计算机科学
COMPUTER SCIENCE

基于日志模板主题特征的日志异常检测

孙雪奎, 戴华, 周建国, 杨庚, 陈燕俐

引用本文

孙雪奎, 戴华, 周建国, 杨庚, 陈燕俐基于日志模板主题特征的日志异常检测[J]. 计算机科学, 2023, 50(6): 313-321.

SUN Xuekui, DAI Hua, ZHOU Jianguo, YANG Geng, CHEN Yanli. LTTFAD:Log Template Topic Feature-based Anomaly Detection [J]. Computer Science, 2023, 50(6): 313-321.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220500020

基于日志模板主题特征的日志异常检测

孙雪奎[1] 戴 华[1,2] 周建国[1] 杨 庚[1,2] 陈燕俐[1]
1 南京邮电大学计算机学院 南京 210023
2 江苏省大数据安全与智能处理重点实验室 南京 210023
(kasonsun@foxmail.com)

摘 要 在系统安全领域,通过日志来检测软件或者系统异常是一种常用的安全防护手段。随着软件和硬件的快速发展,在大规模的日志记录上进行人工标记变得十分困难,目前已有大量的日志异常检测的相关研究。现有的自动化日志检测模型均使用日志模板作为分类,这些模型的性能以及实用性很容易受到日志模板变化的影响。因此,基于日志模板主题特征的日志异常检测模型LTTFAD被提出,LTTFAD首次引入了LDA主题模型以提取日志模板的主题特征并且通过循环神经网络LSTM实现异常检测。实验结果表明,在HDFS和OpenStack数据集上基于日志模板主题特征的日志异常检测模型LTTFAD的查准率、查全率和调和分数等性能指标均明显优于现有基于日志模板的日志异常检测模型。此外,对于新日志模板的注入,LTTFAD模型依然具有较高的稳定性。

关键词:异常检测;日志分析;深度学习;LDA;主题特征
中图法分类号 TP391

LTTFAD:Log Template Topic Feature-based Anomaly Detection

SUN Xuekui[1],DAI Hua[1,2],ZHOU Jianguo[1],YANG Geng[1,2] and CHEN Yanli[1]
1 School of Computer Science,Nanjing University of Posts and Telecommunications,Nanjing 210023,China
2 Jiangsu Key Laboratory of Big Data Security & Intelligent Processing,Nanjing 210023,China

Abstract In the field of system security,using logs to detect software of system anomalies is a very popular method. With the rapid development of software and hardware,it is hard to perform manual detection on the huge scale of logs. There has been a lot of researches on log anomaly detection. Existing automatic log anomaly detection approaches are all based on log template,which is unstable when log template is modified. This paper proposes a log anomaly detection model based on topic feature of log template. Firstly,it utilizes an LDA topic model to extract topic feature of log template and implements anomaly detection through LSTM recurrent neural network. Experimental results show that the proposed anomaly detection model outperforms the existing models on HDFS and OpenStack datasets in most metrics,such as the precision,recall and F1 Score. In addition,LTTFAD model still has high stability for new log template injection.

Keywords Anomaly detection,Log analysis,Deep learning,LDA,Topic feature

1 引言

在系统安全领域,通过日志来检测软件或系统异常是一种常用的安全防护手段。从简单小型的软件系统到大型复杂的软件系统,以及分布式文件系统和高性能的云计算管理平台不可避免地存在漏洞,这就可能导致系统本身运行的异常,此外,攻击者也可能利用软件和系统的漏洞发起危险性攻击来入侵破坏系统。因此,准确地检测出这些异常对于构建安全可信的系统来说至关重要,但现有异常检测模型无法正确地学习正常和异常日志之间的语义差异,导致异常检测模型的泛化能力较差,在实际的应用中并没有取得很好的效果。

日志几乎是所有计算机系统中异常检测模型常见且主要的数据源,它记录了一系列描述软件和系统运行状态的重要事件。现有的以日志为数据源实现异常检测的模型可以被

到稿日期:2022-05-03 返修日期:2022-09-07
基金项目:国家自然科学基金面上项目(61872197,61972209,61902199,61771251);中国博士后自然科学基金(2019M651919);南京邮电大学自然科学基金(NY217119,NY219142)
This work was supported by the National Natural Science Foundation of China(61872197,61972209,61902199,61771251),Postdoctoral Science Foundation of China(2019M651919) and Natural Science Foundation of Nanjing University of Posts and Telecommunications(NY217119,NY219142).
通信作者:戴华(daihua@njupt.edu.cn)

概括为两大类:第一大类是传统基于统计方法或基于规则的机器学习模型;第二大类是基于深度神经网络的深度学习模型。传统的机器学习模型[1-10]在特定应用场景下能够取得很好的效果,但并不能用于检测不同的攻击。深度学习模型[11-13]使用日志模板进行分类,以学习日志序列内的行为模式。但目前基于深度学习的模型无法准确学习日志的语义关系特征,并且大多数模型没有考虑对新增日志类型的处理,模型的鲁棒性不强,此外异常检测模型查准率、查全率、调和分数等相关性能还需进一步提高,以适应复杂多变的软件和系统。

本文所提出的基于日志模板主题特征的 LTTFAD 异常检测模型,能够有效地通过非结构化日志记录检测出进程或系统的异常行为,该模型包含模型训练和异常检测两个阶段。模型训练阶段,原始日志数据集使用日志解析器转换为日志模板集合和结构化日志集合,日志模板集合将会被构造成语料库,用于训练 LDA 主题模型,结构化日志集合生成的日志模板序列会匹配生成对应的日志模板主题序列,进而用于训练 LSTM[14]分类模型;异常检测阶段,使用训练好的 LSTM 模型进行日志异常检测。在大型分布式文件系统 HDFS 和开源的云计算管理平台 OpenStack 的日志记录数据集上进行实验,结果表明,LTTFAD 异常检测模型在查准率、查全率、调和分数上均明显优于现有基于深度学习模型的 Deeplog[11],LogAnomaly[12]以及 RobustLog[13]。

本文的主要贡献如下:

(1)基于 LDA 主题模型构造了日志模板主题分类模型 LTTCM,该模型能够将日志模板转换为最相关的日志模板主题;并且整合了长短期记忆循环神经网络训练生成异常检测分类模型。

(2)基于以上两种模型,提出了日志模板主题匹配算法和异常检测算法,构造了基于日志模板主题特征的日志异常模型 LTTFAD。LTTFAD 是目前率先使用日志模板主题作为分类的日志异常检测模型。

(3)在新日志模板注入场景下,基于日志模板主题特征进行异常检测的 LTTFAD 模型依然能够保持相对较优的检测性能和鲁棒性。

(4)使用大型分布式系统和开源云计算管理平台的日志记录文件作为数据集进行对比实验,结果表明,本文模型明显优于现有的深度学习异常检测模型。

本文第 2 讨论现有的相关工作;第 3 节对本文模型相关的预备知识进行引入;第 4 节介绍了 LTTFAD 模型的相关定义;第 5 节给出了 LTTFAD 模型系统的框架以及实现日志异常检测的详细流程和算法实现;第 6 节通过实验来验证本文模型的有效性;最后总结全文。

2 相关工作

系统日志的初衷是记录系统的运行状态,系统日志包含很多有价值的信息,可以方便用户调试系统并且可以用来跟踪和分析系统。由于日志是半结构化的字符串,在不经过任何预处理的情况下,对其进行异常检测是非常困难的。

针对不同的系统软件环境,大量的日志挖掘工具被设计出来以实现异常检测。尽管一些基于规则的模型[15-22]能够获得较高的准确率,但这些模型往往在很大程度上受到应用场景的限制,并且需要领域专家制定相关的规则。例如,Oprea 等[17]使用信念传播从 DNS 日志中检测早期阶段的企业感染;PerfAugur[20]基于特征挖掘服务日志以检查性能问题。

日志异常检测模型实现通常可以拆分为 3 个步骤:首先,使用日志分析器[23-30]将日志记录拆分为日志模板和参数,日志模板用于描述系统或者进程的行为,而参数部分记录着状态信息,例如时间戳和进程标识符;其次,使用日志记录的时间戳和日志模板构造出系统或进程的行为序列;最后,基于行为序列对产生该序列的系统或者软件进行检测。

传统的机器学习日志异常检测方法包括有监督的机器学习方法和无监督的机器学习方法,有监督的机器学习方法如支持向量分类器(SVM)[1-2]、线性回归(LR)[3-4]、决策树(DT)[5]、K-邻近算法(KNN)[6]。这类方法基于日志频度统计向量记录每个日志模板在日志序列内出现的频度,将频度统计向量作为输入,以二分标签为分类结果。Liang 等[1]和 Wang 等[2]实现了基于 SVM 的日志异常检测方法。SVM 是一种具有完备理论支持的小样本学习方法,其通过线性分类器对数据进行二分类,实际上就是找到能够将样本分开的最优超平面,最大化分类边界,相比一般的二分类方法更简单并且泛化性更强。但是 SVM 难以适用于大规模的数据集并且难以用于实现多分类问题,因此在大规模的日志数据上表现并不理想。Breier 等[3]和 He 等[4]基于线性回归提出了基于数据挖掘的日志异常检测和大规模自动化的日志分析方法。线性回归被广泛用于解决二分类问题,具有可验证性,通常在大规模的数据集上也能保证效率,但在具体应用上需要预判断数据之间的线性关系,这样的预判往往并不能很好地拟合数据之间的非线性关系。Chen 等[5]提出了基于决策树的故障诊断方法,通过构造自上而下的数据树形结构来描述一般判断过程,利用多级判断来决策目标数据的真实类别。日志数据的日志键作为树中每个节点的特征,通过选取最佳特征即具有最大信息增益的特征作为分类特征,递归节点分支以构造完整的决策树实现异常检测。但 DT 忽略了数据特征间的相关性,在样本数据不均匀时决策模型表现不佳。Ying 等[6]提出了一种改进的基于 KNN 的具有自动标记样本的高效日志异常检测方法,拟合原有数据样本点进行类别划分形成簇,对于新来的样本点,计算当前点到每个簇中心的距离,通过 KNN 拟合得到的最佳阈值来判断当前样本点异常与否。日志异常检测过程中需要大量的正常日志数据来进行学习,样本不均衡时,预测往往偏向数量占优的一方,容易出现预测错误。此外基于 KNN 的异常检测方法相比其他方法计算时间和内存消耗更高。

无监督的机器学习方法有主成分分析(PCA)[7]和基于聚类的方法,如孤立森林(IF)[8]、不变量挖掘(IM)[9]、日志聚类(LC)[10]。这类方法使用无标签的数据进行训练,无监督即可

实现日志异常检测。Xu 等[7]基于 PCA 实现通过挖掘控制台日志来检测大规模系统问题，对高维特征数据降维，去除数据噪声和不重要的数据特征，留下相对重要的特征来提高数据处理效率，同时基于训练日志数据生成正常空间和异常空间。对于新的日志，使用投影函数可以得到投影值，若投影值小于某个阈值，则认为日志是正常的。PCA 中主成分的解释模糊，并且降维仅仅只针对线性数据。Xu 等[8]提出了一种基于孤立森林(IF)的改进数据异常检测方法，能够快速发现连续型结构化数据中的异常点，若一些日志数据远离正常日志数据空间，即正常的日志数据点通常以高密度聚集在某个区域，而异常的数据点通常会稀疏地分布，则认为这些稀疏点对应的日志是异常的。在大规模的数据集上会削弱 IF 发现异常点的能力，因为较多样本会干扰隔离过程，不利于隔离出异常点。Lou 等[9]实现从控制台日志中挖掘不变量来检测系统问题。不变量挖掘(IM)需要构造特征矩阵，进行复杂的矩阵变换生成日志向量不变空间，对于不满足不变空间的日志向量，则认为其是异常的执行。Vaarandi 等[10]提出了一种事件日志的数据聚类和模式挖掘算法。将预处理后的训练数据采用聚类法划分为正常聚类簇和异常聚类簇，聚类簇的质心为簇的向量表示，对于新的日志序列，如果其最近的聚类簇是正常的，则认为该序列为正常。当日志类别间差距较大时，日志聚类性能会受到影响。

现有的深度学习日志异常检测方法包括基于递归神经网络(RNN)[11-13]的方法，如 DeepLog，LogAnomaly，RobustLog。这类方法将日志作为自然语言处理的文本，使用具有时间序列长短期信息记忆的 LSTM 模型实现日志异常检测。DeepLog[11]基于日志键和参数训练 LSTM 得到日志键异常检测模型和参数值异常检测模型，两个模型的组合实现异常检测，具有开创意义。但其中的日志键实际上就是日志模板的索引，没有考虑到日志模板的具体语义信息，并且基于日志键的检测需要在检测之前知道日志模板集合的大小，当日志模板发生更新或者出现新增时，模型可能会失效。LogAnomaly[12]基于词向量的同义和反义提出了 template2vec，并使用带有注意力机制的 Bi-LSTM 模型结合日志模板特征和模板内单词特征来得到日志模板语义特征空间向量。当日志模板发生更新时，LogAnomaly 会计算得到该日志模板的语义特征空间向量，并通过选取与其欧氏距离最近的现有的日志模板进行替换，但在日志模板新增较多时性能会急剧下降。RobustLog[13]是针对现有日志异常检测方法对新增日志的处理的不稳定性而提出的一种鲁棒性的深度学习异常检测方法，该方法通过分词、词嵌入并结合 TF-IDF 将日志模板转换为日志模板语义向量，以此训练带有注意力机制的 Bi-LSTM 来实现异常检测。RobustLog 虽然能够在一定程度上解决对于新增日志的鲁棒性问题，但是计算量相对较大，并且在大规模的日志数据上训练很慢。此外，还有基于卷积神经网络的日志异常检测方法。

实际上，现有日志异常检测方法中，无论是传统基于统计方法或基于规则的机器学习方法，还是现有的将日志作为自然语言处理序列的基于神经网络的深度学习方法，都或多或少存在不足。基于统计的方法和基于规则的机器学习方法通常会有应用场景的限制，而基于深度学习的方法大都没有考虑到日志语义特征以及方法对新增日志处理的鲁棒性，因此本文提出了基于日志模板主题特征的日志异常检测方法。

3 预备知识

3.1 日志解析器

系统产生的日志记录是无结构的字符串，需要将日志记录转换为结构化日志以方便日志异常检测模型学习到日志的行为模式。日志解析器能够将系统日志记录分离成日志模板和参数向量两个部分，并将原始日志转换为结构化日志(见图 1)。目前，已有很多准确高效的开源日志解析器，例如 IPLoM[23]，LogSig[24]，Logmine[25]，Spell[26]，Drain[27]，MoLFI[28]和 SHISO[29]等。LTTFAD 模型使用日志解析器提取日志模板集合并构造结构化日志集合，实现对原始日志数据集的预处理，生成训练数据集和异常检测数据集。

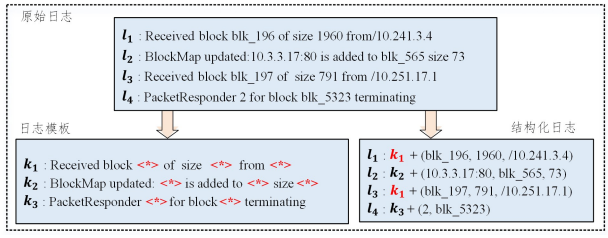

图 1 日志解析示例

Fig. 1 Log parsing

3.2 LDA 主题模型

LDA(Latent Dirichlet Allocation)主题模型[31]是一个 3 层贝叶斯概率模型，扩展到 pLSI(概率潜在语义索引)，即可以用于主题发现的概率图模型。LDA 主题模型将主题视为文本和关键词之间的连接，每个文本都是隐含主题的混合体，并且文本和每个主题的相关度可以用概率分布向量来表示。此外，LDA 是一个无监督的主题模型，基于设定的主题数量和模型输出的文本-主题概率分布向量得到文本主题，日志是文本，日志对应的日志模板也是文本，基于日志模板语料训练的 LDA 模型可以获得每个日志模板对应的日志模板主题。

本文将日志模板视为文本，使用日志模板作为语料训练 LDA 主题模型，并将日志模板 k_i 作为语料输入 LDA 模型，以获得日志模板 k_i 所对应的主题概率分布向量 $\boldsymbol{\Theta} \in R^{1\times r}$，向量 $\boldsymbol{\Theta}$ 内的最大概率值所对应的日志模板主题 t_j 即为 k_i 最符合的日志模板主题。基于 LDA 的日志模板主题分类模型即为 LTTCM(Log Template Topic Classification Model)。

由此可见，LDA 模型将日志模板转换为日志模板主题，实际上就是 LDA 模型进行自然语言处理的下游任务。LDA 模型既可以提取到含有语义信息的日志模板主题，同时对于新增日志模板也可以基于上述的日志模板主题匹配算法(具体算法过程见 5.2 节)得到最相关的日志模板主题，用于日志模板主题映射字典的更新，因此 LDA 模型相比其他模型更适用于日志异常检测。

3.3 LSTM 神经网络模型

LSTM(Long Short Term Memory)[14]是循环神经网络的一种实例模型,它改善了传统循环神经网络存在的长期依赖问题。循环神经网络在自然语言处理领域有着广泛的应用,其最大的优势是能够很好地学习句子内单词的上下文关系模式。由于日志行为同样具有上下文关系,因此可以使用循环神经网络模型学习进程的行为模式,根据已有的日志行为序列片段预测出下一时刻可能出现的日志行为。本文基于 LSTM 网络构造分类模型,以实现日志行为的异常检测。

4 基于日志模板主题特征的异常检测模型(LTTFAD)

假设原始日志集合为 $L=\{log_1, log_2, \cdots, log_n\}$,利用 3.1 节中的日志模板解析得到对应的结构化日志集合为 $D=\{d_1, d_2, \cdots, d_n\}$和日志模板集合为 $K=\{k_1, k_2, \cdots, k_m\}$。

定义 1(结构化日志,Structured Log) 原始日志数据集 L 对应的结构化日志集合为 D,d_i 为 log_i 对应的结构化日志三元组(k, pid, ts),其中 k 为日志模板,pid 为进程标识符,ts 为日志产生的时间戳。

定义 2(日志模板主题,Log Template Topic,LTT) 将 K 中的日志模板输入基于 LDA 的 LTTCM 模型(该模型将在 5.1 节给出),得到对应的日志模板主题集合 $T=\{t_1, t_2, \cdots, t_r\}$,且$|T| \leqslant |K|$。

定义 3(进程日志模板主题序列,LTT Sequence of Process) 对于进程 p_i,设其产生的日志所对应结构化日志序列为 $D_i=\langle d_{i,1}, d_{i,2}, \cdots, d_{i,q}\rangle$,满足 $d_{i,j} \in D_i \wedge D_i \subseteq D \wedge d_{i,j}.ts < d_{i,j+1}.ts (1 \leqslant j \leqslant q)$,则 D_i 对应的进程日志模板主题序列为 $S_i=\langle e_1, e_2, \cdots, e_q\rangle$,其中 $e_j \in T$ 是由日志模板主题匹配算法(具体过程见算法 2)得到的日志模板主题。

定义 4(日志模板主题滑动窗口,LTT Sliding Window)设滑动窗口大小阈值为 h,进程 p_i 的进程日志模板主题序列为 $S_i=\langle e_1, e_2, \cdots, e_q\rangle$,则 $e_{j+1} \in S_i$,e_{j+1} 对应的滑动窗口记为 $Win(S_i, e_j)$,生成方法如下:

(1)若$(h \leqslant j < q)$,则 $Win(S_i, e_j)=\langle e_{j-h+1}, e_{j-h+2}, \cdots, e_j\rangle$;

(2)否则,$Win(S_i, e_j)=\emptyset$。

此外,若满足条件(1),进程 p_i 的日志模板主题序列 S_i 可以得到的 $q-h$ 个滑动窗口形成的集合记为$W_{s_i}=\{Win(S_i, e_j) \mid e_j \in S_i \wedge j \in [h, q)\}$;$W_{s_i}$ 对应的 $q-h$ 个日志模板主题 e_{j+1} 构成的集合记为$T_{e_{j+1}}=\{e_{j+1} \mid e_{j+1} \in S_i \wedge j \in [h, q)\}$。

定义 5(日志模板主题概率分布向量,LTT Probability Distribution Vector) 将进程 p_i 的日志模板主题滑动窗口集合W_{s_i} 输入 LSTM 分类模型,对于任何一个滑动窗口$Win(S_i, e_j)$,其输出为下一时刻的日志对应的日志模板主题 e_{j+1} 在 T 中各日志模板主题的概率所形成的向量,称为日志模板主题概率分布向量,记为 $\boldsymbol{V}_{e_{j+1}}$,满足 $\boldsymbol{V}_{e_{j+1}} \in R^{1\times r}$,其中任一维的数值 $\boldsymbol{V}_{e_{j+1}}[k](1 \leqslant k \leqslant r)$ 表示 e_{j+1} 为 T 中日志模板主题 t_k 的概率。进程 p_i 对应的日志模板主题概率分布向量集合为 $\boldsymbol{V}=\{\boldsymbol{V}_{e_{j+1}} \mid e_{j+1} \in S_i \wedge j \in [h, q)\}$。

定义 6(候选集,Candidate Sets) 对于进程 p_i,设其日志模板主题概率分布向量集合为 $\boldsymbol{V}=\{\boldsymbol{V}_{e_{j+1}} \mid e_{j+1} \in S_i \wedge j \in [h, q)\}$,取任一日志模板主题概率分布向量$V_{e_{j+1}}$ 内最大的 g 个概率值所对应的日志模板主题构成的集合为 e_{j+1} 的候选集,记为$CS_{e_{j+1}}$,满足$CS_{e_{j+1}} \subseteq T$,则进程 p_i 得到的候选集集合 $CS=\{CS_{e_{j+1}} \mid e_{j+1} \in S_i \wedge j \in [h, q) \wedge CS_{e_{j+1}} \subseteq T\}$。

定义 7(进程日志异常检测,Process Log Anomaly Detection) 设进程 p_i 的进程日志模板主题序列为 $S_i=\langle e_1, e_2, \cdots, e_q\rangle$,由 S_i 生成的 $q-h$ 个滑动窗口集合为$W_{s_i}=\{Win(S_i, e_j) \mid e_j \in S_i \wedge j \in [h, q)\}$,$W_{s_i}$ 对应的 $q-h$ 个日志模板主题构成的集合为$T_{e_{j+1}}$,$q-h$ 个日志模板主题的候选集集合为 CS,则进程 p_i 的异常判定方法如下:

(1)若 $\forall e_{j+1} \in T_{e_{j+1}}$,满足 $e_{j+1} \in CS_{e_{j+1}}$,则进程 p_i 无异常;

(2)否则,进程 p_i 异常。

根据上述定义,LTTFAD 模型由日志解析器得到原始日志对应的结构化日志和日志模板,利用基于 LDA 的 LTTCM 模型构建的日志模板主题匹配算法将日志模板转化为语义相关的日志模板主题,并且能够通过提取日志模板的固有属性特征以及日志模板序列的上下文语义特征来学习日志的行为模式,相比现有的基于深度学习的模型方法更能够捕捉到日志语义相关的信息,这种学习模式在理论上能够提高日志异常检测的准确率。第 6 节将给出实验,以验证本文模型的可行性和有效性。

5 LTTFAD 算法的实现

基于日志模板主题特征的日志异常检测模型 LTTFAD 包含模型训练和异常检测两个阶段(见图 2)。在模型训练阶段,模型将会从被专家标记为无异常的原始日志记录中学习正常的行为模式;在异常检测阶段,根据已有的进程日志序列片段做出行为预测,通过对比预测结果和实际进程日志,来判断该进程的行为是否存在异常。该过程的大致流程如下:

(1)模型训练阶段。日志解析器将原始日志集合解析为结构化日志集合和对应的日志模板集合。日志模板集合用于训练基于 LDA 的日志模板主题分类模型(LTTCM)。在 LTTCM 训练完成后,根据结构化日志三元组(见定义 1)中的进程 pid 划分,将结构化日志集合划分为多个子集,并根据结构化日志记录的时间戳 ts 对所有子集内的结构化日志排序,生成进程结构化日志序列集合。日志模板主题匹配算法将进程结构化日志序列集合转换为对应的进程日志模板主题序列集合。最后,输入 LSTM 训练生成 LSTM 分类模型。

(2)异常检测阶段。使用日志解析器将进程日志序列转换为进程结构化日志序列;再调用日志模板主题匹配算法将进程结构化日志序列转换为对应的进程日志模板主题序列;将序列输入基于 LSTM 分类模型的异常检测算法,对该进程日志模板主题序列进行异常检测。

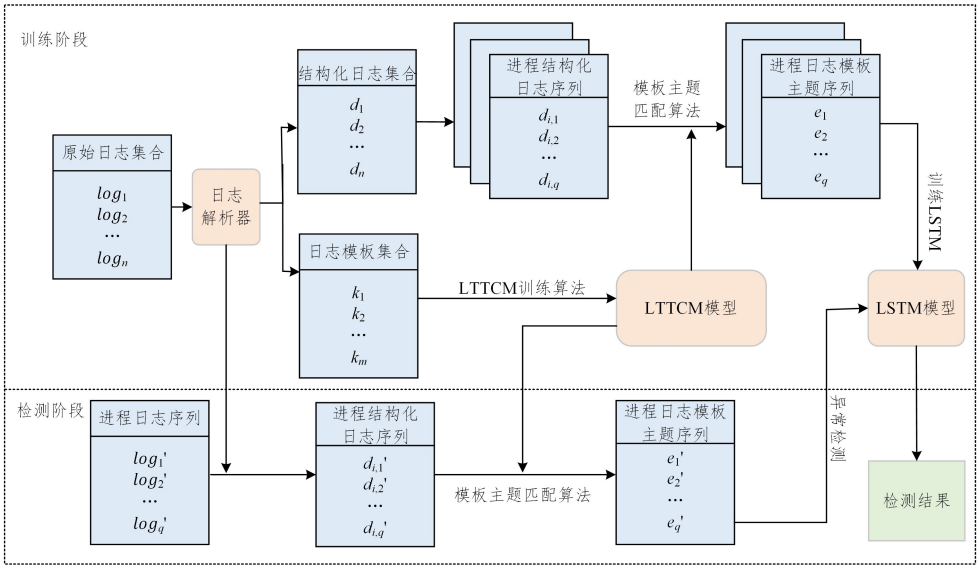

图 2　LTTFAD 的异常检测模型

Fig. 2　Anomaly detection model for LTTFAD

5.1　LTTCM 训练算法

在进行日志模板主题匹配之前，需要对 LDA 模型进行训练，以生成 LTTCM 模型。将日志解析器提取到的日志模板集合视为语料库，对于每一个日志模板 k_i，首先以单词为单位进行拆分，日志模板 k_i 会被拆分为单词列表(Word List，WL)；然后对 WL 内的每一个单词转换为小写、过滤停用词和无语义标识符；最后，使用词袋模型(Bag of Word，BoW)将单词列表 WL 转化为对应的语料 $corpus$，并将其添加到语料列表(Corpus List，CL)中；将 CL 作为语料训练 LDA 主题模型，生成 LTTCM 模型。LTTCM 训练过程如算法 1 所示。

算法 1　LTTCM 训练算法

输入：日志模板集合 K，日志模板主题数量 r

输出：LTTCM 模型

1. 初始化语料列表 CL=∅；
2. 初始化映射字典 TD=∅；
3. FOR k_i∈K DO
4.　将 k_i拆分成单词列表 WL；
5.　FOR EACH word IN WL DO
6.　　lowerCase(word)；
7.　　IF word 为停用词或者无语义标识符 THEN
8.　　将 word 从 WL 内移除；
9.　　END IF；
10.　END FOR；
11.　使用词袋模型(BoW)将单词列表 WL 转换成语料 corpus；
12.　将语料 corpus 加入语料列表 CL；
13. END FOR；
14. 使用语料列表 CL 训练 LDA 主题模型→LTTCM 模型；
15. RETURN LTTCM。

5.2　日志模板主题匹配算法

在 LSTM 模型训练和异常检测之前，需要调用 LTTCM 模型把进程日志模板序列内的每一个日志模板 k_i转换为日志模板主题 $t_j \in T$(见定义 2)。首先，查询映射字典(Topic Dictionary，TD)，若不存在日志模板 k_i和 t_j之间的映射，则调用 LTTCM 模型计算出 k_i和 t_j的映射。具体映射的步骤为：将日志模板 k_i转换为语料 $corpus$，输入 LTTCM 模型获取该日志模板的主题概率分布向量 $\boldsymbol{\Theta} \in R^{1\times r}$，$\Theta[l]$表示日志模板 k_i的主题为日志模板主题 t_l的概率值。选定概率分布向量 $\boldsymbol{\Theta}$ 内最大概率值所对应的日志模板主题 t_j作为日志模板 k_i的匹配结果；将映射关系$\{k_i \rightarrow t_j\}$添加到映射字典 TD 中并返回 t_j。具体日志模板主题匹配过程如算法 2 所示。

算法 2　日志模板主题匹配算法

输入：日志模板 k_i

输出：日志模板主题 t_j

1. IF k_i IN TD THEN
2.　RETURN TD(k_i)；
3. END IF；
4. 对 k_i进行预处理得到 corpus，输入 LTTCM
5. $\boldsymbol{\Theta}$=LTTCM(corpus)；
6. t_j=MaxIndexOf($\boldsymbol{\Theta}$)；/* 获取 $\boldsymbol{\Theta}$ 最大概率值对应的日志模板主题 */
7. 将日志模板主题映射$\{k_i \rightarrow t_j\}$加入映射字典 TD；
8. RETURN t_j。

5.3　LSTM 训练算法

LTTCM 训练完成后，还需要训练 LSTM 以实现正常日志行为模式的学习。在 LSTM 训练过程中，首先由日志模板主题序列(见定义 3)$S_i=\langle e_1, e_2, \cdots, e_q\rangle$生成日志模板主题滑动窗口 $Win(S_i, e_j)$序列，构造训练对(Topic Pair，TP)并将其添加到训练对列表(Topic Pair List，TPL)；最后，使用训练对列表 TPL 训练 LSTM 模型。LSTM 训练过程如算法 3 所示。

算法 3　LSTM 训练算法

输入：进程日志模板主题序列集合 $S=\{\langle e_{1,1}, e_{1,2}, \cdots, e_{1,q1}\rangle, \langle e_{2,1}, e_{2,2}, \cdots, e_{2,q2}\rangle, \cdots, \langle e_{f,1}, e_{f,2}, \cdots, e_{f,qf}\rangle\}$，日志模板主题滑动窗口的长度 h

输出：已训练好的 LSTM 模型

1. 初始化训练对列表 TPL=∅；
2. FOR S_i∈S∧i∈[1，f] DO
3.　FOR j=h，h+1，…，q−1 DO

4. 根据定义 4 生成日志模板主题滑动窗口 Win(S_i,e_j);
5. IF Win(S_i,e_j)=Ø THEN
6. CONTINUE;
7. END IF;
8. 生成训练对 TP=(Win(S_i,e_j),e_{j+1})并加入训练对列表 TPL;
9. END FOR;
10. END FOR;
11. 使用训练对列表 TPL 作为训练集,训练 LSTM 模型;
12. RETURN LSTM。

5.4 异常检测算法

在检测过程中,先由进程 p_i 生成 S_i 所包含的 $q-h$ 个日志模板主题滑动窗口对应的集合 W_{s_i} 以及 W_{s_i} 对应的 $q-h$ 个日志模板主题构成的集合 $T_{e_{j+1}}$(见定义 4)。每一个日志模板主题滑动窗口 $Win(S_i,e_j)$ 都会被输入 LSTM,并得到对应的日志模板主题概率分布向量 $\boldsymbol{V}_{e_{j+1}}$(见定义 5),由 $\boldsymbol{V}_{e_{j+1}}$ 中最大的 g 个概率值所对应的日志模板主题集合获得日志模板主题 e_{j+1} 的候选集 $CS_{e_{j+1}}$,进而得到 $q-h$ 个日志模板主题候选集构成的集合 CS(见定义 6)。最后,由进程日志异常检测算法(见定义 7)来判断进程是否异常,若 $\forall e_{j+1}\in T_{e_{j+1}}$,满足 $e_{j+1}\in CS_{e_{j+1}}$,则进程 p_i 无异常;否则进程 p_i 异常。具体的异常检测算法如算法 4 所示。

算法 4 异常检测算法

输入:进程日志模板主题序列 $S_i=\langle e_1,e_2,\cdots,e_q\rangle$,日志模板主题滑动窗口大小 h,候选集大小 g

输出:检测结果(TRUE 表示正常,FALSE 表示异常)

1. FOR j=h,h+1,…,q−1 DO
2. 根据定义 4 生成日志模板主题滑动窗口 Win(S_i,e_j);
3. IF Win(S_i,e_j)=Ø THEN
4. CONTINUE;
5. END IF;
6. 将 Win(S_i,e_j)输入 LSTM 模型得到 e_{j+1} 日志模板主题概率分布向量 $\mathbf{V}_{e_{j+1}}$;
7. 取 $\mathbf{V}_{e_{j+1}}$ 中最大的 g 个概率值所对应的日志模板主题构成候选集 $CS_{e_{j+1}}$;
8. IF $e_{j+1}\notin CS_{e_{j+1}}$ THEN
9. RETURN FALSE;
10. END IF;
11. END FOR;
12. RETURN TRUE。

上文详细介绍了基于日志模板主题特征的日志异常检测算法步骤,其核心在于 LTTCM 模型的生成、日志模板主题的匹配以及具体的异常检测的判定过程。

6 实验评估

本节将基于 LTTFAD 的日志异常检测算法流程进行实验,下面对实验评价指标、实验环境与超参数以及实验数据集进行具体说明,并通过分析实验结果如查准率、查全率、调和分数以及不同参数对模型的影响来验证 LTTFAD 的有效性。

6.1 实验设置

6.1.1 实验评价指标

(1)误报(False Positive,FP):检测的日志序列中,正常日志被判定为异常的日志序列数量,记为 FP。

(2)漏报(False Negative,FN):检测的日志序列中,异常日志被判定为正常的日志序列数量,记为 FN。

(3)查准率(Precision):被正确检测出的异常日志序列数量,占所有检测为异常的日志序列数量的比例,用 TP 表示被正确检测出的异常日志序列数量。其计算式如下:

$$Precision=\frac{TP}{TP+FP}$$

(4)查全率(Recall):被正确检测出的异常日志序列数量占所有确定异常的日志序列数量的比例。其计算式如下:

$$Recall=\frac{TP}{TP+FN}$$

(5)调和分数(F1 Score):查准率和查全率的调和均值,记为 FS。其计算式如下:

$$FS=\frac{2\times Recall\times Precision}{Recall+Precision}\times 100\%$$

6.1.2 实验环境与超参数

实验设备操作系统采用 Windows 10 64 位,CPU 为 AMD Ryzen 5 3600 4.20GHz 六核心十二线程,内存大小为 32GB,GPU 为 Nvidia GTX 1660S。实验对比方法为 Deeplog,LogAnomaly 以及 RobustLog,LSTM 模型使用 Pytorch 1.4 框架实现,LDA 主题模型采用的是 gensim 库集成的 LDA。实验超参数的设置如表 1 所列。

表 1 实验超参数设置

Table 1 Hyperparameter settings

超参数	数值
Learning rate(学习率)	0.001
Batch size(训练批量大小)	2048
Epoch(训练迭代次数)	300
l(网络层数)	2
u(神经网络隐藏层大小)	64
g(候选集大小)	11
h(滑动窗口大小)	10
r(主题数量)	24

6.1.3 实验数据集

实验数据集包含 HDFS 和 OpenStack,HDFS 数据集源自 200 多个 Amazon EC2 节点,该数据集共有 575 061 个 block,其中共有 16 838 个 block 被 Hadoop 领域的专家标记为异常;OpenStack 数据集共有 1 335 318 条日志记录,并且被注入了 3 种类型的异常。现有日志解析器性能研究实验[32]的结果表明,日志解析器 Spell 和 IPLoM 可以分别准确地处理 HDFS 和 OpenStack 日志数据集。因此,在本实验中,采用 Spell 和 IPLoM 日志解析器分别对 HDFS 和 OpenStack 日志数据集进行分析处理。同时,为了保证实验的准确性,去除了日志数据集内重复的日志序列,具体数据集信息如表 2 所列。

表 2 数据集描述

Table 2 Datasets description

日志数据集	进程日志序列数量			日志模板数量
	训练集	正常测试集	异常测试集	
HDFS	4855	14177	4123	46
OpenStack	386	1064	138	40

6.2 实验结果

为了验证 LTTFAD 的有效性,我们选择 Deeplog,Log-

Anomaly以及 RobustLog 进行对比实验，实验结果如图 3、图 4 和表 3、表 4 所示。此外，实验还对 LTTFAD 的稳定性进行了验证，如图 5—图 9 所示。

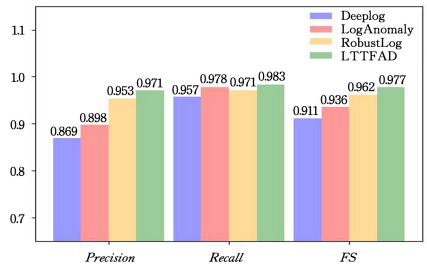

图 3 HDFS 上各方法的准确率

Fig. 3 Accuracy of each method on HDFS

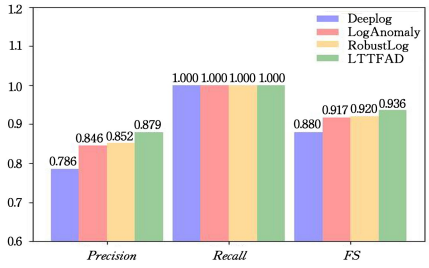

图 4 OpenStack 上各方法的准确率

Fig. 4 Accuracy of each method on OpenStack

表 3 HDFS 数据集上误报和漏报的数量统计

Table 3 Number of FPs and FNs on HDFS log

	Deeplog	LogAnomaly	RobustLog	LTTFAD
误报(FP)	596	460	198	122
漏报(FN)	177	91	120	71

表 4 OpenStack 数据集上的误报和漏报的数量统计

Table 4 Number of FPs and FNs on OpenStack log

	Deeplog	LogAnomaly	RobustLog	LTTFAD
误报(FP)	38	26	24	19
漏报(FN)	0	0	0	0

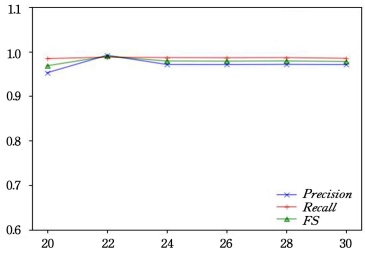

图 5 日志模板主题数量：r

Fig. 5 Number of LTTs：r

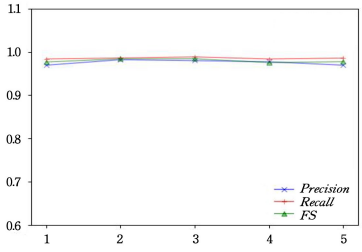

图 6 神经网络层数：l

Fig. 6 Number of layers：l

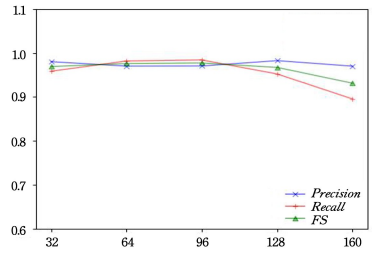

图 7 神经网络隐藏层大小：u

Fig. 7 Hidden layer size：u

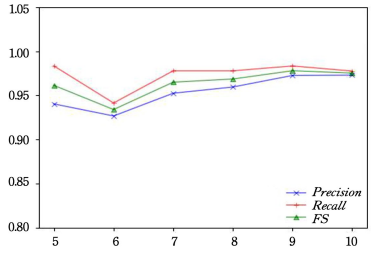

图 8 日志模板主题滑动窗口长度：h

Fig. 8 Window size：h

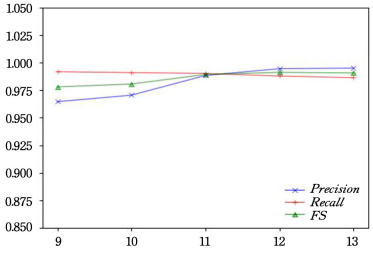

图 9 候选集大小：g

Fig. 9 Candidate sets size：g

6.2.1 查准率、查全率和调和分数

图 3 和图 4 分别给出了 LTTFAD 和 Deeplog，LogAnomaly以及 RobustLog 在 HDFS 和 OpenStack 数据集上的 Precision，Recall 和 FS 的性能指标。LTTFAD 在 HDFS 和 Openstack 数据集上各项性能指标均有明显提升。在 HDFS 数据集上的调和分数 FS 指标，相比 Deeplog，LogAnomaly 和 RobustLog 分别提升了 6.6%，4.1%和 1.5%；而在 OpenStack 数据集上，查准率 Precision 相比之前最高查准率的 RobustLog 提升了 2.7%，调和分数 FS 相比之前的 3 种模型则分别提升了 5.6%，1.9%和 1.6%。实验中 LTTFAD 所使用的 LSTM 模型与 Deeplog 一致，LogAnomaly 和 RobustLog 使用的是带有注意力机制的 Bi-LSTM，实际上 4 种模型都是基于 LSTM 实现的异常检测，但是基于 LDA 模型的 LTTCM 模型提取日志模板的主题特征，相比日志模板本身，能够捕捉到更重要的语义特征，并且对于新增的日志类型也能够基于 LTTCM 模型和日志模板主题匹配算法匹配得到含有语义信息日志模板主题，因此 LTTFAD 可以提升异常检测查准率、查全率和调和分数，能够表现出更高的检测性能。

6.2.2 误报和漏报统计

表 3、表 4 列出了不同方法模型在 HDFS 数据集和 OpenStack 数据集上的误报和漏报的统计。相比现有模型，LTTFAD 能够在很大程度上减少误报和漏报这两种错误出现的

次数，在 HDFS 数据集和 OpenStack 数据集上误报和漏报数量分别为 122，19 和 71，0，均低于现有的 Deeplog，LogAnomaly 和 RobustLog。在实际应用中，误报意味着正常的日志被检测为异常，漏报意味着异常的数据被检测为正常，较多的误报将会导致系统用户被频繁且无意义地通知，而较少的误报能够提高异常检测的效率；而更多的漏报将会导致大量异常潜伏于系统内而没有被检测出来，从而使得系统变得更加不稳定，因此较少的漏报能够提高系统的稳定性。

6.2.3 不同参数对 LTTFAD 的影响

不同参数对 LTTFAD 模型的查准率、查全率以及调和分数性能的影响的实验，需要使用控制变量的思想进行。实验数据集为 HDFS，超参数分别为日志模板主题的数量 r、神经网络层数 l、神经网络隐藏层 u、日志模板主题滑动窗口长度 h 以及候选集大小 g。实验结果如图 5－图 9 所示。

从图 5、图 6 可以看出，日志模板主题数量 r 和神经网络层数 l 的变化对 LTTFAD 的性能影响相对更小，因此不同的日志模板主题数量 r 和神经网络层数 l 下的 LTTFAD 相对更稳定。图 7 给出了不同神经网络隐藏层数 u 对 LTTFAD 的性能影响，当 u 超过 128 后，LTTFAD 的性能整体呈下降趋势，可能的原因是神经网络隐藏层数增加导致参数过多，从而影响模型性能，出现了整体性能下降的情况。图 8 给出了滑动窗口大小 h 对 LTTFAD 的性能影响，可以看出整体性能呈先下降后上升的趋势，当 $h=6$ 时，整体性能最差，当 $h=9$ 时，整体性能最优。图 9 给出了候选集 g 的大小对 LTTFAD 的性能影响，查准率 Precision 随着候选集 g 的增大而提高，原因是候选集中是通过 LSTM 来预测得到下一时刻的日志模板主题，这意味着候选集 g 越大，下一时刻可能出现的日志模板主题就越多，误报数就会在一定程度上减少，查准率因此会提高；查全率 Recall 随着候选集 g 的增大而降低，因为候选集 g 越小，意味着下一时刻可能出现的日志模板主题就越少，漏报数就会在一定程度上减少，因此查全率会提高。总体来说，在不同超参数日志模板主题的数量 r、神经网络层数 l、隐藏单元大小 u、日志模板主题滑动窗口长度 h 以及候选集大小 g 设置下，LTTFAD 模型依旧能保证整体性能稳定，这也意味 LTTFAD 能够方便地部署以适应不同的需求。此外，候选集大小 g 能够平衡查准率 Precision 和查全率 Recall 的表现，即较大的候选集大小会导致误报次数的减少和漏报次数的增加，在实际应用中可以根据需求进行调整。

6.2.4 新日志模板注入性能对比

为了验证 LTTFAD 模型在新日志模板注入场景下的稳定性，在 HDFS 数据集上对比现有深度学习异常检测模型，注入新日志模板进行实验。在训练阶段的日志模板集合仅覆盖原始日志数据集 L，即包含 13 个日志模板，新日志模板注入数量为 33，注入比例达到 254%。由于 Deeplog 没有针对新日志模板的注入提供解决方案，因此本文设定 Deeplog 在检测到日志序列内出现新日志模板时将该日志序列标记为异常。实验结果如表 5 所列。表中，LTTFAD 模型的查准率 Precision 和调和分数 FS 两项性能指标均明显优于 Deeplog，LogAnomaly 和 RobustLog，分别达到了 0.954 和 0.938。由于 LTTFAD 是基于日志模板主题特征进行日志异常检测，注入的新日志模板都将通过基于 LDA 的 LTTCM 模型和日志模板主题匹配算法匹配得到最相近的日志模板主题，因此能够很大程度上降低新日志模板注入对分类模型的影响。本实验设定 Deeplog 将包含新日志模板的日志序列均判定为存在异常，因此 Deeplog 的查全率 Recall 明显优于其他两种方案，但这种方法会导致误报数过多，从而查准率 Precision 的值很低。LogAnomaly 面对新日志模板注入的解决策略是将日志模板转换成语义空间向量，并通过计算欧氏距离将新日志模板替换为语义最相近的已有日志模板，但是当注入的新日志模板数量过多时，性能表现会急速下降。RobustLog 能够通过语义向量化的方式捕获日志事件中的语义信息，从而对注入的新日志模板进行处理，因此其相比 DeepLog 和 LogAnomaly 有更稳定的表现，而整体性能还是 LTTFAD 最优。因此，使用日志模板主题能够有效地降低新日志模板的注入对检测模型性能的影响，保证 LTTFAD 模型的稳定性。

表 5 新日志模板注入性能对比

Table 5 Performance comparison for new log template injection

模型	性能指标		
	Precision	*Recall*	*FS*
Deeplog	0.302	**0.984**	0.462
LogAnomaly	0.446	0.930	0.603
RobustLog	0.796	0.983	0.880
LTTFAD	**0.954**	0.923	**0.938**

结束语 本文首次提出了基于日志模板主题的日志异常检测模型 LTTFAD，其中基于日志模板语料训练得到的 LTTCM 模型不仅能够提取到含有语义信息的日志模板主题，同时对于新增日志模板，也可以基于上述的日志模板主题匹配算法得到最相关的日志模板主题，因此 LDA 的使用和日志模板主题匹配算法优化了异常检测模型。与现有基于日志模板的异常检测模型相比，LTTFAD 各项性能指标的表现均有明显提升，对于新日志模板的注入检测模型也具有较高的鲁棒性，实验也有效地证明了 LTTFAD 的性能表现优于现有模型方法。未来的工作应该着眼于结合日志本身参数以及日志频度，并利用更高性能的深度神经网络模型来完成异常检测。实现通用、有效、全面、稳定的日志异常检测方法仍然是未来值得深入研究并切实需要达到的目标。

参 考 文 献

[1] LIANG Y, ZHANG Y, XIONG H, et al. Failure prediction in ibm bluegene/l event logs[C]// Seventh IEEE International Conference on Data Mining(ICDM 2007). IEEE, 2007: 583-588.

[2] WANG Y, WONG J, MINER A. Anomaly intrusion detection using one class SVM[C]// Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004. IEEE, 2004: 358-364.

[3] BREIER J, BRANIŠOVÁ J. Anomaly detection from log files using data mining techniques[M]// Information Science and Applications. Berlin: Springer, 2015: 449-457.

[4] HE P, ZHU J, HE S, et al. Towards automated log parsing for large-scale log data analysis[J]. IEEE Transactions on Dependable and Secure Computing, 2017, 15(6): 931-944.

[5] CHEN M, ZHENG A X, LLOYD J, et al. Failure diagnosis using

decision trees[C]// International Conference on Autonomic Computing. IEEE,2004:36-43.
[6] YING S,WANG B,WANG L,et al. An Improved KNN-Based Efficient Log Anomaly Detection Method with Automatically Labeled Samples[J]. ACM Transactions on Knowledge Discovery from Data(TKDD),2021,15(3):1-22.
[7] XU W,HUANG L,FOX A,et al. Detecting large-scale system problems by mining console logs[C]// Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles. 2009:117-132.
[8] XU D,WANG Y,MENG Y,et al. An improved data anomaly detection method based on isolation forest[C]// 2017 10th International Symposium on Computational Intelligence and Design (ISCID). IEEE,2017,2:287-291.
[9] LOU J G,FU Q,YANG S,et al. Mining invariants from console logs for system problem detection[C]// 2010 USENIX Annual Technical Conference(USENIX ATC 10). 2010.
[10] VAARANDI R,PIHELGAS M. Logcluster-a data clustering and pattern mining algorithm for event logs[C]// 2015 11th International Conference on Network and Service Management (CNSM). IEEE,2015:1-7.
[11] DU M,LI F,ZHENG G,et al. Deeplog:Anomaly detection and diagnosis from system logs through deep learning[C]// Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017:1285-1298.
[12] MENG W,LIU Y,ZHU Y,et al. LogAnomaly:Unsupervised detection of sequential and quantitative anomalies in unstructured logs[C]// IJCAI. 2019:4739-4745.
[13] ZHANG X,XU Y,LIN Q,et al. Robust log-based anomaly detection on unstable log data[C]// Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2019:807-817.
[14] HOCHREITER S,SCHMIDHUBER J. Long short-term memory[J]. Neural Computation,1997,9(8):1735-1780.
[15] CINQUE M,COTRONEO D,PECCHIA A. Event logs for the analysis of software failures: A rule-basedapproach[J]. IEEE Transactions on Software Engineering,2012,39(6):806-821.
[16] HANSEN S E,ATKINS E T. Automated System Monitoring and Notification with Swatch[C]// LISA. 1993,93:145-152.
[17] OPREA A,LI Z,YEN T F,et al. Detection of early-stage enterprise infection by mining large-scale log data[C]// 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. IEEE,2015:45-56.
[18] PREWETT J E. Analyzing cluster log files using logsurfer[C]// Proceedings of the 4th Annual Conference on Linux Clusters. Citeseer,2003.
[19] ROUILLARD J P. Real-time Log File Analysis Using the Simple Event Correlator(SEC)[C]// LISA. 2004,4:133-150.
[20] ROY S,KÖNIG A C,DVORKIN I,et al. Perfaugur:Robust diagnostics for performance anomalies in cloud services[C]// 2015 IEEE 31st International Conference on Data Engineering. IEEE,2015:1167-1178.
[21] YAMANISHI K,MARUYAMA Y. Dynamic syslog mining for network failure monitoring[C]// Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining. 2005:499-508.
[22] YEN T F,OPREA A,ONARLIOGLU K,et al. Beehive:Large-scale log analysis for detecting suspicious activity in enterprise networks[C]// Proceedings of the 29th Annual Computer Security Applications Conference. 2013:199-208.
[23] MAKANJU A A O,ZINCIR-HEYWOOD A N,MILIOS E E. Clustering event logs using iterative partitioning[C]// Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2009:1255-1264.
[24] TANG L,LI T,PERNG C S. LogSig:Generating system events from raw textual logs[C]// Proceedings of the 20th ACM International Conference on Information and Knowledge Management. 2011:785-794.
[25] HAMOONI H,DEBNATH B,XU J,et al. Logmine:Fast pattern recognition for log analytics[C]// Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. 2016:1573-1582.
[26] DU M,LI F. Spell:Streaming parsing of system event logs[C]// 2016 IEEE 16th International Conference on Data Mining(ICDM). IEEE,2016:859-864.
[27] HE P,ZHU J,ZHENG Z,et al. Drain:An online log parsing approach with fixed depth tree[C]// 2017 IEEE International Conference on Web Services(ICWS). IEEE,2017:33-40.
[28] MESSAOUDI S,PANICHELLA A,BIANCULLI D,et al. A search-based approach for accurate identification of log message formats[C]// 2018 IEEE/ACM 26th International Conference on Program Comprehension(ICPC). IEEE,2018:167-16710.
[29] MIZUTANI M. Incremental mining of system log format[C]// 2013 IEEE International Conference on Services Computing. IEEE,2013:595-602.
[30] SHIMA K. Length matters:Clustering system log messages using length of words[J]. arXiv:1611.03213,2016.
[31] BLEI D M,NG A Y,JORDAN M I. Latent dirichlet allocation [J]. Journal of machine Learning research,2003,3(Jan):993-1022.
[32] ZHU J,HE S,LIU J,et al. Tools and benchmarks for automated log parsing[C]// 2019 IEEE/ACM 41st International Conference on Software Engineering:Software Engineering in Practice (ICSE-SEIP). IEEE,2019:121-130.

SUN Xuekui, born in 1996, master. His main research interests include deep learning and system security.

DAI Hua, born in 1982, Ph.D, professor. His main research interests include data management and security.

(责任编辑:喻藜)