeBERT，各自只有一个候选项。对于 OpenAI 模型，我们请 OCE 选择可能与事件相关的最佳选项。随后，我们请 OCE 按 1 至 5 分为选中的候选项评定正确性和可读性，5 分为最高分。请注意，对于 RoBERTa 和 CodeBERT，我们各只有一个选项。因此，我们只请 OCE 为这些候选项评定正确性和可读性。我们对正确性和可读性的定义如下：

正确性：对于此指标，我们请事件负责人检查模型提供的建议是否有帮助，且相较于实际根本原因/缓解措施是否相关。

可读性：可读性是指读者理解生成文本的难易程度。如果文本语法正确、含义明确且易于理解，那么它就是可读的。请注意，可读的文本不一定正确。

最后，我们请事件负责人给出总体评分（1-5），以表明他们对大语言模型在事件解决中的实用性的看法，并请他们分享对此的想法和意见。

### *B.* 结果

表 VIII 展示了事件负责人给出的正确性和可读性评分。可以看出，对于根本原因和缓解计划，Davinci 和 Code-davinci-002 模型组中的候选项所获均值均高于 Curie 和 Codex 模型组所选候选项（根本原因分别为 2.88 和 2.56，缓解计划分别为 3.04 和 3.16）。所有模型的均值可读性评分介于 2.52 至 4.08 之间。事件负责人对输出的可读性给予了积极评价，并且所有模型的可读性评分均高于正确性评分。我们收到了一些关于如何提升未来可读性的建议（例如，避免使用缩略词，并生成更具体或更有信息量的注释）。

如前所述，基线编码器-解码器模型生成的评论非常笼统，而自动化指标未能识别这一点。我们可以看到，事件负责人给 RoBERTa 和 CodeBERT 模型的正确性评分较低，且多位 OCE 指出，编码器-解码器模型生成的建议十分笼统。尽管 OpenAI 模型的正确性评分介于 2.28 到 3.16 之间，多位 OCE 指出，这些模型建议的根本原因和缓解计划很有帮助。例如，这些模型成功找出了一些难以发现的根本原因：

> **“**我印象非常深刻，因为有一个模型找到了正确的根本原因，而这很难检测到。我们在事后分析阶段找到了它。不过，我有点担心事件网站上的信息可能不够充分。总体而言，我对这些模型的效能印象深刻。**”**

> **“**即使这些建议并非总是正确，也能引导 ***OCE*** 找到真正的根本原因。机器学习模型可以提供方向，也可以给出有价值的建议。**”**

我们还采用了 OpenAI 模型给出的最高分，并报告了正确性和可读性评分的平均值。正确性和可读性评分的均值范围为 3.52 至 4.64（中位数为 3-5），体现了这些模型的整体实力。我们询问了总体评分（1-5），表 IX 显示，事件负责人认为模型的总体贡献很有前景且实用。超过 70% 的事件负责人给模型的建议打了三分或以上。我们发现，至少有一个模型对大多数事件都有效。我们还发现了自动化指标无法提供有价值见解的原因。

凡事都有另一面，我们发现模型的输出对某些事件并无帮助。OCE 对这些事件给出的分数较低，以下是他们提到的一些顾虑：

> **“**仅根据事件数据，模型很难预测根本原因和缓解措施，因为数据库中并未记录所有数据，其中一些数据还被列为机密。**”**

> **“**主要顾虑是，如果建议有误，值班工程师可能需要更长时间来调查问题。**”**

由于缺乏讨论或其他信息，输入中缺少有价值的信号，因此我们发现模型存在一些负样本。不过，模型的整体表现相当可观，可视为未来实现根本原因分析和缓解计划自动化的起点。

### *C.* 两个示例

表 X 展示了两个示例，以说明 GPT-3.x 模型在云事件中生成根本原因和缓解措施计划的有效性。我们并列呈现了 OCE 编写的实际文本和其中一个模型（微调后的 Code-davinci-002）生成的文本。尽管人工编写的文本与生成的文本有所不同，但生成的文本提供了非常相关且有价值的建议，可以帮助 OCE。

表 VIII：事件负责人给出的正确性和可读性评分

| 目标 | 标准 | RoBERTA | | CodeBERT | | Curie | | Codex | | Davinci | | Davinci-002 | | Max OpenAI | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Mean | 中位数 | Mean | 中位数 | Mean | 中位数 | Mean | 中位数 | Mean | 中位数 | Mean | 中位数 | Mean | 中位数 |
| 根本原因 | 正确性 | 1.56 | 1 | 1.72 | 1 | 2.40 | 2 | 2.40 | 2 | 2.88 | **3** | 2.56 | 2 | **3.52** | **3** |
| | 可读性 | 3.56 | **5** | 3.68 | **5** | 3.08 | 4 | 3.52 | 4 | 3.56 | 5 | 3.8 | 4 | **4.52** | **5** |
| 缓解措施 | 正确性 | 1.6 | 1 | 1.52 | 1 | 2.28 | 2 | 2.28 | 1 | 3.04 | 3 | 3.16 | 3 | **4.04** | **4** |
| | 可读性 | 2.88 | 2 | 3.04 | 4 | 2.52 | 2 | 2.8 | 3 | 3.52 | 4 | 4.08 | 4 | **4.64** | **5** |

表 IX：大语言模型在事件解决中的实用性

| 分数 | 事件数量 负责人 | 百分比（%） 总数 |
|---|---|---|
| 5 | 2 | 7.41 |
| 4 | **9** | **33.33** |
| 3 | 8 | 29.63 |
| 2 | 6 | 22.22 |
| 1 | 2 | 7.41 |

这些文本差异也说明了为什么 OCE 认为 GPT-3.x 模型前景可期，而自动评估指标却未能体现这一点。

## VI. 讨论与威胁

### *A.* 自动化指标能反映人类感知吗？

众所周知，自动评估指标能够代表人类感知，因此广泛应用于自然语言翻译等问题 [14], [20], [21]。尽管近期一些研究探讨了这些指标在代码摘要中的有效性，并报告了许多缺陷和不足 [44]–[47]，，研究人员仍在使用它们进行基准测试。自动化指标最好的替代方案是人工验证，或某种形式的自动测试用例评估（用于代码生成任务）。事件管理面临的主要挑战是，如果专家没有参与解决特定事件，即使是专家也难以评估这些事件。有时，OCE 即使在事件发生两个月后，也无法清楚地回忆起相关情况。因此，在这一领域开展大规模研究颇具挑战。不过，我们采访了 25 位事件负责人，发现即使自动化指标得分较低，模型的表现仍相当不错。我们计算了三种词汇指标（即 BLEU-4、ROUGE-L 和 METEOR）与 OCE 给出的正确性和可读性评分之间的皮尔逊系数。我们观察到，系数介于 -0.42 和 +0.62 之间，因此无法从这些数值中归纳出明确的规律。这也表明，对于解决云事件而言，这些自动化指标可能与人类感知并不一致。不过，还需要更多样本案例才能得出任何确切结论。

### *B.* 自然语言还是代码？哪一类模型更适合事件管理？

在选择模型时，我们同时选用了自然语言模型（即，RoBERTa、Curie、Davinci）和代码模型（即，CodeBERT、Codex-cushman、Code-davinci-002），以探究哪一类模型有助于事件管理。我们没有发现这两组模型中有哪一组明显胜出。与其他模型相比，Davinci 和 Code-davinci-002 模型生成的建议正确且易读。需要注意的是这两个模型都拥有 1750 亿个参数。我们将对更大型代码模型进行微调，或使用事件数据从头预训练模型，留待未来研究。

### *C.* 如何提升模型的性能？

我们从事件负责人那里收到了几项建议。最主要的建议是将 OCE 之间的讨论纳入模型。这将引导模型找到更好的建议。我们还舍弃了许多摘要在事件解决时撰写或更新的事件。为了公平地评估模型并防止可能的数据泄漏（如果摘要后来被更新，其中可能会写入根本原因和缓解措施），我们将这些事件从数据集中剔除了。在防止数据泄漏后将它们纳入数据集，可能会提升模型的性能。我们还在清理摘要时丢失了一些关键信息（例如，舍弃图像和表格）。纳入这些信息也可能有所帮助。

### *D.* 有效性威胁

我们的研究存在若干有效性威胁。语义指标的核心使用了预训练模型，而我们在评估中使用的是默认的自然语言模型。使用事件管理文本预训练的模型可能会导致性能评估结果发生一些变化。此外，我们使用组织内部提供的服务来训练和评估模型。如果在其他组织的不同服务集合上进行评估，这些模型可能会表现出意料之外的行为。一些事件负责人担心模型在处理罕见事件时的效能，而微软经常会报告罕见事件。我们研究的另一个有效性威胁是人类受试者研究的样本量。样本如此之小时，很难在正确性和可读性评分上达到统计显著性。不过，根据研究的性质，扩大规模也颇具挑战性。

## VII. 相关工作

### *A.* 事件管理

大型云服务中的事件管理已成为系统与软件工程领域的热门研究课题。该领域以往的工作主要聚焦于两个方向。首先，已有多项实证研究分析生产系统中的事件和服务中断，重点研究由特定类型问题引发的事件 [48]–[51]，或特定服务和系统中的问题 [52]–[54]。其次，与我们的工作更相关的是，利用机器学习和数据驱动技术，自动化事件生命周期的不同环节，例如分诊 [55]–[57]，、诊断 [58]–[60] 以及

表 X：示例　　展示 GPT-3.x 模　　型在事件解决方面效果的

| 示例 | Task | 原文 | 由 DaVinci 模型生成 |
|---|---|---|---|
| I | 根本原因 | 客户发送了大量数据作为有效负载。这些交易的大小以及发送速率导致了最初的网络崩溃。 | 超过账本最大负载大小限制的大型有效负载导致账本崩溃。 |
| | 缓解措施 | 通过代码升级增大负载大小，解决了该问题。 | 以更大的负载大小重新发布了账本 |
| II | 根本原因 | 美国东部发生意外断电，导致多台虚拟机宕机，影响了数据库服务。 | 影响多项服务的数据中心中断 |
| | 缓解措施 | 数据库团队已缓解美国西部区域的上游问题 | 数据库团队通过重启服务缓解了该问题。 |

缓解措施 [5]。与以往工作不同，这是首次尝试利用最先进的语言模型，协助 OCE 解决事件。我们希望这项工作也能推动未来的研究，将传统的、针对特定任务的判别模型与大语言模型结合起来，实现生产环境事件的端到端自动化。

### *B.* 软件工程中的大语言模型

尽管这是首项将大语言模型用于智能运维的研究，但软件工程领域已有多项研究尝试利用大语言模型解决其他具有挑战性的问题。GitHub Copilot 使用 GPT-3 根据自然语言输入自动生成代码 [8]。多位研究人员针对代码生成 [8], [36], 、文档字符串生成 [8], [61], 和代码修复 [62], [63] 等问题开展了研究。Bareiß 等人 [64]展示了少样本学习在以下任务中的有效性：(i) 代码变异；(ii) 根据自然语言文档生成测试预言机；以及 (iii) 测试用例生成。Jain 等人 提出一种方法，通过基于程序分析和综合技术的后处理步骤增强大语言模型，从而取得更好的性能 [65]。然而，与代码生成不同，后者拥有词法和结构信息以及海量训练数据；我们探索的是利用最先进的大语言模型解决事件这一问题，此前尚无人研究。

## VIII. 结论

通过这项工作，我们展示了 GPT-3 和 GPT-3.5 等最先进的大语言模型能够有效帮助进行事件管理，具体而言，可用于识别根本原因和缓解措施。为比较这些模型的有效性，我们在微软针对超过 40,000 起事件开展了严格的大规模研究。为评估该方法的实际实用性，我们邀请了生产环境事件的实际负责人参与其中。我们预计，本文将成为众多利用大语言模型提高事件管理效率的研究中的第一篇。下一步，我们计划将这些模型部署到生产环境中，协助 OCE 进行事件解决。我们还计划探索大语言模型的其他应用场景，例如事件摘要。

## IX. 致谢

我们谨向参与根本原因和缓解措施验证工作的工程师表示感谢。同时，我们也要感谢微软的以下人员所作的贡献：Oleg Losinets、JimJernigan 和 Jim Kleewein。

## 参考文献

[1]S. Wolfe，“亚马逊在 Prime Day 遭遇一小时停机，销售损失可能高达 1 亿美元”，2018年。 [在线]。来源：https://www.businessinsider.com/amazon-prime-day-website-issues-cost-it-millions-in-lost-sales-2018 7 - J. [2] Chen、S. Zhang、X. He、Q. Lin、H. Zhang、D. Hao、Y. Kang、F. Gao、Z. Xu、Y. Dang 等，“这些事件有多偶发？大规模在线服务系统事件的特征分析与优先级排序”，载于《第35届 IEEE/ACM 自动化软件工程国际会议论文集》，2020年，第373–384页。[3] A. Saha 和 S. C. Hoi，“从云服务事件调查中挖掘根本原因知识以实现智能运维”，arXiv 预印本 arXiv:2204 11598，2022年。. [4] J. Chen、X. He、Q. Lin、H. Zhang、D. Hao、F. Gao、Z. Xu、Y. Dang 和 D. Zhang，“大规模在线服务系统的持续事件分诊”，载于《2019年第34届 IEEE/ACM 自动化软件工程国际会议（ASE）》。IEEE，2019年，第364–375页。[5] J. Jiang、W. Lu、J. Chen、Q. Lin、P. Zhao、Y. Kang、H. Zhang、Y. Xiong、F. Gao、Z. Xu 等，“如何缓解事件？面向在线服务系统的有效故障排查指南推荐技术”，载于《第28届 ACM 欧洲软件工程会议与软件工程基础研讨会联合会议论文集》，2020年，第1410–1420页。[6] J. Wei、X. Wang、D. Schuurmans、M. Bosma、E. Chi、Q. Le 和 D. Zhou，“思维链提示能够激发大语言模型的推理能力”，arXiv 预印本 arXiv:2201.11903，2022年。[7] T. Brown、B. Mann、N. Ryder、M. Subbiah、J. D. Kaplan、P. Dhariwal、A. Neelakantan、P. Shyam、G. Sastry、A. Askell 等，“语言模型是少样本学习者”，《神经信息处理系统进展》，第33卷，第1877–1901页，2020年。[8] M. Chen、J. Tworek、H. Jun、Q. Yuan、H. P. d. O. Pinto、J. Kaplan、H. Edwards、Y. Burda、N. Joseph、G. Brockman 等，“对使用代码训练的大语言模型进行评估”，arXiv 预印本 arXiv:2107 03374，2021年。. [9] Z. Chen、Y. Kang、L. Li、X. Zhang、H. Zhang、H. Xu、Y. Zhou、L. Yang、J. Sun、Z. Xu 等，“迈向智能事件管理：为何需要它以及如何实现”，载于《第28届 ACM 欧洲软件工程会议与软件工程基础研讨会联合会议论文集》，2020年，第1487–1497页。[10] “Common Crawl。” [在线]。来源：https://commoncrawl.org/ [11] S. Kulkarni、A. Singh、G. Ramakrishnan 和 S. Chakrabarti，“网页文本中维基百科实体的集体标注”，载于《第15届 ACM SIGKDD知识发现与数据挖掘国际会议论文集》，2009年，第457–466页。[12] “维基百科。” [在线]。来源：https://www.wikipedia.org/ [13] Y. Wang、W. Wang、S. Joty 和 S. C. Hoi，“CodeT5：用于代码理解和生成的标识符感知统一预训练编码器—解码器模型”，arXiv 预印本 arXiv:2109.00859，2021年。[14] A. Vaswani、N. Shazeer、N. Parmar、J. Uszkoreit、L. Jones、A. N. Gomez、Ł. Kaiser 和 I. Polosukhin，“注意力机制即你所需的一切”，载于《神经信息处理系统进展》，2017年，第5998–6008页。[15] S. Hochreiter 和 J. Schmidhuber，“长短期记忆网络”，《神经计算》，第9卷，第8期，第1735–1780页，1997年。[16] J. Chung、C. Gulcehre、K. Cho 和 Y. Bengio，“门控循环神经网络在序列建模中的实证评估”，arXiv 预印本 arXiv:1412.3555，2014年。[17] D. Bahdanau、K. Cho 和 Y. Bengio，“通过联合学习对齐与翻译实现神经机器翻译”，arXiv 预印本 arXiv:1409 0473，2014年。. [18] J. Devlin、M.-W. Chang、K. Lee 和 K. Toutanova，“BERT：用于语言理解的深度双向变换器预训练”，arXiv 预印本 arXiv:1810.04805 ，2018年。

[19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, 和 V. Stoyanov, “RoBERTa：一种稳健优化的 BERT 预训练方法”，arXiv 预印本 arXiv:1907.11692，2019年。[20] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, 和 L. Zettlemoyer, “BART：用于自然语言生成、翻译和理解的去噪序列到序列预训练”，arXiv 预印本 arXiv:1910.13461，2019年。[21] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, 和 P. J. Liu, “探索使用统一的文本到文本转换器进行迁移学习的极限”，arXiv 预印本 arXiv:1910.10683，2019年。[22] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever 等, “通过生成式预训练提升语言理解能力”，2018年。[23] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever 等, “语言模型是无监督的多任务学习者”，OpenAI 博客，第1卷，第8期，第9页，2019年。[24] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin 等, “OPT：开放式预训练变换器语言模型”，arXiv 预印本 arXiv:2205.01068，2022年。[25] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang 等, “CodeBERT：面向编程语言和自然语言的预训练模型”，载于《2020年自然语言处理实证方法会议论文集：研究发现》，2020年，第1536–1547页。[26] D. Guo, S. Ren, S. Lu, Z. Feng, D. Tang, L. Shujie, L. Zhou, N. Duan, A. Svyatkovskiy, S. Fu 等, “GraphCodeBERT：利用数据流预训练代码表示”，载于《国际学习表征会议》，2020年。[27] W. Ahmad, S. Chakraborty, B. Ray, 和 K.-W. Chang, “面向程序理解与生成的统一预训练”，载于《2021年北美计算语言学协会分会：人类语言技术会议论文集》。线上：计算语言学协会，2021年6月，第2655–2668页。 [线上]。网址：https://www.aclweb.org/anthology/ 2021 211 .naacl-main。S. [28] Chakraborty, T. Ahmed, Y. Ding, P. T. Devanbu, 和 B. Ray, “NatGen：通过‘自然化’源代码进行生成式预训练”，载于《第30届ACM联合欧洲软件工程会议暨软件工程基础研讨会论文集》，2022年，第18–30页。[29] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. B. Clement, D. Drain, D. Jiang, D. Tang, G. Li, L. Zhou, L. Shou, L. Zhou, M. Tufano, M. Gong, M. Zhou, N. Duan, N. Sundaresan, S. K. Deng, S. Fu, 和 S. Liu, “CodeXGLUE：用于代码理解与生成的机器学习基准数据集”，CoRR，第 abs/2102 04664卷，2021年。。[30] K. Clark, M.-T. Luong, Q. V. Le, 和 C. D. Manning, “ELECTRA：将文本编码器预训练为判别器而非生成器”，arXiv 预印本 arXiv:2003.10555，2020年。[31] H. Husain, H.-H. Wu, T. Gazit, M. Allamanis, 和 M. Brockschmidt, “CodeSearchNet挑战赛：评估语义代码搜索的现状”，arXiv 预印本 arXiv:1909.09436，2019年。[32] “OpenAI。” [线上]。网址：https://openai.com/ [33] T. Ahmed 和 P. Devanbu, “面向软件工程的多语言训练”，载于《第44届国际软件工程会议论文集》，2022年，第1443–1455页。[34] “CodeXGLUE——代码到文本。” [线上]。网址：https://github.com/microsoft/CodeXGLUE/tree/main/Code-Text/code-to-text[35] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, 和 W. Chen, “LoRA：大语言模型的低秩适配”，arXiv 预印本 arXiv:2106.09685，2021年。[36] F. F. Xu, U. Alon, G. Neubig, 和 V. J. Hellendoorn, “代码大语言模型的系统性评估”，载于《第6届ACM SIGPLAN机器编程国际研讨会论文集》，2022年，第1–10页。[37] C.-Y. Lin 和 F. J. Och, “ORANGE：一种用于评估机器翻译自动评估指标的方法”，载于《COLING 2004：第20届国际计算语言学会议论文集》，2004年，第501–507页。[38] C.-Y. Lin, “ROUGE：自动评估摘要的软件包”，载于《文本摘要新进展》，2004年，第74–81页。[39] D. S. Hirschberg, “最长公共子序列问题的算法”，《ACM学报》（JACM），第24卷，675，1977年4月，第664页。

[40] S. Banerjee 和 A. Lavie, 《Meteor：一种用于机器翻译评估的自动指标，与人工判断的相关性更高》，载于机器翻译和/或摘要内在与外在评估指标 ACL 研讨会论文集，2005年，第 65–72 页。 [41] T. Zhang、V. Kishore、F. Wu、K. Q. Weinberger 和 Y. Artzi, 《BERTScore：使用 BERT 评估文本生成》，arXiv 预印本 arXiv:1904 09675，2019年。.[42] T. Sellam 、 D. Das 和 A. P. Parikh, 《 BLEURT：学习稳健的文本生成指标》，arXiv 预印本 arXiv:2004.04696，2020年。 [43] H. Kane、M. Y. Kocyigit、A. Abdalla、P. Ajanoh 和 M. Coulibali, 《NUBIA：用于文本生成的基于神经网络的可互换性评估器》，2020年。[44] E. Shia、Y. Wangb、L. Dub、J. Chenc、S. Hanb、H. Zhangd、D. Zhangb 和 H. Suna, 《神经代码摘要的评估》，载于第44届国际软件工程会议（ICSE）论文集，2022年。 [45] D. Roy、S. Fakhoury 和 V. Arnaoudova, 《重新评估代码摘要任务的自动评估指标》，载于第29届 ACM 欧洲软件工程会议与软件工程基础研讨会联合会议论文集，2021年，第 1105–1116 页。[46] D. Gros、H. Sezhiyan、P. Devanbu 和 Z. Yu, 《代码到注释的翻译：数据、指标、基线与评估》，载于2020年第35届 IEEE/ACM 自动化软件工程国际会议（ASE）。IEEE，2020年，第 746–757 页。 [47] S. Haque、Z. Eberhart、A. Bansal 和 C. McMillan, 《用于评估源代码摘要的语义相似度指标》，arXiv 预印本 arXiv:2204.01632，2022年。 [48] T. Leesatapornwongsa、J. F. Lukman、S. Lu 和 H. S. Gunawi, 《TaxDC：数据中心分布式系统中非确定性并发错误的分类体系》，载于第二十一届编程语言与操作系统架构支持国际会议论文集，2016年，第 517–530 页。 [49] A. Alquraan、H. Takruri、M. Alfatafta 和 S. Al-Kiswany, 《云系统中 {Network-Partitioning} 故障分析》，载于第13届 USENIX 操作系统设计与实现研讨会（OSDI 18），2018年，第 51–68 页。 [50] Y. Gao、W. Dou、F. Qin、C. Gao、D. Wang、J. Wei、R. Huang、L. Zhou 和 Y. Wu, 《大规模分布式系统中崩溃恢复错误的实证研究》，载于2018年第26届 ACM 欧洲软件工程会议与软件工程基础研讨会联合会议论文集，2018年，第 539–550 页。 [51] Y. Zhang、J. Yang、Z. Jin、U. Sethi、K. Rodrigues、S. Lu 和 D. Yuan, 《理解并检测分布式系统中的软件升级故障》，载于 ACM SIGOPS 第28届操作系统原理研讨会论文集，2021年，第 116–131 页。 [52] S. Ghosh、M. Shetty、C. Bansal 和 S. Nath, 《如何应对生产环境事件？大规模云服务的实证研究》，载于第13届云计算研讨会论文集，2022年，第 126–141 页。[53] H. Liu、S. Lu、M. Musuvathi 和 S. Nath, 《哪些漏洞会导致生产环境中的云事件？》，载于操作系统热点问题研讨会论文集，2019年，第 155–162 页。[54] D. Yuan 、 Y. Luo 、 X. Zhuang 、 G. R. Rodrigues 、 X. Zhao 、 Y. Zhang、P. U. Jain 和 M. Stumm, 《简单测试即可预防大多数严重故障：分布式 {数据密集型} 系统中生产环境故障的分析》，载于第11届 USENIX 操作系统设计与实现研讨会（OSDI 14），2014年，第 249–265 页。 [55] J. Chen、X. He、Q. Lin、Y. Xu、H. Zhang、D. Hao、F. Gao、Z. Xu、Y. Dang 和 D. Zhang, 《在线服务系统事件分诊的实证调查》，载于2019年 IEEE/ACM 第41届国际软件工程会议：软件工程实践（ICSE-SEIP），2019年，第 111–120 页。 [56] J. Chen、X. He、Q. Lin、H. Zhang、D. Hao、F. Gao、Z. Xu、Y. Dang 和 D. Zhang, 《大规模在线服务系统的持续事件分诊》，载于2019年第34届 IEEE/ACM 自动化软件工程国际会议（ASE），2019年，第 364–375 页。 [57] A. P. Azad、S. Ghosh、A. Gupta、H. Kumar、P. Mohapatra、L. Eckstein、L. Posner 和 R. Kern, 《从海底拾取珍珠：从混杂的混合云服务问题分诊协作对话中提取人工制品》，载于 AAAI 人工智能会议论文集，第36卷，第11期，2022年，第 12 440–12 446 页。 [58] V. Nair、A. Raul、S. Khanduja、V. Bahirwani、Q. Shao、S. Sellamanickam、S. Keerthi、S. Herbert 和 S. Dhulipalla, 《学习一种分层

用于检测和诊断服务问题的监控系统》，载于第*21*届 *ACM SIGKDD*知识发现与数据挖掘国际会议，2015年，第 2029–2038 页。 [59] C. Bansal、S. Renganathan、A. Asudani、O. Midy 和 M. Janakiraman，《Decaf：诊断并分诊大规模云服务中的性能问题》，载于*2020*年 *IEEE/ACM* 第*42*届国际软件工程会议：软件工程实践（*ICSE-SEIP*），2020年。[60] C. Luo、J.-G. Lou、Q. Lin、Q. Fu、R. Ding、D. Zhang 和 Z. Wang，《将事件与时间序列关联以进行事件诊断》，载于第*20*届 *ACM SIGKDD*知识发现与数据挖掘国际会议论文集，2014年，第 1583–1592 页。 [61] T. Ahmed 和 P. Devanbu，《针对项目特定代码摘要的少样本大语言模型训练》，载于第*37*届 *IEEE/ACM* 自动化软件工程国际会议，2022年，第 1–5 页。 [62] Z. Fan、X. Gao、A. Roychoudhury 和 S. H. Tan，《通过自动程序修复改进 Codex 自动生成的代码》，*arXiv* 预印本 *arXiv:2205.10583*，2022年。 [63] H. Joshi、J. Cambronero、S. Gulwani、V. Le、I. Radicek 和 G. Verbruggen，《修复几乎就是生成：使用大语言模型进行多语言程序修复》，*arXiv* 预印本 *arXiv:2208.11640*，2022年。 [64] P. Bareiß、B. Souza、M. d’Amorim 和 M. Pradel，《几乎免费的代码生成工具？对代码领域少样本预训练语言模型的研究》，*arXiv* 预印本 *arXiv:2206.01335*，2022年。 [65] N. Jain、S. Vaidyanath、A. Iyer、N. Natarajan、S. Parthasarathy、S. Rajamani 和 R. Sharma，《Jigsaw：大语言模型遇上程序合成》，载于第*44*届国际软件工程会议论文集，2022年，第 1219–1231 页。