解读工业 DevOps 环境中的故障日志

M Abbas[1,2], A Hamayouni[2], M H Moghadam[1],M Saadatmand[1],和 P E Strandberg[3]

[1]瑞典 RISE 研究院，瑞典韦斯特罗斯¨马拉达伦大学，瑞典韦斯特罗斯"Westermo 网络技术有限公司，瑞典韦斯特罗斯

预印本，已被 ITNG 2023 接收

arXiv:2301.03450v1 [cs.SE] 9 Jan 2023

摘要

处理和审查大型工业系统的夜间测试执行失败日志是一项繁琐的工作。此外，在测试执行过程中，多个故障可能有相同的根本原因，因此审查工作可能会造成重复劳动。本文介绍了 LogGrouper 方法，该方法可自动对故障日志进行分组，以辅助根因分析，并支持将每组日志作为一个批次进行处理。LogGrouper 使用先进的自然语言处理和聚类方法，实现有意义的日志分组。我们在工业环境中通过定性和定量两种方式评估了该方法。结果表明，根据我们用于评估聚类质量的两项评估指标（轮廓系数和 Calinski-Harabasz 指数），LogGrouper 得到的分组结果质量良好。定性评估表明，专家认为这些组很有用，并将其视为根因分析和故障分配的初步线索。

关键词：故障聚类、夜间测试、失败嵌入、根因分析、DevOps、测试日志、日志分析、软件测试、词云、自然语言处理

1 引言

大型工业软件系统的增量开发需要持续集成和测试。在不同版本和变体上反复执行回归测试套件，会产生海量测试执行日志，工程师必须对其进行审查 [17] [14]。例如，我们的工业合作伙伴 Westermo Network Technologies AB（Westermo）为各种工业交换机和路由器开发 Westermo 操作系统（WeOS）[1] 。WeOS 包含 Linux 内核、其他自由软件以及专有代码。每项新功能（例如扩展通信协议）会在单独的功能分支中开发，并在合并到主分支之前接受严格的质量保证。除了手动测试和调试之外，WeOS 还会每晚在多个测试系统中进行测试，这些测试系统由构成不同网络拓扑的设备组成。由此会产生海量测试执行日志——其中往往有重复项，或与同一批次中的其他日志非常相似——需要进行审查。通常，工程师会审查测试执行过程中出现的严重警告和故障。这给开发人员和测试工程师带来了大量额外工作，因为大多数故障日志都是重复的，或有着共同的原因。在大多数情况下，工程师可能会漏掉长列表末尾的各类故障，因为他们可能会以为大多数日志都相同。

故障日志包含文本信息，可以根据相似性使用聚类方法对其进行分组。聚类及其他自然语言处理（NLP）方法可用于协助工程师开展夜间日志审查。故障聚类有助于DevOps环境中的日志处理和根因分析，甚至可以协助测试经理进行故障分配。近期研究尝试寻找改进特征提取方法的途径，并提高日志表示准确率，从而生成可接受的簇/组。该领域的多项研究采用了基于自然语言处理的方法，例如词频-逆文档频率（tfidf）、支持向量机模型 [14], 关键词提取 [16], 词嵌入 [6], 以及潜在语义分析（LSA）技术 [13]。文献表明，日志聚类会直接影响故障日志分析的速度和性能（例如， [14]）。然而，上述大多数研究都是从定量角度考察故障聚类，很少涉及工业环境。

贡献。 我们报告了一种名为 LogGrouper 的方法，该方法结合最先进的自然语言处理技术与多种聚类算法，对故障日志进行分组。LogGrouper 旨在通过全面呈现所发生故障的类型（类别），帮助工程师审查夜间故障日志。此外，我们还评估了 LogGrouper 支持的不同聚类方法——

[1]WeOS，在线获取：https://www.westermo.com/solutions/weos

在工业环境中，基于聚类质量指标评估其故障日志聚类效果。最终版，我们还利用在 Westermo 与 11 位专家开展的焦点小组研究中获得的定性数据，对 LogGrouper 方法进行了评估。结果表明，LogGrouper 方法生成了有意义的故障组，其中基于密度的含噪声应用空间聚类（DBSCAN）算法在聚类质量方面表现最佳。定性结果表明，LogGrouper 的故障组很有用，可作为根因分析和故障分配的初步线索。

结构。本文其余部分安排如下：第 2 节简要介绍背景和相关工作。第 3 节介绍本文的方法。第 4 节介绍评估方法，第 5 节展示并讨论获得的定性和定量结果。第 6 节介绍可能影响有效性的因素及局限性。最后，第 7 节总结全文并展望未来研究方向。

2 相关工作

许多研究聚焦于故障聚类和日志分析。以下各节将讨论这一领域近期的几项相关工作。

利用机器学习进行日志嵌入和分析，是辅助日志分组的常见方法。在这方面，Sharp 等人 [26] 使用词袋模型和 Word2vec，提出了一种面向日志的数据驱动标签方法。他们还会在为数据分配标签之前对其进行清理和准备。除了将数据聚类为不同的组之外，该方法还能识别反复出现的长期日志。Xiao 等人 [31] 使用 word2vec 表示文本日志，并设计了一种名为基于向量化的日志解析器（LP）的日志挖掘方法。该研究主要关注离线与在线日志解析。在离线日志解析中，利用提取的日志向量计算两条日志之间的相似度；而在在线解析中，该方法使用平均向量作为日志模板，以检测传入日志消息之间的相似度。

Eljasik 等人 提出了日志分析机器学习器（LAMaLearner）系统，用于对缺乏标准信息的非结构化文本日志文件进行聚类 [14]。他们采用多种人工智能（AI）方法，寻找合适的日志特征表示方式，例如将支持向量机（SVM）模型与词频-逆文档频率向量相结合。Aussel 等人 [5]和 Bertero 等人 [6]也依靠自然语言处理从非结构化日志文件中提取特征，以便为日志分析、挖掘、异常检测和分类等方法提供适当的输入。这些研究使用各种数据集和日志，分析基于自然语言处理的特征提取策略对日志分析的影响。

Itkin 等人 [19] 介绍了一种实用方法，用于识别、分类并清理结算系统的故障日志。这项研究采用了多种数据挖掘方法，例如通过暴力比较进行层次凝聚聚类，以及基于因式分解 tf-idf 词嵌入的 K-Means，以构建日志分析框架。Fu 等人 [16] 提出使用关键词提取对日志纯文本进行聚类并检测异常。在这项研究中，通过预定义规则消除日志消息中的可变部分，这意味着所提出的方法需要掌握有关输入数据集的一些基本信息。Lin等人 [21] 提出了一种用于聚类和识别日志出现模式的日志划分策略，称为 LogCluster。他们的结果表明，使用 LogCluster 后，日志检查过程变得更快，而且 LogCluster 能够在检查过程中发现此前未曾发现的日志。

文献中也有少量关于日志聚类与分析的次级研究。He 等人 [18] 回顾了近期有关日志解析器的研究，并比较和分析了这些解析器在处理大量输入日志时的性能。他们得出结论，现有日志解析器不适用于超大型数据集，而且大多数解析器缺乏必要的执行时间效率。Korzeniowski等人 [20]对日志分析研究开展了系统映射研究。他们以自动化日志分析全景的形式展示了研究发现，界定了各个研究领域，并重点介绍了近期的研究趋势。

现有研究主要从定量角度探讨故障聚类，且很少在工业环境中开展研究。本文则在工业环境中从定性和定量角度探讨这一问题。

3 LogGrouper——方法

本节介绍 LogGrouper 方法，该方法用于在DevOps环境中对故障日志进行分组。如图 1 所示，LogGrouper 包含三个不同的步骤，并将其封装为表述性状态转移 (REST) 应用程序编程接口 (API)。首先，LogGrouper 从日志存储库中筛选出日志文件的相关部分（错误和严重警告），并将其用作输入。随后，故障日志会经过预处理流水线——图 1 中的步骤 A ——对故障日志内容进行清理，以便用于后续步骤。然后，将清理后的故障日志文本转换为数值特征向量——图 1 中的步骤 B ——以用于聚类。在对故障日志进行聚类之前，我们根据从故障日志中推断出的特征向量，计算最佳参数，例如通过肘部法 为 K均值聚类确定簇的数量 [11]。然后，LogGrouper 在步骤 C 中将故障日志（包括经过预处理和未经预处理的日志）聚为多个簇（组），如图 1 所示。接下来，LogGrouper 针对每种向量化方法评估多种聚类方法，并根据簇的质量输出效果最佳的聚类算法的结果。然而，终端用户可能不容易理解这些簇。因此，LogGrouper 使用关键词短语提取

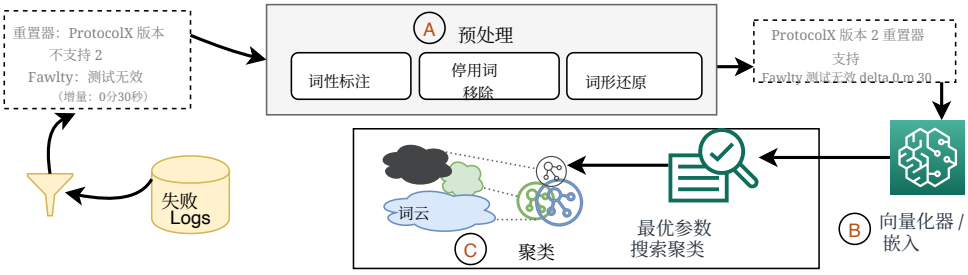


图 1：用于对回归测试故障日志进行分组的 LogGrouper 方法

算法从每组故障日志中提取关键短语，并以词云的形式呈现。下面将详细介绍 LogGrouper 的每个步骤。

A. 预处理： 日志消息具有噪声的特性，这可能会使后续的自然语言处理任务变得复杂。日志文件包含多种对日志聚类无用的属性。作为预处理的第一步，我们将日志消息文本转换为小写，并移除时间戳、符号和缩进。为避免将相同术语以不同语言形式解读，我们对日志文件中的每个词元进行词形还原。这有助于后续任务中进行有意义的相似度计算。图 1 中虚线边框文本框展示了预处理前后的日志消息示例片段。

B. 特征向量提取： 文本数据的特征向量提取通常通过将文本映射为向量空间模型中的向量表示来完成。具体做法是先从数据中学习一种语言模型表示，再推断出日志消息的特征向量。因此，这类方法也称为语言模型、表示学习、向量化或词嵌入方法。然而，随着迁移学习的兴起，大型预训练语言模型逐渐应用于软件工程任务。在这方面，LogGrouper 将多种不同的向量化方法与预训练语言模型结合起来，对日志消息进行向量化，以便对日志文件进行分组/聚类。目前，LogGrouper 支持信息检索（IR）模型词频-逆文档频率、基于机器学习的 FastText [7], 以及基于深度学习的基于变换器的双向编码器表示（BERT） [12]。

用于信息检索（*IR*）的特征向量降维：
LogGrouper采用的信息检索方法基于提取词项-文档矩阵，并将tfidf分数作为特征值。从矩阵中提取的向量会形成包含大量零值的稀疏向量。因此，LogGrouper会对向量进行归一化，并采用广为人知的主成分分析（PCA）方法进行降维 [4]。由此得到的向量维度合理。

C. 聚类及其可视化：我们考虑了三类不同算法中的四种广泛使用的聚类算法，用于对故障日志进行分组，具体如下。凝聚式聚类是一种自底向上的层次聚类技术——它首先将每个项目视为一个独立的簇，然后反复找出最相似的两个簇并将其合并，直至达到预先设定的簇数量。K-均值是一种基于划分（以质心为基础）的聚类算法，它从一组（K 个）随机选取的质心开始，这些质心作为簇的起始点。随后，它反复计算数据点与簇质心之间距离的平方和，将每个项目分配给最近的质心，并计算每个簇的新质心，直至质心收敛。基于密度的带噪声应用空间聚类（*DB-SCAN*）根据项目之间的距离，将彼此接近的项目围绕一个核心点聚集成簇。谱聚类是一种先使用拉普拉斯矩阵降低输入向量维度，再进行实际聚类的技术，之后使用其他算法进行聚类。

LogGrouper使用上述算法进行聚类，并采用不同策略为每种聚类算法寻找最佳参数。在K-Means、凝聚聚类和谱聚类中，使用肘部法 [11]绘制平方距离和随k 变化的曲线，并根据Calinski-Harabasz指数和轮廓系数（见第4节）选择曲线的肘部作为最佳k 值。对于DBSCAN，需要计算两个重要参数：Epsilon（eps），即将其中一个项视为另一个项的邻居时两者之间的最大距离；以及最小样本数，即构成一个簇所需的最少项数。LogGrouper根据文献建议 [25] 设置最小样本数，并使用贪心搜索计算Epsilon。聚类完成后，该方法根据Calinski-Harabasz指数和轮廓系数输出表现最佳的聚类算法的结果。不过，也可以获取所有其他聚类流程的结果。使用快速应用关键词提取（RAKE）算法 [22]，分别提取所有簇中的关键短语。随后将关键短语以词云的形式进行可视化。

4 评估

A. 背景与研究问题：本研究可归类为一项探索性案例研究，重点是改进 Westermo 的日志处理。总体目标

是辅助分析夜间测试执行日志的根因，并帮助批量处理故障日志。LogGrouper 是实现这一总体目标的一个步骤。为评估 LogGrouper 方法，我们探讨以下研究问题（RQs）。

RQ1： 从评估指标来看，哪些聚类方法能够对失败日志消息进行高质量分组?

由于分组质量在很大程度上取决于所采用的聚类方法，我们旨在结合数据集的具体情况，根据评估指标，确定有无预处理时的最佳聚类方法。需要注意的是，评估向量化方法的质量不在本研究范围之内，将留待未来研究。

研究问题 ***2***： 工程师对 LogGrouper 的结果有何看法?

在这个研究问题中，我们旨在收集定性洞见，了解工程师对 LogGrouper 结果的看法、他们对分组结果正确性的判断，以及该方法及其采用过程中面临的挑战。

B. 数据收集： Westermo 提供了为期两个月的夜间测试结果数据，用于设计和评估该方法。数据包括测试结果数据库、夜间测试生成的日志文件，以及一个用于浏览和查询测试结果的工具 [28]。为在所研究的工业环境中评估该方法，我们使用了三个时间窗口内被标记为错误和关键的日志消息。这些时间窗口是工程师在日常工作中进行日志审查时通常会采用的，具体如下：第一个时间窗口涵盖一次夜间测试会话中一个或多个代码分支；第二个时间窗口涵盖一个或多个代码分支；最后，第三个时间窗口涵盖所有可用的日志文件。这模拟了典型的夜间测试执行日志审查流程。此外，我们还使用生成的分组结果、关键短语和词云对该方法进行定性评估。

C. 指标：由于无法获得分组结果的真实标注，我们采用为无标签数据的无监督聚类方法质量评估而设计的标准指标，具体如下。

轮廓系数（*SC*）： [23] 是一个介于 -1 和 1 之间的值，表示所得簇之间的分离程度。理想情况下，轮廓系数越接近 1，聚类质量就越高。轮廓分数通过平均值计算

簇内距离与每个样本到最近簇的平均距离。

Calinski-Harabasz 指数（*CH*）： [10] 是一种用于评估缺少真实标签的聚类技术的准则（也称为方差比准则、CH 得分）。CH 得分表示某个对象与其所在簇的相似程度，相对于它与其他簇的相似程度。其计算方法是比较对象与其所在簇质心之间的距离，以及簇质心与所有数据点质心之间的距离。CH 指数值越高，

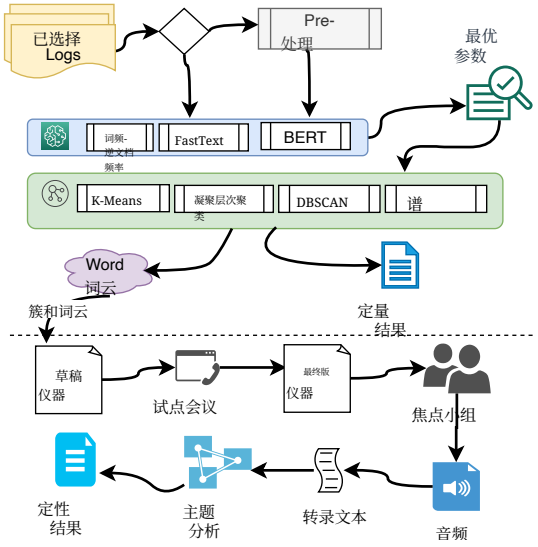


图 2：评估流程

簇越纯，簇之间的区分度越高，因此质量也越高。

D. 流程：整体流程如图 2 所示。如数据收集部分所述，我们考察了 Westermo 三个时间窗口的日志，并将其作为向量化的输入，分别采用预处理和未预处理的数据。在向量化过程中，我们使用以下语言模型及其特定设置。

- 词频-逆文档频率采用 1 至 3 的 n元语法范围，并通过 PCA 进行降维
- 在 Wikipedia 的一百万个词上以子词级别训练的预训练 FastText 模型
- 预训练的 BERT-base无大小写模型，具有 12 层、768 个隐藏单元、12 个注意力头和 110M 个参数

对于每组向量，我们使用 K均值、层次凝聚式聚类、DBSCAN 和谱聚类，并通过肘部法和贪心搜索计算最佳参数。聚类后，我们计算具体的评估指标并进行归一化，以便更好地报告结果。我们提取所有簇中的关键短语，然后将其可视化为词云。

我们计划开展一项焦点小组研究，以收集有关结果的定性见解，遵循 Breen等人提出的指南 [9] ，并结合我们在行业中开展和报告焦点小组的经验 [1–3]（见图 2 下半部分）。我们起草了一份仪器，随后邀请公司的一位专家参与试点会议，对其进行验证和完善。该仪器包含七个问题，涉及对结果正确性的感知、采用 LogGrouper 所面临的挑战及其益处。我们在实际会议前一天将部分结果发送给参与者审阅，其中包含三个时间窗口的簇、各簇元素的 ID、关键短语和词云。向参与者发送结果文档的目的在于

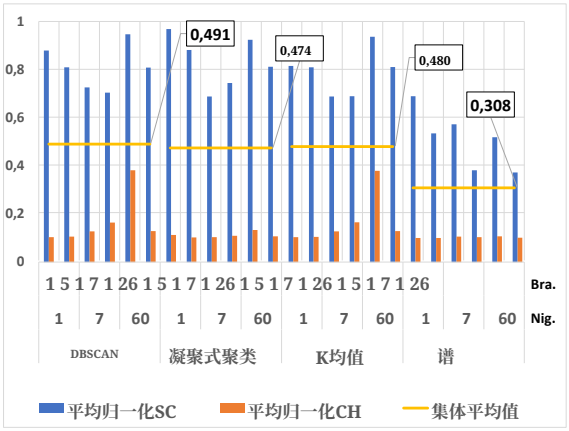


图3：归一化平均结果的整体视图

并非评估任何特定指标，而是引发关于总体正确性感知的讨论。实际会议由公司十一位专家参加，其中包括本文的一位作者。我们选择这些专家，是为了确保其角色和经验具有多样性。在一场简短的介绍中，我们说明了研究目的，并演示了 LogGrouper API，随后开展了实际会议。征得专家同意后，我们录制了会议，并在匿名处理机密数据的同时将其转录为文本 [27]。我们遵循 Braun 和 Clarke 提出的指南 [8], , 对转录文本进行了主题分析，并由本文三位作者进行验证。主题分析完成后，出于保密考虑，我们删除了音频文件。最终形成的主题和子主题构成了定性结果，详见第 5 节。

5 结果与讨论

5.1 定量结果（RQ1）

在本小节中，我们呈现并讨论定量结果。我们针对时间窗口、向量化方法和聚类算法的每种独特组合计算评估指标。不过，受篇幅限制，我们仅在表1中以一个代码分支的一周时间窗口为例报告结果。尽管如此，我们还是在图3中全面展示了所有组合的平均汇总结果，包括经过预处理和未经预处理的输入。此外，我们还将所有补充结果整理在电子表格中，供感兴趣的读者查阅 [2]。

表1展示了在一个代码分支上进行一周故障日志聚类的结果。ANSC 列显示了不同向量化方法下 SC 的平均归一化得分。此外，ANCH 列显示了不同向量化方法下 CH 得分的归一化平均值。以“p”开头的向量化方法表示使用了预处理。从表1的结果可以看出，对于一个代码分支的一周数据，凝聚聚类方法在归一化平均 SC 和 CH 得分方面似乎表现最佳。不过，K-Means 和 DBSCAN 的表现也紧随其后。在不同的时间窗口和代码分支中，我们发现，K均值、DBSCAN 和凝聚式聚类在归一化 SC 和 CH 指标方面的表现差距要小得多。相比之下，在我们的数据集中，谱聚类方法在所有时间窗口中的表现都最差，与其他方法有显著差距。此外，SC 和 CH 指标的平均结果都表明，预处理会导致聚类质量较低。大多数日志消息通常较短，采用更正式的语言并遵循一定的结构。因此，我们观察到，与对原始日志消息进行聚类相比，预处理可能会导致分组质量降低。在需求工程等其他领域中也观察到了词汇有限这一现象 [15]。我们的定性评估（见下文小节）表明，专家更倾向于经过预处理的组摘要，而不是未去除停用词的摘要。尽管如此，在我们的场景中，平均而言，将经过预处理的输入用于聚类方法会降低聚类质量（以 SC 和 CH 衡量）。我们认为，可以在实际聚类完成后进行预处理，以词云和关键短语的形式呈现摘要。

我们还在图3中全面展示了三个时间窗口内评估指标的总体平均值。在图3中，横轴表示各时间窗口中可用代码分支的数量（图3中的 Bra. 行），而 Nig. 行则表示夜晚数量。由图3可见，平均而言（黄色线），对于 LogGrouper流水线，DBSCAN聚类方法的归一化SC和CH表现更好。此外，与表1中的每周结果类似，凝聚式聚类和KMeans聚类方法的表现与DBSCAN聚类方法非常接近。不过，在我们的具体情境中，谱方法的结果最差。

基于我们的定量结果，我们将对 RQ1 的回答总结如下。

> ***RQ1*** 摘要：平均而言，DBSCAN 聚类方法在分组故障日志方面表现最佳，归一化 SC 和 CH 得分的综合平均值为 0,491。然而，根据时间窗口和代码分支数量的不同，凝聚式聚类和 K均值在某些情况下也可能优于 DBSCAN。在本研究中，就我们的评估指标而言，谱聚类方法的表现最差。预处理似乎对故障聚类的质量产生了负面影响。

[2]复现包：https://doi.org/10.5281/zenodo. 6405969

表1：一个分支在一周内所选算法的聚类质量对比 ch

聚类	向量化器	SC	CH	ANSC	ANCH
DBSCAN	BERT	0,915	2481,469	0,909	0,133
	pBERT	0,905	2296,911		
	FastText	0,798	1766,108		
	pFastText	0,811	1607,540		
	词频-逆文档频率	0,799	815,786		
	pTFIDF	0,687	543,144		
聚类-式的	BERT	0,867	42439,533	**0,939**	**0,381**
	pBERT	0,850	32037,364		
	FastText	0,848	1535,555		
	pFastText	0,811	1607,540		
	词频-逆文档频率	0,884	1503,334		
	pTFIDF	0,867	1419,293		
K均值	BERT	0,857	42268,074	0,925	0,380
	pBERT	0,850	32037,364		
	FastText	0,845	1500,804		
	pFastText	0,728	1402,890		
	词频-逆文档频率	0,884	1503,334		
	pTFIDF	0,867	1419,293		
谱	BERT	-0,133	506,165	0,366	0,106
	pBERT	0,077	617,049		
	FastText	-0,060	31,014		
	pFastText	0,388	197,845		
	词频-逆文档频率	0,607	621,170		
	pTFIDF	0,199	66,624		

表2：从转录文本中提取的主题

主题	子主题和编码
1. 益处	1.1 提供概览，以便快速响应故障分配 1.2 突显原本可能被遗漏的各类故障 1.3 有助于指出原因并改进日志记录 1.4 协助分析，以检测间歇性故障
2. 正确性的感知	2.1 提供至关重要的分组，其重要性高于呈现方式 2.2 假阳性可能会降低对 LogGrouper 的信任
3. 挑战与改进	3.1 与其他制品（如提交和代码）的关联 3.2 分组和参数的历史信息 面向终端用户的控制 3.3 工具的集成与维护

5.2 定性结果（研究问题 2）

主题分析完成后，我们对提取出的主题和子主题进行了审阅、逻辑上的重新排序，并在表2中进行了报告。在本小节中，我们将呈现并讨论定性结果，同时提供支持性引文——这些引文根据转录文本进行了改写，以提高可读性；转录文本以“斜体文本”呈现，以示强调。我们还将本小节的文本与表2中以粗体显示的子主题关联起来，如（**子主题编号**）所示。

5.2.1 感知收益

焦点小组中的专家一致认为，LogGrouper 方法的结果可以提供故障的高层概览，有助于改进日志记录，并为查找根本原因提供方向。

(1.1). 人们还一致认为，这样的故障概览有助于快速响应故障，并能帮助开发人员了解其代码更改对整个系统造成了哪些影响。然而，开发人员认为，当他们只在单个分支上工作时，这种方法并无益处。“作为一名软件开发人员，确实，这更像是一种通用指导..... [它] 可以快速提示我了解自己的提交造成了什么问题.....”另一方面，测试团队认为，这些结果有助于将故障分配给开发人员。高层次的故障分组被认为非常有用，可以作为识别可能负责调查故障的团队的依据。此外，这种分组也被认为有助于识别相似故障，并将它们分配给同一个团队。“这种分组让我省去了查看大量日志文件的工夫.....问题出在测试框架还是测试系统?我是说，首先要弄清楚应该由哪个团队来查看这个故障.....有时，如果有多个测试失败，你正在考虑 [分配] 其中一个，突然有人又问你能不能看看其他类似的故障；但如果我有这些簇，就只需分配一次。”

(1.2). 通常，夜间故障日志可能包含大量日志消息，专家可能会跳过列表末尾的几条日志。不过，将故障聚类为数量有限的组，有助于凸显多种不同的故障，并促使工程师查看多个组。参与研究的环境中的专家也认同，LogGrouper 有助于凸显不同的故障组，进而避免漏看末尾的日志。“早上如果有三十条新日志要看，其中二十九条都一样，我可能会先看前三条，然后跳过其余的；这样就会漏掉那条不同的日志。 [LogGrouper 可能]很有用，因为我可以直接查看这些簇，这样就不会漏掉最后一条了。”

(1.3). 此外，专家还认为，这些分组结果可以作为开展常见原因/根本原因分析的初步指引。不过，对于根因分析而言，实际日志比其分组结果或代表这些组的词云更为重要。“通过那个词云，我们直接了解了错误是什么；然而，要找到根本原因，可能需要查看实际日志，以重现该错误。”专家还强调，日志的质量取决于其自身的质量。日志消息的质量越高，分组结果就可能越有意义。因此，在工业环境中对故障进行聚类可能会促使日志记录得到改进。“日志的质量取决于其自身。如果我们没有在日志中打印高质量的日志或恰当的短语，就不会对聚类产生任何 [好的作用]。这也可能很好地表明，你的日志并不充分。”

(1.4). 最后，专家还强调，LogGrouper 可能有助于检测间歇性故障。不过，为此，LogGrouper 必须保留有关分组结果的历史信息。专家认为，随着时间推移，故障组可能与发现间歇性故障相关联。“我们仍然存在间歇性故障问题。因此，这些簇如何随时间推移运作，是一个非常值得探讨的问题。假设我们有一个不太常出现的簇，也许它并不是 [间歇性的]。所以，我认为这可能是该方法非常有用的一个方面。”

5.2.2 感知到的正确性

(2.1). 上述讨论的益处应以正确性为核心加以实现。因此，我们询问专家对 LogGrouper 生成结果正确性的看法。专家一致认为，LogGrouper 初始生成的分组比组的可视化更重要。不过，一位专家指出，组的词云呈现可以直接为测试提供初步线索。“如果我们看这里的短语，它实际上是对 [通信协议功能], 的概述，我马上就明白这是哪个测试用例了.....”

(2.2). 不过，专家指出，分组中的假阳性和模糊性可能会降低人们对 LogGrouper 的信任。对此，专家还提出了几项改进建议（参见“挑战与改进”主题）。当被问及更倾向于模糊还是严格的词法分组时，专家表示，对于一晚的故障日志，他们更倾向于严格的词法分组；而对于较长的时间窗口，他们则更倾向于细致一些的语义分组。“假阳性可能会导致我们花更多时间处理某件事，而如果没有被带偏，我们本可以更快地弄清楚。（.....）我认为，推广应用面临的最大问题是信任.....随着时间推移，我们需要更模糊的分组，但对于一晚的日志，则需要严格分组。”

5.2.3 挑战、局限性及未来改进

在诸多其他挑战中，我们在此报告了讨论最多的挑战，以及专家为 LogGrouper 提出的潜在改进建议。

(3.1). 仅凭日志分组结果并不够具体；若能关联 DevOps 环境中的其他制品，就可以改进结果，从而支持更详尽的根因分析。例如，专家建议提取源代码中导致某个组内大多数故障的高频位置，并将其关联到该组。专家认为，这将为开发人员和测试人员开展后续根因分析和故障复现提供指引。“[从开发人员的角度来看] ，你总是会想查看源代码；如果能从这里链接到源代码，那就太好了。”此外，专家还提出一个值得关注的建议：将故障组关联到提交。然而，这可能不像 Westermo 的测试选择过程那么简单 [30]，因为该过程可能会跳过许多本可触发近期提交中故障的测试用例；之后，在该提交合并到分支后，新一轮测试选择才可能触发该故障。这会使确定可能导致故障的提交变得复杂，因为在实际导致故障的提交与故障被触发的时间之间，可能存在多个提交。尽管如此，仍可将可能相关的提交列表关联到故障组，为根因分析提供有用的方向。“我们希望将原因关联到提交，因为我们在分析并查找根本原因时，实际就是这样做的。”

(3.2). 当前版本的 LogGrouper 不会保留任何分组历史信息，也忽略了故障日志的时效性。然而，历史分组信息有助于检测间歇性故障，也可能促成故障与解决方案对提取。尽管如此，Westermo 已经有一种检测间歇性故障的解决方案 [29], ，但通过补充故障分组历史，可以进一步完善该方案。此外，所研究系统的分布式特性以及日志文件的时效性，也使得有必要对故障日志及其所属组进行额外的时间序列可视化。专家认为，可以通过为分组补充分组历史记录以及对故障进行时间序列可视化，来改进 LogGrouper。“这里没有历史记录，但 LogGrouper 中的历史记录对间歇性故障会非常有用 [.....]历史信息可以帮助 [更好地理解，随着时间推移，哪些组可能确实是很好的组]

。”

（**3.3**）。然而，LogGrouper 能否被采用，很大程度上取决于它集成到哪里。公司拥有许多用于测试结果可视化和 DevOps 运维的工具。集成 LogGrouper API 的工具可能会影响它在所研究场景中的潜在用途。此外，完成集成后，还必须维护该工具。因此，必须有一个团队负责 LogGrouper API，并维护 LogGrouper 的运行环境和源代码。“[至于采用]，这取决于我们如何集成 [LogGrouper]。[它会是一个]独立工具，还是集成到我们自己的工具中？我认为这会影响我们的使用程度。 [关于集成]，我们会以某种方式保留源代码或获得访问权限，然后由公司内部的某个团队负责（.....）。我们需要指定负责人，否则它就只是放在那里，却没人 [会使用和维护它。] ”

基于定性结果，我们总结了对研究问题 2 的回答如下。

> 研究问题 ***2*** 总结：在行业 DevOps环境中，故障日志分组有助于故障分配，并可为根本原因分析提供初步线索。人们认为 LogGrouper 生成的分组结果很有用。然而，历史信息、将故障组与提交关联起来以及工具集成，都是具有挑战性的限制因素；若要推动该工具在行业中的应用，就必须改进这些方面。

6 有效性威胁

本节根据 Runeson 和 Hoest [24] 提出的分类，介绍相关的有效性威胁。

构念效度 威胁与研究人员对所研究现象的理解有关。在这方面，我们针对聚类质量表现最佳的聚类算法，对 LogGrouper 进行了定量评估。然而，结果也取决于作为聚类方法输入的表示向量。尽管如此，我们认为，将向量化方法与聚类算法结合起来进行评估会很有意义，但这需要获取有关分组的真实标注数据，因此可作为未来的研究方向。

内部效度威胁与我们结果的可信度有关。为减轻潜在的内部效度威胁，我们采用了这些算法的标准开源实现，并遵循了评估设计的标准指南。焦点小组仪器也在试点会议中得到了验证。此外，我们在研究设计中纳入了背景多元的作者和行业专家。

外部效度威胁关系到研究结果的普遍适用性。本研究是在一家开发路由器和交换机操作系统的瑞典公司内进行的。我们认为，仅研究一家公司可能无法构成良好的样本。不过，我们认为，拥有大型回归测试套件、会产生大量夜间日志的公司仍可从研究结果中受益。尽管如此，我们并未声称研究结果可推广到所研究情境之外。

7 结论与未来方向

本文介绍了 LogGrouper 方法，用于在工业 DevOps 环境中进行故障分组。在我们的研究背景下，结果表明，平均而言，基于密度的聚类方法 DBSCAN 在分组质量方面表现更好。不过，K均值和凝聚聚类方法的表现也紧随 DBSCAN。此外，预处理，尤其是停用词移除，会对聚类质量产生负面影响。定性结果表明，LogGrouper 生成的故障组很有用，也可作为根因分析和故障分配的起点。此外，故障分组有助于凸显工程师可能忽略的多种故障组。

然而，若干挑战和局限可能会阻碍 LogGrouper 在行业中的应用。未来，我们计划应对其中一些挑战，并拓展 LogGrouper 方法，用于故障与解决方案对提取等。此外，我们还计划探索利用历史聚类信息来检测间歇性故障。

致谢。 本研究部分由 AIDOaRt (KDT) 和 SmartDelta (ITEA) 项目资助。

参考资料

[1] M. Abbas、A. Ferrari、A. Shatnawi、E. Enoiu、M. Saadat- mand 和 D. Sundmark。关于相似需求与相似软件之间关系的研究。需求工程，第 1–25 页，2022年。[2] M. Abbas、R. Jongeling、C. Lindskog、E. P. Enoiu、M. Saa- datmand 和 D. Sundmark。工业中的产品线采用：来自铁路领域的经验报告。载于第 *24* 届 *ACM* 系统与软件产品线会议论文集：*A* 卷，SPLC ’20，美国纽约州纽约，2020年。ACM。[3] M. Abbas、M. Saadatmand、E. Enoiu、D. Sundamark 和 C. Lindskog。基于自然语言需求的产品线资产自动化复用推荐。载于 S. Ben Sassi、S. Ducasse 和 H. Mili 编，《style id='8'>新兴软件工程实践中的复用，第 173–189 页，Cham，2020年。Springer International Publishing。[4] H. Abdi 和 L. J. Williams。主成分分析。*Wiley*跨学科评论：计算统计学，2(4):433–459，2010年。

[5] N. Aussel、Y. Petetin 和 S. Chabridon。通过基于自然语言处理的日志解析，提高用于异常预测的日志挖掘性能。载于*2018*年 *IEEE* 第 *26* 届计算机与电信系统建模、分析与仿真国际研讨会（*MASCOTS*），第 237–243 页。IEEE，2018年。

[6] C. Bertero、M. Roy、C. Sauvanaud 和 G. Tr´edan。经验报告：使用自然语言处理进行日志挖掘并将其应用于异常检测。载于*2017*年*IEEE*第*28*届软件可靠性工程国际研讨会（*ISSRE*），第351–360页。IEEE，2017年。[7] P. Bojanowski、E. Grave、A. Joulin 和 T. Mikolov。利用子词信息丰富词向量。计算语言学协会汇刊，第5卷146页，2017年135：–。[8] V. Braun 和 V. Clarke。心理学中的主题分析方法。《心理学定性研究》，3(2):77–101，2006年。[9] R. L. Breen。焦点小组研究实用指南。《高等教育地理学杂志》，30 475，2006年3 463( ): –。[10] T. Cali´nski 和 J. Harabasz。用于聚类分析的树突法。《统计学通讯：理论与方法》，3(1):1–27，1974年。[11] P. Dangeti。机器学习统计学。Packt Publishing Ltd，2017年。[12] 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。[13] N. DiGiuseppe 和 J. A. Jones。基于概念的故障聚类。载于*ACM SIGSOFT*第*20*届软件工程基础国际研讨会论文集，第1–4页，2012年。[14] T. Eljasik-Swoboda 和 W. Demuth。利用聚类和自然语言处理克服日志管理中的多样性问题。载于*ICAART*（*2*），第281–288页，2020年。[15] A. Ferrari、G. O. Spagnolo 和 S. Gnesi。PURE：公共需求文档数据集。载于*2017*年*IEEE*第*25*届国际需求工程会议（*RE*），第502–505页，2017年。[16] Q. Fu、J.-G. Lou、Y. Wang 和 J. Li。通过非结构化日志分析检测分布式系统中的执行异常。载于*2009*年第九届*IEEE*国际数据挖掘会议，第149–158页。IEEE，2009年。[17] M. Golagha、C. Lehnhoff、A. Pretschner 和 H. Ilmberger。无需覆盖率的故障聚类。载于第*28*届*ACM SIGSOFT*软件测试与分析国际研讨会论文集，第134–145页，2019年。[18] P. He、J. Zhu、S. He、J. Li 和 M. R. Lyu。日志解析及其在日志挖掘中的应用评估研究。载于*2016*年第*46*届*IEEE/IFIP*国际可信系统与网络会议（*DSN*），第654–661页。IEEE，2016年。[19] I. Itkin、A. Gromova、A. Sitnikov、D. Legchikov、E. Tsym-b alov、R. Yavorskiy、A. Novikov 和 K. Rudakov。用户辅助日志分析：分布式金融科技应用的质量控制。载于*2019*年*IEEE*人工智能测试国际会议（*AITest*），第45–51页。IEEE，2019年。

[20] Ł. Korzeniowski 和 K. Goczyła。自动化日志分析全景：系统性文献综述与映射研究。*IEEEAccess*，2022年。[21] Q. Lin、H. Zhang、J.-G. Lou、Y. Zhang 和 X. Chen。面向在线服务系统的问题识别的日志聚类。载于第*38*届软件工程国际会议附属会议论文集，第102–111页，2016年。[22] S. Rose、D. Engel、N. Cramer 和 W. Cowley。从单篇文档中自动提取关键词。文本挖掘：应用与理论，1:1–20，2010年。[23] P. J. Rousseeuw。轮廓系数：用于解释和验证聚类分析的图形辅助工具。《计算与应用数学杂志》，20:53–65，1987年。[24] P. Runeson 和 M. H¨ost。软件工程案例研究的开展与报告指南。实证软件工程，14(2):131–164，2009年。[25] J. Sander、M. Ester、H.-P. Kriegel 和 X. Xu。空间数据库中的基于密度的聚类：GDBSCAN算法及其应用。数据挖掘与知识发现，2(2):169–194，1998年。[26] M. E. Sharp、T. B. Sexton、M. P. Brundage 等。对非结构化维护日志数据进行半自动标注，以支持诊断根因分析。载于生产管理系统进展（*APMS 2017*），2016年。[27] P. E. Strandberg。软件工程中的伦理访谈。载于*2019*年*ACM/IEEE*实证软件工程与度量国际研讨会（*ESEM*），第1–11页。IEEE，2019年。[28] P. E. Strandberg、W. Afzal 和 D. Sundmark。结合持续集成与夜间测试，对软件测试结果进行探索与可视化。软件技术转移工具国际期刊，第1–25页，2022年。[29] P. E. Strandberg、T. J. Ostrand、E. J. Weyuker、W. Afzal 和 D. Sundmark。嵌入式系统领域中的间歇性失败测试。载于第*29*届*ACM SIGSOFT*软件测试与分析国际研讨会论文集，第337–348页，2020年。[30] P. E. Strandberg、D. Sundmark、W. Afzal、T. J. Ostrand 和 E. J. Weyuker。经验报告：使用多种因素进行自动化系统级回归测试优先级排序。载于*2016*年*IEEE*第*27*届软件可靠性工程国际研讨会（*ISSRE*），第12–23页。IEEE，2016年。[31] T. Xiao、Z. Quan、Z.-J. Wang、K. Zhao 和 X. Liao。LPV：基于向量化的日志解析器，用于离线与在线日志解析。载于*2020*年*IEEE*国际数据挖掘会议（*ICDM*），第1346–1351页。IEEE，2020年。