基于双向时间卷积网络的半监督日志异常检测*

尹春勇†, 孔 娴
(南京信息工程大学 计算机学院, 南京 210044)

摘 要: 由于日志解析准确率不高以及标记样本不足降低了异常检测的准确率,所以提出了一种新的基于日志的半监督异常检测方法。首先,通过改进字典的日志解析方法,保留了日志事件中的部分参数信息,从而提高日志信息的利用率和日志解析的准确率;然后,使用 BERT 对模板中的语义信息进行编码,获得日志的语义向量;接着采用聚类的方法进行标签估计,缓解了数据标注不足的问题,有效提高了模型对不稳定数据的检测;最后,使用带有残差块的双向时间卷积网络(Bi-TCN)从两个方向捕获上下文信息,提高了异常检测的精度和效率。为了评估该方法的性能,在两个数据集上进行了评估,最终实验结果表明,该方法与最新的三个基准模型 LogBERT、PLELog 和 LogEncoder 相比,F_1 值平均提高了 7%、14.1% 和 8.04%,能够高效精准地进行日志解析和日志异常检测。

关键词: 日志解析; 异常检测; 半监督学习; 双向时间卷积网络; 上下文相关性
中图分类号: TP391.1 **文献标志码**: A **文章编号**: 1001-3695(2024)07-026-2110-08
doi:10.19734/j.issn.1001-3695.2023.10.0507

Semi-supervised log anomaly detection based on bidirectional temporal convolution network

Yin Chunyong†, Kong Xian
(*School of Computer, Nanjing University of Information Science & Technology, Nanjing* 210044, *China*)

Abstract: Because the accuracy of log parsing is not high and the lack of tag samples reduces the accuracy of anomaly detection, this paper proposed a new semi-supervised anomaly detection method based on logs. Firstly, the method enhanced the log parsing method of the dictionary to retain parameter information in log events, improving the utilization and accuracy of log resolution. Next, the method utilized BERT to encode semantic information in the template, obtaining the semantic vector of the log. Then, the method employed the clustering method to estimate the tag, which effectively alleviated the problem of insufficient data labeling and enhanced the model's ability of detecting unstable data. Finally, the method captured context information from two directions based on the bidirectional temporal convolution network (Bi-TCN) with residual blocks, which enhanced the accuracy and efficiency of anomaly detection. To evaluate the method's performance, it conducted extensive experiments on two datasets. The results demonstrate that the proposed method achieves an average improvement of 7%, 14.1% and 8.04% in F_1 value compared to the latest three benchmark models, LogBERT, PLELog and LogEncoder, enabling efficient and accurate log parsing and log anomaly detection.

Key words: log parsing; anomaly detection; semi-supervised learning; bidirectional temporal convolution network; contextual correlation

0 引言

随着大型系统的规模和复杂性日益增加,一些外部故障或内部错误都可能引起系统崩溃,有必要及时并准确地检测出系统中的异常,以确保系统的可靠性和稳定性,从而减轻不必要的损失[1]。系统日志是指现代网络设备、系统及服务程序等在运行过程中产生的具有记载系统状态和事件信息的记录。通过对日志的分析,可以识别偏离预期行为或模式的日志序列,进行故障排除和问题诊断。因此,日志文件在监控网络情况、维护系统稳定、调试性能问题和保护软件安全等方面有重要的作用。

日志数据通常是由源代码中日志语句打印的半结构化的文本字符串组成,详细记录了操作信息,包含应用状态、程序执行逻辑、事件信息等。随着系统的发展,日志事件和日志序列的更迭导致日志语句不断演化。不同类型的设备包含多样的数据元素和不同的日志语法,因此日志具有多样性和非结构化的特点。通常在日志数据的采集、检索和预处理过程中,不可避免地会在原始日志数据中引入一定程度的噪声,导致日志数据具有不稳定性[2]。除此之外,分布式软件系统会产生大量具有复杂依赖关系的日志信息。因此日志数据的非结构性、不稳定性以及庞大的数据量都会给异常日志检测带来巨大的挑战。

早期,系统日志的异常检测依赖于运维人员的个人知识、技术及经验,人工检查分析大量日志信息,难以得出有效的分析和检测。随着人工智能技术的不断发展,越来越多基于机器学习的数据挖掘与异常检测方法应运而生,这些方法能够自动地从运维数据中学习日志事件之间的内在关系并总结规则,实现了日志异常检测的自动化,增加了异常检测的效率和准确率。

现有采用深度学习技术的异常检测方法分为监督学习、无

收稿日期: 2023-10-07; **修回日期**: 2023-12-06 **基金项目**: 国家自然科学基金面上项目(6177282)
作者简介: 尹春勇(1977—),男(通信作者),山东潍坊人,教授,博导,博士,主要研究方向为网络空间安全、大数据挖掘、隐私保护、人工智能、新型计算等(yinchunyong@hotmail.com);孔娴(1999—),女,江苏南京人,硕士研究生,主要研究方向为异常检测、深度学习.

监督学习和半监督学习三大类。监督学习方法 SwissLog[3,4]、LightLog[5]依赖于大量已标注的训练数据,可以有效学习历史日志信息,但基于日志数据异构、大量且不稳定的特性,需要大量人工标记工作的监督学习方法并不实用。无监督异常检测方法如 DeepLog[6]、LogAnomaly[7],基于训练数据大部分是正常日志的先验条件,对噪声敏感,在实际中遇到看不见的日志事件或序列有效性会降低。而半监督学习使用部分标记数据和大量未标记的数据进行模式识别工作,尽可能减少人工标记成本,同时又能带来比较高的准确性。

日志数据集作为时序数据的一种,大多深度学习的异常检测方法 DeepLog、LogRobust、SwissLog 等使用循环神经网络(recurrent neural network,RNN)以及相关变体进行日志序列特征学习。然而 RNN 每两个相邻神经元是相互连接的,计算当前的隐藏状态需要计算之前的隐藏状态,因此不支持并行计算,训练时间开销大,且处理长序列时可能出现梯度消失或爆炸的问题,虽然长短期记忆网络(long short-term memory,LSTM)和门控循环单元网络(gate recurrent unit,GRU)可以减缓但也不能完全杜绝。此外,RNN 在从日志序列获取信息时,由于每个隐藏状态仅记住部分信息,会随着时间的增长被遗忘,所以稳定性不佳。时间卷积网络(temporal convolution network,TCN)[8]不仅可以通过一些层的堆叠来捕捉序列数据的时间属性,而且可以并行化处理,解决了建模远程关系时的梯度爆炸和消失问题,在内存消耗和处理速度方面具有显著优势,能够更快地获得稳定的模型。

与此同时,基于深度学习的日志异常检测方法性能对日志解析的正确性依赖较高,然而现有的日志解析方法无法应对频繁变化的日志语句,普遍需要经过预先调参,存在过度解析丢失日志事件中价值信息的问题。基于频繁项挖掘的方法 FT-tree[9]、基于最长公共子序列的方法 Spell[10]、基于启发式方法 Drain[11]是日志异常检测使用最频繁的几种解析方法。FT-tree 方法在时间、空间上开销大;Spell 仅基于最长公共子序列的策略会导致分区不足,且没有限制其前缀树的深度,直接计算两条日志事件之间的最长公共子序列,比较耗时;Drain 在准确性、健壮性和效率性方面具有优势,但其将日志过度解析为可变部分,会丢失重要信息,容易导致误报。在 SwissLog 中提出了一种基于字典且无须参数调整的日志解析方法,可以保留日志语义含义,并可以高效地正确解析日志。

本文提出了一种新的基于日志的半监督异常检测方法,命名为 BTCNLog(log anomaly detection based on Bi-TCN)。首先,通过改进的基于字典的日志解析方法进行日志解析;然后,使用 BERT[12]对日志模板进行编码,充分表示日志语义信息;接着,受 PLELog[13]方法的启发,采用层次聚类方法初步估计未标记的日志序列的标签并赋予标签概率,可以学习历史异常日志信息;最后,采用带有残差块的双向时间卷积网络从两个方向捕获上下文信息,对日志数据进行有效异常检测。

本文的主要贡献有:a)提出了一种新的基于日志的半监督异常检测方法 BTCNLog,引入带有残差块的双向时间卷积网络从两个方向捕获上下文信息,从而更好地捕获日志序列的特征,提高了异常检测的精度和效率;b)结合日志数据的特性改进了日志解析方法,保留了部分参数语义信息,提高了日志信息的利用率和日志解析的准确率;c)为了缓解数据标注不足问题,采用聚类的方法进行标签估计,学习历史异常日志信息;d)通过三个公共数据集上对模型进行对比实验和消融实验,证明了本文方法的有效性。

1　相关工作

1.1　日志解析

日志解析是一个将非结构化数据转为结构化数据的过程,具体而言,通过技术将已知日志事件中日志关键字和参数拆分并统一模板的过程。目前,日志解析是许多日志异常检测方法的基本步骤,现有方法可以分为以下几类:

a)基于聚类的方法。主要假设是同类的日志事件或签名之间具有相似性,通过计算它们之间的距离,根据相似性进行聚类,在每个聚类中提取日志模板,代表方法有 LKE[14]和 LogMine[15]。

b)基于频繁模式挖掘的方法。通过挖掘日志事件中项的频率,提取日志中频繁出现的一组常量项形成日志模板,代表方法有 LFA[16]和 FT-tree。

c)基于日志结构启发式方法。利用日志结构中不同的属性,使用机器学习等启发式方法提取通用模板,代表方法有 Drain、IPLoM[17],其中 Drain 是一种基于固定深度树的在线日志解析方法,能够以流的方式准确、高效地从原始日志事件中自动提取日志模板。

d)基于字典的方法。通过现有日志模板来构造字典,再用于日志解析的方法。Logparse[18]将模板生成问题转换为词分类问题,并学习模板词和变量词的特征。Logram[19]是利用 *n*-gram字典实现自动日志解析方法。

e)其他方法。Spell 使用最长公共序列算法在线搜索日志组解析流中的日志。NuLog[20]是一种基于掩码语言建模分析上下文中出现的标记来学习日志数据的一般语义表示。

1.2　日志异常检测

异常检测过程是使用机器学习等方法对特征向量进行学习,从而生成异常检测的模型,该模型可用于对新的日志进行检测[21]。现有日志异常检测方法可以分为数据挖掘和深度学习方法。

数据挖掘的方法有逻辑回归(logistic regression,LR)[22],其将日志序列构造成事件计数向量,利用计数向量和标签训练逻辑回归模型,计算测试用例的异常可能性并设置阈值区分出异常日志。支持向量机[23]同样将日志序列表示为事件计数向量,如果测试用例位于超平面之上则其为异常日志。基于聚类的无监督学习方法 LogCluster[24]通过事件计数向量表示日志序列,对日志序列进行聚类,每个簇的质心为代表性向量。通过计算新日志序列向量与所有现有集群代表向量之间的距离来检测异常。此类基于数据挖掘的日志异常检测方法通常将日志表示为日志事件计数向量,无法捕获日志事件的语义信息,异常检测效果不稳定。

基于深度学习的日志异常检测方法可以更好地学习日志序列中日志事件之间的内在关系,对异常日志的检测有较高的精确率和鲁棒性。监督方法中,LogRobust 采用 Drain 方法进行日志解析,通过预处理、词向量化并结合词频-逆文档频率(term frequency-inverse document frequency,TF-IDF)[25]将日志事件转变为语义向量,使用基于注意力的双向长短期记忆神经网络(bidirectional long-short term memory,Bi-LSTM)学习不同日志事件的重要性,可以处理不稳定的日志序列。SwissLog 利用基于字典无须参数调整的日志解析方法进行解析,并使用 BERT 编码器对模板中的语义信息进行编码,最后基于注意力的 Bi-LSTM 模型进行日志异常检测。LightLog 方法基于 word2vec[26]和后处理算法获得低维语义向量,通过轻量级时间

卷积网络实现在边缘设备上的日志异常检测。上述监督学习方法需要人工对大量日志进行标注，十分耗时。无监督方法中，Logsy[27]方法使用辅助数据集的正常日志序列增强数据表示，将注意力机制与 Transformer 编码器相结合来进行异常检测。OC4Seq[28]将一类分类器与 GRU 网络相结合，在日志序列上分别进行全局和局部的异常检测。此类无监督学习的方法只关注历史正常日志序列的信息，忽略了历史异常日志序列的信息。

2 BTCNLog 模型总体框架

现有基于日志异常检测的方法通常都包含日志收集与预处理、日志解析、特征提取和异常检测四个步骤。本文 BTCNLog 日志异常检测模型总体框架如图 1 所示。在训练阶段，首先对原始日志条目进行预处理、日志解析提取模板，并将模板进行向量化提取日志语义向量。然后，将日志条目分组成日志序列，对训练日志序列数据进行聚类标签估计，得到伪标签。最后将带有伪标签的日志序列语义向量输入 Bi-TCN，进行异常检测模型训练。在测试阶段，同样对新的原始日志进行处理获得日志序列及对应的日志序列语义向量，输入训练好的模型进行异常检测。

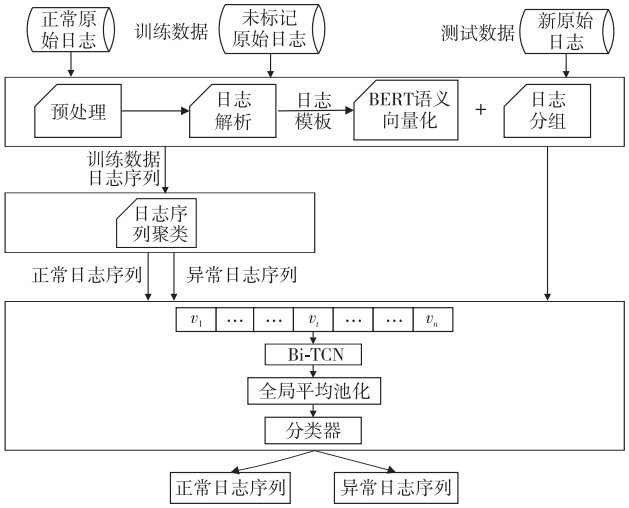


图 1 BTCNLog 模型总框架

Fig. 1 General framework of BTCNLog model

2.1 日志解析和向量化

日志解析是将原始日志经过预处理，提取日志模板的过程。日志的向量化是指将日志提取的模板语句嵌入语义向量，得到可以进行模型训练的日志数据，最后结合日志分组得到日志序列及对应的日志序列向量。

2.1.1 预处理

预处理日志数据是日志处理的第一步。不同系统日志条目会包含时间戳、日志消息等级和日志事件等不同的日志数据项，因此需要根据日志数据集的框架，使用正则表达式匹配各个数据项，如图 2 所示。

其中显示了一条 BGL 数据集日志条目，当标签 Label 为“-”时表示正常日志，否则为异常日志。日志类型 level 可以粗略区分日志的等级，但不足以将异常日志和正常日志完全区分，而日志事件 event 是描述事件具体发生的非结构化文本，蕴涵丰富的日志信息，因此是进行日志分析和异常检测的主要数据来源。如果将原始日志事件直接向量化，则会出现词汇量过多、无效词过多的问题，给日志异常检测带来许多噪声，影响训练效果。因此将事件中的日志公共变量，例如 IP 地址、八进制数、日期等，利用正则表达式分别替换成对应标记，而不是被统一的变量通配符替换，这样不仅可以保留部分变量的语义信息，还可以通过正则表达式筛选出不符合规则的变量。接下来对日志事件进行日志解析，将描述系统事件的日志关键字部分和日志参数部分进行分离。

APPSEV 1124722088 2005.08.22 R41-M0-N4-I:J18-U01 2005-08-22-07.48.08.401638 R41-M0-N4-I:J18-U01 RAS APP FATAL ciod : Error reading message prefix after LOGIN_MESSAGE on CioStream socket to 172.16.96.116:37564: Link has been severed

Label	APPSEV	Timestamp	1124722088	Date	2005.08.22
Node	R41-M0-N4-I:J18-U01	Time	2005-08-22-07.48.08.401638	NodeRepeat	R41-M0-N4-I:J18-U01
Type	RAS	Component	APP	Level	FATAL
Event	ciod: Error reading message prefix after LOGIN_MESSAGE on CioStream socket to 172.16.96.116:37564: Link has been severed				

图 2 日志条目匹配数据项

Fig. 2 Log entries matching data items

2.1.2 日志解析

传统日志解析方法例如使用最频繁的 Spell 和 Drain 方法都需要对不同日志分别进行预训练调参才能达到最佳解析结果。此外，还会因正常日志和异常日志存在过多相同内容而导致日志关键字部分被过度解析成参数部分。目前日志事件大多都是基于英文单词组成具有一定可读性的语句，因此基于字典的日志解析方法，可以保留有效词的语义信息，缓解过度解析的问题。其中 SwissLog 使用了此类日志解析方法。

首先，对每个日志事件进行分隔并形成一组单词标记。由于日志事件中包含许多特殊符号，所以采用空格及日志中常用的特殊分隔符来分割日志事件字符串。根据引入的字典，将所有的单词标记进行字典化，有效单词放入单词集中。然后，将日志事件根据单词集进行集群，具有相同单词集并且每个单词词频相同的日志事件被归为同一集群，每个集群使用最长公共序列提取维护一个模板。接下来，将所有不同集群获得的粗略模板按照顺序生成前缀树，用来整合最终的模板。

然而 SwissLog 在依据有效词的单词集进行集群时，当日志事件中无有效词或有效词单一较少时，有被解析为 0 模板或其他模板的可能性，导致大量 0 模板或模板解析错误，影响日志序列上下文依赖性的提取。因此本文在选择日志事件单词集时，额外加入了由下画线或其他连字符组成的有效单词，并将其存入分词缓存字典。此外，针对分词器 wordninja 会产生错误拆分单词的问题，本文对其拆分后的单词作进一步筛选，确保拆分的正确性，可以大大减少解析为 0 模板和其他模板的可能性。图 3(a)是 SwissLog 日志解析处理过程，其将单词“ciod”错误拆分为“cio”和“d”，并且将“LOGIN_MESSAGE”视为无效词，而该单词在一定程度上蕴涵语义。图 3(b)是 BTCNLog 日志解析处理过程，可以看出本文的日志解析方法，结合日志事件中单词的特性，保留了日志事件中的部分参数信息，从而减少了传统日志解析中，因解析不准确降低日志异常检测方法的性能。

2.1.3 BERT 语义向量化

下面对每个日志事件转换的日志模板进行向量化。传统上，日志事件的单词通过使用 word2vec 或 GloVe[29]方法进一步转换为向量，然后根据单词向量计算每个句子的表示向量。但是，此类方法生成的向量不包含上下语义信息，不能解决同义词问题。为了更好地捕捉单词的上下语义信息和词性差异，BTCNLog 使用 BERT 作为词嵌入的方法。BERT 词嵌入模型的核心是 Transformer 编码器，使用双向自注意力机制能够学习到单词上下文的相关性，能够根据周围的单词动态地产生单词表示形式。

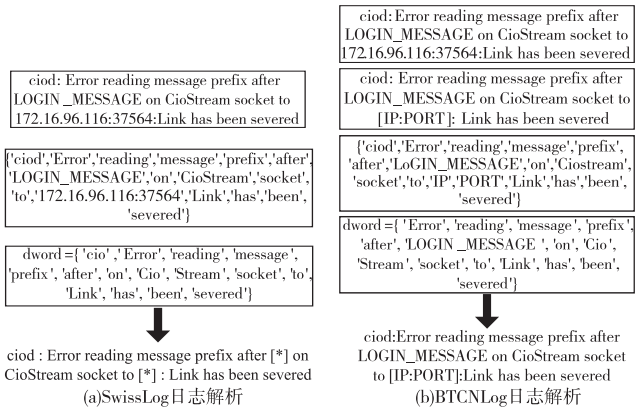


图 3　日志解析过程对比

Fig. 3　Log parsing process comparison

在本模型中,利用预先训练好的 BERT 模型来获取日志事件的语义信息,该模型在维基百科语料库和图书语料库上进行了训练,包含 12 层 Transformer 的编码器,其中每个 Transformer 有 768 个隐藏单元。

2.1.4　日志分组

本文使用了滚动窗口和会话窗口结合的分组策略对日志序列进行分组,可以减少同一个事务的日志数据被分到不同日志序列中的情况,从而提高本模型的日志序列上下文依赖性的提取性能。

常规的日志分组方法包括滚动窗口、滑动窗口和会话窗口。滚动窗口是基于日志时间戳数据取相同窗口大小的日志,互不重叠。滑动窗口也是基于日志时间戳数据,依据窗口大小和步长划分,其前后窗口会重叠。会话窗口是基于日志的会话标识符进行分组。然而系统会多个任务并发执行,导致日志的执行顺序可能会交错,因此基于日志时间戳分组的滚动窗口和滑动窗口会导致同一个事务的日志数据被分到不同日志序列中,影响日志序列上下文依赖性的提取。会话窗口会导致日志序列长度有较大差别。

本文采取基于日志时间戳结合日志条目中某个特定数据项将日志重排并固定窗口大小 w 进行日志分组。此日志分组方法基于数据项可以粗略划分日志事务的前提来选取数据项,例如 node 或 user。可以一定程度上保证相同日志事务被划分到相同日志序列,并且不会产生日志序列长度相差太大的情况。若日志序列包含异常日志事件,则该日志序列为异常日志序列,否则为正常日志序列。

2.2　聚类估计标签

在已知部分正常日志序列的基础上,基于语义相似的日志序列更有可能共享相同标签的性质,通过聚类,进一步估计训练集中未标记日志序列的标签,从而融合监督方法的优点,可以学习到历史异常日志的信息。

基于密度群集的 HDBSCAN[30] 聚类算法可以处理密度不同的聚类问题,无须预定义聚类数量且参数量少。因此采用 HDBSCAN 算法将训练集中的所有日志序列聚类到不同的簇,使每个簇更可能包含语义相似的日志序列。由于日志序列语义向量维度较大影响聚类效率,所以采用了 FastICA[31] 算法降低日志语义向量的维度,保留下最重要的一些特征,去除噪声和不重要的特征,从而实现提升数据处理速度的目的。

本文的日志序列聚类过程如下:首先,将日志序列中每条日志语句语义向量求和作为图的顶点,日志序列之间的相互可达度量距离作为顶点之前边的权重,构建了一个带权连通无向图。日志序列之前相互可达度量距离如式(1)所示。

$$d_{\text{mreach-}m}(a,b)=\max\{\text{core}_m(a),\text{core}_m(b),d(a,b)\} \tag{1}$$

其中:$d_{\text{mreach-}m}(a,b)$ 表示日志序列样本 a 和 b 的相互可达度量距离;$\text{core}_m(a)$ 表示日志序列样本 a 与第 m 个最近邻样本点的核心距离;$d(a,b)$ 是两个日志序列样本点 a 和 b 之间的直接距离。

其次,将该带权图通过 Prim 算法构建其最小生成树。根据最小生成树,通过按距离升序排列树的边,并为每条边创建一个新的合并簇进行迭代,将其转换为连接组件的层次结构。然后,根据设置的最小簇数值,将大而复杂的簇层次结构缩小成一个较小的树。最后,根据每个簇的稳定性值,从压缩树中提取出稳定的簇。簇的稳定性 s_{cluster} 公式如式(2)所示。

$$s_{\text{cluster}}=\sum_{p\in\text{cluster}}(\lambda_a-\lambda_{\text{birth}}) \tag{2}$$

其中:$\lambda=1/distance$;λ_{birth} 表示当节点被分裂成两个子节点时,对应断开边的长度的倒数;λ_a 表示日志序列样本点 a 因为分裂离开该节点时,对应断开边长度的倒数。

这样,将训练集中的日志序列分成若干簇,语义相似的日志序列划分到同一簇中,同簇的日志序列共享相同的标签。因此,如果一个簇包含已知的正常日志序列,则该簇中的其余未标签的日志序列更有可能也是正常日志序列,否则有可能是异常的。通过测量未标记日志序列属于每个标签的概率来为每个未标记日志序列指定概率标签,以减少噪声的影响。具体而言,HDBSCAN 聚类结果会给每个日志序列样本分配一个 0 ~ 1 的分数 *score*,用于表示每个日志序列属于该簇的不确定性,*score* 越小说明该日志序列越接近该簇的中心。因此,可以将该分数划为属于正常或异常的概率,若预测为正常日志序列,则伪标签如式(3)所示。

$$P(\text{normal})=1-\frac{score}{2},P(\text{anomalous})=\frac{score}{2} \tag{3}$$

2.3　异常检测

在日志异常检测阶段中,使用了残差块的双向时间卷积网络从两个方向捕获上下文信息,从而提高捕获日志数据的特征性能。首先通过日志序列 $S=\{e_1,e_2,\cdots,e_n\}$ 得到日志序列语义向量表示为 $\boldsymbol{V}=\{\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_n\}$,其中 n 为该日志序列长度,$e_t(1\leqslant t\leqslant n)$ 是第 t 个日志事件,将其中每个日志事件 e_t 都对应地表示为语义向量 $\boldsymbol{v}_t$,从而将其放入到 Bi-TCN 中。

Bi-TCN 是由 TCN 扩展而来的。TCN 是一种基于 CNN 的新型神经网络模型,旨在处理时间序列预测任务,其遵循一维全卷积网络的架构,具有相同的输入和输出长度。与普通卷积不同的是,TCN 使用因果卷积来确保只捕获过去的信息,利用膨胀卷积在卷积核中增加空洞来增加感受野,因此具有更灵活的感受野,有利于捕捉序列数据的短期和长期时间依赖性。与 RNN 相比,其易于并行化处理,解决了建模远程关系时的梯度爆炸和消失问题,在内存消耗和处理速度方面具有显著优势。

然而原始的 TCN 只关注当前日志序列的局部特征与其上文信息之间的相关性,完全忽略了日志序列与其下文信息之间的相关性。因此,将 TCN 扩展为双向网络即 Bi-TCN,它可以从日志序列上下文的信息中捕获每个日志的长期上下文依赖关系,从而有效地增强上下文表示。在 TCN 中,因果卷积操作保证只有当前时间戳之前的输入数据被用来进行学习,因此将因果卷积改为一维卷积以实现双向的信息学习。

将 V 输入 Bi-TCN,进一步提取长日志上下文的相关性。给定序列 $\boldsymbol{V}=\{\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_n\}$,Bi-TCN 输出一个相同长度的输出结果 $Y=\{y_1,y_2,\cdots,y_n\}$。其中,每个 $y_t(1\leqslant t\leqslant n)$ 不仅依赖于该时刻之前的输入值,而且与未来的输入值有关,从而提取

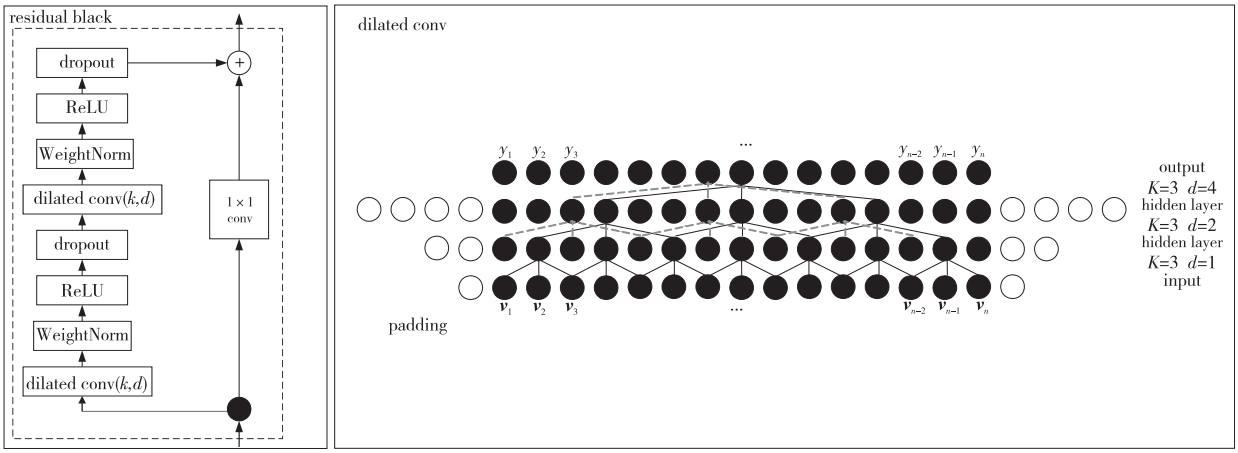
了日志序列的上下文相关性。Bi-TCN 残差块如图 4(a)所示，一个残差块包含两层膨胀卷积，以使模型可以获得更大的感受野，在每层之后加入了 WeightNorm 层，并使用 dropout 层来防止模型过拟合。通过残差块可以实现跨层连接的恒等映射，解决了深层网络的训练问题可以大大增加网络的层数。卷积过程如图 4(b)所示，Bi-TCN 使用一维全卷积，并在每层加入 0-padding 实现输入和输出长度相等，填充公式如式(4)所示，膨胀卷积计算公式如式(5)所示。

$$padding = \frac{(k-1)\times d}{2} \tag{4}$$

$$F(\boldsymbol{v}_t) = (\boldsymbol{V}\times F(d))(\boldsymbol{v}_t) = \sum_{i=0}^{(k-1)/2} f(i)\cdot \boldsymbol{v}_{t-d\cdot i} + \sum_{i=1}^{(k-1)/2} f(i)\cdot \boldsymbol{v}_{t+d\cdot i} \tag{5}$$

其中：d 表示膨胀率，以 2 的指数增长($d=1,2,\cdots,2^n$)；k 表示卷积核的大小；f 表示滤波器；$t-d\cdot i$ 表示过去的时间方向；$t+d\cdot i$表示未来的时间方向。

提取特征后，使用全局平均池化代替全连接层来平均最后一个卷积层的输出特征，进一步减少全连接层中的参数，从而实现紧凑和高效的特征表示。最后，经过全连接层和 softmax 分类器判定异常。



(a)Bi-TCN残差块 (b)Bi-TCN膨胀卷积

图 4 Bi-TCN 残差块以及膨胀卷积

Fig. 4 Bi-TCN residual block and dilated convolution

算法 1 BTCNLog 算法

输入：正常原始日志数据 $S_1=\{e_1,e_2,\cdots,e_n\}$，未标记原始日志数据 $S_2=\{e_1,e_2,\cdots,e_n\}$。

输出：异常检测结果 $Y=\{y_1,y_2,\cdots,y_n\}$。

a)使用正则表达式对 S_1 和 S_2 进行预处理得到日志序列 S_1'和 S_2'。

b)对 S_1'和 S_2'进行基于字典的日志解析，获得日志模板序列 X_1 和 X_2。

c)BERT 对 X_1 和 X_2 进行语义提取得到语义向量 $\boldsymbol{V}_1$ 和 $\boldsymbol{V}_2$。

d)将训练序列 $\boldsymbol{V}_1$ 和 $\boldsymbol{V}_2$ 使用 FastICA 算法降维，并使用 HDBSCAN 聚类算法获得 $\boldsymbol{V}_2$ 的伪标签 P。

e)将 $\boldsymbol{V}_1$ 和 $\boldsymbol{V}_2$ 以及 P 输入到 Bi-TCN，多层计算结果得到内在特征表示 O。

f)对 O 进行全局平均池化得到 O'。

g)将 O'输入全连接层和 softmax 分类器，得到异常检测结果 $Y=\{y_1,y_2,\cdots,y_n\}$。

3 实验及分析

3.1 数据集

日志解析实验部分采用了 Loghub[32] 提供的真实世界日志数据集，其中包含从分布式系统、超级计算机、操作系统、移动系统、服务应用程序收集的 16 个数据集，每个数据集包含 2 000个日志条目。

日志异常检测实验部分选择了三个公共日志数据集(BGL、Thunderbird 和 Spirit)用于验证实验的有效性。BGL 是从 BlueGene/L 超级计算机收集的日志数据集，部署在劳伦斯利弗莫尔国家实验室，时间跨度 251 d，包含 4 747 963 条日志记录，其中 348 460 条是异常的。Thunderbird 和 Spirit 都是在美国新墨西哥州阿尔伯克基的桑迪亚国家实验室的两台真实世界的超级计算机系统收集的大型日志数据集。选取 Thunderbird 原始数据集中前 5 000 000 条连续日志数据和 Spirit 数据集的 1 GB 日志数据进行实验。这三个数据集中的每条日志消息都被手动标记为异常或正常。

3.2 评估指标

大部分日志异常检测的研究只使用精确率(precision)、召回率(recall)和 F_1 值作为评估指标。由于日志数据集正负样例不均衡的特性，使用特异性(specificity)即被正确识别为正常的日志序列在所有真实正常序列中的百分比，可以衡量误报的概率，更全面地评估日志异常检测模型。日志解析实验的度量标准是解析准确率(parsing accuracy)，正确解析的日志事件与日志事件总数之比，其中日志模板的部分匹配情况也被认定为不正确。

TP 表示模型正确检测出的异常日志序列数量；FP 表示被模型错误识别为异常的正常日志序列的数量；FN 表示被模型判别为正常的异常日志序列数量，即未被检测到的异常日志序列的数量；TN 表示模型正确识别的正常日志序列数量。

$$Precision = \frac{TP}{TP+FP} \tag{6}$$

$$Recall = \frac{TP}{TP+FN} \tag{7}$$

$$F_1 = \frac{2\times Precision\times Recall}{Precision+Recall} \tag{8}$$

$$Specificity = \frac{TN}{TN+FP} \tag{9}$$

$$Parsing\ Accuracy = \frac{\text{count}(correct\ event\ ID\ group)}{\text{count}(all\ event\ ID\ group)} \tag{10}$$

3.3 基准模型

日志解析实验部分将改进的日志解析方法与使用频率最高的两个日志解析方法 Spell 和 Drain 以及 SwissLog 日志解析方法进行比较。同时，日志异常检测实验部分将 BTCNLog 与七个基准方法作比较，其中包括两个基于数据挖掘的日志异常检测方法，以及五种基于深度学习的日志异常检测方法，详情如表 1 所示。

表 1　基准模型详情

Tab. 1　Details of baselines

模型	技术分类	日志解析	语义表示	检测方法
PCA	无监督	代码分析	无	PCA
IM	无监督	LKE	无	IM
DeepLog	无监督	Spell	无	LSTM
LogAnomaly	无监督	FT-Tree	Template2Vec	LSTM
LogBERT	自监督	Drain	无	BERT
PLELog	半监督	Drain	word2vec	GRU
LogEncoder	半监督	Drain	BERT	LSTM

a)主成分分析(PCA)[33]。将日志序列矢量化为事件计数向量,利用 PCA 构建对事件计数向量的正常空间和异常空间,若测试用例远离正常空间则为异常日志。

b)不变量挖掘(invariant mining,IM)[34]。从日志计数向量中挖掘日志事件之间的不变量,不满足这些不变量的日志序列被视为异常。

c)DeepLog。将系统日志表示为日志模板的索引,通过 LSTM 网络学习正常日志序列并预测下一个日志事件来判定异常。

d)LogAnomaly。在 template2vec 的支持下提取隐藏在日志模板中的语义信息,使用 LSTM 网络从正常日志序列中提取顺序和定量特征并通过预测判定异常。

e)LogBERT[35]。使用 BERT 来捕获整个日志序列的信息,并通过屏蔽日志键预测和超球体最小化体积两个新的自监督训练任务学习正常日志序列。训练后,利用模型预测结果推导出日志序列的异常分数用于区分出异常日志。

f)PLELog。通过概率标签估计结合历史异常日志信息,并使用 word2vec 语义嵌入处理不稳定的数据,最后使用基于注意力的 GRU 进行异常检测。

g)LogEncoder[36]。一种半监督异常检测的日志序列编码框架,利用预先训练的模型来获得每个日志事件的模型,并将单类和对比目标学习方法结合,使用基于注意力的模型来捕获日志事件的信息。

3.4　实验坏境和配置

在配备 Intel® Core™ i5-10400F 2.90 GHz CPU、128 GB 内存、具有 6 GB GPU 内存的 GTX1660Ti 和 Windows 操作系统的服务器上,使用了 Python 3.8.3、PyTorch 1.5.1 等进行日志解析、日志异常检测和消融实验。

本文设置日志序列窗口大小 w 为 60,根据训练环境设置模型训练 *epochs* 为 5,*batch_size* 为 100,测试 *batch_size* 为1 024。根据先验知识,FastICA 组件数量设置为 100,HDBSCAN 中参数 *min_cluster_size* 设置为 100,*min_samples* 为 100。根据调参结果,使用 4 层 Bi-TCN 模型,每层的卷积核大小 k 为 3,*dropout* 设置为 0.2。使用 Adam 优化器和 LambdaLR 学习率调整方法,初始学习率设置为 2E-3。根据以下比例拆分所有数据集:60% 的数据用于训练,10% 用于验证,30% 用于测试。

3.5　结果和分析

3.5.1　日志解析实验

日志解析部分使用与 SwissLog 方法所提供的相同字典进行实验,除此之外,为了与解析结果相匹配,在预处理时将正则表达式替换的参数部分改为统一标记。表 2 显示了 BTCNLog 模型日志检测部分与其余三个基准方法在 16 个日志数据集上的实验结果以及平均准确率,其中加粗字体显示每个数据集最优结果。实验结果表明,在每个数据集上,BTCNLog 模型都能取得优于 Spell 和 Drain 方法的结果,因为基于字典的解析方法不会将日志关键字过度解析为日志参数。较 SwissLog 而言,结合日志单词特点的改进使得日志解析准确率在 Hadoop、Thunderbird、Mac、Linux、Android 五个数据集上都提高了。值得一提的是,在 SwissLog 论文中提到 Mac 日志数据集上解析的效果较差,在经过改进后,Mac 日志数据集中含有语义信息的合成词被保留,准确率提高了 4.45%。

表 2　日志解析实验结果对比

Tab. 2　Comparison of log analysis experiment results　/%

dataset		HDFS	Hadoop	Spark	Zookeeper	BGL	HPC	Thunderbird	OpenStack
parsing accuracy	Spell	100.00	77.75	90.50	96.35	78.65	65.40	84.35	76.40
	Drain	99.75	94.75	92.00	96.65	96.25	88.70	95.50	73.25
	SwissLog	100.00	99.20	99.65	98.45	96.95	90.95	99.20	100.00
	BTCNLog	100.00	99.35	99.65	98.45	96.95	90.95	99.40	100.00

dataset		Windows	Linux	Andriod	HealthApp	Apache	Proxifier	OpenSSH	Mac	AVG
parsing accuracy	Spell	98.85	60.50	91.85	63.90	100.00	52.65	55.40	75.65	79.26
	Drain	99.70	69.00	91.10	78.00	100.00	52.65	78.75	78.65	86.54
	SwissLog	100.00	86.90	95.35	90.10	100.00	99.00	100.00	84.00	96.23
	BTCNLog	100.00	87.05	95.55	90.10	100.00	99.00	100.00	88.45	96.56

3.5.2　日志异常检测实验

图 5(a)(b)分别显示了 BTCNLog 与七个基准在 BGL 和 Thunderbird 日志数据集上的对比实验结果。由于很多基准实验未在 Spirit 日志数据集上进行,并且使用 Drain 解析 Spirit 数据集准确率低,所以只在消融实验中对该日志实验结果进行分析。

实验结果表明,基于数据挖掘的方法 IM 和 PCA 表现低于基于深度学习的方法,因为此类方法将日志序列表示为日志计数向量,即每个维度表示日志事件在日志序列中出现的次数,忽略了日志事件中的语义信息,无法应对不断演化的日志语句。基于正常日志进行训练的无监督方法 DeepLog 和 LogAnomaly旨在预测日志序列中的下一个日志事件,所以它们只能预测其训练集中出现的日志事件,很容易将看不见的日志事件视为异常,降低了异常检测的准确率。此外,DeepLog 基于日志模板索引进行异常检测,忽略了模板语义信息,效果不如 LogAnomaly。至于方法 LogBERT,虽在 BGL 数据集上表现一般,但其使用 BERT 可以有效捕获整个日志序列的信息,因此其有效性优于 Deeplog 和 LogAnomaly。PLELog 由于其语义嵌入和整合历史异常知识的能力,在 BGL 数据集上表现良好,但是在 Thunderbird 数据集上表现较差,与其采用的日志解析方法 Drain 解析效果不佳有关。半监督模型 LogEncoder 同样使用 BERT 提取模板中的语义信息,结合注意力机制和对比学习具有鲁棒性,在两个数据集上都能取得较好的检测结果。而 BTCNLog 在日志解析方法上进行了改进,结合 BERT 更能保留日志事件语义信息,使用 Bi-TCN 从日志序列上下文提取相关性,可以更好地捕获异常,因此在两个数据集上都取得了较高的 F_1 分数,优于其他基准模型。

为了进一步评估不同日志序列窗口大小对日志异常检测有效性的影响,窗口大小 w 分别选取 60、120、180、240 进行实验,实验结果如图 6 所示。结果表明,实验窗口对异常检测有

效性结果影响不大,各个评估指标都能达到较高水平。因此,使用 BTCNLog 模型可以有效学习不同长度日志序列中的依赖关系。

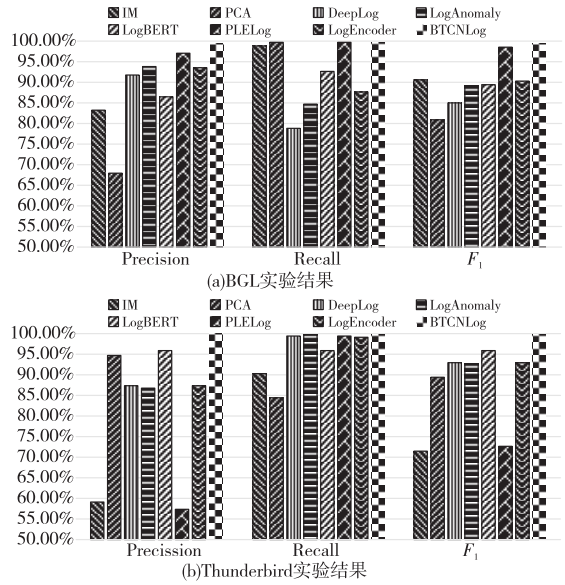


图 5 日志异常检测实验对比

Fig. 5 Comparison of log anomaly detection experiments

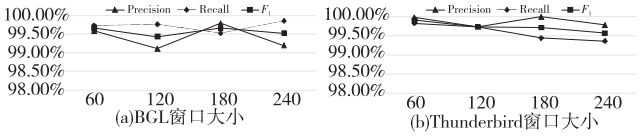


图 6 不同窗口大小实验

Fig. 6 Experiment with different window sizes

3.5.3 消融实验

消融实验可以验证模型各个部分的有效性,因此分别在 BGL、Thunderbird 和 Spirit 日志数据集上进行消融实验,验证日志解析、日志语义嵌入和日志异常检测模型的有效性。表 3 显示了在三个数据集上进行三项消融实验的实验结果。

1)日志解析的消融实验

将日志解析部分改为 SwissLog 原始解析方法,实验效果如表 3 的模型 Bi-TCN + SwissLog 所示,在三个数据集上的实验结果都略有下降。因为 BTCNLog 模型的解析部分结合了日志单词的特性,并保留了部分参数的语义信息,充分利用日志事件的语义信息。在 Spirit 数据集上,实验效果普遍较差,这与该数据集中包含较多专业领域缩写、可读性较差有关。

2)异常检测模型的消融实验

表 3 的模型 TCN + BERT 显示了将日志异常检测模型部分改为 TCN 的实验结果,结果显示,使用 TCN 进行异常检测仍然可以得到很好的效果,但与 Bi-TCN 相比,实验效果略微下降。此外,使用具有 100 层隐藏层的 Bi-LSTM 替换 Bi-TCN,实验结果如表 3 的 Bi-LSTM + BERT 模型所示,虽召回率可以保持较好水平,但精确率有所下降,从而证实了 Bi-TCN 有效学习日志序列上下文信息,对日志异常检测有改进的效果。

3)语义向量的消融实验

在保持模型其余部分相同的情况下,将日志模板语义嵌入模块改为 GloVe 和 TF-IDF 进行实验,其中语义向量维度为 300。实验结果如表 3 的 Bi-TCN + GloVe 模型所示,虽然在召回率上 BTCNLog 模型结果略低些,但在精确度、F_1 分数和特异性上比使用 GloVe 作为语义嵌入的效果好。这是因为使用 GloVe 方法会有很多 OOV(out of vocabulary)问题,无法准确表示日志序列语义信息。因此结果证实了 BERT 语义嵌入的有效性,可以有效保留日志的语义信息。

表 3 消融实验结果

Tab. 3 Results of ablation experiment /%

数据集	方法	Precision	Recall	F_1	specificity
BGL	BTCNLog	99.584	99.737	99.660	99.910
	Bi-TCN + SwissLog	93.888	99.827	96.767	98.637
	TCN + BERT	99.518	99.737	99.627	99.896
	Bi-LSTM + BERT	92.936	99.868	96.278	98.361
	Bi-TCN + GloVe	96.035	89.719	92.770	99.223
Thunderbird	BTCNLog	99.987	99.831	99.909	99.994
	Bi-TCN + SwissLog	96.253	99.417	97.809	98.535
	TCN + BERT	99.831	99.883	99.857	99.928
	Bi-LSTM + BERT	93.213	99.896	96.439	96.902
	Bi-TCN + GloVe	93.703	99.896	96.701	97.140
Spirit	BTCNLog	93.695	94.597	94.144	98.778
	Bi-TCN + SwissLog	87.810	94.162	90.875	97.491
	TCN + BERT	91.120	95.591	93.302	98.212
	Bi-LSTM + BERT	89.724	93.805	91.719	97.938
	Bi-TCN + GloVe	91.108	97.671	94.275	98.170

4 结束语

本文提出了一种新的基于日志的半监督异常检测方法 BTCNLog。首先通过改进的基于字典的日志解析方法进行日志解析,使用 BERT 对模板中的语义信息进行编码获得日志的语义向量;通过新的日志分组方式进行分组;然后通过聚类进行标签估计;最后,基于双向的时间卷积网络进行异常检测。该模型能够提高日志解析准确性,充分利用日志事件的语义信息,并高效提取日志序列上下文信息,提高了日志异常检测的效率和精度。同时,半监督学习方法减少人工标注,可以学习到历史异常日志的信息。

今后工作的主要目标在于在现实大型软件系统中部署,测试其在更多日志数据集上的表现。同时,将进一步改进模型,解决 BTCNLog 对于部分可读性较差的系统日志事件异常检测效果不佳的问题。

参考文献:

[1] 贾统, 李影, 吴中海. 基于日志数据的分布式软件系统故障诊断综述[J]. 软件学报, 2020, 31(7): 1997-2018. (Jia Tong, Li Ying, Wu Zhonghai. Review of fault diagnosis of distributed software system based on log data[J]. Journal of Software, 2020, 31(7): 1997-2018.)

[2] Zhang Xu, Xu Yong, Lin Qingwei, *et al*. Robust log-based anomaly detection on unstable log data[C]//Proc of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. New York: ACM Press, 2019: 807-817.

[3] Li Xiaoyun, Chen Pengfei, Jing Linxiao, *et al*. SwissLog: robust and unified deep learning based log anomaly detection for diverse faults [C]//Proc of the 31st International Symposium on Software Reliability Engineering. Piscataway, NJ: IEEE Press, 2020: 92-103.

[4] Li Xiaoyun, Chen Pengfei, Jing Linxiao, *et al*. SwissLog: robust anomaly detection and localization for interleaved unstructured logs [J]. IEEE Trans on Dependable and Secure Computing, 2023, 20(4): 2762-2780.

[5] Wang Zumin, Tian Jiyu, Fang Hui, *et al*. LightLog: a lightweight temporal convolutional network for log anomaly detection on the edge [J]. Computer Networks, 2022, 203: 108616.

[6] Du Min, Li Feifei, Zheng Guineng, *et al*. DeepLog: anomaly detection and diagnosis from system logs through deep learning[C]//Proc of ACM SIGSAC Conference on Computer and Communications Secu-

rity. New York: ACM Press, 2017: 1285-1298.

[7] Meng Weibin, Liu Ying, Zhu Yichen, *et al*. LogAnomaly: unsupervised detection of sequential and quantitative anomalies in unstructured logs[C]//Proc of the 28th International Joint Conference on Artificial Intelligence. San Francisco: Morgan Kaufmann Publishers, 2019: 4739-4745.

[8] Bai Shaojie, Kolter J Z, Koltun V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling[EB/OL]. (2018-04-19). https://arxiv.org/abs/1803.01271.

[9] Zhang Shenglin, Meng Weibin, Bu Jiahao, *et al*. Syslog processing for switch failure diagnosis and prediction in datacenter networks [C]//Proc of the 25th International Symposium on Quality of Service. Piscataway, NJ: IEEE Press, 2017: 1-10.

[10] Du Min, Li Feifei. Spell: online streaming parsing of large unstructured system logs[J]. IEEE Trans on Knowledge and Data Engineering, 2018, 31(11): 2213-2227.

[11] He Pinjia, Zhu Jieming, Zheng Zibin, *et al*. Drain: an online log parsing approach with fixed depth tree[C]//Proc of the 24th IEEE International Conference on Web Services. Piscataway, NJ: IEEE Press, 2017: 33-40.

[12] Devlin J, Chang Mingwei, Lee K, *et al*. BERT: pre-training of deep bidirectional transformers for language understanding [EB/OL]. (2019-05-24). https://arxiv.org/abs/1810.04805.

[13] Yang Lin, Chen Junjie, Wang Zan, *et al*. Semi-supervised log-based anomaly detection via probabilistic label estimation[C]//Proc of the 43rd International Conference on Software Engineering. Piscataway, NJ: IEEE Press, 2021: 1448-1460.

[14] Fu Qiang, Lou Jianguang, Wang Yi, *et al*. Execution anomaly detection in distributed systems through unstructured log analysis[C]// Proc of the 9th IEEE International Conference on Data Mining, Piscataway, NJ: IEEE Press, 2009: 149-158.

[15] Hossein H, Debnath B, Xu Jianwu, *et al*. LogMine: fast pattern recognition for log analytics[C]//Proc of the 25th ACM International on Conference on Information and Knowledge Management. New York: ACM Press, 2016: 1573-1582.

[16] Nagappan M, Vouk M. Abstracting log lines to log event types for mining software system logs[C]//Proc of the 7th IEEE Working Conference on Mining Software Repositories. Piscataway, NJ: IEEE Press, 2010: 114-117.

[17] Makanju A, Zincir-Heywood A N, Milios E. Clustering event logs using iterative partitioning[C]//Proc of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2009: 1255-1264.

[18] Meng Weibin, Liu Ying, Zaiter F, *et al*. Logparse: making log parsing adaptive through word classification[C]//Proc of the 29th International Conference on Computer Communications and Networks. Piscataway, NJ: IEEE Press, 2020: 1-9.

[19] Dai Hetong, Li Heng, Shang Weiyi, *et al*. Logram: efficient log parsing using *n*-gram dictionaries[J]. IEEE Trans on Software Engineering, 2022, 48(3): 879-892.

[20] Nedelkoski S, Bogatinovski J, Acker A, *et al*. Self-supervised log parsing[C]//Proc of Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Cham: Springer, 2020: 122-138.

[21] 张颖君, 刘尚奇, 杨牧, 等. 基于日志的异常检测技术综述[J]. 网络与信息安全学报, 2020,6(6): 1-12. (Zhang Yingjun, Liu Shangqi, Yang Mu, *et al*. Overview of log based anomaly detection technology[J]. Journal of Network and Information Security, 2020, 6(6): 1-12.)

[22] Bodík P, Goldszmidt M, Fox A, *et al*. Fingerprinting the datacenter: automated classification of performance crises[C]//Proc of the 5th European Conference on Computer Systems. New York: ACM Press, 2010: 111-124.

[23] Liang Yinglung, Zhang Yanyong, Xiong Hui, *et al*. Failure prediction in IBM BlueGene/L event logs[C]//Proc of the 7th IEEE International Conference on Data Mining Workshops. Piscataway, NJ: IEEE Press, 2007: 583-588.

[24] Zhu Jieming, He Pinjia, Fu Qiang, *et al*. Learning to log: helping developers make informed logging decisions[C]//Proc of the 37th IEEE International Conference on Software Engineering. Piscataway, NJ: IEEE Press, 2015: 415-425.

[25] Soucy P, Mineau G. Beyond TFIDF weighting for text categorization in the vector space model[C]//Proc of International Joint Conference on Artificial Intelligence. San Francisco: Morgan Kaufmann Publishers, 2005: 1130-1135.

[26] Ling Wang, Dyer C, Black A, *et al*. Two/too simple adaptations of word2vec for syntax problems[C]//Proc of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Stroudsburg, PA: ACL Press, 2015: 1299-1304.

[27] Nedelkoski S, Bogatinovski J, Acker A, *et al*. Self-attentive classification-based anomaly detection in unstructured logs[C]//Proc of IEEE International Conference on Data Mining. Piscataway, NJ: IEEE Press, 2020: 1196-1201.

[28] Wang Zhiwei, Chen Zhengzhang, Ni Jingchao, *et al*. Multi-scale one-class recurrent neural networks for discrete event sequence anomaly detection[C]//Proc of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. New York: ACM Press, 2021: 3726-3734.

[29] Pennington J, Socher R, Christopher D. Manning. GloVe: global vectors for word representation[C]//Proc of Conference on Empirical Methods in Natural Language Processing. Stroudsburg, PA: ACL Press, 2014: 1532-1543.

[30] McInnes L, Healy J, Astels S. HDBSCAN: hierarchical density based clustering[J]. Journal of Open Source Software, 2017, 2(11): 205.

[31] Oja E, Yuan Zhijian. The FastICA algorithm revisited: convergence analysis[J]. IEEE Trans on Neural Networks, 2006, 17(6): 1370-1381.

[32] Zhu Jieming, He Shilin, He Pinjia, *et al*. Loghub: a large collection of system log datasets for AI-driven log analytics[C]//Proc of the 34th International Symposium on Software Reliability Engineering. Piscataway, NJ: IEEE Press, 2023: 355-366.

[33] Xu Wei, Huang Ling, Fox A, *et al*. Detecting large-scale system problems by mining console logs[C]//Proc of the 22nd ACM SIGOPS Symposium on Operating Systems Principles. New York: ACM Press, 2009: 117-132.

[34] Lou Jianguang, Fu Qiang, Yang Shengqi, *et al*. Mining invariants from console logs for system problem detection[C]//Proc of USENIX Annual Technical Conference. Berkeley: USENIX Press, 2010: 24.

[35] Guo Haixuan, Yuan Shuhan, Wu Xintao. LogBERT: log anomaly detection via BERT[C]//Proc of International Joint Conference on Neural Networks. Piscataway, NJ: IEEE Press, 2021: 1-8.

[36] Qi Jiaxing, Luan Zhongzhi, Huang Shaohan, *et al*. LogEncoder: log-based contrastive representation learning for anomaly detection[J]. IEEE Trans on Network and Service Management, 2023, 20(2): 1378-1391.