计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(9):57−69 [doi: 10.15888/j.cnki.csa.008661]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于机器学习的日志异常检测综述①

闫　力, 夏　伟

(江南计算技术研究所, 无锡 214084)
通信作者: 闫　力, E-mail: yanli_mail@yeah.net

摘　要: 日志异常检测是当前数据中心智能运维管理的典型核心应用场景. 随着机器学习技术的快速发展和逐步成熟, 将机器学习技术应用于日志异常检测任务已经形成热点. 首先, 文章介绍了日志异常检测任务的一般流程, 并指出了相关过程中的技术分类和典型方法. 其次, 论述了日志分析任务中机器学习技术应用的分类及特点, 并从日志不稳定性、噪声干扰、计算存储要求、算法可移植性等方面分析了日志分析任务的技术难点. 再次, 对领域内相关研究成果进行了梳理总结和技术特点的比较分析. 最后, 文章从日志语义表征、模型在线更新、算法并行度和通用性 3 个方面讨论了日志异常检测今后的研究重点及思考.

关键词: 日志; 异常检测; 机器学习; 智能运维技术; 深度学习

引用格式: 闫力,夏伟.基于机器学习的日志异常检测综述.计算机系统应用,2022,31(9):57–69. http://www.c-s-a.org.cn/1003-3254/8661.html

Survey on Log Anomaly Detection Based on Machine Learning

YAN Li, XIA Wei

(Jiangnan Institute of Computing Technology, Wuxi 214084, China)

Abstract: Log anomaly detection is a typical core application scenario of artificial intelligence for IT operations (AIOPS) in the current data center. With the rapid development and gradual maturity of machine learning technology, the application of machine learning to log anomaly detection has become a hot spot. Firstly, this study introduces the general procedure of log anomaly detection and points out the technical classifications and typical methods in the related process. Secondly, the classifications and characteristics of the application of machine learning technology in log analysis tasks are discussed, and we probe into the technical difficulties of log analysis tasks in terms of log instability, noise interference, computation & storage requirements, and algorithm portability. Thirdly, the related research productions in the field are summarized and their technical characteristics are compared and analyzed. Finally, the study discusses the future research focus and thinking of log anomaly detection from three aspects: log semantic representation, online model update, algorithm parallelism and versatility.

Key words: log; anomaly detection; machine learning; artificial intelligence for IT operations (AIOPS); deep learning

1　引言

日志是各类软硬件记录系统运行状态的一类重要数据, 它描述了系统的历史运行状态和详情. 运维工程师在进行故障排除、安全性检查等工作场景下, 经常需要对日志数据进行检查和分析. 而大数据技术的迅猛发展使得现代数据中心规模日益扩大, 复杂的系统每天都会产生数以千万计的日志记录, 一个运行在中等规模网络中的系统每天的日志量也能轻松超过 TB 级[1]. 体量巨大且种类繁多的日志数据使得运维工程师难以再依赖简单的关键词搜索或者正则匹配等方

① 收稿时间: 2021-11-26; 修改时间: 2021-12-28; 采用时间: 2022-01-05; csa 在线出版时间: 2022-06-17

式来进行手工分析. 这不仅会带来繁重而枯燥工作量, 同时也是对运维人员领域专业知识的极大考验——只有对某类日志特征具有丰富的处理经验才能设定出合理的过滤规则. 因此, 应用机器学习技术为机器自身赋能, 实时或准实时捕获系统运行状态异常以实现日志分析自动化, 是未来数据中心实现智能自主运维的重要途径之一.

日志、指标、链路追踪数据是运维工作中最为核心的监控和分析对象[2], 相较于指标和链路追踪数据, 日志是内容最为丰富且来源充足的一类数据, 系统的异常或者性能下降一般会在日志中优先体现出来. 文章阐述了实现在线自动日志异常检测的主要步骤、关键技术、方法分类等内容, 分析了日志分析的技术难点, 并介绍了近期领域内的最新研究成果. 最后, 提出今后的一些研究重点和思路.

2 日志异常检测任务简介

2.1 日志异常检测流程

日志异常检测任务一般分为日志采集、日志解析、特征表示、异常判别 4 个步骤.

日志采集: 日志作为一种信息系统中广泛存在的数据, 往往散落于系统各处, 特别是对于分布式软件、大型软件等, 不同节点、不同组件都在持续生成和积累日志, 所以日志采集是日志异常检测任务的第一步. 较为常见的日志采集工具包括 Logstash、Flume、Fluentd、Kafka 等, 通过持续、实时的数据采集, 可以将数据集中入库或者通过“发布-订阅”的模式推送给消费端进行后续处理.

日志解析: 日志属于半结构化数据, 需要分析其一般构成, 并利用解析技术将其中的常量和变量分离出来, 为后续特征表示提供良好基础. 一条日志数据可以分解为正则消息部分和特征消息部分[3]. 正则消息包括时间戳、日志等级、产生日志的类名称等日志基本组成, 这部分是可以用正则表达式进行提取和分解的, 属于日志数据的基础信息. 特征消息部分则是描述了日志内容的核心部分, 由文本、数字及特殊符号等组成. 一条日志数据是由程序开发者在代码中预先埋点编写的各种输出语句在满足特定条件下产生的, 例如 printf、log.info、log.warn 等. 它包括了字符串常量和参数变量, 其中字符串常量被称为日志模板 (或事件、键值), 表明了该条日志数据的类属, 是日志消息的语义描述; 参数变量则表明了系统状态关键变量的取值, 例如对象 ID、内存消耗、执行时间等. 其中日志模板由于是开发者对系统状态的语义描述, 蕴含丰富的信息, 往往作为重要的分析对象输入后续检测模型进行异常判别. 表 1 展示了截取自多个系统的日志片段, 并标识了各个组成部分, 需要注意的是由于应用软件开发规范和开发者的编程习惯各异, 日志格式上也存在千差万别.

数据解析的精度将显著影响检测模型的性能表现, 有研究试验表明数据解析阶段 4% 的错误可能会在异常检测阶段导致性能下降扩大一个数量级[4]. 日志解析从分析对象来源上区分, 可分为基于源代码的模板抽取和基于日志文本的模板抽取. 有研究工作使用了基于源代码中类似 printf 等语句来抽取模板[5,6], 但这种方式在多数情况下并不可行——因为源代码分析难度大且难以获得. 目前, 就基于日志文本来抽取模板的方式, 领域内已经提出了大量算法, 典型的有 SLCT[7]、IPLoM[8]、LKE[9]、Spell[10]、Drain[11]、LPV[12] 等. SLCT 是最早提出的日志解析算法, 通过 2 次遍历日志消息分别构造日志词汇表和候选簇, 再从簇中进行模板抽取; IPLoM 在模板生成前采用 3 步式分层划分方法对日志消息进行处理; LKE 是由微软提出的一种离线日志解析算法, 主要利用了分层聚类和启发式规则; Spell 基于最长公共序列来以在线方式解析日志模板; Drain 使用一棵固定深度解析树, 通过持续更新解析树来在线获得日志模板; LPV 是最近提出的一种日志解析算法, 它使用 Word2Vec 词嵌入技术[13] 对日志消息向量化, 基于向量相似度来聚类, 日志模板从所聚类的簇中提取出来.

特征表示: 这部分工作的主要目的在于构造机器学习模型可以处理的特征数据, 借此来学习日志的正常或者异常模式. 所提取特征的质量决定了后续模型检测效果所能达到的精度. 日志分析领域, 前一阶段提取的日志模板一般被称为事件或者键值. 在进行特征提取前, 需要对连续的日志进行切分, 通常有时间窗口、会话 ID 等方式: (1) 日志是一种具有时间属性的连续文本, 一条日志数据是否提示异常不仅取决于自身所蕴含的信息, 还受到其上下文的影响, 因此可采用固定或者滑动时间窗口将日志切分成数据片段, 便于输入模型进行处理. 最优时间窗口的大小可根据数据的特点进行多次尝试得到, 过小或过大的窗口都会对检测精度带来不利影响——过小的窗口会使得上下文

信息不足, 而过大的窗口又会带来冗余信息; (2) 某些日志的变量参数标记了某一特定会话的执行路径, 例如 HDFS 日志中的 block_id, OpenStack 日志中的 instance_id 等, 可以根据会话 ID 将并行进程产生的日志进行剥离, 但这种方法仅限于具有此类变量参数的日志.

表 1 典型日志样例解析

类别	系统	日志样例	正则消息	特征消息	
				日志模板	参数变量
操作系统	Linux	[OK] Started File System Check on /dev/disk/by-uuid/6d329…5-88652b684ca0.	[OK]	Started File System Check on	/dev/disk/by-uuid/6d329…5-88652b684ca0
数据库	Oracle	ORA-00603:ORACLE server session terminated by fatal error	ORA-00603	ORACLE server session terminated by fatal error	—
硬件	交换机	Nov 9 2018 09:49:03 HuaWei03 %%01SHELL/4/LOGINFAILED(s)[12]:Failed to login. (Ip=192.168.1.199, UserName=admin, Times=3, AccessType=TELNET, VpnName=)	Nov 9 2018 09:49:03 HuaWei03 %%01SHELL/4/LOGINFAILED(s)[12]	Failed to login. (Ip=*, UserName=*, Times=*, AccessType=*, VpnName=*)	(192.168.1.199, admin, 3, TELNET)
分布式系统	HDFS	081109 203519 147 INFO dfs.DataNode$PacketResponder: Received block blk_-1608999687919862906 of size 91178 from /10.250.14.224	081109 203519 147 INFO dfs.DataNode$ PacketResponder:	Received block * of size * from *	(blk_-1608999687919862906, 91178, 10.250.14.224)
Web服务	Apache	[Fri Aug 18 22:36:26 2000] [error] [client 192.168.1.6] File does not exist: /usr/local/apache/bugletdocs/Img/south-korea.gif	[Fri Aug 18 22:36:26 2000] [error]	[client *] File does not exist:	(192.168.1.6, /usr/local/apache/bugletdocs/Img/south-korea.gif)
消息队列	Rabbit MQ	=INFO REPORT==== 3-Jul-2017::11:45:14 === rabbit on node rabbit@node2 up	=INFO REPORT==== 3-Jul-2017::11:45:14 ===	rabbit on node * up	rabbit@node2
应用程序	Health APP	201812-20:5:32:205|Step_ExtSDM|30002312|calculateCaloriesWithCache totalCalories=18289	201812-20:5:32:205|Step_ExtSDM|30002312|	calculateCaloriesWithCache totalCalories=*	18289

日志异常检测通过采用的主要特征包括: 事件计数、事件序列、文本语义、时间间隔、变量取值、变量分布等[14], 这些特征的变化往往反映出系统状态的异常. 事件计数表示日志中某一类别的事件数量大幅增加或者减少, 如 Web 应用日志中某一时间段内突然出现大量“Login failed”事件, 可能是系统出现安全事件的标志; 由于具有时序属性, 正常日志事件出现的顺序也会遵循一定的规律, 如 OpenStack 管理日志中, 一台虚拟机的生命周期是从“VM Created”开始, 中间可能经过若干次“Pause/Unpause”和“Suspend/Resume”的事件组合, 最终以“VM deleted”结束; 文本语义是通过自然语言处理技术解析日志文本自身携带的语义来判别日志是否出现异常; 时间间隔则是两条日志消息相继出现所经过的时长, 由于绝大多数日志都带有时间戳, 时间间隔很容易计算, 如果特定两条日志消息的时间间隔过大, 可能预示着服务性能的下降; 变量取值指的是模板抽取后分离出的参数变量如果取值超过正常阈值范围, 则提示了系统的状态异常, 例如某进程消耗的内存过大, 可能是出现了内存泄漏. 变量分布则是对于某些参数变量的取值进行分布统计, 典型的如 Web 服务器日志中某一时间段同一源 IP 地址分布密度远大于其他地址, 则提示潜在威胁的出现. 现有的研究成果提出的算法往往用到其中一种或者多种的组合, 这些特征表示从不同维度反映出系统的当前状态.

异常判别: 经过特征提取, 原始日志数据已经转换为模型可以处理的特征数据, 可以输入判别模型进行异常检测. 在检测模型的设计上, 包括传统机器学习方法和深度学习方法. 传统机器学习方法具有硬件依赖性低、可解释性好等特点, 典型的有基于主成分分析的算法[5]、基于支持向量机的算法[15]、基于隐马尔科夫模型的算法[16]、基于 K 最近邻算法[17,18]、各种聚类算法[3,19] 等. 传统机器学习算法提取高级特征或者全局特征的能力相对有限, 特别是日志文本的语义识别、长距离依赖等问题上表现不如深度学习, 所以有大量研究将深度学习引入日志异常检测任务. 基于深度模型的日志分析算法中包括基于长短期记忆网络的算法[20–24]、基于双向长短期记忆网络的算法[25]、基于变分自编码器的算法[6,26]、基于生成对抗网络的算法[27]、

基于 Transformer 网络的算法[28,29] 等. 但深度学习方法也并非“完美”: 首先, 深度学习作为一种端到端的解决方案, 属于黑盒模型, 可解释性较差; 其次, 为了获得满意的模型容量, 深度模型包含数层神经网络, 参数量往往十分庞大, 训练时间长且消耗资源多. 本质上, 异常判别模型是一个二分类器, 通过对新到达的日志消息进行分析, 来推断其属于正常或异常.

2.2 评价标准

关于日志异常检测的研究普遍采用分类任务中的常用评价准则, 即精确率 (*precision*)、召回率 (*recall*) 和 *F*1-score. 精确率表示在所有判别为异常的结果中正确判别结果所占比例, 精确率过低表示算法的误检率较高, 会带来大量的系统误警; 召回率表示在所有实际为异常的结果中正确判别结果所占比例, 召回率过低表示算法的漏检率较高, 系统将不能识别大部分的日志异常; *F*1-score 为精确率和召回率的调和平均数, 兼顾了精确率和召回率对算法整体检测效果的影响, 过低的精确率或召回率都会导致 *F*1-score 性能下降. 准确率 (*Accuracy*) 表示所有判别正确的结果占总结果数的比例, 少部分文献采用了准确率作为评价指标, 但在正常和异常样本数据比例严重失衡的日志异常检测任务中, 占比大的样本 (正常日志) 对准确率的影响更大, 一般不能很好反映算法的性能. 正式地, 有:

$$precision = \frac{TP}{TP+FP} \tag{1}$$

$$recall = \frac{TP}{TP+FN} \tag{2}$$

$$F1\text{-score} = 2\times\frac{precision\times recall}{precision+recall} \tag{3}$$

$$Accuracy = \frac{TP+TN}{TP+FN+TN+FP} \tag{4}$$

其中, *TP* (true positive) 代表实际为真 (异常日志) 且判别也为正的结果数; *FP* (false positive) 代表实际为假 (正常日志) 但判别为正的结果数; *TN* (true negative) 代表实际为假且判别也为负的结果数; *FN* (false negative) 代表实际为真但判别为负的结果数.

3 日志分析技术难点分析

3.1 技术分类及其特点

根据是否需要对特征数据打标签来进行模型训练, 可分为有监督方法、半监督方法和无监督方法. 有监督方法利用预先打过标签的特征数据对模型进行训练, 同时学习正常数据与异常数据的特征, 训练完成的模型即可对测试数据进行判别 (分类). 然而, 有监督方法在日志分析中的热度不如半监督方法和无监督方法, 原因有以下几点: (1) 日志中的异常数据是相较稀少的, 导致训练数据集失衡现象严重, 影响模型训练效果; (2) 为每条日志打上标签是非常耗时耗力的, 且需要对该类日志有较深的领域认知, 才能做出正确判断; (3) 有监督方法得到的模型只能识别已知的异常日志, 对于未见过的异常无法判断. 相比于有监督方法, 无监督方法避免了上述问题, 主要通过聚类、降维等方法寻找具有相同或相近特征的数据, 进而识别出数据异常点, 但也面临着准确率较低、容易受噪声影响等问题. 半监督方法介于二者之间, 通常只需要系统正常样本即可完成训练过程, 实现前两者的优势互补.

3.2 日志分析难点

指标类数据属于单变量或者多变量的时序数据, 链路追踪数据属于结构化数据, 输出格式也相对固定且有限, 依据采用的收集工具 (如谷歌的 Dapper、推特的 Zipkin 等) 即可确定. 但是, 因日志具有非结构化特点, 且没有统一标准, 使得日志输出格式上自由度非常大, 这就使日志分析难度更大.

3.2.1 日志的不稳定性

由于系统升级、应用更新等原因, 源代码的日志输出语句会持续变化, 包括添加、删除和修改等操作, 致使前期模型训练数据当中未发生变化的内容越来越少, 进而使异常检测模型性能急剧下降甚至失效. 举例来说, 一款来自微软的软件经过数次版本迭代, 生成的日志中未发生变化的日志事件只占总数的 30% 左右[25].

3.2.2 易受噪声干扰

噪声并非日志消息自身携带, 而是在输入模型处理前的各个环节引入的. 例如, 采集日志消息时由传输网络不稳定、系统断电、软件 bug 等引起的日志消息整体缺失或部分缺失; 又如, 解析过程中出现的日志模板抽取不准确, 导致特征数据质量降低. 日志解析错误主要来自于两种: (1) 语义理解偏差, 将参数变量解析为模板组成, 或反之. 如图 1 所示, 在日志解析时将“hadoop”误判为模板组成. (2) 词表外词汇 (out of vocabulary) 引入的解析错误, 由于日志不稳定性或者训练数据划分等问题, 测试数据中出现训练过程中从未出现的词汇, 致使日志解析出未知模板 (事件), 影响

模型性能. 文献 [28] 就日志解析对模型性能的影响做了详细的量化分析.

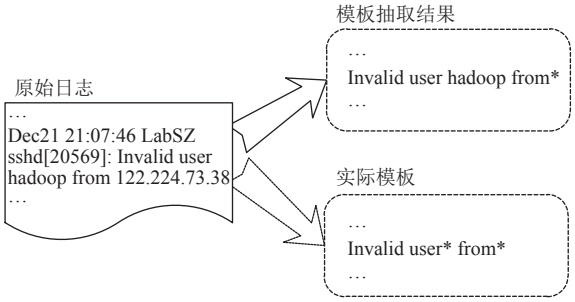

图 1 日志模板解析错误示例

3.2.3 计算和存储要求高

如前文所述, 日志种类多样且规模日益增大, 涵盖了从底层硬件、服务中间件、数据库和上层应用等多种来源. 海量的日志需要足够大的存储资源来承载数据, 和快速实时处理能力来满足时效性要求, 并且基于深度学习的算法还需昂贵的 GPU 资源来训练模型. 因此, 在做相关研究设计时, 应当考虑尽量降低算法的存储要求, 同时可利用大数据分布式处理工具 (如 Spark Streaming, Storm 等) 来实现并行处理, 这样有利于算法更好地应用于实际生产中. 文献 [1] 提出的 ADA 算法通过在线训练的方式解决了深度模型需要大量数据来训练的问题, 进而极大减少了对于存储空间的需求. 文献 [3] 提出分布式日志处理框架, 通过 Spark Streaming 完成日志解析、特征提取、数据归一化等步骤, 增强了算法面对海量实时日志的分析能力. 文献 [30] 针对经典日志解析算法面对海量日志解析能力不足的问题, 基于 Spark 分布式计算框架提出了 POP 算法. 文献 [31] 则基于 GPU 的并行能力改进了 LKE 算法.

3.2.4 算法和模型可移植性差

由于日志格式自由度大, 因此不同的系统日志呈现出的特征模式不同, 同时根据应用目的的不同, 即使同一类型日志关注的特征也可能有所差异. 现有研究多数基于有限的公开日志数据集[32] 来开展算法和模型设计, 所以其可移植性差, 难以实现成果复用. 文献 [14] 详细分析了日志异常模式的复杂性, 并提出一种组合式算法 LogAD 来针对不同异常模式来选择合适的算法来处理.

4 相关研究

该领域已经有大量研究工作就如何实现自动化的日志异常检测提出了诸多算法和框架模型. 根据前述分类, 将现有成果进行简要说明, 并在附录中以表格形式 (表 A1) 对比了各个算法特点.

4.1 基于传统机器学习的方法

文献 [5] 使用 PCA 的方法来实现对日志的离线异常检测. 其主要步骤是在完成对整个日志文件的解析后, 构建状态变量比率向量和消息计数向量 2 种特征, 再应用 PCA 来识别出异常数据. 文献 [15] 同样构造了以会话 ID 为行、以日志模板为列的特征矩阵, 只是通过专家知识对语义相近的模板进行了合并, 减少了噪声干扰. 最后, 应用 SVM 对日志数据是否异常进行判别. 文献 [16] 针对 Web 日志提出二级机器学习算法实现异常检测. 首先, 利用带有标签的训练数据训练一颗决策树, 用来对测试数据分类, 生成正常数据集和异常数据集, 最后将正常数据分解参数后用于隐马尔科夫模型的训练, 提取出所有训练数据中的正常状态用以识别异常. 文献 [17] 利用 TF-IDF 技术对日志模板向量化, 并使用 meanshift 聚类等方法为样本打上标签, 最后利用标签化后的数据训练 KNN 模型来进行新样本的异常判别. 文献 [18] 提出由 K-prototype 聚类和 KNN 分类组合的日志异常检测算法, 先利用 K-prototype 来筛选出疑似异常对象, 再基于 KNN 给出最终判别结果. 文献 [19] 基于字符串度量和数值度量设计了一种对日志消息进行实时在线聚类的算法, 用于检测信息与通信网络中的攻击. 文献 [3] 构建并基于 Spark Streaming 持续更新日志消息计数向量 (message count vector), 利用 K-means 聚类算法将这些向量分为正常和异常两类.

上述算法在相应的各类数据集上取得了最高约 0.99 的 $F1$-score 性能表现.

4.2 基于深度学习的方法

日志数据的内在复杂模式和长距离依赖等特点, 使得不少研究工作将深度学习引入日志异常检测领域.

长短期记忆网络 (long short-term memory, LSTM)[33] 作为一种特殊的循环神经网络, 最早应用于机器翻译领域, 由于 LSTM 擅于保存序列中的长距离依赖关系, 使其也适合应用于日志分析当中——捕捉日志消息中的远距离依赖关系. DeepLog[20] 作为最经典的基于深度学习的日志分析算法之一, 利用 2 层叠加的 LSTM 网络实现了日志模板序列和参数变量的异常检测. nLSALOG[21] 同样使用 2 层 LSTM 网络, 并在其中引

入自注意力机制, 使得模型能更好地捕获序列内部的依赖关系, 提高异常检测性能. LogAnomaly[22] 在日志模板语义表达上做了改进, 利用 dLCE 自然语言处理模型通过词嵌入技术生成模板向量, 再输入 LSTM 进行模型训练, 最终用于预测推断. 这种方法可通过计算模板向量间的近似度将系统新产生的陌生日志模板匹配到模型已经学习到的模板向量. LogNL[23] 借鉴了 DeepLog 和 LogAnomaly 的优势, 通过 2 层 LSTM 完成日志模板和参数变量的异常判断, 改善了有模型性能. LogMerge[24] 对提取的日志模板基于词向量加权求和后得到模板向量, 通过对 2 种类型日志的模板向量进行聚类操作获取簇中心向量, 实现了跨日志类型的异常检测. 模型方面采用了 CNN+LSTM 的组合. LogRobust[25] 利用一个双向 LSTM 捕获来自日志模板序列前向和后向两个方向的依赖关系, 并应用注意力机制识别出那些对判别结果更重要的日志模板. 同时, 由于也使用了日志模板的语义表达, 对于噪声干扰有一定的鲁棒性. NoTIL[34] 是一种用于检测日志消息时间间隔异常的算法, 通过滑动时间窗口对日志序列进行切分, 并计数时间窗口内的每类事件生成该窗口的表示向量, 输入 LSTM 来预测未来的时间窗口, 并基于实际结果与预测值间的误差判别异常. ADA[1] 引入在线深度学习技术[35], 基于 LSTM 提出了一种自适应式的异常检测算法, 具备在线学习能力的同时, 还能够根据当前异常检测结果自动调整模型层数.

门控循环单元 (gated recurrent unit, GRU)[36] 是另一种重要的循环神经网络, 基本原理和 LSTM 十分相似. PLELog[37] 结合了有监督方法和无监督方法的优势, 通过对日志模板向量化后进行聚类, 然后基于概率标签估计的方法为训练集中没有标签的数据打上概率化的标签. 再以此输入带有注意力机制的 GRU 网络中训练.

采用 LSTM 或者 GRU 搭建日志异常检测模型是一个较普遍的做法, 但 RNN 类的模型需要顺序输入数据, 致使其无法实现数据的并行化处理, 也就限制了其在大流量日志数据条件下的性能表现. 近年来, 谷歌的 Transformer 模型[38] 因其强大的快速并行能力在自然语言处理领域获得了广泛应用, Transformer 在处理大流量输入数据方面的优势使有些学者将其引入日志分析任务中. HitAnomaly 算法[29] 基于分层式的 Transformer 结构分别将日志序列中的模板 (事件) 序列和参数 (变量) 序列转化为两个向量表示, 再利用注意力机制对二者加权求和, 最后通过 Softmax 层得到异常概率分布. NeuralLog 算法[28] 采用了 BERT 模型[39] 对日志消息所有单词进行词嵌入, 并对日志消息中所有的词向量求均值得到日志消息的特征向量, 最后输入基于 Transformer 的模型进行训练及预测. 值得注意的是, NeuralLog 并未使用日志解析技术, 避免了日志解析错误给异常检测模型结果带来的影响.

变分自编码器 (variational auto-encoder, VAE)[40] 是深度学习领域重要的生成模型之一, 广泛用于图像生成任务. VAE 通过编码器将训练数据映射为隐空间内的低维变量, 再由解码器从隐空间中随机采样来生成训练数据的重构. 基于 VAE 的日志异常检测算法一般是以重构误差来判断异常或者直接给出异常概率. VeLog[6] 以会话 ID 切分日志数据, 并构造日志执行计数矩阵和日志执行顺序矩阵, 输入 VAE 进行训练, 以此来识别日志模板在数量和执行路径上的异常. 该方法对于训练集中未出现的日志模板会判断为异常, 需将新出现的模板加入训练集对模型重新训练. 文献 [26] 提出的 hybrid CAE and VAE framework 将日志模板序列进行独热编码 (one-hot encoding) 后, 使用卷积自编码器 (convolutional auto-encoder, CAE) 将其压缩以提取序列高层特征, 再输入 VAE 进行训练, 学习日志模板序列的正常模式.

生成对抗网络 (generative adversarial network, GAN)[41] 作为另一种重要的深度生成模型, 也有研究人员将其用于日志分析任务. 文献 [27] 为解决日志数据不平衡的问题基于生成对抗机制设计了生成器和判别器, 通过二者博弈来相互提高性能, 直至收敛. 最后, 利用生成器可产生出逼近真实异常样本的模拟数据, 进而在检测时输出当前日志模板序列模式下后续可能事件正常或异常的概率分布.

上述算法在相应的各类数据集上取得了最高约 0.99 的 $F1$-score 性能表现.

5 研究重点及方向

5.1 日志消息的语义表征

日志属于软件开发者对系统实时状态的描述文本, 这种特点令自然语言处理中的诸多词嵌入技术可以应用于日志分析当中并变得越来越重要. 独热编码属于最简单的编码方式, DeepLog、ADA、hybrid CAE and

VAE framework 等都采用了这种方式, 但独热编码存在维度灾难和语义鸿沟 (即编码向量由于词汇表庞大而变得非常稀疏, 且所有向量均正交, 不能体现语义相似性), 使其应用受限. 相比之下, Word2Vec、FastText[42]、Glove[43] 等词向量相关技术能更好地表现日志语义关系, 也被大量研究工作所采用, 如 LogRobust、PLELog、LogMerge 等, 但这几种技术并不能很好地解决一词多义的现象. 词表外词汇的出现在日志中十分普遍, 据文献 [28] 研究, 某些数据集 (如 BGL) 中即使训练集达到 80% 时, 仍有超过 80% 的词汇在训练集中从未出现. 能否处理好词表外词汇是日志语义表征能力的重要体现. 有研究专门为日志分析提出了一种语义表征框架 Log2Vec[44], 通过 MIMICK 算法[45] 解决词表外词汇的向量表示, 但其字符级推算的方式难以获得有实际意义的词向量. NeuralLog 算法采用 BERT 模型来表征日志语义, 不仅能够解决一词多义问题, 也可以在子词级别推算词表外词汇的含义, 获得相对更有意义的词向量. 总之, 语义表征是日志分析中的重要一环, 能够提高算法的鲁棒性, 在一定程度上抵抗日志不稳定性带来的影响.

5.2 模型在线更新机制

如前文所述, 现代软件产品多采用 DevOps 敏捷开发模式, 产品的持续集成、持续部署加速了其迭代更新, 从而也使得日志模式持续演进, 产生不稳定性. 而日志不稳定性会导致持续出现新的日志模板和更多的词表外词汇. 所以在设计有关算法时, 不得不考虑模型的持续更新能力, 以适应新出现的日志模式. 过往研究中, 主要有 3 种方法来进行模型更新: 一是定期对模型重新训练以适应新的日志模式; 二是基于运维人员对检测结果的反馈, 对模型就行及时修正, 如 DeepLog、VeLog 等; 三是引入在线学习机制, 如 ADA 等. 模型定期重新训练的方式不仅在实时性上满足不了生产要求, 也会带来较大的计算和时间开销 (特别是深度学习模型). 应该说运维人员参与到日志异常检测任务当中来是必要的, 特别是通过人工反馈的方式可以引入领域专家知识, 而这些是无法单纯凭借算法获得的. 但应考虑模型的在线更新机制的设计, 尽量减少重复训练带来的开销. 文献 [46] 提出了终身学习的日志异常检测方法, 通过反馈结果来对模型进行更新 (而不是重新训练). 在线学习是一种重要的研究思路, 可以将数据以流的方式输入模型进行逐步优化的动态训练和预测, 令模型更具扩展性.

5.3 算法的并行度和通用性

日志大体量和多类型的特点对检测算法提出了并行度和通用性要求. 首先, 不论是基于传统机器学习算法还是深度学习算法, 均需考虑算法部署于生产环境下的并行化能力, 以满足海量日志处理的需求, 例如文献 [3] 利用大数据相关技术实现日志的全流程并行处理, 使其具有实际应用能力. 同时, 在某些计算、存储资源不充沛的条件下, 还应当尽量减少日志分析处理过程中的计算和存储开销, 如 ADA 算法通过在线深度学习减少了在空间存储上的需求. 其次, 结合日志数据的异常模式和应用场景的多样化, 算法设计应尽可能具有一定的通用性, 或者只需要根据数据特征或场景需求进行微调即可快速应用, 如 LogAD 即是一种典型的组合式方法, 其应对复杂场景下多种日志模式的异常检测综合能力明显提高.

5.4 结果的可解释性和辅助决策

为了在智能运维场景下更好地进行故障预警和辅助运维人员进行事件处理, 日志异常检测任务不应停留在只检测当前日志是否出现异常, 也应注重检测结果的可解释性. 例如, 当前异常由何种特征提示、如何以运维人员能够看得懂的方式呈现、异常的相互关联问题、异常可能的影响范围如何等问题. 即在更高层次赋予检测结果良好的可解释性和辅助决策能力, 是该项技术的深化和延展. DeepLog、LogAD 中都试图构建任务执行的工作流, 很好地印证了填补机器算法输出与人之间理解鸿沟的重要性. 特别地, 由于深度学习一般被认为是端到端的黑盒模型, 基于深度模型的算法可解释性较差, 目前已经有相关研究致力于获得检测结果的可解释性[47].

6 结束语

日志异常检测任务是智能运维发展的重要落地场景, 是机器学习和运维管理结合的重要一环. 大量研究已经表明, 将机器学习应用于日志分析当中, 可以有效应对当下数据中心日志体量急速膨胀导致分析难、管理难的问题. 文章对日志异常检测任务的典型流程、技术难点、相关研究工作和后续研究重点都做了详细阐述和分析, 对于该领域研究具有一定的参考价值.

参考文献

1 Yuan YL, Adhatarao SS, Lin MK, *et al*. ADA: Adaptive

deep log anomaly detector. IEEE Conference on Computer Communications (IEEE INFOCOM 2020). Toronto: IEEE, 2020. 2449–2458. [doi: 10.1109/INFOCOM41043.2020.9155487]

2 Nedelkoski S, Cardoso J, Kao O. Anomaly detection and classification using distributed tracing and deep learning. 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). Larnaca: IEEE, 2019. 241–250. [doi: 10.1109/CCGRID.2019.00038]

3 Astekin M, Özcan S, Sözer H. Incremental analysis of large-scale system logs for anomaly detection. 2019 IEEE International Conference on Big Data (Big Data). Los Angeles: IEEE, 2019. 2119–2127. [doi: 10.1109/BigData47090.2019.9006593]

4 He PJ, Zhu JM, He SL, *et al*. An evaluation study on log parsing and its use in log mining. 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Toulouse: IEEE, 2016. 654–661. [doi: 10.1109/DSN.2016.66]

5 Xu W, Huang L, Fox A, *et al*. Detecting large-scale system problems by mining console logs. Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles. Montana: ACM, 2009. 117–132. [doi: 10.1145/1629575.1629587]

6 Qian Y, Ying S, Wang BM. Anomaly detection in distributed systems via variational autoencoders. 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC). Toronto: IEEE, 2020. 2822–2829. [doi: 10.1109/SMC42975.2020.9283078]

7 Vaarandi R. A data clustering algorithm for mining patterns from event logs. Proceedings of the 3rd IEEE Workshop on IP Operations & Management (IPOM 2003) (IEEE Cat. No. 03EX764). Kansas City: IEEE, 2003. 119–126. [doi: 10.1109/IPOM.2003.1251233]

8 Makanju A, Zincir-Heywood AN, Milios EE. A lightweight algorithm for message type extraction in system application logs. IEEE Transactions on Knowledge and Data Engineering, 2011, 24(11): 1921–1936. [doi: 10.1109/TKDE.2011.138]

9 Fu Q, Lou JG, Wang Y, *et al*. Execution anomaly detection in distributed systems through unstructured log analysis. 2009 9th IEEE International Conference on Data Mining. Miami Beach: IEEE, 2009. 149–158. [doi: 10.1109/ICDM.2009.60]

10 Du M, Li FF. Spell: Online streaming parsing of large unstructured system logs. IEEE Transactions on Knowledge and Data Engineering, 2019, 31(11): 2213–2227. [doi: 10.1109/TKDE.2018.2875442]

11 He PJ, Zhu JM, Zheng ZB, *et al*. Drain: An online log parsing approach with fixed depth tree. 2017 IEEE International Conference on Web Services (ICWS). Honolulu: IEEE, 2017. 33–40. [doi: 10.1109/ICWS.2017.13]

12 Xiao T, Quan Z, Wang ZJ, *et al*. LPV: A log parser based on vectorization for offline and online log parsing. 2020 IEEE International Conference on Data Mining (ICDM). Sorrento: IEEE, 2020. 1346–1351. [doi: 10.1109/ICDM50108.2020.00175]

13 Mikolov T, Chen K, Corrado G, *et al*. Efficient estimation of word representations in vector space. arXiv: 1301.3781, 2013.

14 Zhao NW, Wang HL, Li ZY, *et al*. An empirical investigation of practical log anomaly detection for online service systems. Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Singapore: ACM, 2021. 1404–1415. [doi: 10.1145/3468264.3473933]

15 Han SB, Wu QH, Zhang H, *et al*. Log-based anomaly detection with robust feature extraction and online learning. IEEE Transactions on Information Forensics and Security, 2021, 16: 2300–2311. [doi: 10.1109/TIFS.2021.3053371]

16 Cao QM, Qiao YR, Lyu Z. Machine learning to detect anomalies in Web log analysis. 2017 3rd IEEE International Conference on Computer and Communications (ICCC). Chengdu: IEEE, 2017. 519–523. [doi: 10.1109/CompComm.2017.8322600]

17 Ying S, Wang BM, Wang L, *et al*. An improved KNN-based efficient log anomaly detection method with automatically labeled samples. ACM Transactions on Knowledge Discovery from Data, 2021, 15(3): 34. [doi: 10.1145/3441448]

18 Liu ZL, Qin T, Guan XH, *et al*. An integrated method for anomaly detection from massive system logs. IEEE Access, 2018, 6: 30602–30611. [doi: 10.1109/ACCESS.2018.2843336]

19 Wurzenberger M, Skopik F, Landauer M, *et al*. Incremental clustering for semi-supervised anomaly detection applied on log data. Proceedings of the 12th International Conference on Availability, Reliability and Security. Reggio Calabria: ACM, 2017. 31. [doi: 10.1145/3098954.3098973]

20 Du M, Li FF, Zheng GN, *et al*. DeepLog: Anomaly detection and diagnosis from system logs through deep learning.

Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. Texas: ACM, 2017. 1285–1298. [doi: 10.1145/3133956.3134015]

21 Yang RP, Qu D, Gao Y, *et al*. nLSALog: An anomaly detection framework for log sequence in security management. IEEE Access, 2019, 7: 181152–181164. [doi: 10.1109/ACCESS.2019.2953981]

22 Meng WB, Liu Y, Zhu YC, *et al*. LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. Proceedings of the 28th International Joint Conference on Artificial Intelligence. Macao: AAAI Press, 2019. 4739–4745. [doi: 10.24963/ijcai.2019/658]

23 Zhu B, Li J, Gu RB, *et al*. An approach to cloud platform log anomaly detection based on natural language processing and LSTM. 2020 3rd International Conference on Algorithms, Computing and Artificial Intelligence (ACAI 2020). Sanya: ACM, 2020. 88. [doi: 10.1145/3446132.3446415]

24 张圣林, 李东闻, 孙永谦, 等. 面向云数据中心多语法日志通用异常检测机制. 计算机研究与发展, 2020, 57(4): 778–790. [doi: 10.7544/issn1000-1239.2020.20190875]

25 Zhang X, Xu Y, Lin QW, *et al*. Robust log-based anomaly detection on unstable log data. Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Singapore: ACM, 2019. 807–817. [doi: 10.1145/3338906.3338931]

26 Wadekar A, Gupta T, Vijan R, *et al*. Hybrid CAE-VAE for unsupervised anomaly detection in log file systems. 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT). Kanpur: IEEE, 2019. 1–7. [doi: 10.1109/ICCCNT45670.2019.8944863]

27 夏彬, 白宇轩, 殷俊杰. 基于生成对抗网络的系统日志级异常检测算法. 计算机应用, 2020, 40(10): 2960–2966. [doi: 10.11772/j.issn.1001-9081.2020020270]

28 Le VH, Zhang HY. Log-based anomaly detection without log parsing. arXiv: 2108.01955, 2021.

29 Huang SH, Liu Y, Fung C, *et al*. HitAnomaly: Hierarchical transformers for anomaly detection in system log. IEEE Transactions on Network and Service Management, 2020, 17(4): 2064–2076. [doi: 10.1109/TNSM.2020.3034647]

30 He PJ, Zhu JM, He SL, *et al*. Towards automated log parsing for large-scale log data analysis. IEEE Transactions on Dependable and Secure Computing, 2017, 15(6): 931–944. [doi: 10.1109/TDSC.2017.2762673]

31 Ren XY, Zhang L, Xie KP, *et al*. A parallel approach of weighted edit distance calculation for log parsing. 2019 IEEE 2nd International Conference on Computer and Communication Engineering Technology (CCET). Beijing: IEEE, 2019. 101–104. [doi: 10.1109/CCET48361.2019.8989069]

32 He SL, Zhu JM, He PJ, *et al*. Loghub: A large collection of system log datasets towards automated log analytics. arXiv: 2008.06448, 2020.

33 Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9(8): 1735–1780. [doi: 10.1162/neco.1997.9.8.1735]

34 Baril X, Coustié O, Mothe J, *et al*. Application performance anomaly detection with LSTM on temporal irregularities in logs. Proceedings of the 29th ACM International Conference on Information & Knowledge Management. Atlanta: ACM, 2020. 1961–1964. [doi: 10.1145/3340531.3412157]

35 Sahoo D, Pham Q, Lu J, *et al*. Online deep learning: Learning deep neural networks on the fly. Proceedings of the 27th International Joint Conference on Artificial Intelligence. Stockholm: IJCAI, 2018. 2660–2666.

36 Cho K, van Merriënboer B, Gulcehre C, *et al*. Learning phrase representations using RNN encoder-decoder for statistical machine translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha: Association for Computational Linguistics, 2014. 1724–1734. [doi: 10.3115/v1/D14-1179]

37 Yang L, Chen JJ, Wang Z, *et al*. Semi-supervised log-based anomaly detection via probabilistic label estimation. 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). Madrid: IEEE, 2021. 1448–1460. [doi: 10.1109/ICSE43902.2021.00130]

38 Vaswani A, Shazeer N, Parmar N, *et al*. Attention is all you need. Advances in Neural Information Processing Systems. 2017, 30: 25998–6008.

39 Devlin J, Chang MW, Lee K, *et al*. BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Minneapolis: Association for Computational Linguistics, 2018. 4171–4186.

40 Kingma DP, Welling M. Auto-encoding variational Bayes. arXiv: 1312.6114, 2013.

41 Goodfellow IJ, Pouget-Abadie J, Mirza M, *et al*. Generative

adversarial nets. Proceedings of the 27th International Conference on Neural Information Processing Systems. Montreal: MIT Press, 2014. 2672–2680.

42 Joulin A, Grave E, Bojanowski P, *et al*. Bag of tricks for efficient text classification. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers. Valencia: Association for Computational Linguistics, 2016. 427–431.

43 Pennington J, Socher R, Manning C. Glove: Global vectors for word representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha: Association for Computational Linguistics, 2014. 1532–1543. [doi: 10.3115/v1/D14-1162]

44 Meng WB, Liu Y, Huang YH, *et al*. A semantic-aware representation framework for online log analysis. 2020 29th International Conference on Computer Communications and Networks (ICCCN). Honolulu: IEEE, 2020. 1–7. [doi: 10.1109/ICCCN49398.2020.9209707]

45 Pinter Y, Guthrie R, Eisenstein J. Mimicking word embeddings using subword RNNs. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. Copenhagen: Association for Computational Linguistics, 2017. 102–112. [doi: 10.18653/v1/D17-1010]

46 Du M, Chen Z, Liu C, *et al*. Lifelong anomaly detection through unlearning. Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. London: ACM, 2019. 1283–1297. [doi: 10.1145/3319535.3363226]

47 Pang GS, Aggarwal C. Toward explainable deep anomaly detection. Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. Online: ACM, 2021. 4056–4057. [doi: 10.1145/3447548.3470794]

附录

为了更清晰直观地比较各个算法的技术路径和优劣势, 文章给出了算法对比表, 如表 A1 所示.

表 A1 各算法对比详情表

算法	年份	技术分类	日志解析/预处理	特征表示	检测技术/模型	数据集	性能表现 (最佳参数设定)	技术特点分析
控制台日志挖掘算法[5]	2009	无监督	源码分析	事件计数、变量分布	PCA	DarkStar (在线游戏服务日志)、HDFS	DarkStar: 未提供具体数据; HDFS: 精确率=0.91, 召回率=0.99, *F*1-score=0.95*	计算复杂度低, 不需要人工干预和相关领域知识.
Web日志分析算法[16]	2017	有监督	字段解析	变量取值	DT、HMM	自有Web日志	根据论文实验结果只可以推算出召回率=0.96, 该算法采用了准确率作为评价指标, 准确率为0.94.	分析对象是结构相对简单的Web日志, 具有结构化数据特点, 且需要人工对数据打标签.
一种基于增量式聚类的日志异常检测算法[19]	2017	半监督	无	事件计数	增量式聚类法	自有系统日志	针对在不同攻击场景下的*F*1-score会有所不同: SQL注入场景最高可达0.82, XSS攻击场景最高可达0.99, 暴力登录攻击场景最高可达0.75	该方法主要基于日志异常检测来识别SQL注入、XSS攻击、暴力登录攻击等安全事件. 通过增量式聚类来实现在线检测功能.
一种组合式的大规模系统日志异常检测算法[18]	2018	有监督	字段解析	变量取值	K-prototype、KNN	自有系统日志	算法在手工生成的6类数据集上精确率、召回率达到平均0.90左右的水平.	属于会话级异常检测, 通过对用户登录活动和会话统计数据两种类别的属性来应用机器学习算法进行异常检测, 不具体到某一行日志消息.
一种增量日志分析算法[3]	2019	无监督	未介绍	事件计数	K-means聚类	HDFS	算法对间歇性到达的日志流数据进行处理, 在整个过程中基本保持了性能指标的稳定, 在HDFS数据集上最终达到的*F*1-score约为0.93.	该技术根据实时到来的新数据更新判别模型, 并基于Spark Streaming实现并行计算, 可有效应对大规模日志.

注: *表示论文未直接给出相关评价指标数据, 根据论文实验数据计算得出.

表 A1 (续) 各算法对比详情表

算法	年份	技术分类	日志解析/预处理	特征表示	检测技术/模型	数据集	性能表现 (最佳参数设定)	技术特点分析
基于改进KNN的日志异常检测算法[17]	2021	有监督	N-gram和频繁模式挖掘	事件计数	meanshift、KNN	Liberty、BGL, Thunderbird、Spirit, HDFS、Zookeeper	采用了准确率、召回率和*F*1-score作为评价指标, 未明确给出在各数据集上的性能数据, 但从实验结果来看, 该算法在HDFS、Zookeeper上各项指标均达到0.95左右, 其余数据集上达到0.90左右.	为利用有监督方法KNN, 文中使用聚类算法对数据分类, 并分别打上标签, 并对KNN进行了改进, 以解决数据不平衡问题.
OES[15]	2021	有监督	未介绍	事件计数	SVM	HDFS、OpenStack、BGL (蓝色基因超级计算机日志)	未明确给出在各数据集上的评价指标数据, 但从实验结果上可看到在人为注入噪声后, 相比于其他算法, OES的精确率、召回率、*F*1- score等指标仍可保持在0.90以上.	通过人工干预合并语义相近的日志模板, 该算法虽无需专门的训练过程, 但必须依赖人工确认结果在线修正检测模型参数, 所以仍归类为有监督方法, 其对于噪声干扰具有鲁棒性
DeepLog[20]	2017	半监督	Spell	事件序列、变量取值、时间间隔	LSTM	HDFS、OpenStack	HDFS: 精确率=0.95, 召回率=0.96, *F*1-score=0.96; OpenStack:精确率=0.96, 召回率=1.0, *F*1-score=0.98	充分利用了LSTM在处理长距离依赖方面的优势, 实现对日志序列和参数变量同时检测, 但缺点是需要为每一种日志模板的参数变量都构建一个LSTM网络.
nLSALOG[21]	2019	半监督	未介绍	事件序列	LSTM with self-attention	HDFS、BGL	HDFS: 精确率=0.97, 召回率=0.99, *F*1-score=0.98; BGL: 精确率=0.82, 召回率=0.95, *F*1-score=0.88**	在模型中引入了词嵌入层和自注意力机制层, 提高了日志语义表征能力和长距离依赖捕获能力.
LogAnomaly[22]	2019	半监督	FT-tree	事件序列、事件计数	LSTM	HDFS、BGL	HDFS: 精确率=0.98, 召回率=0.97, *F*1-score=0.98; OpenStack: 精确率=0.95, 召回率=0.92, *F*1-score=0.94	利用dLCE模型, 对日志消息模板进行了语义表征, 提取了日志语义信息.
LogRobust[25]	2019	有监督	Drain	事件序列	Bi-LSTM with attention	Original HDFS、synthetic HDFS、Microsoft Service X	Original HDFS: 精确率=0.98, 召回率=1.0, *F*1-score=0.99; Synthetic HDFS: 在原始HDFS数据集上模拟各种不稳定性, 实验表明*F*1-score均能保持在0.90以上; Microsoft Service X: 精确率=0.69, 召回率=0.99, *F*1-score=0.81	基于TF-IDF技术对词向量进行加权求和得到模板语义向量, 并应用注意力机制以对抗日志不稳定性.
Hybrid CAE and VAE framework[26]	2019	半监督	未介绍	事件序列	CAE+VAE	HDFS	HDFS: 精确率=0.98, 召回率=1.0, *F*1-score=0.99	对日志模板进行one-hot编码, 并基于block id生成日志事件序列, 输入CAE和VAE的混合模型提取日志正常模式, 但由于CNN输入要求所有序列长度一致, 该方法对超出一定长度的日志序列进行截断, 使序列信息缺失.

注: **表示论文采用准确率、真正率 (召回率)、假正率作为评价标准, 但为了方便比较, 这里通过计算转换为精确率、召回率和 *F*1-score.

表 A1 (续) 各算法对比详情表

算法	年份	技术分类	日志解析/预处理	特征表示	检测技术/模型	数据集	性能表现 (最佳参数设定)	技术特点分析
LogNL[23]	2020	半监督	Drain	事件序列、变量取值	LSTM	HDFS、OpenStack	HDFS: 精确率=0.96, 召回率=0.94, *F*1-score=0.95; BGL: 精确率=0.97, 召回率=0.94, *F*1-score=0.96	实现日志事件序列和变量取值2种特征异常检测, 同时也考虑了日志语义信息.
LogMerge[24]	2020	半监督	FT-tree	事件序列	CNN+LSTM	WordCount、PageRank、HDFS	HDFS: 精确率=0.73, 召回率=0.84, *F*1-score=0.78; PageRank: 精确率=0.82, 召回率=0.89, *F*1-score=0.86; WordCount: 未给出具体指标, 在跨类型检测算法下, 该数据集上取得了0.80左右的*F*1-score.	论文中HDFS和PageRank分别作为源日志数据, WordCount作为目标日志数据, 作者基于聚类算法, 并结合CNN+LSTM实现了跨日志类型的日志异常检测.
NoTIL[34]	2020	半监督	未介绍	时间间隔	LSTM	OpenStack	OpenStack: *F*1-score=0.45	通过对滑动时间窗口内的日志事件计数来表示该窗口, 并基于LSTM来对日志时间间隔上的异常进行检测.
VeLog[6]	2020	半监督	源码分析	事件序列、事件计数	VAE	HDFS、OpenStack	HDFS: 精确率=0.99, 召回率=0.99, *F*1-score=0.99; OpenStack: 精确率=0.98, 召回率=0.99, *F*1-score=0.99	利用VAE来学习日志的正常模式, 根据重构误差来判别异常.
一种基于GAN的日志异常检测算法[27]	2020	有监督	正则匹配	事件序列	GAN	BGL	BGL: 精确率=0.15, 召回率=0.98, *F*1-score=0.26	通过生成对抗机制来生成逼近真实样本的假样本, 以此获得当前日志序列下后续日志模板的概率分布.但是,基于GAN的模型往往具有不稳定、难以收敛的特点.
ADA[1]	2020	半监督	无模板抽取, 基于标点符号对日志切分, 形成词汇表	事件序列	LSTM	LANL实验室赛博安全日志	LANL实验室赛博安全日志: 精确率=0.98, 召回率=0.92, *F*1-score=0.95	引入在线深度学习技术, 以实时流的方式训练多种层深的模型, 根据当前检测事件是否异常, 自动选择不同深度的模型用于检测.
HitAnomaly[29]	2020	有监督	Dain	事件序列、变量取值	Transformer	HDFS、BGL、OpenStack	HDFS: 精确率=1.0, 召回率=0.97, *F*1-score=0.98; BGL:精确率=0.95, 召回率=0.90, *F*1-score=0.92; OpenStack:精确率=0.84, 召回率=0.92, *F*1-score=0.86	提出了利用分层式的Transformer结构, 分别对日志模板解析后的事件序列和变量取值进行向量化, 基于二者的加权求和来预测结果的概率分布.
PLELog[37]	2021	半监督	Drain	事件序列	GRU with attention	HDFS、BGL	HDFS: 精确率=0.95, 召回率=0.96, *F*1-score=0.96; BGL:精确率=0.97, 召回率=1.0, *F*1-score=0.98	基于日志事件序列的语义向量表示相似性, 利用HDBSCAN聚类算法对部分未知标签的训练数据打标签, 为后续的模型训练提供带有标签的训练数据, 该算法结合了有监督方法和无监督方法的优势.

表 A1 (续) 各算法对比详情表

算法	年份	技术分类	日志解析/预处理	特征表示	检测技术/模型	数据集	性能表现 (最佳参数设定)	技术特点分析
NeuralLog[28]	2021	有监督	无模板抽取, 只删除数字、标点符号的预处理	事件序列	Transformer	HDFS、BGL、Thunderbird、Spirit	HDFS: 精确率=0.96, 召回率=1.0, *F*1-BGL: 精确率=0.98, 召回率=0.98, *F*1-score=0.98; *F*1-score=0.98; Thunderbird:精确率=0.93, 召回率=1.0, *F*1-score=0.96; Spirit:精确率=0.98, 召回率=0.96, *F*1-score=0.97	首次提出了摒弃日志解析过程的新思路, 并基于Transformer来进行异常识别.
LogAD[14]	2021	半监督	结合知识库的组合式日志解析方法	关键词、事件计数、事件序列、量取值、量分布	针对不同特征, 采用关键事词提取、基于LSTM的变多变量时序变异常检测、有限状态自动机等多种组合方法	10种自有的实际系统日志 (Nginx、JVM、DB2等)	在10种实际系统日志集上, LogAD取得了平均0.83的*F*1-score.	深入研究实际场景中日志特征, 提出了组合式的日志异常检测算法以应对多种需求.

(校对责编: 孙君艳)