Computer Science and Application 计算机科学与应用, 2023, 13(5), 1006-1018
Published Online May 2023 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2023.135098

Hans汉斯

基于GRU-TextCNN的日志序列异常检测方法

胡　标[1,2]，徐　克[1,2]，简　文[1,2]，姜咏绮[1,2]，刘　军[1,2*]

[1]温州大学计算机与人工智能学院，浙江　温州
[2]温州市智能网络重点实验室，浙江　温州

收稿日期：2023年4月12日；录用日期：2023年5月12日；发布日期：2023年5月22日

摘　要

系统详细记录着系统的运行情况和事件，因此系统维护人员常常基于日志对系统状态进行分析，判断系统有无出现异常，以此更好地维护系统。由于现代系统日志数据大规模增加，传统的日志异常检测方法已经不适用现代系统日志。本文基于深度学习技术，提出了一种基于GRU-TextCNN的日志异常检测方法。该方法首先通过预处理将日志处理成日志语句，然后利用SBERT模型将日志语句转换成相应的句向量，随后利用滑动窗口提取日志序列，最后利用本文提出的基于GRU-TextCNN的日志序列异常检测模型检测日志序列。通过在两个数据集上的实验结果表明，该方法能够有效检测出日志序列异常。

关键词

日志异常检测，深度学习，GRU，TextCNN

GRU-TextCNN-Based Anomaly Detection Method for Log Sequences

Biao Hu[1,2], Ke Xu[1,2], Wen Jian[1,2], Yongqi Jiang[1,2], Jun Liu[1,2*]

[1]College of Computer and Artificial Intelligence, Wenzhou University, Wenzhou Zhejiang
[2]Wenzhou Key Laboratory for Intelligent Networking, Wenzhou Zhejiang

Received: Apr. 12th, 2023; accepted: May 12th, 2023; published: May 22nd, 2023

Abstract

The system meticulously records its operations and events, allowing system maintenance personnel to frequently analyze its status based on log data. This analysis is crucial for determining any abnormalities and ensuring optimal system maintenance. However, with the immense growth

*通讯作者。

文章引用: 胡标, 徐克, 简文, 姜咏绮, 刘军. 基于 GRU-TextCNN 的日志序列异常检测方法[J]. 计算机科学与应用, 2023, 13(5): 1006-1018. DOI: 10.12677/csa.2023.135098

in modern system log data, traditional log anomaly detection methods have become inadequate for contemporary systems. In this paper, we introduce a deep learning-based log anomaly detection method utilizing GRU-TextCNN. This method begins by preprocessing logs into log statements, followed by converting these statements into corresponding sentence vectors using the SBERT model. Next, log sequences are extracted through sliding windows, and the log sequence anomaly detection model, based on GRU-TextCNN, is applied. Experimental results from two datasets demonstrate the effectiveness of this method in detecting log sequence anomalies.

Keywords

Log Anomaly Detection, Deep Learning, GRU, TextCNN

1. 引言

随着现代计算机技术的不断发展，计算机系统规模逐渐扩大，但同时，系统异常的概率也在上升。例如，当前的分布式系统规模庞大且运行逻辑复杂，容易导致系统出现异常。此外，系统上运行的各种软件服务也给系统运行带来挑战。随着软件服务的增加，系统故障的风险也随之上升。因此，需要相应的异常检测手段来确保系统的正常运行。

日志详细记录了系统运行状况和各种事件，因此基于日志的异常检测方法具有重要的实际意义，并成为众多学者和专家的研究方向[1]。早期的日志异常检测主要依赖人工分析。例如，在系统出现异常时，运维人员通过查看日志并搜索关键词，如“fail”和“error”，以找到可能与异常状态相关的信息。然而，随着系统规模的扩大，日志规模也相应增加，人工分析日志的成本和效率变得不尽如人意。因此，设计针对日志信息的异常检测算法具有重要实际价值。随着机器学习技术的发展，许多专家将其应用于日志异常检测领域，并提出了众多方法。这些方法利用机器学习的能力，以提高异常检测的效率和准确性，为系统运行提供更强大的保障。

Xu 等人[2]首次将主成分分析(PCA)方法应用于日志异常检测领域。他们首先结合源代码分析和信息检索来解析日志，随后将每个日志序列转换为日志计数向量，并应用 PCA 进行异常检测。最后，结果以决策树形式可视化。Lou 等人[3]通过应用不变量挖掘方法来寻找日志中的异常。程序不变量指的是在系统运行过程中始终存在的线性关系。通过不变量挖掘，可以表示系统运行时的正常执行线性关系。如果被检测的日志序列破坏了这种关系，则将该日志序列判断为异常。例如，用户的登录和登出是对应的，如果登录和登出表示的日志数量不相等，则说明出现了异常。文献[4]首次将决策树应用于 WEB 请求日志的故障检测，基于日志计数向量及其标签构建决策树，并通过遍历决策树来判断新实例的状态。Liang 等人[5]使用支持向量机算法(SVM)进行故障检测，同样基于日志计数向量及其标签进行训练。

早期基于传统机器学习的日志异常检测方法主要利用日志计数向量进行异常检测，但此类检测方式具有局限性。一旦日志解析出错，将产生新的日志模板，从而影响模型检测。随着深度学习技术的发展，该领域取得了显著成果，因此许多学者将深度学习技术应用于日志异常检测领域。

Lu 等人[6]将卷积神经网络(CNN)应用于大数据系统日志的异常检测。该方法首先通过日志解析将日志转换为日志键，接着利用日志键生成日志序列，然后将日志序列向量化，最后输入到 CNN 模型中进行

异常检测。Du 等人[7]基于 LSTM 网络提出了一种新的日志异常检测模型 DeepLog，将系统日志视为自然语言建模。异常检测分为执行路径异常检测和参数异常检测。通过结合这两种不同的检测，以达到更好的检测效果。并采用在线方式增量地更新 DeepLog 模型，以便模型能够适应新的日志。Zhang 等人[8]提出了一种基于双向长短时记忆网络(Bi-LSTM)的日志路径异常检测模型，可以有效进行日志路径异常检测。文献[9]提出了一种并行 GRU 模型，该方法利用日志解析生成的日志模板，进而得到相应的日志模板序列和对应的日志模板频度向量。然后将两者分别输入到两个 GRU 模型中，接着将两个 GRU 模型的输出进行拼接，最后通过 softmax 函数得到输出。

尽管现有的基于深度学习的日志异常检测方法取得了一定的成绩，但仍存在以下不足：大多数基于深度学习的日志异常检测方法采用日志解析来处理日志，但研究表明日志解析可能带来噪声影响，从而影响模型检测性能[10] [11] [12]。此外，日志格式的多样性进一步影响日志解析器对日志的解析。某些解析器可能在某个数据集上表现良好，但在其他数据集上表现较差。目前尚无通用的日志解析器能完美解决这些问题。另外，现有的日志序列异常检测方法大多基于日志模板索引编码进行日志序列化，这容易导致日志语义丢失，从而影响模型的检测性能。

针对以上不足，本文提出了一种基于 GRU 和 TextCNN 的日志序列异常检测方法。该方法基于日志语义信息，通过 SBERT 将日志转换为句向量，然后利用滑动窗口提取日志序列向量，最后利用本文提出的日志序列异常检测模型 GTCLog 对日志序列进行异常检测。GRU 能够有效捕获序列中的依赖关系，而 TextCNN 能够对 GRU 捕获的依赖关系进行进一步的特征提取。GTCLog 通过结合 GRU 和 TextCNN 的优势，从而能够有效地检测出日志序列异常。

2. 理论基础

2.1. 向量表示

尽管日志是非结构化文本，但从语义角度来看，可以将日志视为具有特殊意义的语句。通过将日志语句向量化，可以计算向量之间的距离来表示其相似性，从而加以区分。

在文本向量化技术中，有许多优秀的词嵌入模型，如 Word2Vec [13]，Glove [14]，和 ELMo [15]等。尽管这些模型能够有效地将词表示为词向量，但对于句子向量化而言，仍存在一定的不足。简单地对词向量进行加权求和会导致上下文语义的缺失。为了更好地保留日志语义信息，本文采用 SBERT [16]模型来对日志进行句向量化处理。

SBERT 模型是近年来较为优秀的句子嵌入模型。通过 SBERT，可以快速地将句子转换成句向量，并且保留丰富的语义信息。SBERT 基于 BERT 模型，并进行了一定修改：使用 Siamese 和 triplet 网络结构来获得具有语义的句子向量。模型结构如图 1 所示。

训练过程如下：句子 A 和句子 B 通过 BERT 模型之后，经过池化得到向量 u 和 v。接着，将 u、v 和两者差的绝对值 $|u-v|$ 进行拼接，并乘以一个可训练的权重 W，输入 softmax 函数得到预测结果。计算公式如下：

$$o = \text{softmax}\left(W\left(u, v, |u-v|\right)\right) \tag{1}$$

2.2. GRU 神经网络

GRU [18]是 LSTM 的一种变种，相较于 LSTM 有三个门(遗忘门，输入门，输出门)，GRU 只有两个门(更新门，重置门)。此外，GRU 与标准 RNN 相似，只有 h_t，没有 LSTM 中的 CELL 状态。其结构如图 2 所示：

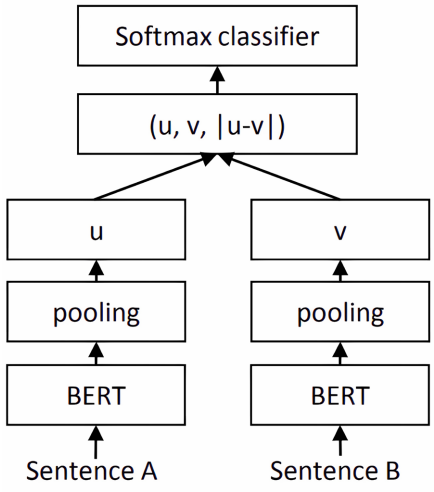

Figure 1. SBERT model structure diagram [17]
图 1. SBERT 模型结构图[17]

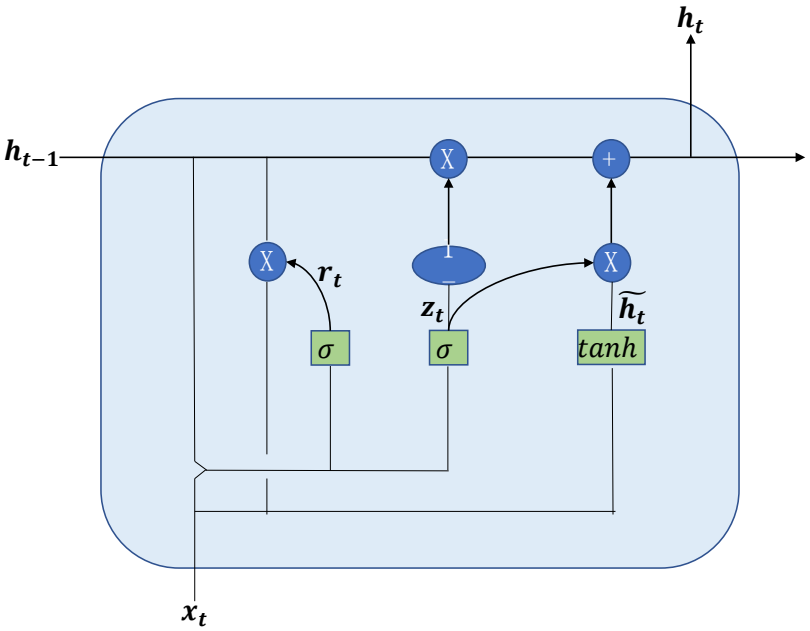

Figure 2. GRU model structure diagram
图 2. GRU 模型结构图

图中 r_t 表示重置门，z_t 表示更新门。重置门 r_t 决定 h_{t-1} 传入候选状态 $\tilde{h}_t$ 的比例；r_t 的值越小，则 h_{t-1} 的信息添加到候选状态 $\tilde{h}_t$ 越少。更新门 z_t 决定前一状态的信息 h_{t-1} 和候选状态 $\tilde{h}_t$ 在新状态 h_t 中的保留程度。其计算公式如下所示：

$$z_t = \text{sigmoid}\left(W_{hz}h_{t-1} + W_{xz}x_t + b_z\right) \tag{2}$$

$$r_t = \text{sigmoid}\left(W_{hr}h_{t-1} + W_{xr}x_t + b_r\right) \tag{3}$$

$$\tilde{h}_t = \tanh\left(W_{hh}\left(h_{t-1} \odot r_t\right) + W_{xh}x_t + b_h\right) \tag{4}$$

$$h_t = \left(1 - z_t\right) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{5}$$

2.3. 文本卷积神经网络

除了在图像处理领域广泛应用，CNN 同样在自然语言处理领域也取得了广泛的成功。TextCNN [19] (有研究称之为 CNN-text)通过改变卷积方式，使得 CNN 模型可以有效处理文本数据。

由于模型不能直接接受文本数据，因此在模型训练之前，需要将文本数据转换为向量数据。文本由单词组成，因此通常首先将单词转换为词向量，然后基于词向量完成对文本的转换。转换完成后，便得到文本的词向量矩阵。这使得 TextCNN 可以像处理图像数据一样处理文本词向量矩阵。其网络结构如图 3 所示：

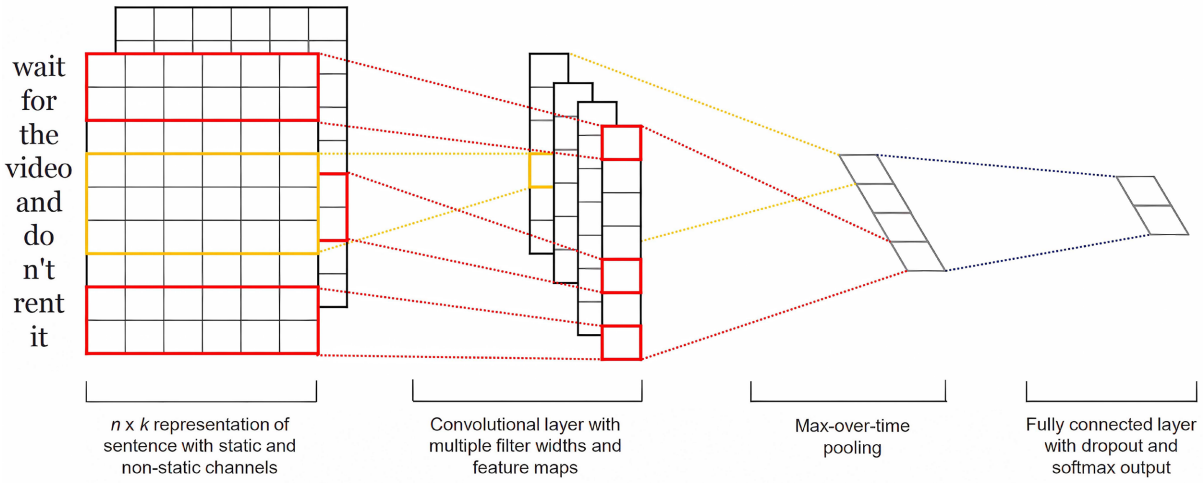

Figure 3. TextCNN model structure diagram [19]
图 3. TextCNN 模型结构图[19]

在输入层，每条文本都通过词嵌入技术转换为一个二维词向量矩阵，该词向量矩阵可以视为具有相同大小的图像。在卷积层，TextCNN 使用半定长卷积核提取特征，其长度通常设置为(2, 3, 4)，宽度设置为词向量维度。例如，若词向量维度为 d，则卷积核宽度也为 d。这是因为一行词向量表示一个单词，若宽度未设定为词向量维度，词向量将被截断，从而丢失部分信息。在池化层，通过最大池化将特征压缩，然后拼接在一起。最后，通过全连接层和 softmax 函数得到输出。

3. 模型方法

3.1. 方法框架

本文所提出的基于 GRU-TextCNN 的日志序列异常检测方法，从实现角度来分，可以分为四个模块：预处理模块、日志语句向量化模块、日志序列提取模块与异常检测模块。整体框架如图 4 所示：

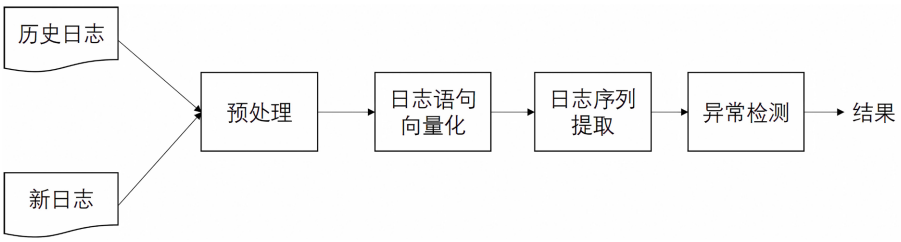

Figure 4. GTCLogmethodology framework diagram
图 4. GTCLog 方法框架图

3.2. 预处理

现有的日志序列异常检测方法主要采用日志解析器对日志进行预处理，通过日志解析器解析出日志模板，然后利用模板索引编码或模板语义进行日志序列异常检测。但这种预处理手段可能导致噪音产生，进而影响模型检测效果。本文提出的预处理方法将日志视为具有特殊语义的句子，并采用文本处理算法进行预处理，具体过程如算法 1 所示。

算法 1：日志预处理

```
输入：原始日志集合 RLS(Raw Log Set)
输出：预处理完成的日志集合 PLSS(Pre-processed Log Statement Set)
1.    初始化 PLSS 列表：PLSS=[]
2.    FOR log in RLS：
3.            将 log 转化成单词集合 LW
4.            FOR word in LW:
5.                IF word  为时间、空格、冒号等:
6.                    删除 word
7.                END IF
8.                word = Lower(word)
9.            END FOR
10.           将处理完成的 LW 转换成字符串 LS
11.           将 LS 添加到 PLSS 列表中
12.   END FOR
13.   RETURN PLSS
```

上述算法中，Lower 函数的作用是将单词转换为小写。以日志：1117838570 2005.06.03 R02-M1-N0-C:J12-U112005-06-03-15.42.50.363779 RAS KERNEL INFO instruction cache parity error corrected 为例。首先通过分词将日志处理成为一组单词集合 LW。然后去除时间戳、日期等时间信息，将时间信息去除之后，接着去除空格、冒号、逗号等符号以及数字。然后将处理完成的单词集合 LW，转换成字符串格式。处理完成之后，最终得到一行日志语句 LS = {RAS KERNEL INFO instruction cache parity error corrected}。PLSS 对应定义如下，其中 LS 为预处理完成的日志语句：

$$\mathrm{PLSS}=\left[\mathrm{LS}_1,\mathrm{LS}_2,\mathrm{LS}_3,\cdots,\mathrm{LS}_n\right]$$

3.3. 日志语句向量化

尽管日志属于非结构化文本，但从语义角度来看，可以将日志视为具有特殊意义的语句。通过将日志语句向量化，可以计算向量之间的距离以表示其相似性，进而实现区分。

算法 2：日志语句向量化

```
输入：预处理完成的日志语句集合 PLSS
输出：日志语句向量集合 LSVS(Log Statement Vector Set)
1.    FOR batch in PLSS:
2.            SBERT 训练
3.    END FOR
4.    RETURN LSVS
```

本文的实验基于 SBERT 模型将日志语句直接转换为句向量，具体过程如算法 2 所示。通过 SBERT，每条日志语句 LS 都被处理成了一条日志语句向量 LSV，将日志语句集合 PLSS 转换为日志语句向量集合 LSVS。以下是 LSVS 的定义，其中 LSV 代表日志语句向量：

$$LSVS=\left[LSV_1,LSV_2,LSV_3,\cdots,LSV_n\right]$$

3.4. 日志序列提取

系统中的某个事件或状态可能需要多条日志来记录，由此产生了许多日志序列。日志序列也表明了其执行路径，因此基于日志序列的异常检测任务也被称为基于日志执行路径的异常检测任务。该任务的输入是日志向量序列。首先，通过预处理模块将原始日志集合处理成日志语句集合；接着，通过句子向量化技术将日志语句集合转换成日志语句向量集合。此时，日志语句向量集合是离散的，不包含任何执行序列。因此，需要通过日志序列提取模块提取日志序列。

现有的日志序列提取方式主要有两种：一种是根据会话提取，另一种是根据窗口提取。会话提取日志序列需要满足一定条件，即日志语句中需包含相关会话标识，例如 HDFS 数据中的 blk 信息。然后根据会话标识提取属于同一个会话的日志，从而提取出相应的会话序列。窗口提取序列根据窗口的选择又有多种形式，如固定窗口提取、滑动窗口提取等。固定窗口通过一个固定大小的窗口提取日志，例如设定固定窗口大小为 5，则每次提取日志数量为 5，且下一次提取是当前提取日志序列的后五条日志。滑动窗口与固定窗口类似，都是提前设置好一个固定大小的窗口。与固定窗口不同的是，滑动窗口每次的滑动步长也是通过人为设定。例如设置窗口大小为 5，步长为 1，有日志集合 $S=\{L1,L2,L3,\cdots,L10\}$。第一次提取的日志序列为{L1, L2, L3, L4, L5}，第二次提取的日志序列为{L2, L3, L4, L5, L6}，以此类推，每次滑动都只往下滑动一步。

由于本文实验所采用的数据集没有会话标识，且固定窗口的提取方式较为僵化，可能导致遗漏部分日志序列，因此本实验采用滑动窗口进行序列提取，提取方式如算法 3 所示：

算法 3：滑动窗口提取算法

输入：日志语句向量集合 LSVS
输出：日志语句序列集合 LSSS(Log Statement Sequence Set)
1. 设定窗口大小 w
2. 设定滑动步长 s
3. 初始化 LSSS 列表：LSSS=[]
4. WHILE 未遍历完LSVS DO:
5. 根据 w 在 LSVS 中取相应数量的日志语句向量
6. 然后将获取到的序列添加到LSSS
7. 然后根据滑动步长 s 往后滑动
8. END WHILE
9. RETURN LSSS

通过滑动窗口，日志语句向量集合 LSVS 中的语句向量被提取出来，成为一个日志序列，然后放入日志语句序列集合 LSSS 中。LSSS 的定义如下，其中 LSS 为日志语句序列：

$$LSSS=\left[LSS_1,LSS_2,LSS_3,\cdots,LSS_n\right]$$

3.5. 异常检测

为了更好地检测日志序列异常，本文提出了一种基于 GRU-TextCNN 的日志序列异常检测模型：GTCLog 模型。日志序列为序列数据，包含一定的序列信息。实践中已证实，GRU 模型能有效捕获序列中的信息，而 TextCNN 则能对文本特征进行有效处理。因此，本文方法通过结合两者优势，提出 GTCLog 模型，以实现对日志序列异常更精准的检测。模型结构如图 5 所示。GTCLog 模型可分为四个部分：输入层、GRU 层、TextCNN 层和输出层。

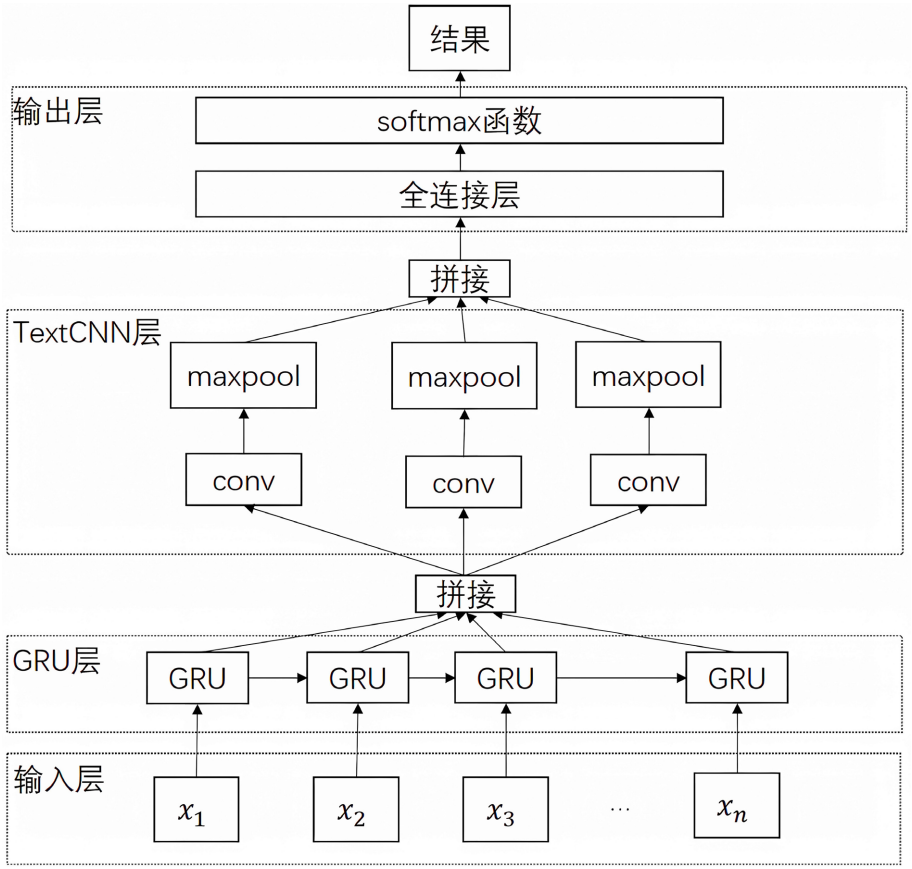

Figure 5. GTCLog structure diagram
图 5. GTCLog 结构图

在数据流入输入层之前，需将数据处理成相应格式。通过日志预处理模块、日志语句向量化模块和日志序列提取模块，日志集合转换成日志语句向量序列集合。输入层的输入维度为(N, l, d)，其中 N 为样本数，l 为日志序列长度等于滑动窗口大小，d 为句向量维度。GRU 层由 GRU 神经网络构成，利用 GRU 网络对输入的序列数据进行初步特征提取。在得到 GRU 层输出后，由 TextCNN 层进行更深入的特征挖掘。TextCNN 通过设置多个不同大小的卷积核，以学习多种不同特征。卷积操作后，每个卷积核会得到一个特征向量。池化层将每个卷积核得到的特征向量进行最大池化，然后拼接起来作为池化层输出。最后，将经过 TextCNN 处理的特征通过全连接层进行特征映射，接着通过 softmax 函数得到最终概率。

4. 实验结果与分析

4.1. 实验设置

为了评估模型，实验取 BGL [20]数据集的全部数据和 Thunderbird [20]数据集前 1100 万条日志数据进行实验。BGL 数据集来自于劳伦斯利弗莫尔国家实验室的 BlueGene/L 超级计算机，该数据集包含 4747963 条日志，其中 348460 条日志为异常日志。Thunderbird 数据集来自位于阿尔伯克基桑迪亚国家实验室(SNL)的雷鸟超级计算机系统。BGL 数据集和 Thunderbird 数据集均来自文献[21]开源的项目。

由于日志序列异常检测可以看作是一个二分类问题，因此实验采用精确率、召回率和 F1 值作为实验的评价指标，来评估本文模型的有效性。

1) 精确率(Precision)：

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

2) 召回率(Recall)：

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

3) F1 值(F1-Score)：

$$\text{F1} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{8}$$

其中 TP (True Positive)为模型正确检测到的异常日志序列数量。FP (假阳性)是被错误识别为异常的正常日志序列的数量。FN (False Negative)是错误地判定为正常的日志序列数量。

实验选用三个机器学习算法(朴素贝叶斯算法、决策树算法、随机森林算法)和一个深度学习算法 Deeplog 作为基准模型。

4.2. 异常检测效果对比实验

为了验证本文模型在日志序列检测任务上的有效性，实验分别基于 BGL 数据集和 Thunderbird 数据集与四种基准模型进行比较。实验结果分别如表 1 和表 2 所示。从两个实验的结果可见，不同的滑动窗口对模型的检测性能有影响。三种机器学习方法容易受到滑动窗口的影响，导致模型检测性能波动较大。然而，本文模型在不同滑动窗口下均能保持出色的检测性能，有效检测日志序列中的异常。接下来将对两个实验进行更深入地分析。

Table 1. Experimental results on the BGL dataset
表 1. BGL 数据集上的实验结果

模型	评价指标	滑动窗口大小			
		5	10	15	20
NB	精确率	0.81	0.67	0.54	0.47
	召回率	0.80	0.88	0.88	0.87
	F1 值	0.80	0.76	0.67	0.61
DT	精确率	0.78	0.56	0.61	0.22
	召回率	0.81	0.70	0.32	0.74
	F1 值	0.80	0.62	0.42	0.34
RF	精确率	0.24	0.24	0.38	0.24
	召回率	0.92	**0.94**	0.92	0.91
	F1 值	0.38	0.38	0.54	0.38
Deeplog	精确率	0.98	0.97	**0.99**	**0.99**
	召回率	0.94	0.92	0.92	0.90
	F1 值	0.96	0.94	0.95	0.94
GTCLog	精确率	**0.99**	**0.98**	**0.99**	**0.99**
	召回率	**0.95**	**0.94**	**0.94**	**0.92**
	F1 值	**0.97**	**0.96**	**0.96**	**0.95**

Table 2. Experimental results on Thunderbird dataset
表 2. Thunderbird 数据集上的实验结果

模型	评价指标	滑动窗口大小		
		10	20	30
NB	精确率	0.96	0.93	0.93
	召回率	0.82	0.83	0.85
	F1 值	0.88	0.88	0.89
DT	精确率	0.49	0.73	0.33
	召回率	0.70	0.85	0.93
	F1 值	0.58	0.78	0.49
RF	精确率	0.98	0.96	0.56
	召回率	0.94	0.93	0.97
	F1 值	0.96	0.94	0.71
Deeplog	精确率	0.98	0.95	0.95
	召回率	**0.99**	**0.98**	0.97
	F1 值	0.98	0.96	**0.97**
GTCLog	精确率	**0.99**	**0.97**	**0.96**
	召回率	**0.99**	**0.98**	**0.98**
	F1 值	**0.99**	**0.97**	**0.97**

将 BGL 数据集以 8:2 进行划分，前 80%的数据当作训练集，后 20%的数据当作测试集，滑动窗口大小分别取 5，10，15，20，步长分别为滑动窗口大小的一半，步长值不为整数的，则向下取整，最终实验结果如表 1 所示。

从表 1 可见，模型的检测性能受到滑动窗口的影响。特别是三种机器学习方法，更容易受到滑动窗口的影响。例如，当滑动窗口为 5 时，NB 算法精确率为 0.81，但当滑动窗口大小为 20 时，其精确率下降到 0.47。GTCLog 具有良好的鲁棒性，即使滑动窗口大小改变，其检测性能仍保持在出色水平。与四个基线模型相比，本文模型在各指标上均取得了出色成绩。三种传统机器学习方法在 BGL 数据集上表现不理想，整体性能的 F1 值均未达到 0.85 以上。当滑动窗口为 5 时，GTCLog 的各指标均优于四个基准模型，如图 6 所示。

实验结果在 Thunderbird 数据集上如表 2 所示。将 Thunderbird 数据集以 7:3 划分，前 70%的数据作为训练集，后 30%的数据作为测试集。滑动窗口大小分别取 10，20，30，步长为滑动窗口大小的一半。从表 2 可见，GTCLog 在 Thunderbird 数据集上表现出色，在三个滑动窗口的实验中，各指标均达到最优。特别是在滑动窗口大小为 10 的实验中，本文模型的精确率、召回率和 F1 值均达到了 0.99，如图 7 所示。四个基线模型中，Deeplog 表现最好，其精确率达到 0.98，召回率达到 0.99，F1 值达到 0.98。表中同样可见，三个机器学习方法更容易受到滑动窗口大小的影响。例如，DT 算法的精确率波动较大，其值在三个滑动窗口下分别为 0.49，0.73，0.33，而 Deeplog 和 GTCLog 能保持稳定的检测性能，指标值均能达到 0.96 以上。

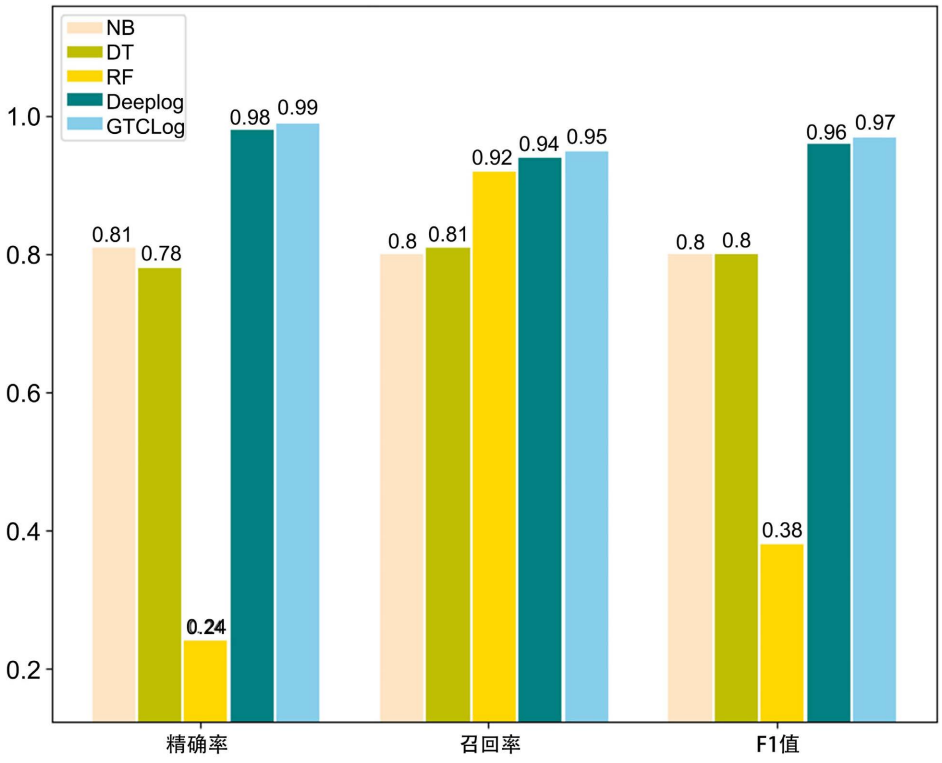

Figure 6. Experimental results for a sliding window of 5 on the BGL dataset
图 6. BGL 数据集上滑动窗口为 5 的实验结果

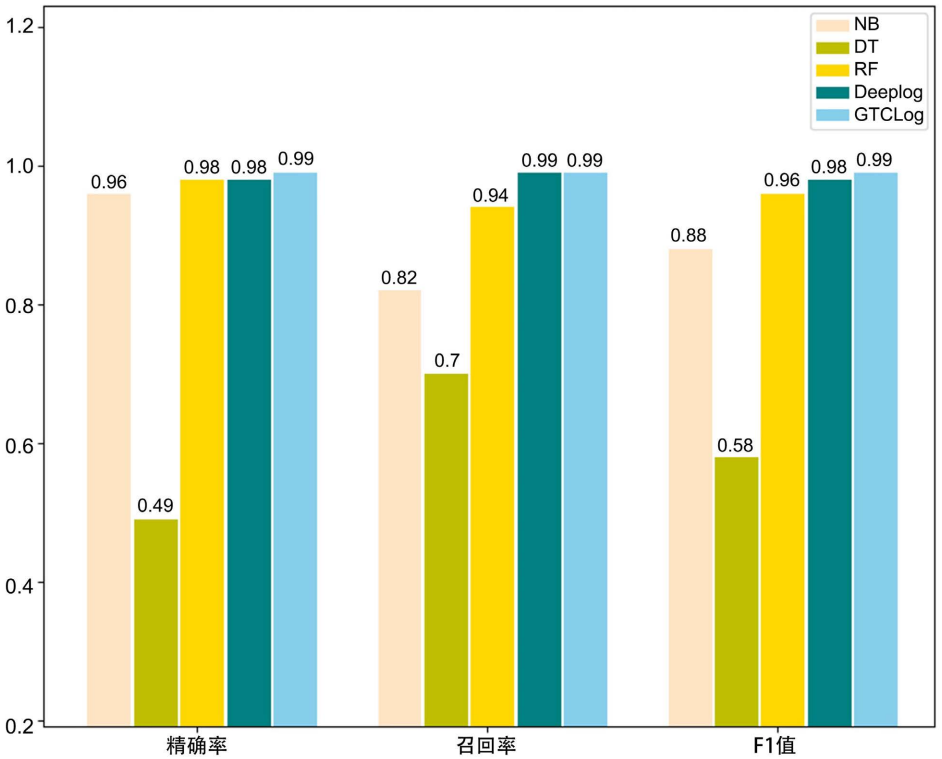

Figure 7. Experimental results for a sliding window of 10 on the Thunderbird dataset
图 7. Thunderbird 数据集上滑动窗口为 10 的实验结果

4.3. 消融实验

本文通过结合 GRU 和 TextCNN 提出了一种新的日志序列异常检测模型。为了验证其结合机制对模型检测性能的提升，本节通过消融实验进行实证研究。实验数据集选取 BGL，同样以 8:2 进行划分，滑动窗口大小设为 5，滑动步长设为 2。实验分别选取 TextCNN 和 GRU 作为基准模型，且对应的参数与本文模型保持相同。实验结果如表 3 所示。

Table 3. Ablation experiments
表 3. 消融实验

评价指标	TextCNN	GRU	GTCLog
精确率	0.98	0.90	**0.99**
召回率	0.86	0.92	**0.93**
F1 值	0.92	0.91	**0.96**

从表 3 可见，TextCNN 在精确率方面优于 GRU 模型，达到了 0.98。然而，在召回率方面，GRU 模型优于 TextCNN 模型，GRU 召回率达到了 0.92，而 TextCNN 模型的召回率仅为 0.86。本文模型综合了 TextCNN 模型和 GRU 模型的优势，使其性能得到提升。无论是精确率还是召回率，均实现了一定程度的提升。在 BGL 数据集上，准确率达到了 0.99，召回率达到了 0.93，F1 值达到了 0.96，各个指标均优于单一模型。

5. 结束语

本文首先介绍了日志序列异常检测存在的问题，然后针对这些问题提出了本文方法，并对其进行了详细阐述。本文方法将日志视为具有特殊语义的句子，通过句向量模型 SBERT 进行向量化，接着利用滑动窗口提取日志序列向量，最后基于本文模型 GTCLog 对日志序列向量进行异常检测。通过在两个公开数据集上评估本文模型，实验结果表明，本文提出的方法能够有效检测日志序列异常。

基金项目

本论文由温州市智能网络重点实验室开放课题资助。

参考文献

[1] 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.

[2] Xu, W., Huang, L., Fox, A., *et al*. (2009) Detecting Large-Scale System Problems by Mining Console Logs. *Proceedings of the ACM SIGOPS* 22*nd Symposium on Operating Systems Principles* (*SOSP*'09), Big Sky, 11-14 October 2009, 117-132. https://doi.org/10.1145/1629575.1629587

[3] Lou, J.G., Fu, Q., Yang, S., *et al*. (2010) Mining Invariants from Console Logs for System Problem Detection. *Proceedings of* 2010 *USENIX Annual Technical Conference*, Boston, 23-25 June 2010, 1-14.

[4] Chen, M., Zheng, A.X., Lloyd, J., *et al*. (2004) Failure Diagnosis Using Decision Trees. *Proceedings of* 2004 *International Conference on Autonomic Computing*, New York, 17-18 May 2004, 36-43. https://doi.org/10.1109/ICAC.2004.1301345

[5] Liang, Y., Zhang, Y., Xiong, H., *et al*. (2007) Failure Prediction in IBM BlueGene/L Event Logs. *Proceedings of Seventh IEEE International Conference on Data Mining* (*ICDM*), Omaha, 28-31 October 2007, 583-588. https://doi.org/10.1109/ICDM.2007.46

[6] Lu, S., Wei, X., Li, Y., *et al*. (2018) Detecting Anomaly in Big Data System Logs Using Convolutional Neural Network. *Proceedings of* 2018 *IEEE* 16*th International Conference on Dependable, Autonomic and Secure Computing*, 16*th International Conference on Pervasive Intelligence and Computing*, 4*th International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress* (*DASC*/*PiCom*/*DataCom*/*CyberSciTech*),

Athens, 12-15 August 2018, 151-158. https://doi.org/10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00037

[7] Du, M., Li, F., Zheng, G., *et al*. (2017) DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. *Proceedings of the* 2017 *ACM SIGSAC Conference on Computer & Communications Security* (*CCS*'17), Dallas, 30 October-3 November 2017, 1285-1298. https://doi.org/10.1145/3133956.3134015

[8] 张林栋, 鲁燃, 刘培玉. 基于双向长短时记忆网络的系统异常检测方法[J]. 计算机应用与软件, 2020, 37(12): 297-303+333.

[9] 周建国, 戴华, 杨庚, 等. 基于并列 GRU 分类模型的日志异常检测方法[J]. 南京理工大学学报, 2022, 46(2): 198-204.

[10] He, P.J., Zhu, J.M., He, S.L., *et al*. (2016) An Evaluation Study on Log Parsing and Its Use in Log Mining. *Proceeding of* 2016 46*th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (*DSN*), Toulouse, 28 June-1 July 2016, 654-661. https://doi.org/10.1109/DSN.2016.66

[11] Zhu, J., He, S., Liu, J., *et al*. (2019) Tools and Benchmarks for Automated Log Parsing. *Proceeding of* 2019 *IEEE*/*ACM* 41*st International Conference on Software Engineering*: *Software Engineering in Practice* (*ICSE-SEIP*), Montreal, 25-31 May 2019, 121-130. https://doi.org/10.1109/ICSE-SEIP.2019.00021

[12] Le, V.H. and Zhang, H. (2021) Log-Based Anomaly Detection without Log Parsing. *Proceedings of* 2021 36*th IEEE*/*ACM International Conference on Automated Software Engineering* (*ASE*), Melbourne, 15-19 November 2021, 492-504. https://doi.org/10.1109/ASE51524.2021.9678773

[13] Mikolov, T., Chen, K., Corrado, G., *et al*. (2013) Efficient Estimation of Word Representations in Vector Space. (Preprint)

[14] Pennington, J., Socher, R. and Manning, C.D. (2014) Glove: Global Vectors for Word Representation. *Proceedings of the* 2014 *Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), Doha, 26-28 October 2014, 1532-1543. https://doi.org/10.3115/v1/D14-1162

[15] Peters, M.E., Neumann, M., Iyyer, M., *et al*. (2018) Deep Contextualized Word Representations. *Proceedings of the* 2018 *Conference of the North American Chapter of the Association for Computational Linguistics*: *Human Language Technologies* (*NAACL-HLT* 2018), New Orleans, 1-6 June 2018, 2227-2237. (Preprint) https://doi.org/10.18653/v1/N18-1202

[16] Thakur, N., Reimers, N., Daxenberger, J., *et al*. (2021) Augmented Sbert: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks. *Proceedings of the* 2021 *Conference of the North American Chapter of the Association for Computational Linguistics*: *Human Language Technologies*, Online, 6-11 June 2021, 296-310. https://doi.org/10.18653/v1/2021.naacl-main.28

[17] Reimers, N. and Gurevych, I. (2019) Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks. *Proceedings of the* 2019 *Conference on Empirical Methods in Natural Language Processing and the* 9*th International Joint Conference on Natural Language Processing* (*EMNLP-IJCNLP*), Hong Kong, 3-7 November 2019, 3982-3992. (preprint) https://doi.org/10.18653/v1/D19-1410

[18] Cho, K., Van Merriënboer, B., Bahdanau, D., *et al*. (2014) On the Properties of Neural Machine Translation: Encoder—Decoder Approaches. https://arxiv.org/abs/1409.1259

[19] Kim, Y. (2014) Convolutional Neural Networks for Sentence Classification. *Proceedings of the* 2014 *Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), Doha, 25-29 October 2014, 1746-1751. https://doi.org/10.3115/v1/D14-1181

[20] Oliner, A. and Stearley, J. (2007) What Supercomputers Say: A Study of Five System Logs. *Proceedings of* 37*th Annual IEEE/IFIP International conference on Dependable Systems and Networks* (*DSN*'07), Edinburgh, 25-28 June 2007, 575-584. https://doi.org/10.1109/DSN.2007.103

[21] He, S., Zhu, J., He, P., *et al*. (2020) Loghub: A Large Collection of System Log Datasets towards Automated Log Analytics. (Preprint)