

苏州大学

本科毕业设计(论文)

学院(部) 计算机科学与技术学院

题 目 日志异常检测算法研究与实现

年 级 2017 级 专业 计算机科学与技术

班 级 人工智能实
验班 学号 1727405145

姓 名 陈健

指导老师 贾俊铖 职称 副教授

论文提交日期 2021/5/11

苏州大学
本科毕业设计（论文）独创性声明

本人郑重声明：所提交的本科毕业设计（论文）是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本设计（论文）不含其他个人或集体已经发表或撰写过的研究成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人承担本声明的法律责任。

作者签名：_____ 日 期：2021/5/13

苏州大学

本科毕业设计（论文）使用授权声明

本人完全了解苏州大学关于收集、保存和使用本科毕业设计（论文）的规定，即：本科毕业设计（论文）的著作权以及文中研究成果的知识产权归属苏州大学。苏州大学有权向国家有关部门或第三方机构送交毕业设计（论文）的复印件和电子文档，允许毕业设计（论文）被查阅和借阅，可以采用影印、缩印或其他复制手段保存和汇编毕业设计（论文），可以将毕业设计（论文）的全部或部分内容编入有关数据库进行检索。

涉密设计（论文）☐

本设计（论文）属 _____ 在 _____ 年 _____ 月解密后适用本规定。

非涉密设计（论文）☒

论文作者签名： _____ 日 期： 2021/5/13

导 师 签 名：  _____ 日 期： 2021/5/13

摘要

异常检测在大规模系统中起着至关重要的作用。日志可以由技术人员对其进行分析，及时察觉到系统出现的问题，并且通过技术手段进行修复。在初期阶段，技术人员通常使用正则表达式和基于代码等方式手动检查日志的异常情况。但是，随着技术的快速发展，系统的复杂度变得越来越高，相当多的日志产生出来，给技术人员带来了巨大的麻烦。为了使人工工作量降低，许多关于自动分析日志的方法被提出并被加以实验。

日志异常检测的研究包含诸多领域，例如基于规则和时间序列的异常检测等。近年来机器学习快速崛起，深度学习也随之成熟，日志异常检测中关于人工智能的检测方法正被不断提出。本文对常见的日志异常检测算法进行了简要概括，介绍了日志异常检测算法的步骤，分别介绍了现阶段日志解析和日志异常检测的相关技术，对基于最长公共子序列(Longest Common Subsequence, 简称 LCS)的日志解析技术的原理进行了详细说明，并进行了相关实验进行验证。此外，本文还对基于长短时记忆(Long Short-Term Memory, 简称 LSTM)的日志异常检测算法进行了相关实验，探讨了在不同的参数设置下对其效率和有效性的影响。

关键词：日志异常；日志解析；异常检测

Abstract

Anomaly detection plays an important role in large-scale systems. The log can be analyzed by technicians to detect the system problems in time and repair them by technical means. In the early stage, technicians usually use regular expressions and code-based methods to manually check the log for exceptions. However, with the rapid development of technology, the complexity of the system becomes higher and higher, a considerable number of logs are generated, which brings great trouble to the technicians. In order to reduce the manual workload, many methods of automatic log analysis are proposed and tested.

The research of log anomaly detection includes many fields, such as anomaly detection based on rules and time series. In recent years, with the rapid rise of machine learning and the maturity of deep learning, the detection methods of artificial intelligence in log anomaly detection are constantly proposed. This paper briefly introduces the common log anomaly detection algorithms, analyzes the steps of the log anomaly detection algorithm, introduces the related technologies of log parsing and log anomaly detection at this stage, explains the principle of log parsing technology based on the LCS(longest common subsequence) in detail, and carries out relevant experiments to verify. In addition, this paper also makes experiments on log anomaly detection algorithm based on LSTM(long short term memory), and discusses the influence of different parameter settings on its efficiency and effectiveness.

Keywords: log anomaly; log analysis; abnormal detection

目录

前言.....	1
第 1 章 绪论.....	2
1.1 研究背景及意义	2
1.2 研究现状	2
1.3 本文的主要工作	3
1.4 本文的组织结构	3
第 2 章 日志异常检测技术.....	5
2.1 日志收集	5
2.2 日志解析	6
2.3 特征提取	8
2.4 异常检测	8
2.5 本章小结	9
第 3 章 基于 LCS 的日志解析算法研究.....	11
3.1 实验原理	11
3.2 实验过程	13
3.2.1 实验环境	13
3.2.2 实验数据	13
3.2.3 实验结果与分析	13
3.3 本章小结	14
第 4 章 基于 LSTM 的日志异常检测算法研究.....	16
4.1 实验原理	16
4.1.1 LSTM 简介	16
4.1.2 基于 LSTM 的异常检测	19
4.2 实验过程	21

4.2.1 实验环境	21
4.2.2 实验数据	21
4.2.3 实验指标	22
4.2.4 实验参数	22
4.2.5 实验结果	23
4.2.6 实验分析	26
4.3 本章小结	27
第 5 章 总结与展望.....	28
5.1 本文总结	28
5.2 后续工作与展望	28
参考文献.....	30
致谢.....	32

前言

主机系统的运行过程会产生诸多类别的日志，日志保存了系统的工作状态和操作指令，可以用于实时观测系统和进行异常检测，因此对系统日志进行相关的检测可以作为判别系统是否异常的手段之一。

当前市场存在各式各样的日志检测系统，然而在实际使用中，由于系统类型具有差别，其产生的日志也多种多样，因此检测的结果也不尽如人意，经常不够准确。

近年来，随着机器学习等领域的快速发展，相关技术得到成熟，与机器学习相关的日志检测技术也得到了进一步的发展。但是对于不同的系统，其日志类型有着巨大的差异，而机器学习需要对其特征进行提取，这要求使用者有着严格的专业知识。而实际中，由于日志的种类和数量都在快速扩大，没有一个方法是通用的，因此日志的匹配工作需要消耗大量的人工成本。

相比之下，使用深度学习来研究异常检测具有良好的效果。系统的数据量在不断地扩增，深度学习的结果依赖大量数据，因此如果设置好参数，可以不需要人工进行提取，因此深度学习模型可以对日志检测进行良好的测试。

本文主要以日志异常检测流程作为研究对象，介绍现有的日志解析和异常检测算法，对比各种算法的优缺点。在深入分析各种日志异常算法的基础上，对日志解析和日志异常检测算法的有效性和效率进行了系统评估，并进行了实验验证。本文完成的主要工作有：

（1）对目前常见的日志异常检测算法进行介绍。

（2）通过实验验证了基于最长公共子序列(Longest Common Subsequence, 简称 LCS)^[1]的日志解析算法^[2]。

（3）利用 HDFS(Hadoop Distributed File System)数据集研究了基于长短时记忆(Long Short-Term Memory, 简称 LSTM)^[3]的日志异常检测算法^[4]的效率。

第 1 章 绪论

本章首先介绍了日志异常检测算法的研究背景及意义和研究现状，概述了本文所做的主要工作，在本章的最后介绍了论文的组织结构。

1.1 研究背景及意义

现代操作系统和网络应用等会产生大量的日志，这些日志可以记录系统的状态变化，包含巨量的信息。技术人员可以依据系统日志对系统的状态进行判断，从而对错误快速定位并改正，提高了系统的稳定性。

传统的日志异常检测方法主要是基于人工检测和正则表达式规则。在小型日志中，人工检测可以起到不错的效果，但是缺点是劳动重复且对人有较高的经验要求，并且检测速度很难让人满意。基于正则表达式的规则可以识别出错误日志，速度得到进一步提升，但是问题是对于每种类型的日志语句，都要设置对应的正则表达式，实现难度和工程量都比较大，且对人员的专业性要求极高。

由于计算机能力的提升和网络的普及与发展，各个系统、网络和应用程序规模不断扩大，日志数据量激增，并且系统行为过于复杂，导致开发人员无法理解，传统的检测方法已无法适应当前日志的异常检测。机器学习已经成为日志异常检测的常用方法，各种相关领域的日志异常检测算法被不断提出并加以实验，取得了不错的效果。

1.2 研究现状

日志记录了系统的相关信息，通过对日志的异常检测可以发现系统中存在的问题，为技术人员解决系统错误提供了一个有效的途径。日志异常检测的重点为日志解析和异常检测。在日志解析方面主要的问题有：分布式系统产生的日志信息量巨大，对日志解析的性能要求极高；不同类型的日志包含的内容不同，多种系统产生的日志有不同的模板，因此多样化是日志的特点之一；开发人员使用的日志结构不同，导致日志具有非结构化的特点，因此自动地进行日志的难度很高。目前研究人员采用的策略是

分离出日志的模板和日志参数，以实现日志的解析。日志解析技术已经相对成熟，目前常见的有基于正则表达式的日志解析、基于代码的日志解析、基于机器学习和深度学习的日志解析等，并且已经成功地应用于实践之中。

然而由于日志的多样性和复杂性，现在仍然缺少通用的日志解析方法可以对现实中所有的日志进行解析，更多地是面对各种日志进行针对性的解析设置，以取得良好的结果。

非结构化的文本通过日志解析可以转化为结构化的文本，其具有广泛的应用，例如异常检测、程序分析和性能诊断等。日志的异常检测是指根据系统产生的日志寻找出数据中不符合预期的日志，这种情况表明系统发生了故障，因此日志异常检测可以帮助技术人员发现系统问题并修复系统。

当前基于日志的异常检测已经有很多，在系统日志日渐复杂的背景下，人工检查日志已经变得不可行。因此自动化的进行日志异常检测已经变得非常迫切，当前日志异常检测的相关方法已经有很多，主要是基于机器学习的相关技术，例如基于规则的日志异常检测、基于图的日志异常检测、基于时间序列的日志异常检测等，并且得到不错的检测效果。

1.3 本文的主要工作

本文主要以日志异常检测算法为研究对象，分别介绍现有的各种日志解析和异常检测算法，对比各种检测算法的优缺点，并对有效性和效率进行对比。本文重点介绍了日志解析和异常检测技术及原理，基于最长公共子序列方法对 HDFS 日志进行日志解析，得出对应的日志键和日志参数，并使用 HDFS 数据集对基于 LSTM 的日志异常检测算法进行实验，研究其效率和有效性。

1.4 本文的组织结构

本文共分为五章，各章内容安排如下：

第一章：绪论。本章介绍了课题的研究背景及意义、研究现状、日志异常检测技术的各个步骤、本文的主要工作，最后介绍了本文的组织结构。

第二章：日志异常检测流程介绍。本章重点阐述了常见的日志解析技术和异常检

测算法，并对比分析了各自领域相关技术的发展情况。

第三章：主要介绍了基于最长公共子序列 LCS 的日志解析算法，并通过 HDFS 日志对日志解析算法进行了测试。

第四章：主要介绍了深度学习中基于 LSTM 的日志异常检测算法。对该算法的工作原理进行详细介绍，对该算法进行实验验证，并对比不同参数设置对该算法的效率和有效性的影响。

第五章：总结全文，提出未来工作的设想与展望。

第 2 章 日志异常检测技术

日志的异常检测由 4 部分组成^[5]，分别是日志收集和预处理、日志解析、特征提取和异常检测。该过程在图 2.1 中示出，下面分别对其进行描述。

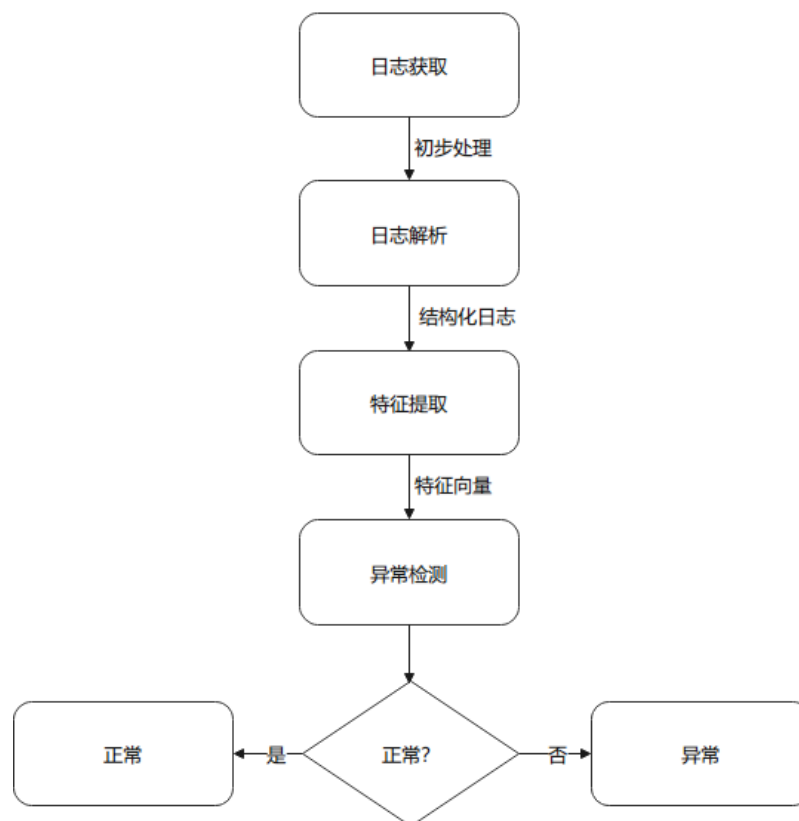


图 2.1 日志异常检测流程图

2.1 日志收集

对于日志异常检测，首先进行的是日志收集，当前大型系统会实时地产生日志来记录系统的当前状态，这些信息可以用于分析系统的状态，异常检测等。图 2.2 描绘了某 Linux 系统的部分数据。在收集完数据后，根据需求对日志中无效的信息（如重复信息）进行删除。

```

Jun 15 02:04:59 combo sshd(pam_unix)[20882]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net user=root
Jun 15 02:04:59 combo sshd(pam_unix)[20884]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net user=root
Jun 15 02:04:59 combo sshd(pam_unix)[20883]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net user=root
Jun 15 02:04:59 combo sshd(pam_unix)[20885]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net user=root
Jun 15 02:04:59 combo sshd(pam_unix)[20886]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net user=root
Jun 15 02:04:59 combo sshd(pam_unix)[20892]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net user=root
Jun 15 02:04:59 combo sshd(pam_unix)[20893]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net user=root
Jun 15 02:04:59 combo sshd(pam_unix)[20896]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net user=root
Jun 15 02:04:59 combo sshd(pam_unix)[20897]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net user=root
Jun 15 02:04:59 combo sshd(pam_unix)[20898]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=220-135-151-1.hinet-ip.hinet.net user=root
Jun 15 04:06:18 combo su(pam_unix)[21416]: session opened for user cyrus by (uid=0)
Jun 15 04:06:19 combo su(pam_unix)[21416]: session closed for user cyrus
Jun 15 04:06:20 combo logrotate: ALERT exited abnormally with [1]
Jun 15 04:12:42 combo su(pam_unix)[22644]: session opened for user news by (uid=0)
Jun 15 04:12:43 combo su(pam_unix)[22644]: session closed for user news
Jun 15 12:12:34 combo sshd(pam_unix)[23397]: check pass; user unknown
Jun 15 12:12:34 combo sshd(pam_unix)[23397]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 15 12:12:34 combo sshd(pam_unix)[23395]: check pass; user unknown
Jun 15 12:12:34 combo sshd(pam_unix)[23395]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 15 12:12:34 combo sshd(pam_unix)[23404]: check pass; user unknown
Jun 15 12:12:34 combo sshd(pam_unix)[23404]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 15 12:12:34 combo sshd(pam_unix)[23399]: check pass; user unknown
Jun 15 12:12:34 combo sshd(pam_unix)[23399]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 15 12:12:34 combo sshd(pam_unix)[23406]: check pass; user unknown
Jun 15 12:12:34 combo sshd(pam_unix)[23406]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 15 12:12:34 combo sshd(pam_unix)[23396]: check pass; user unknown
Jun 15 12:12:34 combo sshd(pam_unix)[23394]: check pass; user unknown
Jun 15 12:12:34 combo sshd(pam_unix)[23407]: check pass; user unknown
Jun 15 12:12:34 combo sshd(pam_unix)[23394]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 15 12:12:34 combo sshd(pam_unix)[23403]: check pass; user unknown
Jun 15 12:12:34 combo sshd(pam_unix)[23396]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 15 12:12:34 combo sshd(pam_unix)[23407]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 15 12:12:34 combo sshd(pam_unix)[23403]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 15 12:12:34 combo sshd(pam_unix)[23412]: check pass; user unknown
Jun 15 12:12:34 combo sshd(pam_unix)[23412]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Jun 15 12:13:19 combo sshd(pam_unix)[23414]: check pass; user unknown

```

图 2.2 Linux 系统日志

2.2 日志解析

在常见的系统中，日志都是非结构化的文本，日志解析可以从系统产生的日志中进行提取，提取出某条日志对应的日志模板。日志可以分为日志键和日志参数，因此在进行异常检测前，我们需要将两者分开，将日志解析成结构化。每一行日志都是源代码的输出语句生成的。

如图 2.3 所示，某个进程的源代码中的日志打印语句为：

```
Print("Accept password for %s port from %s port %s ssh",user,host,port)
```

那么在程序的运行过程中，可能会产生：

```
Accept password for root port from 112.64.243.186 port 2371 ssh
```

一条源代码打印出的日志为同一类型，其代码称为日志键，该日志键的形式即为：

```
Accept password for root port from * port * ssh
```

它的日志参数为：

```
['root','112.64.243.186','2371']
```

将日志键和日志参数进行分离可以为后续的特征向量提取进行铺垫。

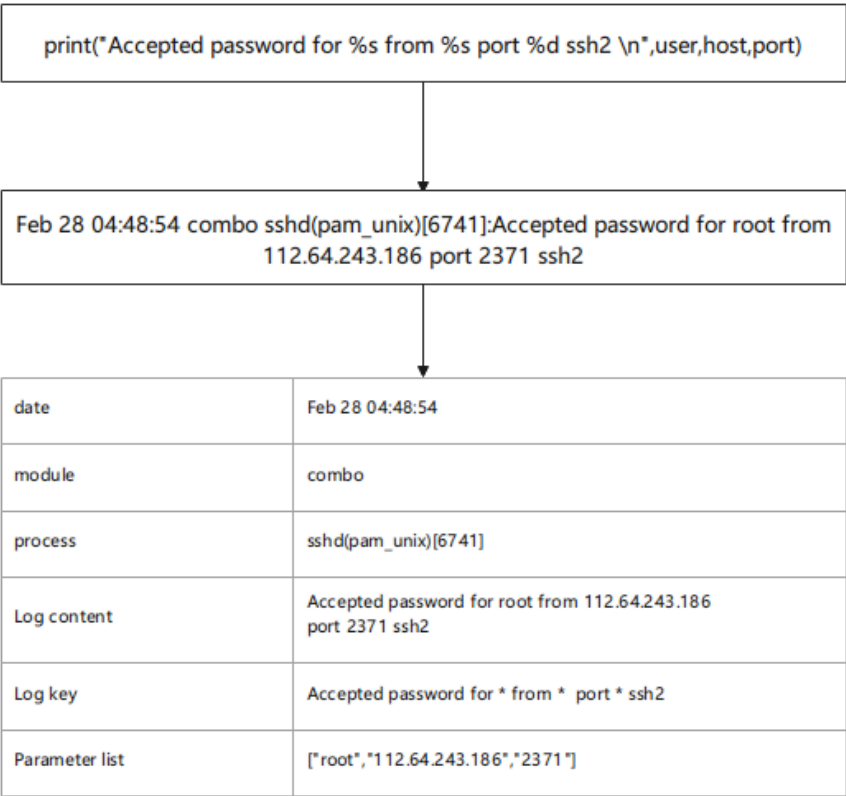


图 2.3 日志解析示例

在现代大规模日志系统中，想要尽可能地理解日志的内容，日志解析是必要的。随着技术的发展，日志解析的效率也不断得到提升，当前已经提出了许多日志解析技术。不同的日志解析技术各有其特点，常见的日志解析技术优缺点如表 2.1 所示。

表 2.1 常见日志解析技术对比

日志解析技术	优点	缺点
基于正则表达式的日志解析技术 ^[6]	方法简单、技术成熟	依赖人力、效率低
基于代码分析的日志解析技术	适用于离线解析、技术成熟	许多情况下不容易获取代码
IPLoM	算法效率高、不会受到阈值影响	——
Drain	流式及时解析、小规模日志在线解析中效率较高	大规模日志处理效率低
SLCT	最早自动化的解析工具	存在过拟合

可以看出，现有的日志解析技术各有其优缺点，在对于简单的日志、复杂离线日

志和复杂在线日志的问题处理上，不同的日志解析技术有不同的适用情况，但是至今为止还没有一个适用于各种情况的日志解析技术可供使用。日志解析可以导出日志的日志键和日志参数，为后续的特征提取和异常检测提供了方便。

2.3 特征提取

基于机器学习的日志异常检测，需要的输入是特征向量，日志解析后的输出是日志键和日志参数，因此需要提取特征，以便转换为可用于机器学习的数据形式，一般转换为事件计数矩阵^[7]。

如图 2.4 所示，日志特征提取有三种类型：固定窗口，滑动窗口和会话窗口，它们以不同的方式形成事件计数矩阵 X 。矩阵代表日志事件发生的次数。固定窗口和滑动窗口均采用时间戳对日志键进行编码，不同的是固定窗口采用的为定长窗口，每个固定窗口内部发生的日志被视为一个日志序列；滑动窗口由窗口大小和步长共同决定，同一个滑动窗口内部的日志也被视为一个日志序列，但是可能会存在多个重叠。会话窗口采用的不是时间戳作为分组依据，而是采用标识符，标识符存储日志序列中不同的执行路径，例如分配、复制或删除等，因此可以根据标识符将日志分为不同的日志序列，每个会话窗口的标识符唯一。

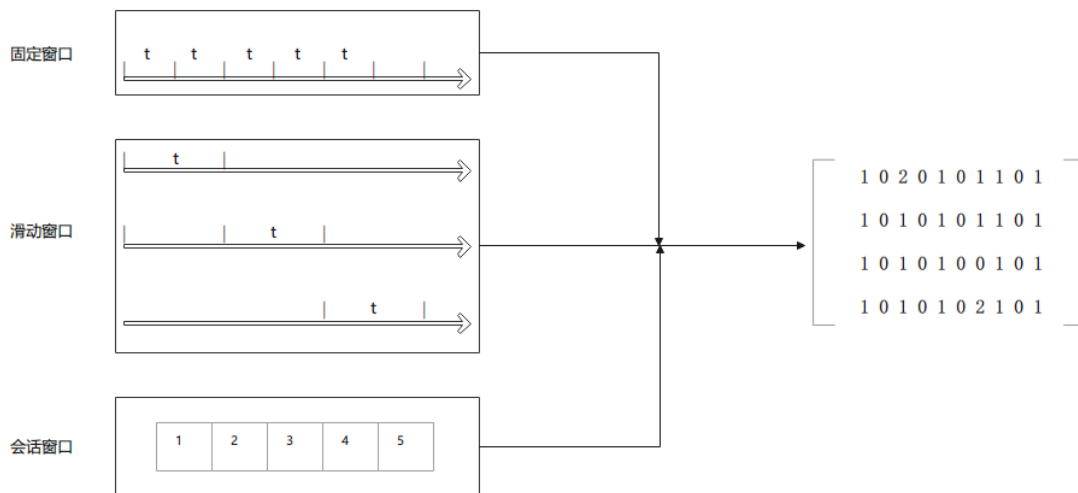


图 2.4 日志特征提取分类

2.4 异常检测

异常检测为整个日志异常检测流程的最后一步，其过程主要为设计一个模型，针

对输入的日志序列进行检测，判定该日志序列属于正常日志还是异常日志。

目前，对日志异常检测方法的研究已经有了很多尝试。由于当前系统产生的日志的规模和复杂程度均不断扩大和提高，基于人工的人工日志异常检测变得不可行。因此，对复杂日志异常检测的需求不断扩大，并提出了许多相应的自动异常检测方法，包括基于规则，工作流和图的异常检测方法^[8]。

在当前机器学习领域蓬勃发展的背景下，相关的日志异常检测算法也得到了广泛的研究和应用，诸多机器学习方法如聚类^[9]，生成对策网络^[10]，卷积神经网络^[11,12]都被相关学者进行异常检测方面的尝试。部分异常检测算法的效率如表 2.2 所示。

表 2.2 部分日志异常检测算法效率对比

名称	分类	基于算法	模式	准确率	召回率
ACLog	监督学习	SVM、LCS	离线	92.4%	80%
IM	监督学习	K-prototype、KNN	离线	89%	85%
LogCluster	无监督学习	聚类算法	在线	60%	36.2%
LogGAN	深度学习	LSTM	离线	100%	35.6%
LogRobust	深度学习	Bi-LSTM	在线	98%	100%
DeepLog	深度学习	LSTM	在线	95%	96%

日志异常检测算法的应用已经十分广泛，越来越多的机器学习相关技术正在不断发展，并融入其中。总而言之，现在的日志异常检测算法仍在不断发展，并取得了不错的成果，但仍存在着许多问题。例如部分算法只是针对特定的场景和数据集进行处理，适用性受到限制且大部分算法均为离线处理，不能够及时地对新产生的日志进行处理，仍缺少相对通用的算法技术。

2.5 本章小结

本章详细介绍了日志异常检测算法的日志获取、日志解析、特征提取和异常检测四个步骤，其中日志解析和异常检测是日志异常检测算法的重点和难点部分，当前日志异常检测的相关研究也主要针对这两个部分进行。

总的来说，对于日志异常检测流程，首先日志获取读取系统产生的日志，并进行初步的处理，将无效的信息进行剔除。日志解析将非结构化的文本处理为结构化，分

离为日志键和日志参数，日志键是一条日志的模板，日志参数为模板对应的参数。结构化的日志仍为字符串，因此需要对结构化的日志进行特征提取，将结构化日志转换为特征向量。日志异常检测的最后一步为异常检测，将处理好的特征向量作为输入，对异常检测模型进行训练，训练好的模型可以对新输入的测试数据进行判断，输出为该测试日志为正常日志还是异常日志。

第 3 章 基于 LCS 的日志解析算法研究

近年来，由于机器学习的发展，对系统日志的分析也取得了很大的成果。分析系统日志的第一步通常是日志解析，现在已经有很多系统日志解析工具。面对不同的机器学习方法，不同的日志解析方法各有优势，本章采用的算法是一种基于 LCS 的异常检测算法^[14]，通过比较多条日志序列，找到其最长公共子序列，以此提取出日志的日志键和日志参数。

3.1 实验原理

日志都是由程序生成的，则日志的格式也是按照一定的模板生成的，例如日志：

```
connection from 222.33.90.199
```

则其必定是基于程序代码：

```
printf(connection from %s,tmp)
```

那么其模板为：

```
connection from *
```

日志可以分为日志键（日志模板）和日志参数，以进行进一步的操作。

问题在于，当输入一条新的日志时：

```
connection from 222.33.90.200
```

如何提取出该日志的日志键，并寻找与之相同的日志键是一个难点。本实验提供了一种思路解决这种问题，即在与其他日志序列的比较中提取相同的部分作为日志键。

首先定义日志对象列表为 `LCSMap`，该日志对象列表内保存着一系列日志对象 `LCSObject`，日志对象的属性包括日志键 `LCSseq` 和行数列表 `lineIds`。

假设现在输入的日志序列为：

```
connection from 222.33.90.199
```

遍历 `LCSMap` 后，假设有一个 `LCSObject` 的 `LCSseq` 为：

```
connection from 222.33.90.200
```

则可以得到这两个日志的最长公共子序列为：

connection from *

如果最长公共子序列的长度介于日志长度的 1/2 至 1 之间，则可以判定这两个日志的日志键是相同的，然后将其合并，并将该日志的行数添加到日志对象 LCSObject 的 lineIds 属性中。

过程图如图 3.1 所示：

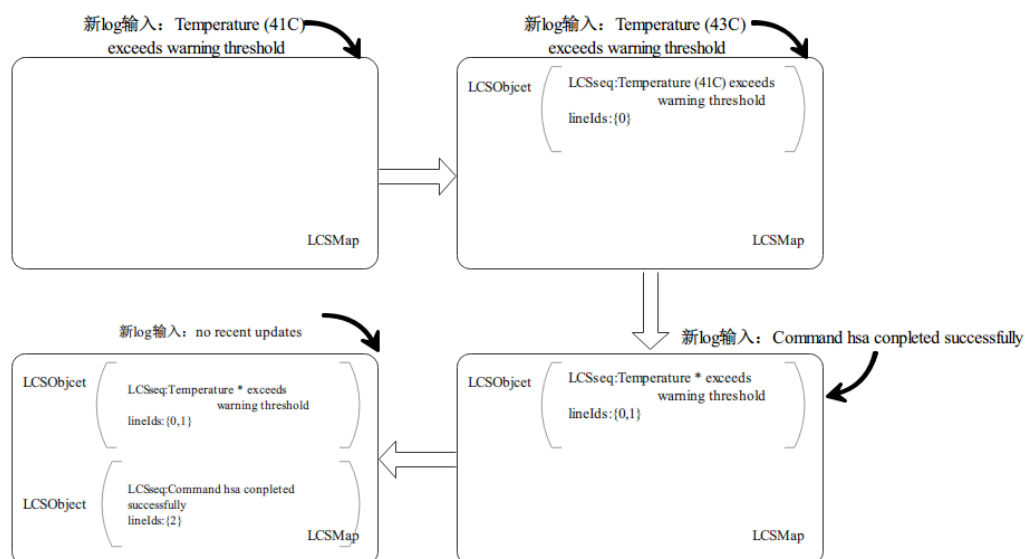


图 3.1 基于 LCS 的日志解析流程

因此基于 LCS 解析日志的完整过程如下：

- （1）对日志对象 LCSObject 和日志对象 LCSMap 进行初始化。
- （2）对日志进行按行读取。
- （3）对于一条日志，在 LCSMap 中的 LCSObject 寻找与其具有最长公共子序列的序列，若最长公共子序列的长度介于该日志长度的 1/2 至 1 之间，则认为匹配成功，进行步骤（5），若匹配失败则进行步骤（4）。
- （4）为该日志新建一个日志对象 LCSObject，并置入 LCSMap 中。
- （5）将日志的行数放入到匹配到的日志对象 LCSObject 的 lineIds 中，并更新日志对象的 LCSseq。
- （6）重新运行第 2 步，直到日志读取完毕。

3.2 实验过程

3.2.1 实验环境

实验环境基于 Windows10，编程语言为 Python3.7.5。

3.2.2 实验数据

实验的数据采用了公开的 HDFS 数据集，它是亚马逊 EC2 平台收集到的 Hadoop 分布式系统日志，共包含了 11175629 条日志消息，本次实验选取其中 1000 条作为实验数据对日志进行解析，部分实验数据如图 3.2 所示。

```
081109 204005 35 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.73.220:50010 is added to blk_7128370237687728475 size 67108864
081109 204015 308 INFO dfs.DataNode$PacketResponder: PacketResponder 2 for block blk_822919380324955061 terminating
081109 204106 329 INFO dfs.DataNode$PacketResponder: PacketResponder 2 for block blk_-6670958622368987959 terminating
081109 204132 26 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.43.115:50010 is added to blk_3050920587428079149 size 67108864
081109 204324 34 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.203.80:50010 is added to blk_7888946331804732825 size 67108864
081109 204453 34 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.11.85:50010 is added to blk_2377150260128098806 size 67108864
081109 204525 512 INFO dfs.DataNode$PacketResponder: PacketResponder 2 for block blk_572492839287299681 terminating
081109 204655 556 INFO dfs.DataNode$PacketResponder: Received block blk_3587508140051953248 of size 67108864 from /10.251.42.84
081109 204722 567 INFO dfs.DataNode$PacketResponder: Received block blk_5402003568334525940 of size 67108864 from /10.251.214.112
081109 204815 653 INFO dfs.DataNode$DataXceiver: Receiving block blk_579248908079196128 src: /10.251.30.6:33145 dest: /10.251.30.6:50010
081109 204842 663 INFO dfs.DataNode$DataXceiver: Receiving block blk_1724757848743533110 src: /10.251.111.130:49851 dest: /10.251.111.130:50010
081109 204908 31 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.110.8:50010 is added to blk_8015913224713045110 size 67108864
081109 204925 673 INFO dfs.DataNode$DataXceiver: Receiving block blk_-562317679330377570 src: /10.251.75.228:53725 dest: /10.251.75.228:50010
081109 205035 28 INFO dfs.FSNamesystem: BLOCK* NameSystem.allocateBlock: /user/root/./rand/./temporary/_task_200811092030_0001_m_000590_0/part-00590. blk_-1727475099218
081109 205056 710 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_5017373558217225674 terminating
081109 205157 752 INFO dfs.DataNode$PacketResponder: Received block blk_9212264480425680329 of size 67108864 from /10.251.123.1
081109 205315 29 INFO dfs.FSNamesystem: BLOCK* NameSystem.allocateBlock: /user/root/./rand/./temporary/_task_200811092030_0001_m_000742_0/part-00742. blk_-7878121102358
081109 205409 28 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.111.130:50010 is added to blk_4568434182693165549 size 67108864
081109 205412 832 INFO dfs.DataNode$PacketResponder: Received block blk_-570489971262113150 of size 67108864 from /10.251.91.229
081109 205632 28 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.74.79:50010 is added to blk_-4794867979917102672 size 67108864
081109 205739 29 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.38.197:50010 is added to blk_8763662564934652249 size 67108864
081109 205742 1001 INFO dfs.DataNode$PacketResponder: Received block blk_-5861636720645142679 of size 67108864 from /10.251.70.211
081109 205746 29 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.74.134:50010 is added to blk_7453815855294711849 size 67108864
081109 205749 997 INFO dfs.DataNode$DataXceiver: Receiving block blk_-28342503914935090 src: /10.251.123.132:57542 dest: /10.251.123.132:50010
081109 205754 952 INFO dfs.DataNode$PacketResponder: Received block blk_8291449241650212794 of size 67108864 from /10.251.89.155
081109 205858 31 INFO dfs.FSNamesystem: BLOCK* NameSystem.allocateBlock: /user/root/./rand/./temporary/_task_200811092030_0001_m_000487_0/part-00487. blk_-5319073033164
081109 205931 13 INFO dfs.DataBlockScanner: Verification succeeded for blk_-4980316519894289629
081109 210022 1110 INFO dfs.DataNode$PacketResponder: Received block blk_-5974833545991408899 of size 67108864 from /10.251.31.180
081109 210037 1084 INFO dfs.DataNode$DataXceiver: Receiving block blk_-5009020203888190378 src: /10.251.199.19:52622 dest: /10.251.199.19:50010
081109 210248 1138 INFO dfs.DataNode$PacketResponder: Received block blk_692167471195988070 of size 67108864 from /10.251.65.203
081109 210407 33 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.7.244:50010 is added to blk_5165786360127153975 size 67108864
081109 210458 1278 INFO dfs.DataNode$DataXceiver: Receiving block blk_2937758977269298350 src: /10.251.194.129:37476 dest: /10.251.194.129:50010
081109 210551 32 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.6.191:50010 is added to blk_673825774073966710 size 67108864
081109 210637 1283 INFO dfs.DataNode$PacketResponder: Received block blk_-752694544867194862 of size 67108864 from /10.251.203.80
081109 210656 1334 INFO dfs.DataNode$PacketResponder: Received block blk_-2094397855762091248 of size 67108864 from /10.251.126.83
081109 210712 1333 INFO dfs.DataNode$PacketResponder: Received block blk_-8523968015014407246 of size 67108864 from /10.251.214.225
081109 210743 27 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.89.155:50010 is added to blk_8181993091797661153 size 67108864
```

图 3.2 HDFS 系统日志

3.2.3 实验结果与分析

实验结果的模板为图 3.3 所示：

“日志键序号{“lcsseq”:日志模板,“lineids”:行数列表,“position”:日志参数位置}”

图 3.3 实验结果模板

该实验结果模板的解释如下：

- (1) 日志键序号：代表了当前共有多少个日志键存在，若有一个新的日志键出现，则日志键序号加 1；
- (2) lcsseq：日志模板为日志键对应日志的模板，如 “* combo * authentication failure”，该日志键具有两个日志参数；

(3) **lineids**: 符合本日志键的日志所在的行数, 每发现一个符合日志模板的日志, 便将其写入对应的日志键序号所在的 **lineids** 的行数列表中;

(4) **posotion**: 日志参数在该条日志模板中对应的位置, 以列表形式给出, 如日志模板“* combo * authentication failure”对应的参数位置列表为[0,2]。

数据集日志解析部分运行情况如图 3.4 所示。

```
the lcsmap are:
0 {"lcsseq": "* INFO dfs.DataNode$PacketResponder: PacketResponder * for block * terminating ", "lineids": [1, 2, 4, 5, 9, 17, 46, 49, 53, 54, 56, 68, 72, 187
1 {"lcsseq": "* INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: * is added to * size * ", "lineids": [3, 6, 7, 8, 14, 20, 22, 23, 2
2 {"lcsseq": "* INFO dfs.DataNode$PacketResponder: Received block * of size * from * ", "lineids": [10, 11, 18, 21, 24, 27, 30, 32, 36, 37, 38, 41, 42, 44, 47
3 {"lcsseq": "* INFO dfs.DataNode$DataXceiver: Receiving block * src: * dest: * ", "lineids": [12, 13, 15, 26, 31, 34, 40, 43, 57, 71, 106, 112, 114, 115, 116
4 {"lcsseq": "081109 * INFO dfs.FSNamesystem: BLOCK* NameSystem.allocateBlock: * ", "lineids": [16, 19, 28, 58, 59, 121, 138, 140], "postion": [1, 6]}
5 {"lcsseq": "* 13 INFO dfs.DataBlockScanner: Verification succeeded for * ", "lineids": [29, 70, 176, 197, 346, 347, 348, 358, 569, 646, 699, 755, 781, 790,
6 {"lcsseq": "* INFO dfs.FSDataset: Deleting block * file * ", "lineids": [73, 426, 428, 429, 430, 431, 432, 433, 435, 438, 439, 440, 441, 442, 443, 445, 446,
7 {"lcsseq": "081109 * INFO dfs.DataNode$DataXceiver: * Served block * to * ", "lineids": [74, 75, 76, 77, 80, 83, 87, 90, 97], "postion": [1, 4, 7, 9]}
8 {"lcsseq": "* WARN dfs.DataNode$DataXceiver: * exception while serving * to * ", "lineids": [78, 79, 81, 82, 84, 85, 86, 88, 89, 91, 92, 93, 94, 95, 96, 98,
9 {"lcsseq": "081110 * INFO dfs.FSNamesystem: BLOCK* NameSystem.allocateBlock: * ", "lineids": [151, 154, 175, 185, 187, 201, 204, 220, 243, 250, 253, 254, 25
10 {"lcsseq": "* INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: * is added to invalidSet of * ", "lineids": [292, 362, 363, 364, 365, 366, 367, 368, 369, 37
11 {"lcsseq": "081110 * INFO dfs.DataNode$DataXceiver: * Served block * to * ", "lineids": [293, 295, 298, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 3
12 {"lcsseq": "081110 211541 18 INFO dfs.DataNode: 10.250.15.198:50010 Starting thread to transfer block blk_4292382298896622412 to 10.250.15.240:50010 ", "li
13 {"lcsseq": "081110 * 19 INFO dfs.FSNamesystem: BLOCK* ask * to delete * ", "lineids": [928, 1029], "postion": [1, 7, 10]}
14 {"lcsseq": "081111 * INFO dfs.DataNode$DataXceiver: * Served block * to * ", "lineids": [1116, 1117, 1118, 1119, 1121, 1124, 1125, 1126, 1128, 1129, 1157,
15 {"lcsseq": "081111 * INFO dfs.FSNamesystem: BLOCK* NameSystem.allocateBlock: * ", "lineids": [1185, 1186, 1219, 1224, 1234, 1244, 1246, 1272, 1295, 1345, 1
16 {"lcsseq": "081111 * INFO dfs.DataNode$DataXceiver: Received block * src: * dest: * of size 67108864 ", "lineids": [1439, 1768], "postion": [1, 6, 8, 10]}
```

图 3.4 日志解析结果

从结果可以看到, 该算法能够将较好的将日志进行解析, 分离出日志的日志键和日志参数, 将非结构化的日志转换为结构化的形式, 且准确程度较高, 基本能够实现日志的解析, 并能够对日志键进行编码, 方便后续进行日志键特征向量的提取, 并且返回出日志参数的位置, 这对于部分异常检测模型中使用日志参数训练的算法提供了方便。

该算法也存在部分问题尚待解决, 首先如果数据量不够大, 某些日志仅存在一条, 则只会保存该日志而不能保存其模板, 因此如果在使用时需要确保相似的日志出现两次及以上才能达到理想效果。此外在提取过程中, 需要保证寻找到的最长公共子序列的长度介于日志序列的 $1/2$ 至 1 之间, 否则会因为日志参数过多而无法保证该最长公共子序列即为日志模板, 因而出现提取失败的问题。

3.3 本章小结

本章主要介绍了基于 LCS 的日志解析算法对日志进行解析。一般来说, 每条日志都是由程序生成的, 因此其具有一定的模板, 但是在没有源代码的情况下, 仅凭单条日志机器很难成功分离其日志模板和日志键, 这就需要开发人员根据日志手动地对日

志进行区分。

为了解决这个问题，本章采用的思想是基于最长公共子序列，通过寻找两个以上的日志序列的最长公共子序列来对日志的日志键和日志参数进行分离，从而实现日志的解析，将非结构化的日志转化为结构化的日志，实验结果表明，尽管存在一些问题，整个实验仍取得了不错的效果。

第 4 章 基于 LSTM 的日志异常检测算法研究

LSTM 是一种递归神经网络（Recurrent Neural Network，简称 RNN）^[15] 模型，并且保留了 RNN 的大部分特性，同时也解决了 RNN 参数回传过程中的梯度消失问题。LSTM 十分适合处理自然语言等与时间序列^[16]紧密相连的问题。系统日志可以看成更加规范化的自然语言序列，因此可以使用 LSTM 对其进行训练。本章将详细讲述基于 LSTM 的日志异常检测的原理，并进行相关实验测试其性能。

4.1 实验原理

4.1.1 LSTM 简介

LSTM 是一种特殊的 RNN。LSTM 与标准 RNN 的区别在于其内部结构不同。在标准 RNN 中，RNN 模块由多个简单模块组成。LSTM 同样由多个重复块共同组成，相比于只有 \tanh 层^[17]的标准 RNN，它的特点是单个模块的内部结构较为复杂。图 4.1 和图 4.2 分别展示了标准 RNN 和 LSTM 单个模块的结构示意图，其中 RNN 的输入 H_{t-1} 为上一层的输出和本层的输入，与之相比，LSTM 的内部结构更为复杂，且输入和输出多了一个状态 C_t ，该状态的引入可以使得相邻的 LSTM 的信息不加改变的流动，从而解决了 RNN 的长期依赖的问题。

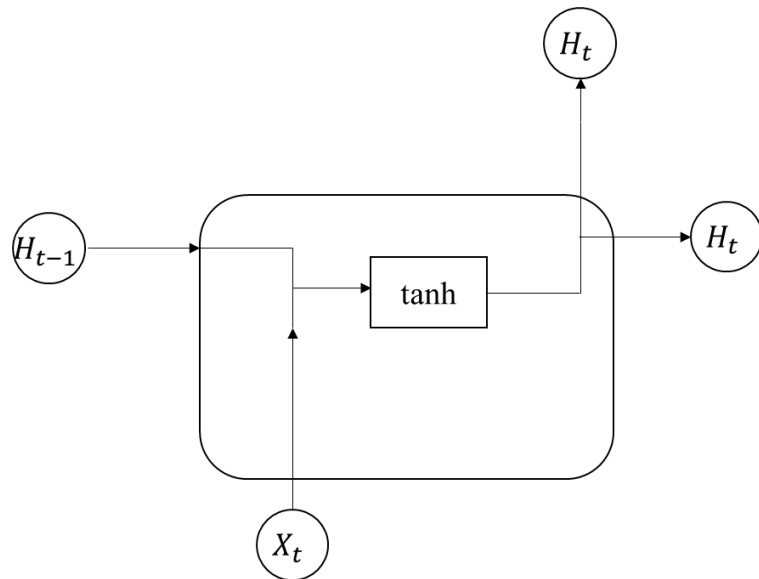


图 4.1 标准 RNN 示意图

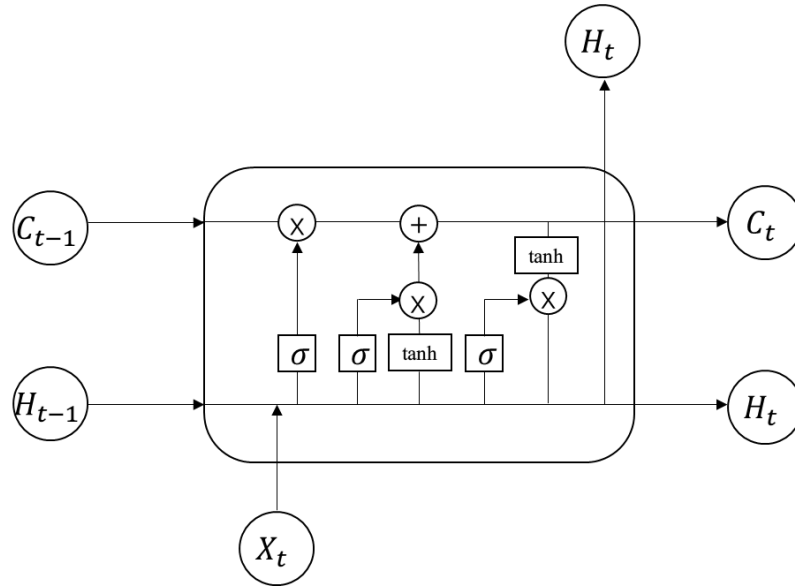


图 4.2 LSTM 示意图

LSTM 在内部结构中设计了遗忘门、输入门、状态更新单元和输出门，这些单元函数通过前一个 LSTM 块的状态和输出以及当前 LSTM 块的输入来确定状态每个门函数都单独设置了参数权重进行更新，以解决 RNN 中参数共享导致的梯度消失问题。下面分别对这四个部分进行简要介绍。

（1）遗忘门

遗忘门的部分如图 4.3 所示，其作用是决定从该模型中丢弃什么信息，将上一步的模型中的状态选择性遗忘。一般通过 sigmoid 层来实现遗忘门的作用，该层的输入为上一层 LSTM 的输出和该层 LSTM 的输入，输出为一个 0 至 1 之间的数字，记为 f_t ，其含义为保留信息的多少。

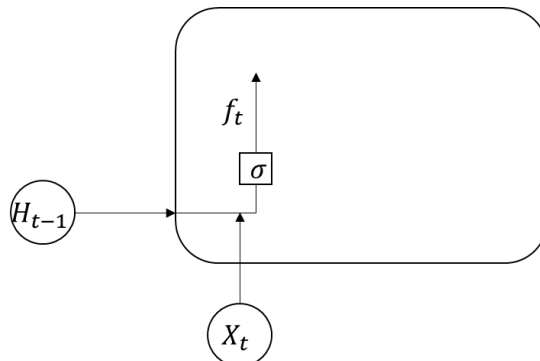


图 4.3 遗忘门结构示意图

遗忘门的数学表达式为：

$$f_t = \sigma(w_f \cdot [H_{t-1}, X_t] + b_f) \quad (4.1)$$

（2）输入门

输入门的结构如图 4.4 所示，其作用是决定该模型的存储内容，将新的输入信息选择性地记录到模型中。输入门包含两部分结构：sigmoid 层和 tanh 层，其中 sigmoid 层决定更新值的类型，tanh 层则会创建一个向量 \tilde{C}_t ，其被添加到状态更新单元中。

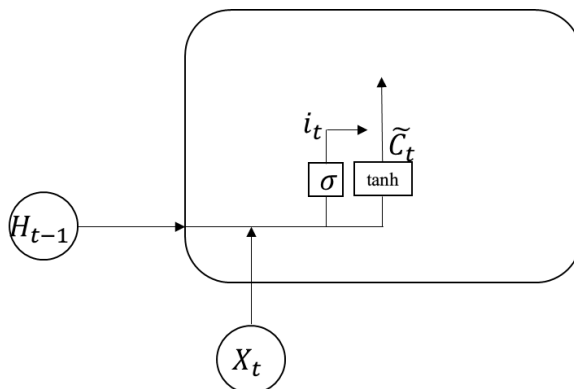


图 4.4 输入门结构示意图

输入门的数学表达式为：

$$i_t = \sigma(w_i \cdot [H_{t-1}, X_t] + b_i) \quad (4.2)$$

$$\tilde{C}_t = \sigma(w_c \cdot [H_{t-1}, X_t] + b_c) \quad (4.3)$$

（3）状态更新单元

状态更新单元的结构如图 4.5 所示，作用为实现 LSTM 的状态更新，由前面的遗忘门和输入门处理后的输出共同决定， f_t 决定了上一层 C_t 的遗忘程度， i_t 决定了候选值 \tilde{C}_t 的加入程度，并且通过状态更新单元可以实现删除部分旧的信息、添加新的信息。

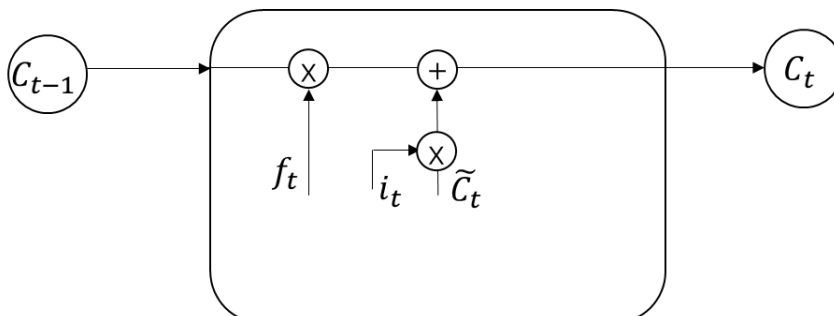


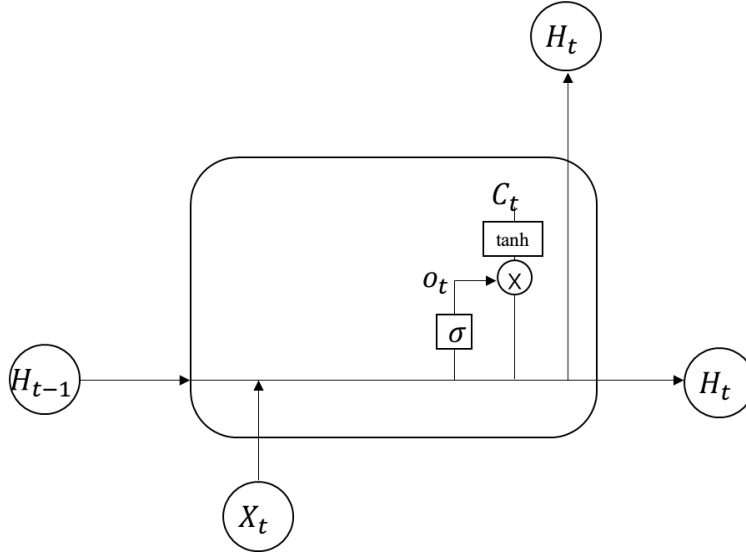
图 4.5 状态更新单元结构示意图

状态更新单元的数学表达式为：

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (4.4)$$

（4）输出门

输出门的结构如图 4.6 所示，其决定做出怎样的预测。输出门有 sigmoid 层决定本模型输出 C_t 的何种信息，状态更新单元通过 tanh 层与 sigmoid 层的输出相乘，可以确定最终的输出 H_{t-1} 。



4.6 输出门结构示意图

输出门的数学表达式为：

$$o_t = \sigma(w_o \cdot [H_{t-1}, X_t] + b_o) \quad (4.5)$$

$$H_t = o_t * \tanh(C_t) \quad (4.6)$$

通过上述的遗忘门、输入门、状态更新单元和输出门可以决定 LSTM 模型要保留的来自上一模型的信息，为处理自然语言相关的问题提供了方便，而日志可以看作更为规范化的自然语言，因此也可以尝试使用 LSTM 对日志异常进行检测。

4.1.2 基于 LSTM 的异常检测

系统的日志一般是相互联系的，例如图 4.7 的几条日志：

```

1 Jun 26 04:10:02 combo su(pam_unix)[1546] session opened for user news by (uid=0)
2 Jun 26 04:10:04 combo su(pam_unix)[1546] session closed for user news
3 Jun 27 04:02:47 combo su(pam_unix)[7031] session opened for user cyrus by (uid=0)
4 Jun 27 04:02:48 combo su(pam_unix)[7031] session closed for user cyrus

```

图 4.7 日志示例

从其中可以发现，某几条连续的日志具有很高的关联度，若这种关联因为某条日志的出现而被破坏，则可以判定该日志出现了异常^[18]。

因此日志的异常检测被转化为多分类的问题，在训练过程中，输入为要进行正常的日志键，通过训练到的分类器输出由训练模型对输入的日志键进行预测后的概率分布图，如图 4.8 所示。若输出结果与实际的日志相差过大，则可以判定输入的日志为异常日志。

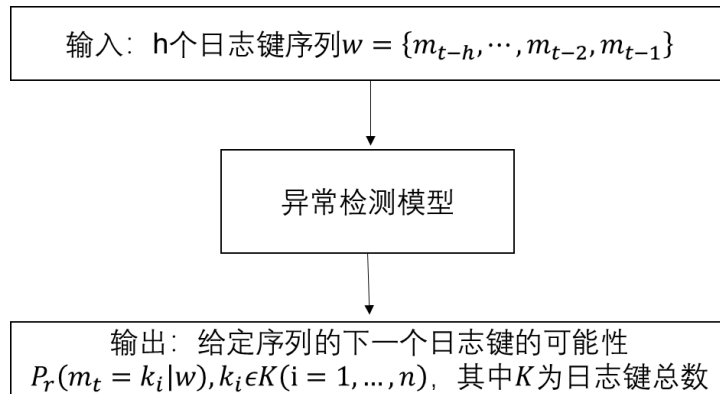


图 4.8 异常检测模型流程

对于异常检测模块，首先构造好一个 LSTM 模型，将正常的日志键序列作为输入，结果会产生一个异常检测模型。系统日志经过数据处理后的日志键作为检测模型的输入，利用生成的异常检测模型预测之后的日志键的可能情况，若接下来的实际的日志在预测结果中占有较大的可能性，达到了设定的阈值，则可以判定该日志属于正常的日志，反之为异常日志。

对于多个 LSTM 模块中的一个模块来说，其前一个模块的输出会作为本模块的输入之一，这样多个模块之间就产生了关联，可以进行语言序列方面的学习。多个 LSTM 模块按照一定的顺序进行连接，构成了一个完整的 LSTM 模型。本 LSTM 模

块的输入由来自前一个 LSTM 模块的状态和外部输入共同组成，输出为新的状态情况，这使得日志序列的信息得到保留，传递到下一个模块。基于 LSTM 的日志异常检测框架如图 4.9 所示，该检测框架最后使用一个全连接层进行分类输出。

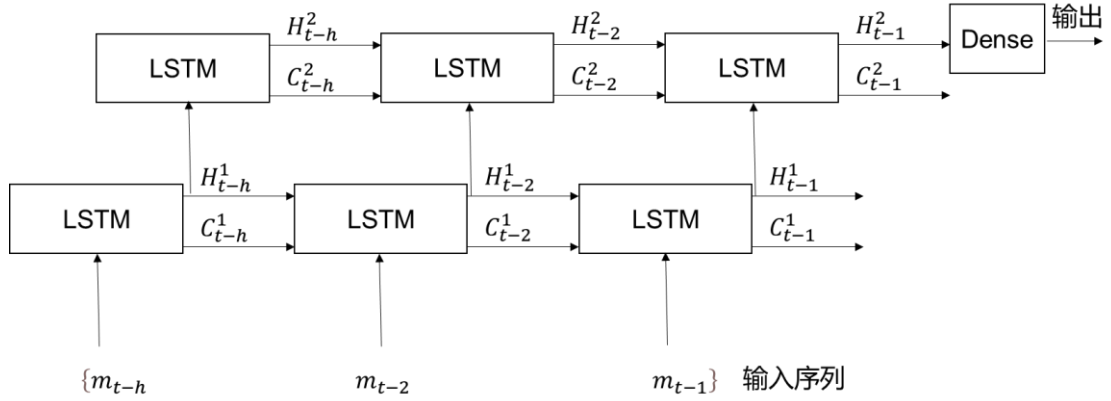


图 4.9 基于 LSTM 的日志异常检测框架

4.2 实验过程

4.2.1 实验环境

操作系统为 Windows10，编程环境为 Python3.7.5，基于 keras2、tensorflow2(cpu)。

4.2.2 实验数据

数据仍然使用 HDFS 数据集，不同的是此次数据集已经被相关人员完成日志解析并进行了特征编码（如图 4.10），并且被相关领域专家标记为正常或者异常。经过程序统计，本次实验的训练集共有 4855 行，即有 4855 个任务流，共有 46575 个日志键序列。

```

22 5 5 5 26 26 26 11 9 11 9 11 9 2 3 23 23 23 21 21 21
22 5 5 5 11 9 11 9 11 9 26 26 26
22 5 5 5 26 26 26 11 9 11 9 11 9 4 3 3 3 4 3 3 4 3 3 23 23 23 21 21 21
22 5 5 5 26 26 26 11 9 11 9 11 9 3 3 4 3 4 3 3 3 4 4 3 3 23 23 23 21 21 21
5 22 5 5 26 26 11 9 11 9 11 9 26 23 23 23 21 21 21
22 5 5 5 26 26 26 11 9 11 9 11 9 4 4 3 2 23 23 23 21 21 21
5 22 5 5 11 9 11 9 11 9 26 26 26 23 23 23 21 21 21
5 5 5 22 11 9 11 9 11 9 26 26 26 23 23 23 21 21 21
5 22 5 5 11 9 11 9 11 9 26 26 26 3 3 4 3 3 4 23 23 23 21 21 21
5 5 5 22 11 9 11 9 11 9 26 26 26 23 23 23 21 21 21
5 5 22 5 11 9 26 26 11 9 11 9 26 23 23 23 21 21 21
22 5 5 5 26 26 26 11 9 11 9 11 9 4 3 4 23 23 23 21 21 21
5 22 5 5 11 9 11 9 11 9 26 26 26 23 23 23 21 21 21
5 22 5 5 11 9 11 9 11 9 26 26 26 23 23 23 21 21 21
5 5 22 5 11 9 11 9 11 9 26 26 26 23 23 23 21 21 21
5 5 5 22 11 9 11 9 11 9 26 26 26 23 23 23 21 21 21
5 5 5 22 11 9 11 9 11 9 26 26 26 23 23 23 21 21 21
5 5 5 22 11 9 11 9 26 26 11 9 26 23 23 23 21 21 21
5 5 5 22 11 9 11 9 11 9 26 26 26 3 3 4 3 3 4 23 23 23 21 21 21
5 22 5 5 11 9 11 9 11 9 26 26 26 23 23 23 21 21 21
5 5 5 22 11 9 11 9 11 9 26 26 26 23 23 23 21 21 21
5 5 5 22 11 9 11 9 11 9 26 26 26 23 23 23 21 21 21
5 5 22 5 11 9 26 26 11 9 11 9 26
5 5 5 22 11 9 11 9 11 9 26 26 26 23 23 23 21 21 21
22 5 5 5 11 9 11 9 11 9 26 26 26 23 23 23 21 21 21
5 22 5 5 11 9 11 9 11 9 26 26 26 3 4 3 3 4 23 23 23 21 21 21

```

图 4.10 HDFS 数据集样例

4.2.3 实验指标

本次实验的模型采用三个指标进行评估，分别是精确率(Precision, P)，召回率(Recall, R)，以及 F1 值(F1-measure, F1)。各评价指标的意义和计算公式如下，其中 TP 代表了正确例子中预测为正确的个数，FP 为错误例子中预测为正确的个数，TN 为错误例子中预测为错误的个数，FN 为正确例子中预测为错误的个数。

精确率：是指预测为正的样本中有多少是正确的。

$$P = TP / (TP + FP) \quad (4.7)$$

召回率：是指样本中有多少正样本被预测正确了。

$$R = TP / (TP + FN) \quad (4.8)$$

F1 是用来综合精确率和召回率的。

$$F1 = 2 \times (P \times R) / (P + R) \quad (4.9)$$

4.2.4 实验参数

本实验使用了多个 LSTM 层,日志键在输入时会被编码为 one-hot 向量，输出层为全连接层，利用 softmax 函数将输出转换为一个概率分布函数。最终判断预测值与实际值是否相差过大，若超过阈值则判定异常。

为了得到良好的输出结果，需要对训练阶段的模型调整适当的参数权重^[19]。训练

过程中,权重更新的方法采用梯度下降法。模型的损失函数为 `categorical_crossentropy`,优化器采用 `Adam`,评价指标为 `accuracy`,训练次数 `epoch` 为 500, `batch_size` 设置为 2048。检测阶段使用训练好的 LSTM 模型对输入序列进行预测。

实验中需要调整的参数为预测正常输出范围 `g`,日志序列长度 `h`,LSTM 模型的模型层数 `num_layer` 和各个 LSTM 层的维度 `hidden_size`,通过调整该部分参数对实验性能进行评价,实验的默认参数值为 `h=10,g=10,num_layers=2` 和 `hidden_size=64`。

4.2.5 实验结果

通过图 4.11 可以看到,随着训练的不断进行,模型的损失在不断降低,正确率不断提高,取得不错的效果。

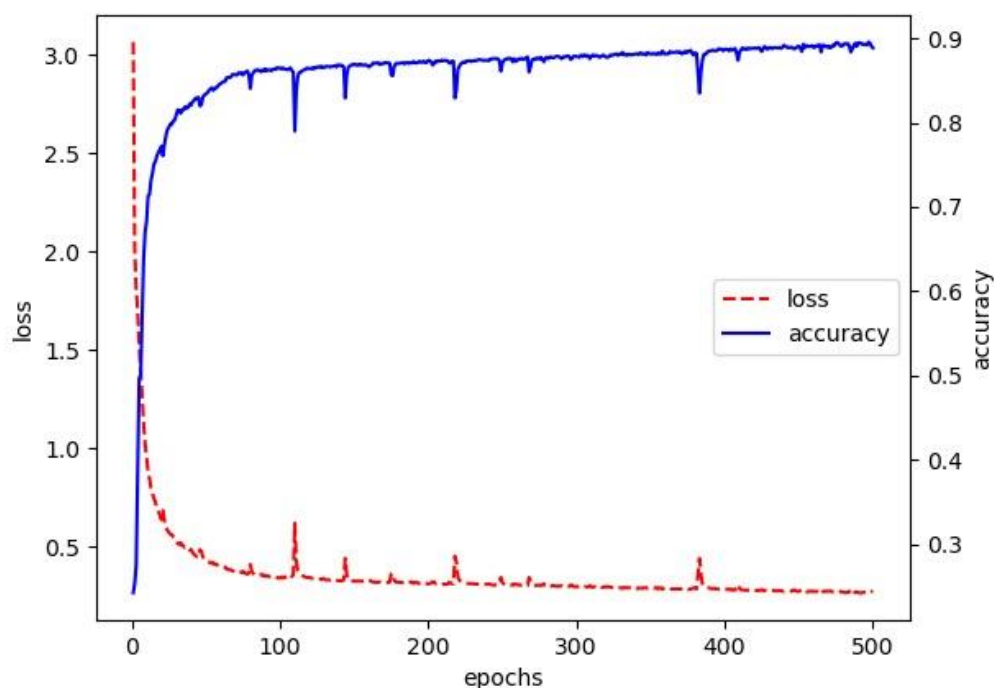


图 4.11 训练过程中 loss 与正确率变化情况

对于其中一次训练的网络结构如图 4.12 所示,该网络结构共有两层 LSTM 和一层用于分类的全连接层,全连接层的激活函数为 `softmax`。

Model: "sequential_1"

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 10, 64)	16896
lstm_2 (LSTM)	(None, 64)	33024
dense_1 (Dense)	(None, 28)	1820
Total params: 51,740		
Trainable params: 51,740		
Non-trainable params: 0		

图 4.12 实验网络结构示意图

通过控制变量法进行参数调整，结果图 4.13、4.14、4.15 和 4.16 所示。

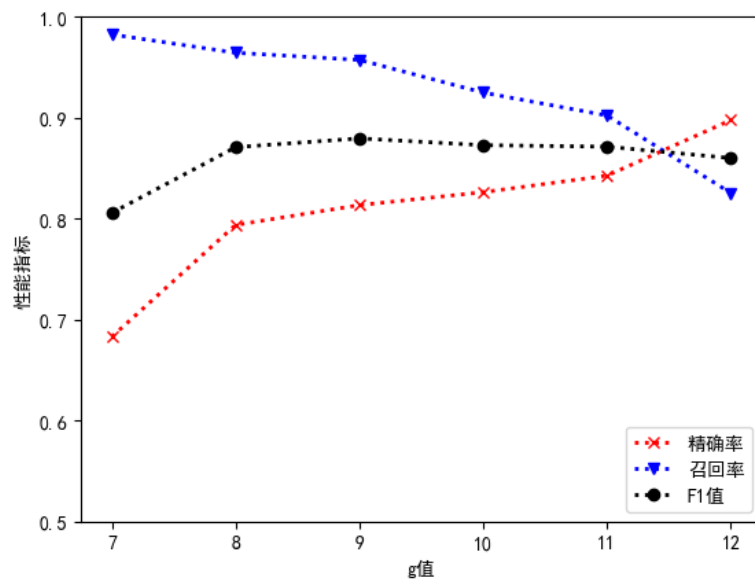


图 4.13 性能指标随 g 值变化情况

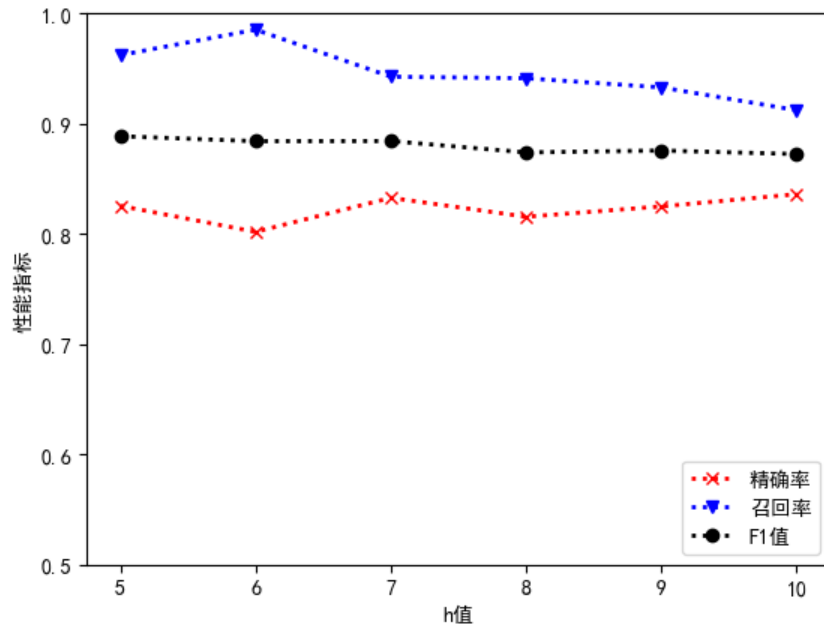


图 4.14 性能指标随 h 值变化情况

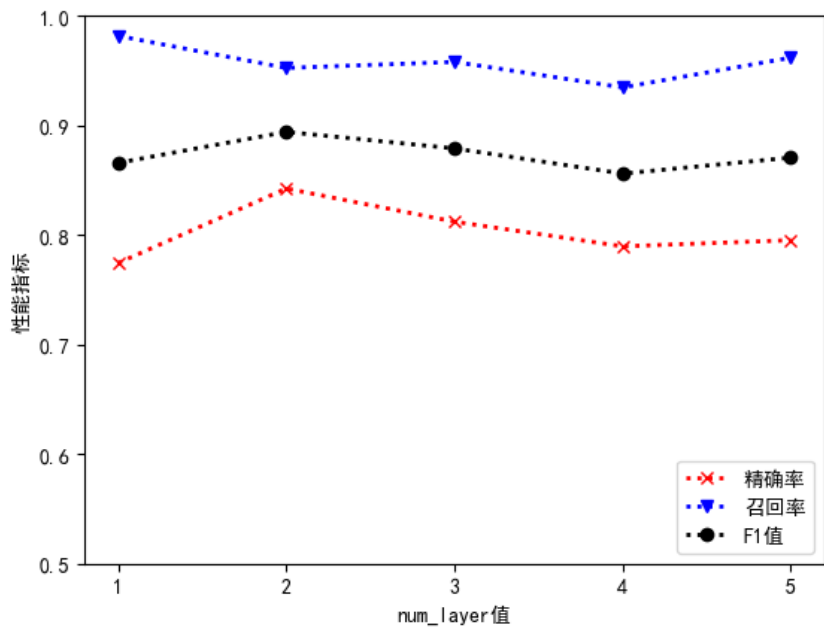


图 4.15 性能指标随 num_layers 变化情况

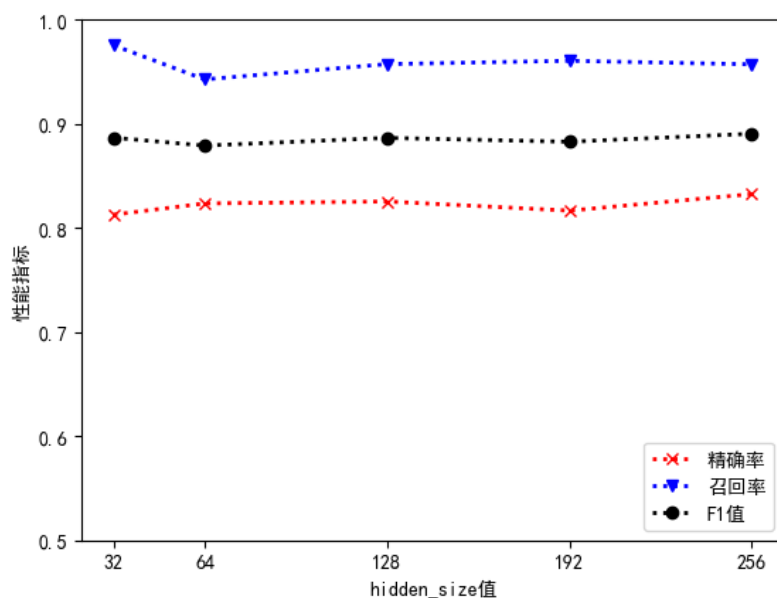


图 4.16 性能指标随 hidden_size 变化情况

4.2.6 实验分析

从结果来看，不同的参数设置对于算法性能有着不同的影响。

在 $h=10, num_layers=2$ 和 $hidden_size=64$ 的条件下随着 g 值的增大，精确率有着较大的提升，而召回率变化趋势相反，呈现下降趋势，F 值没有明显变化。在 $g=10, num_layers=2$ 和 $hidden_size=64$ 的条件下随着 h 值的增大，精确率和 F 值均无明显变化，召回率略有下降。在 $g=10, h=10$ 和 $hidden_size=64$ 的条件下， $num_layers=2$ 时，准确率达到最高值，随着 num_layers 的增大，各项性能都有所下降，但下降幅度不大。在 $g=10, h=10$ 和 $num_layers=2$ 的条件下随着 $hidden_size$ 值的增大，各项指标有一定的上升趋势，但是上升趋势不明显。

由上述结果得出，不同的参数设置会对模型结果产生不同的影响，但是由于随着 h 值、 num_layers 或者 $hidden_size$ ，模型复杂度不断提高，训练时间和系统负载都会变大，综合选择下， $h=10, g=10, num_layers=2$ 和 $hidden_size=64$ 即可以取得不错的性能。

4.3 本章小结

异常检测是日志异常检测流程中的最后一步，许多相关的算法被研究人员提出，并进行了不同测试。本章尝试将日志视为更为规范的自然语言序列，对 LSTM 的原理进行了简要介绍，并基于 LSTM 对 HDFS 数据集进行了实验，对比了不同的参数设置对模型性能的影响，总体取得了不错的效果。

第 5 章 总结与展望

5.1 本文总结

由于现代系统变得愈发复杂，针对系统的运行情况而产生的系统日志也随之变得越来越复杂，通过传统人工和简单的正则表达式等方法对日志异常进行检测已经变得不可取。近年来，机器学习等领域蓬勃兴起，许多相关日志异常检测算法被相关学者研究出来，一定程度上解决了日志异常检测的自动化问题^[20]。

本文主要对日志异常检测算法的步骤进行了详细解读，分别介绍了日志解析和异常检测部分的工作原理和常见算法，并且分别对日志解析和异常检测部分进行了相关实验，对比不同参数下对异常检测效率的影响。

本文主要贡献如下：

（1）对日志异常检测框架及其流程进行了详细介绍，其中的关键部分是日志解析与异常检测。业界已经有许多日志解析工具可供选择。异常检测算法目前仍缺少通用性方法，大多数已经提出的异常检测算法具有较高的特异性，仅适用于特定场景^[21]。

（2）针对基于 LCS 的日志解析算法进行讲解，详细阐述日志解析的原理，利用 Linux 系统日志进行相关的日志解析实验，将日志解析为日志键和日志参数，对日志键进行编码，并确定日志参数的位置，得到了较好的结果。

（3）基于 LSTM 对日志进行异常检测实验，介绍基于 LSTM 的检测算法流程，利用 HDFS 数据集对日志进行异常检测，对模型进行不同的参数设置，并对比不同的参数设置下的算法效率。

5.2 后续工作展望

然而，本文讨论的方法也存在以下不足：

（1）基于 LCS 的日志解析算法还存在一定的问题尚未解决，主要体现在当数据量较少时，可能会出现一个单独的日志记录，此时因为无法找到与其他的最长公共子序列而导致无法对其进行解析。上述情况需要思考如何进行解决。

（2）由于机器性能的限制，每次实验参数的改变均需要重新进行实验，耗费时间过长，因此在参数调教过程中仅针对部分参数进行调整。并且本实验只选定一个数据集，没有再选择其他数据集，可能存在一定的偶然性。

由于当前仍缺少通用的日志异常检测算法，当前研究人员提出的检测算法仅针对特定场景或者离线状态进行检测，具有较大的局限性。本文将日志视为更为规范化的自然语言文本进行处理，一定程度上取得了不错的效果，未来可以尝试使用其他的RNN网络^[22]，以测试它们在异常检测中的效率。

参考文献

- [1] 郑翠玲.最长公共子序列算法的分析与实现[J].武夷学院学报,2010,29(02):44-48.
- [2] Wei Shao,Zhi Wang,Xiaolu Wang,Kefan Qiu,Chunfu Jia,Chong Jiang. LSC: Online auto-update smart contracts for fortifying blockchain-based log systems[J]. Information Sciences,2020,512.
- [3] 杨丽,吴雨茜,王俊丽,刘义理.循环神经网络研究综述[J].计算机应用,2018,38(S2):1-6+26.
- [4] Min Du,Feifei Li,Guineng Zheng,Vivek Srikumar. DeepLog[P]. Computer and Communications Security,2017.
- [5] 张颖君,刘尚奇,杨牧,张海霞,黄克振.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(06):1-12.
- [6] 张军,王芬芬.基于正则表达式的日志解析系统构建研究[J].无线互联科技,2020,17(03):48-49.
- [7] 刘凯. 基于日志特征的异常检测系统的设计与实现[D].西安电子科技大学,2014.
- [8] 王易东,刘培顺,王彬.基于深度学习的系统日志异常检测研究[J].网络与信息安全学报,2019,5(05):105-118.
- [9] 冯士龙,台宪青,马治杰.改进的基于日志聚类的异常检测方法[J].计算机工程与设计,2020,41(04):1087-1092.
- [10] 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[J].计算机应用,2020,40(10):2960-2966.
- [11] 杨瑞朋,屈丹,朱少卫,钱叶魁,唐永旺.基于改进时间卷积网络的日志序列异常检测[J].计算机工程,2020,46(08):50-57.
- [12] 梅御东,陈旭,孙毓忠,牛逸翔,肖立,王海荣,冯百明.一种基于日志信息和 CNN-text 的软件系统异常检测方法[J].计算机学报,2020,43(02):366-380.
- [13] Jin Wang,Yangning Tang,Shiming He,Changqing Zhao,Pradip Kumar Sharma,Osama Alfarraj,Amr Tolba. LogEvent2vec: LogEvent-to-Vector Based Anomaly Detection for Large-Scale Logs in Internet of Things[J]. Sensors,2020,20(9).

- [14] Wei Shao,Zhi Wang,Xiaolu Wang,Kefan Qiu,Chunfu Jia,Chong Jiang. LSC: Online auto-update smart contracts for fortifying blockchain-based log systems[J]. Information Sciences,2020,512.
- [15] 张孝慈. 递归神经网络模的若干关键问题研究[D].中国科学技术大学,2019.
- [16] 杨海民,潘志松,白玮.时间序列预测方法综述[J].计算机科学,2019,46(01):21-28.
- [17] 张尧.激活函数导向的 RNN 算法优化[D].浙江大学,2017.
- [18] 张晓箐. 基于海量日志消息的软件系统异常检测技术研究 with 实现[D].西安电子科技大学,2015.
- [19] 王盛玉,曾碧卿,胡翩翩.基于卷积神经网络参数优化的中文情感分析[J].计算机工程,2017,43(08):200-207+214.
- [20] 沈潇军,葛亚男,沈志豪,倪阳旦,吕明琪,翁正秋.一种基于 LSTM 自动编码器的工业系统异常检测方法[J].电信科学,2020,36(07):136-145.
- [21] 冯秋燕. 基于 Web 应用的日志异常检测与用户行为分析研究[D].华南理工大学,2019.
- [22] Andy Brown,Aaron Tuor,Brian Hutchinson,Nicole Nichols. Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection[P]. Machine Learning for Computing Systems,2018.

致谢

回首四年的本科学习生活，首先要感谢的是我的指导老师贾俊铖老师。“师者，所以传道授业解惑也。”贾老师为人幽默，善于指导学生，在本科阶段多次受到他的指点，他用身体力行向我展示了一位言传身教的学者形象。同时在毕业设计阶段，不论是课题的选择，还是文献阅读、论文撰写，贾老师都及时给予了悉心的指导，十分感谢几年来贾老师对我学生道路上的帮助和指导。

其次，我要感谢培养我的母校苏州大学，母校有着浓厚的学术氛围，塑造了良好的学习环境，计算机科学与技术学院有着雄厚的技术底蕴，给了我一个能够快速成长的平台，使我能够不断地蜕变成长。

同时，我要感谢我的父母，父母用他们勤劳的双手抚平了我在他乡求学的担忧，有他们的陪伴和支持，我才能没有后顾之忧地完成四年的学业。

最后，我要感谢我在母校结交的各位朋友和室友，“三人行，必有我师焉。”在求学道路上以及完成论文的过程中，正是有着朋友的陪伴和指点，我才能顺利完成此论文。

苏州大学本科生毕业设计（论文）任务书

学院（部）：计算机科学与技术学院

设计（论文）题目：日志异常检测算法研究与实现					
指导教师姓名	贾俊铖	职 称	副教授	类 别	毕业设计
学 生 姓 名	陈健	学 号	172705145	设计（论文）类型	应用型
年 级	2017 级	专 业	计算机科学与技术（人工智能）	是否隶属科研项目	否
<p>1、设计（论文）的主要任务及目标</p> <p>目标：</p> <p>本课题对日志异常检测算法进行研究，基于开源机器学习平台实现 DeepLog 日志异常检测算法，并与其他常见模型进行性能比较。</p> <p>主要任务：</p> <ul style="list-style-type: none">（1） 搭建机器学习框架。（2） 掌握日志解析的过程。（3） 掌握 DeepLog 工作流程。（4） 利用 python 程序实现 DeepLog 算法并进行性能测试。					
<p>2、设计（论文）的主要内容</p> <ul style="list-style-type: none">（1） 阅读文献资料，调研日志异常检测的相关算法，掌握 DeepLog 算法流程。（2） 对海量数据进行处理，提取出需要的数据。（3） 实现 DeepLog 程序，对模型进行训练。（4） 使用测试集对训练完的模型进行测试，并判断其性能。					
<p>3、设计（论文）的基本要求</p> <ul style="list-style-type: none">（1） 熟悉模型以及数据准备与处理。（2） 运行模型得到检测结果，并进行统计与分析。（3） 完成 1000 字左右的文献综述、不少于 3000 个单词的外文翻译。（4） 按时提交毕业论文和源代码。（5） 完成毕业答辩。					

4、主要参考文献

- [1] 张颖君. 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(06).
- [2] 张林栋. 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.
- [3] 任怡彤. 系统日志的异常检测及分析研究[D].中国民航大学, 2020.
- [4] 王易东. 基于深度学习的系统日志异常检测研究[J].网络与信息安全学报, 2019,5(05).
- [5] Min Du. DeepLog[P]. Computer and Communications Security, 2017.

5、进度安排

	设计（论文）各阶段任务	起 止 日 期
1	熟悉日志检测算法相关模型	2021-01-16 至 2021-02-27
2	数据准备和处理	2021-02-28 至 2021-03-16
3	实现 DeepLog 日志检测算法	2021-03-17 至 2021-04-12
4	测试 DeepLog 算法与其他日志检测算法性能差异	2021-04-13 至 2021-04-20
5	完善改进算法，提高算法性能并撰写毕业论文	2021-04-21 至 2021-05-10
6	修改论文，准备毕业答辩	2021-05-11 至 2021-05-19



A LSTM-Based Anomaly Detection Model for Log Analysis

Zhijun Zhao¹ · Chen Xu¹ · Bo Li²

Received: 22 October 2020 / Revised: 2 January 2021 / Accepted: 26 January 2021
© The Author(s) 2021

Abstract

Security devices produce huge number of logs which are far beyond the processing speed of human beings. This paper introduces an unsupervised approach to detecting anomalous behavior in large scale security logs. We propose a novel feature extracting mechanism and could precisely characterize the features of malicious behaviors. We design a LSTM-based anomaly detection approach and could successfully identify attacks on two widely-used datasets. Our approach outperforms three popular anomaly detection algorithms, one-class SVM, GMM and Principal Components Analysis, in terms of accuracy and efficiency.

Keywords Anomaly detection · Log analysis

1 Introduction

The running state of the system is usually recorded in a log file, used for debugging and fault detection, therefore the log data is a valuable resource for anomaly detection. Log data is natural time series data, contents and types of events recorded by the log file also tend to be stable. Except for some highly covert apt attacks, most of the attacks are not instantaneous and have a fixed pattern, the log data will produce a pattern when recording malicious behaviors, which provides the possibility to detect anomaly from the log sequence.

The traditional methods rely on the administrator to manually analyze the log text. This kind of processes lead to a large number of human power costs, and requires the system administrator to understand the network environment and be proficient in system architecture.

However, in order to avoid tracking by the security administrator, the logs generated by attacks are getting similar to the logs generated by the normal access behaviors. In addition, because of the large variety of applications and services, each web node will generate a large number of logs, which results in the log data file becoming extremely large. It may not be possible to directly process these logs manually. These logs

may contain signals of malicious behaviors, so it is necessary to use some anomaly detection methods for analysis.

The existing automatic methods of anomaly detection based on log data can be divided into two categories: supervised learning methods relying on tags, such as decision tree, LR, SVM and unsupervised learning methods based on PCA, clustering and invariant mining. Supervised learning has a very good effect in detecting known malicious behavior or abnormal state, but it cannot detect unknown attacks, as it depends on prior knowledge. Unsupervised method can be used to detect unknown exceptions, but most of the methods need to improve the accuracy.

A exists research introduced use concept of the longest common subsequence to reduce the number of matching patterns obtained during the calculation [1], enabling simple classification of logs. Besides, Xu et al. uses association rules, which are generated by trust scores, to mine frequent item sets, and then detects attack behavior based on the association rules [2]. Zhao et al. used a method based on character matching to study the classification of system log, and determined the correspondence between log type and character [3]. Seker. S.E et al. took the occurrence frequency of letters as key elements to recognize log sequence, abstracting different types of logs into different characters, and selects adaptive k-value according to their characteristics [4]. The existing research shows that there is a strong correlation between logs and their character composition.

This model is based on LSTM sequence mining, through data-driven anomaly detection method, it can learn the sequence pattern of normal log, and detect unknown malicious behaviors, identify red team attacks in a large number of log sequences. The model performs character-level analysis of the

✉ Bo Li
13121239987@163.com

¹ Jiaxing Hengchuang Electric Group Co.,Ltd, Information Technology Branch, Zhejiang, China

² School of Computer Science and Engineering, Beihang University, Beijing, China

log text directly, so there is no need to perform excessive log processing, such as log classification, matching, etc., which greatly simplifies the calculation complexity.

However, there are still some problems in the current model. For instance, it only performs character level analysis, which may ignore some high-level features. A natural idea is to analyze the logs hierarchically, but research finds that this measure has not improved its performance [5], thus the relevant methods need further study. In addition, since no log correlation matching is performed, only abnormal log lines can be alerted, and the anomaly level of each user cannot be detected directly. A possible solution is to trace users through log entries that are determined to be abnormal, but obviously it is slightly verbose.

Contribution Through joint efforts, after discussing the experimental together, Wenhao ZHOU completed the modification of LSTM code and training of the model, Jiuyao ZHANG completed the comparative experiments, and Ziqi YUAN completed the writing of the article.

The first section of this paper introduces the background, the goal of the model and the existing problems. The second section introduces the methods used in the model, including data processing methods, the specific structure of the LSTM network, and the adjustment of parameters. The third section compares this model with the methods in [5, 6] and illustrates the advantages. The fourth section introduces the experiment, including a detailed description of the data set, test indicators, and comparison results compared with other methods such as one-class SVM, GMM and PCA.

2 Anomaly Detection Approach

Our approach learns character-level behaviors for normal logs, processing a stream of log-lines as follows:

- (1) Initialize weights randomly
- (2) Train the model with log data of the first n days
- (3) For each day k ($k > n$) in chronological order, firstly based on model M_{k-1} , which is trained by with log data of the first $k-1$ days, produce anomaly scores for all events in day k . Secondly, record per-user-event anomaly scores in rank order to analysts for inspection. Thirdly, update model M_{k-1} to M_k , as logs in day k are used.

2.1 Log-Line Tokenization

The network log cloud be obtained from many sources. The logs obtained from different sources are naturally generated in different formats and record different information. In order to

expand the application scope of the model as much as possible, we consider each line of the logs as a string directly, and take the character-level data as the input of LSTM directly.

Since only printable characters are considered, whose range is from 0×20 to $0 \times 7e$, there are 95 characters in total. Convert each word into its corresponding ASCII and subtract 30, so we get 0 for the beginning and 1 for the end. Then fill the space with -1 to ensure the same length of each sequence for training. An example is shown in Fig. 1.

2.2 LSTM Model

In order to calculate the anomaly score of each record, we consider a calculation model for a single log line. RNN is a common tool to deal with that kind of data. Specifically, we use an LSTM network, whose input is the characters of each log line, so as to predict the probability distribution of the next character and infer the abnormal possibility of this log line, or the user account.

Bidirectional Event Mode (BEM) For a log line with a character length of K , let $x(t)$ be the character in position t , and $h(t)$ be the hidden representation of the corresponding position of the character. According to the relevant theory of LSTM [7], we have the following relations

$$h(t) = o(t) \cdot \tanh(c(t)) \quad (2.1)$$

$$c(t) = f(t) \cdot c(t-1) + i(t) \cdot g(t) \quad (2.2)$$

$$g(t) = \tanh(x(t)W(g, x) + h(t-1)W(g, h)) \quad (2.3)$$

$$f(t) = \sigma(x(t)W(f, x) + h(t-1)W(f, h) + b(f)) \quad (2.4)$$

$$i(t) = \sigma(x(t)W(i, x) + h(t-1)W(i, h) + b(i)) \quad (2.5)$$

$$o(t) = \sigma(x(t)W(o, x) + h(t-1)W(o, h) + b(o)) \quad (2.6)$$

Among them, initial state $h(0)$ and initial cell state $c(0)$ are preset as zero vectors, bullet symbol denotes element-wise multiplication, and sigma represents logistic function with standard parameters, that is

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.7)$$

0	55	19	18	19	34	38	49	47	19	14	37	19	26	24	20	6
34	38	49	47	19	14	37	19	26	24	20	14	37	19	26	24	
20	14	33	14	33	14	35	87	86	74	47	67	82	14	53	87	
69	69	71	85	85	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	

Figure 1 Example of processed features of LSTM model.

Vector \mathbf{g} represents the hidden value inferred from the current input and the previous hidden state. Vectors \mathbf{f} , \mathbf{i} , \mathbf{o} are the standard forgetting gate, input gate and output gate in the LSTM model. Matrix \mathbf{W} and deviation vector \mathbf{b} are the parameters used in the model.

In addition, since we can get all the contents of a log line directly, and the characters in a certain position in a log line are obviously related to their context content, we build a bidirectional LSTM, or BiLSTM, which infers the character probability distribution in each position from the back to the front meanwhile. For this reason, we add a new set of hidden vectors $\mathbf{h}^b(K+1), \mathbf{h}^b(K), \dots, \mathbf{h}^b(1)$, so the model could run the LSTM equations in reverse at the same time. The superscript b of reverse LSTM parameters shows its parameter matrix \mathbf{W} .

The hidden value \mathbf{h} is used to predict the characters in the new position, and the result is p , specifically, we have

$$\mathbf{p}(t) = \text{softmax}(\mathbf{h}(t-1)\mathbf{W}(p) + \mathbf{h}^b(t+1)\mathbf{W}^b(p) + \mathbf{b}(p)) \quad (2.8)$$

Compared with the ordinary one-way LSTM, it is obvious that the hidden value \mathbf{h} and matrix \mathbf{W} of the reverse LSTM are added to the prediction function, which enables us to predict the character in a certain position through the positive and negative directions at the same time.

Finally, the cross-entropy loss is defined as

$$\frac{1}{K} \sum_{t=1}^K H(x(t), \mathbf{p}(t)) \quad (2.9)$$

to update the weights. We train this model using stochastic mini-batch (non-truncated) back-propagation through time.

3 Related Work

The most relevant works are [5, 6].

Among them, [5] constructs a double-layer LSTM model for Knet2016 dataset. Its lower layer is composed of a LSTM network. On this basis, the hidden state of each token can be used as the feature of this line to get the type of log-line, and a LSTM model is constructed to complete the prediction of the next log-line type. However, the experimental results show that the double-layer structure does not improve the prediction performance, its metrics are lower than the simple single-layer model.

Due to the limitation of computing power, we have to use less data, so we did not build a double-layer LSTM model, but used a single-layer BiLSTM model, through which the characters that may appear can be predicted bidirectionally at the same time. On the one hand, this ensures the detection effect

to a certain extent, on the other hand, it reduces the demand for computing power directly.

[6] discusses the value and potential problems of many logs in detail, and gives a new, comprehensive, real network security data overview. This paper enumerates some kinds of logs and counts word frequency and other information, but does not give a detailed and feasible processing method. Each kind of log has different format and reserved words. Even if we can make a detailed analysis, it is difficult to find a common and rapid method to make exception detection for different kinds of logs.

Therefore, it is necessary for us to conduct character level analysis and prediction, which can directly avoid the great differences between different logs, and because the normal log-line and anomaly log-line are obviously different in character level, it is guaranteed that this kind of method is correct.

4 Experimental Results

4.1 Data

We used the Los Alamos National Laboratory (LANL) cyber security dataset (Kent 2016), which collected event logs of LANL's internal computer network for 58 consecutive days, with more than 1 billion log lines, including authentication, network traffic and other records. Fields involving privacy have been anonymous. Besides normal network activities, 30 days of red team attacks were recorded.

[5] gives some statistical descriptions, and the parts we use, which are the authentication event logs, are shown Table 1. We used 3,028,187 loglines in 20 days, the first 14 days for training, and the rest 6 days for testing, with 51 anomalous loglines and 26 anomalous user-days.

After tokenization, the length of all log lines is filled to 112.

We also used insider thread test dataset (R6.2), which is a collection of synthetic insider thread test datasets that provide both background and malicious actor synthetic data. It covers

Table 1 Dataset statistics of Knet2016.

Field	Example	#unique labels
time	1	5,011,198
source user	C625@DOM1	80,553
dest. User	U147@DOM1	98,563
source pc	C625	16,230
dest. pc	C625	15,895
auth.type	Negotiate	29
logon type	Batch	10
auth. Orient	LogOn	7
success	Success	2

device interaction, e-mail, file system and other aspects of the log content. We choose device access log and net access log to use in the experiments. The fields and statistics of net access log are summarized in Table 2a, while those of device access log are shown in Table 2b.

We used 1,676,485 loglines in 21 days, the first 14 days for training, and the rest 7 days for testing, with 530 anomalous loglines and 530 anomalous user-days.

In the process of tokenization, it is found that length of the loglines of net access dataset is up to 2500 characters. If all the lines are filled to such a long sequence, it will undoubtedly cost a very large amount of computation. Therefore, the URL column only extracts domain name, and the introduction column of the page extracts its key phrases through rake algorithm [8]. Finally, the length of the sequences is determined to be 830.

Timescale For this work, we consider the following timescales.

First of all, each line of the log records a relatively independent action, for example, a user makes an identity authentication, and the result is success or failure, so line is regarded as a set of independent input. We call this timescale logline level. As its name, logline level analysis will calculate the anomaly score of each line.

Secondly, in order to compare with the baseline experiments, we integrate all the actions recorded by the log of each user in each day, the anomaly score of each logline of a user in each day is aggregated to calculate the anomaly score for the user in that day. As we use the maximum anomaly score of a user, it is called user-day-max level. This level indicates the possibility of anomaly of a user in a specific day.

In addition, we use a normalization strategy. For the anomaly score of each line, it first subtracts the average exception score of the corresponding user on the same day, and then be

calculated normally, so as to realize the normalization of the original score. This normalized calculation method is known as diff, so it is called user-day-diff level, marking the difference between maximum anomaly score and average score of a user in a specific day.

4.2 Metric

We use AUC as the evaluation metric, which represents the area under the receiver operator characteristic curve. It characterizes the trade off in model detection performance between true positives and false positives.

The AUC value will not be greater than 1, the higher the value, the better the performance of the model.

4.3 Baselines

According to [5], for the data we used, we define a multidimensional aggregate feature vector for each user, as the basis of comparative experiments.

We consider three baseline models, which are one class SVM, GMM and PCA.

Data Processing In terms of data, we first filter out the abnormal logs in the data set, keep only the normal logs, and take 70% as the training set; then take all the remaining 30% in the original data set as the test set.

Specifically, the possibility of field value occurrence is divided into common and uncommon, the object is divided into single user and all users. Source PC name, target PC name, target user, process name and the PC name of the process are taken as the key fields of statistical information. The time is divided into all day, 0–6, 6–12, 12–18 and 18–24.

Make Cartesian product on the probability, object-oriented, key fields and time to get 100 features of

Table 2 Dataset statistics of R6.2: (a) Network access log fields and statistics and (b) device access log fields and statistics.

Field	Example	#unique labels
date	01/02/2010 6:21:31	726,611
user	ANC1950	3867
pc	PC-4921	3867
url	http://icio.us/John.asp	132,352
activity	WWW Visit	6
content	"Further consultation with post-production team"	54,946
Field	Example	#unique labels
date	01/02/2010 07:17:18	62,420
user	SDH2394	756
pc	PC-5849	1287
file_tree	R:\R\JKS2444	763
activity	Connect	2

statistical information, and then the fields that often appear in the log, such as login result, which is success or failure, are taken as the features to get 134-dimensional feature vector in the end.

The daily log of each user is summarized. If there are only a few different values in a certain field, such as login result, the frequency of these values will be counted. If there are a large number of different values in a certain field, such as PC name, user name, etc., these values will be divided into common or uncommon.

For a user, if the frequency of a value is less than 5%, it will be classified as uncommon; otherwise, it will be classified as common. For all users, if the frequency of a value is less than the average in its field, it will be classified as uncommon; otherwise, it will be classified as common. The information of each dimension of the feature vector is counted to get the feature vector of each user every day.

An example is shown in Fig. 2.

A. principal components analysis

PCA is used to learn the position representation of the extracted feature vector, project the original data from the original space to the principal component space, and then reconstruct the projection to the original space. If only the first principal component is used for projection and reconstruction, for most data, the error after reconstruction is small; but for outliers, the error after reconstruction is still relatively large.

B. one class SVM

In the detection of logs, there are only two categories: normal and abnormal, and the normal data is much more than the other. Therefore, one class SVM can be used to classify the extracted features and complete the exception detection.

C. Gaussian Mixture Model

Gaussian Mixture Model (GMM) is one of the most prevalent statistical approaches used to detect anomaly by using the Maximum Likelihood Estimates (MLE) method to perform the mean and variance estimates of Gaussian distribution

Field	Example	# unique labels
time	1	5011198
source user	C625@DOM1	80553
dest. user	U147@DOM1	98563
source pc	C625	16230
dest. pc	C625	15895
auth. type	Negotiate	29
logon type	Batch	10
auth. orient	LogOn	7
success	Success	2

Figure 2 Example of processed features of baselines.

[9]. Several Gaussian distributions are joint together to express the extracted eigenvectors and find out the outliers.

5 Results and Analysis

Table 3a summarizes the detection performance on Knet2016 dataset, while the ROC curves are shown in Fig. 3.

Among all the methods, the best one is log-line level detection of BEM, and that of user-day level are also satisfactory. The feature vectors used by baselines can be equivalent to user-day level detection, it is shown that performance of BEM model is better than that of baselines.

For baselines, the used feature vectors determine their performance. At present, the vectors focus on the statistics of log generated by a user in a day in different time periods [5]. If this statistical method can't directly reflect the pattern of anomalous logs, the baselines trained with these vectors will not get satisfactory results.

For BEM, it has achieved much better performance at logline level than user-day level. A good performance in logline level is easy to achieve, because the number of normal logs is hundreds of thousands of times that of anomalous logs, a few false positives will not make reduce the score too much. That is also the reason why the AUC score of logline level in [5] is so high.

In the calculation process at user-day level, we aggregate the anomaly score of each logline as the anomaly score of a user in a day. Proportion of normal and abnormal is reduced to several thousand times after the aggregation, which makes it more difficult to achieve higher AUC score. Besides, it may be the aggregation of anomaly scores which causes the

Table 3 Detection performance: (a) Performance on Knet2016 and (b) performance on R6.2 Detection performance: (a) Performance on Knet2016 and (b) performance on R6.2.

Model or Level	Tokenization	AUC
PCA	Vector	0.693
One-Class SVM	Vector	0.684
GMM	Vector	0.500
BEM logline	Characters	0.913
BEM user-day-max	Characters	0.821
BEM user-day-diff	Characters	0.711
Model or Level	Tokenization	AUC
PCA	Vector	0.954
One-Class SVM	Vector	0.869
GMM	Vector	0.500
BEM logline	Characters	0.984
BEM user-day-max	Characters	0.984
BEM user-day-diff	Characters	0.502

Field	Example	#unique labels
date	01/02/2010 6:21:31	726611
user	ANC1950	3867
pc	PC-4921	3867
url	http://icio.us/John.asp	132352
activity	WWW Visit	6
content	"Further consultation with post-production team"	54946

(a)

Field	Example	#unique labels
date	01/02/2010 07:17:18	62420
user	SDH2394	756
pc	PC-5849	1287
file_tree	R:\R\JKS2444	763
activity	Connect	2

(b)

Figure 3 ROC curves for dataset Knet2016.

performance. A better computing method of users' anomaly score needs to be explored.

Table 5b summarizes the detection performance on R6.2 dataset, while the ROC curves are shown in Fig. 4. Notice that several curves are completely coincident. So we recognize the dataset used in this experiment is too small.

The result of user-day-diff method is not satisfactory. One of the reasons may be that the proportion of normal and abnormal is too small.

Compared with the results of [5], we also come to the conclusion that the effect of the model at logline level is better than user-day level. However, [5] did not find the significant difference between logline level and user-day level when using the same data for training. This may be caused by the large difference in the amount of training data. Due to the limitation of computing power, we only use a small part of the dataset to train the model, so only methods with IR6.2le demand for data can maintain the best effect.

The two kinds of logs are also analyzed respectively, and the results are shown in Fig. 5. It can be seen that the results of user-day-diff method are not caused by the combination of the results of different logs.

44,7,0,19,18,74,15,8,27,24,2,0,0,0,2,0,0,0,0,0,0,0,
0,0,0,183,42,36,46,59,2,6,2,2,1,5,330,81,59,86,1,
0,14,2,2,2,8,16,4,2,2,8,317,97,67,67,86,287,89,5,
9,59,80,359,104,67,86,102,16,4,2,2,8,16,4,2,2,8,1,
78,62,31,40,45,336,98,65,84,89,31,23,8,0,0,0,2,2,0,
0,0,0,0,0,0,0,0,3,13,324,0,114,0,7,0,0,0,0,0,0,0,
265,0,0,0,0,0,0,12,0,57,21,0,0,156,121,6,361,0

Figure 4 ROC curves for dataset R6.2

Model or Level	Tokenization	AUC
PCA	Vector	0.693
One-Class SVM	Vector	0.684
GMM	Vector	0.500
BEM logline	Characters	0.913
BEM user-day-max	Characters	0.821
BEM user-day-diff	Characters	0.711

(a)

Model or Level	Tokenization	AUC
PCA	Vector	0.954
One-Class SVM	Vector	0.869
GMM	Vector	0.500
BEM logline	Characters	0.984
BEM user-day-max	Characters	0.984
BEM user-day-diff	Characters	0.502

(b)

Figure 5 Separate ROC curves for dataset R6.2: (a) Device data and (b) http data.

6 Conclusion

Based on the analysis of the logs content in datasets, we build an anomaly detection model based on LSTM. Processing the logs in Knet2016 and R6.2 datasets, training and testing the model on the extracted log-line text, the results show that it can correctly detect exceptions. Trough contrast experiments we can say that its effect is better than other models, including PCA, one class SVM and GMM.

However, the comparison between this model and other classical model still needs further testing. On one hand, training and testing should be carried out on datasets with wider sources and larger amount to obtain results which are more general; on the other hand, isolation forest should be introduced as a baseline for comparative experiments, because it is not as the best performing anomaly measurement algorithm in the recent DARPA insider threat detection program [10].

For the BEM itself, the use of dataset still needs to be adjusted. For each user in a day, a fixed number of logs could be selected for training to avoid oversized influence of any single user. At present, there are only 64 cells in the LSTM layer, more cells could be used in the next step.

A better calculation method of the anomalous score of a user needs to be explored. Anomalous loglines contain information of anomalous users, so a better integration method could improve the detection effect of the model on user-day level.

Through theoretical analysis, we give up the construction of multi-layer model. This enables the model to work at a lower computational power level. However, this does not mean that multi-layer model has no significance.

[11] propose another training and prediction method for log-line level, which also uses LSTM model. Through the

drain approach proposed by [12], the log-line is parsed to obtain the category, which is used in sequence to predict the distribution of the next log-line category after training by LSTM.

Obviously, that kind of anomaly detection mode works in log-line level, which is different from the model in this paper. These two levels of prediction do not interfere with each other, so it can be considered to a certain extent. For example, directly weight the results of the two, or use any resolution method which is different from relying on the hidden value of BiLSTM to extract log-line features to parse the log. In the training of log-line feature sequence, this training can also be conducted through BiLSTM.

Due to the limitation of computing power, the datasets are not fully used, so the scale of the training data should be expanded to obtain better results. In addition, in R6.2 dataset, besides device access log and net access log, which are used, there are also logon log, file access log, etc. The comprehensive use of these logs will enhance the ability to detect anomalous users, and especially useful for identifying red team users.

However, the comprehensive use of logs from different sources requires further research. Obviously, logs from different sources have different structures. Mixing these logs directly and training through character-level models may not yield a good result. A natural idea is to detect these logs separately, and then integrate these anomaly scores after obtaining the anomaly scores of each user; another idea is to parse the logs to category code [12], sort these codes by time, then use several models to make predictions on the sequence. Obviously, this method will lose part of the information.

In some sense, this is also a kind of hierarchical structure. Whether calculating anomaly score of users by log category and then integrate, or parsing by logline and then detect, it is to synthesize existing information at a certain level, and then use the comprehensive result to detect anomaly at a higher level.

All in all, the way of building hierarchical structure model is worth further study.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included

in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Y. Zhao, X. Wang, H. Xiao and X. Chi, Improvement of the Log Pattern Extracting Algorithm Using Text Similarity, 2018 IEEE International Parallel And Distributed Processing Symposium Workshops (IPDPSW), Vancouver, BC, 2018, pp. 507–514.
2. Xu, K. Y., Gong, X. R., & Cheng, M. C. (2016). Audit log association rule mining based on improved Apriori algorithm. *Computer Application*, 36(7), 1847–1851.
3. Y. Zhao and H. Xiao, Extracting Log Patterns from System Logs in LARGE, 2016 IEEE international parallel and distributed processing symposium workshops (IPDPSW), Chicago, IL, 2016, pp. 1645–1652.
4. Seker, S. E., Altun, O., Ayan, U., & Mert, C. (2014). A novel string distance function based on Most frequent K characters. *International Journal of Machine Learning & Computing*, 4(2), 177–183.
5. Tuor A, Baerwolf R, Knowles N, et al. Recurrent neural network language models for open vocabulary event-level cyber anomaly detection. 2017.
6. Kent and Alexander D. Cyber security data sources for dynamic network research, Dynamic Networks and Cyber-Security. 2016.
7. Hochreiter S. and Schmidhuber J. Long Short-Term Memory, *Neural computation* 9(8):1735–1780.
8. Rose S, Engel D, Cramer N, et al. Automatic keyword extraction from individual documents, Text Mining: Applications and Theory. John Wiley & Sons, Ltd, 2010, Automatic Keyword Extraction from Individual Documents.
9. W. Contributors. Maximum Likelihood Estimation, available: https://en.wikipedia.org/w/index.php?title=Maximum_likelihood_estimation&oldid=857905834, (2015).
10. Gavai, G., Sricharan, K., Gunning, D., Hanley, J., Singhal, M., & Rolleston, R. (2015). Supervised and unsupervised methods to detect insider threat from enterprise social and online activity data. *Journal Of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 6(4), 47–63.
11. Du M, Li F, Zheng G, et al. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning, *Acm SigSAC Conference on Computer & Communications Security ACM*, 2017.
12. He P, Zhu J, Zheng Z, et al. Drain: An online log parsing approach with fixed depth tree, 2017 IEEE international conference on web services (ICWS). IEEE, 2017.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

一种基于 LSTM 的日志分析异常检测模型

摘要 安全设备产生的大量日志远远超出了人类的处理速度。介绍了一种无监督的方法来检测大规模安全日志中的异常行为。我们提出了一种新的特征提取机制，能够准确地描述恶意行为的特征。我们设计了一种基于 LSTM 的异常检测方法，可以成功识别对两个广泛使用的数据集的攻击。在准确性和效率方面，我们的方法优于三种流行的异常检测算法，一类 SVM, GMM 和主成分分析。

关键词：异常检测。日志分析

1.简介

系统的运行状态通常记录在日志文件中，用于调试和故障检测，因此日志数据是异常检测的宝贵资源。日志数据是自然时间序列数据，日志文件记录的事件内容和类型也趋于稳定。除了一些隐蔽性很强的 apt 攻击之外，大多数攻击都不是瞬时的，具有固定的模式，日志数据在记录恶意行为时会产生模式，这为从日志序列中检测异常提供了可能性。

传统方法依赖管理员手动分析日志文本。这类流程导致大量人力成本，需要系统管理员了解网络环境，精通系统架构。

但是，为了避免被安全管理员跟踪，攻击生成的日志与正常访问行为生成的日志越来越相似。此外，由于应用程序和服务的种类繁多，每个 web 节点都会生成大量的日志，这导致日志数据文件变得非常大。可能无法直接手动处理这些日志。这些日志可能包含恶意行为的信号，因此有必要使用一些异常检测方法进行分析。

现有的基于日志数据的异常自动检测方法可以分为两类：基于标签的监督学习方法，如决策树、LR、SVM 和基于主成分分析、聚类和不交挖掘的非监督学习方法。监督学习在检测已知的恶意行为或异常状态方面有很好的效果，但它不能检测未知的攻击，因为它依赖于先验知识。无监督方法可以用于检测未知异常，但大多数方法需要提高准确性。

现有的一项研究引入了使用最长公共子序列的概念，以减少在计算过程中获得的匹配模式的数量[1]，从而能够对日志进行简单的分类。此外，徐等人利用信任度产生的关联规则挖掘频繁项集，并基于关联规则检测攻击行为[2]。赵等人用基于字符匹配的方法研究了系统日志的分类，确定了日志类型与字符的对应关系[3]。谢克尔.S.E.等人以字母的出现频率作为识别日志序列的关键要素，将不同类型的日志抽象成不同的字符，并根据其特征选择自适应的 k 值[4]。现有的研究表明，日志与其字符组成有很强的相关性。

该模型基于 **LSTM** 序列挖掘，通过数据驱动的异常检测方法，可以学习正常日志的序列模式，检测未知的恶意行为，识别大量日志序列中的红队攻击。该模型直接对日志文本进行字符级分析，因此不需要执行过多的日志处理，如日志分类、匹配等，大大简化了计算复杂度。

然而，目前的模式仍然存在一些问题。例如，它只执行字符级分析，这可能会忽略一些高级功能。一个自然的想法是对日志进行分层分析，但研究发现这种方法并没有提高其性能[5]，因此相关方法需要进一步研究。此外，由于没有进行日志相关匹配，所以只能对异常的日志行进行报警，不能直接检测每个用户的异常级别。一种可能的解决方案是通过被确定为异常的日志条目来跟踪用户，但是显然它稍微有点冗长。

贡献 通过共同努力，共同讨论实验后，周文浩完成了 **LSTM** 代码的修改和模型的训练，张九耀完成了对比实验，子琪元完成了文章的写作。

本文第一部分介绍了模型的背景、目标和存在的问题。第二部分介绍了模型中使用的方法，包括数据处理方法、**LSTM** 网络的具体结构以及参数的调整。第三部分将该模型与[5, 6]中的方法进行了比较，并说明了其优点。第四部分介绍了实验，包括对数据集、测试指标的详细描述，以及与 **SVM**、**GMM** 和主成分分析等其他方法的比较结果。

2 异常检测方法

我们的方法学习正常日志的字符级行为，处理日志行流如下：

(1) 随机初始化权重

(2) 用前 n 天的日志数据训练模型

(3) 对于按时间顺序排列的每一天 k ($k > n$)，首先基于用前 $k-1$ 天的日志数据训练的模型 M_{k-1} ，为第 k 天的所有事件产生异常分数。其次，按等级顺序记录每个用户事件的异常分数，供分析人员检查。第三，因为使用的是 k 日的日志，所以更新模型 M_{k-1} 到 M_k 。

2.1 日志线标记化

网络日志云可以从许多来源获得。从不同来源获得的日志自然会以不同的格式生成，并记录不同的信息。为了尽可能扩大模型的应用范围，我们将日志的每一行都直接看作一个字符串，并将字符级数据直接作为 **LSTM** 的输入。由于只考虑可打印字符，其范围从 0×20 到 $0 \times 7e$ ，因此总共有 95 个字符。把每个单词转换成它对应的 **ASCII** 码，减去 30，那么我们得到的开头是 0，结

尾是 1。然后用 -1 填充空格，以确保每个训练序列的长度相同。一个例子如图 1 所示。

2.2 LSTM 模型

为了计算每个记录的异常分数，我们考虑单个测井曲线的计算模型。RNN 是处理这类数据的常用工具。具体来说，我们使用 LSTM 网络，其输入是每个日志行的字符，以便预测下一个字符的概率分布，并推断该日志行或用户帐户的异常可能性。双向事件模式(BEM)对于字符长度为 K 的日志行，设 $x(t)$ 为位置 t 的字符， $h(t)$ 为该字符对应位置的隐藏表示。根据 LSTM 的相关理论 [7]，我们有以下关系

$$h(t) = o(t) \cdot \tanh(c(t)) \quad (2.1)$$

$$c(t) = f(t) \cdot c(t-1) + i(t) \cdot g(t) \quad (2.2)$$

$$g(t) = \tanh(x(t)W(g,x) + h(t-1)W(g,h)) \quad (2.3)$$

$$f(t) = \sigma(x(t)W(f,x) + h(t-1)W(f,h) + b(f)) \quad (2.4)$$

$$i(t) = \sigma(x(t)W(i,x) + h(t-1)W(i,h) + b(i)) \quad (2.5)$$

$$o(t) = \sigma(x(t)W(o,x) + h(t-1)W(o,h) + b(o)) \quad (2.6)$$

其中，初始状态 $h(0)$ 和初始单元状态 $c(0)$ 被预设为零向量，项目符号表示元素式乘法， σ 表示标准参数的逻辑函数，即

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.7)$$

0	55	19	18	19	34	38	49	47	19	14	37	19	26	24	20	6
34	38	49	47	19	14	37	19	26	24	20	14	37	19	26	24	
20	14	33	14	33	14	35	87	86	74	47	67	82	14	53	87	
69	69	71	85	85	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

图 1 LSTM 模型的已处理特征示例

向量 g 表示从当前输入和先前隐藏状态推断出的隐藏值。向量 f 、 i 、 o 是 LSTM 模型中的标准遗忘门、输入门和输出门。矩阵 W 和偏差向量 b 是模型中使用的参数。此外，由于我们可以直接获得日志行的所有内容，并且日志行中某个位置的字符显然与其上下文内容相关，因此我们构建了一个双向 LSTM，或称 BiLSTM，它同时从后向前推断每个位置的字符概率分布。为此，我们添

加了一组新的隐藏向量 $h^b(K+1)$, $h^b(K) \dots h^b(1)$, 因此该模型可以同时反向运行 LSTM 方程。逆 LSTM 参数的上标 b 表示其参数矩阵 W

隐藏值 h 用于预测新位置的字符, 结果是 p, 具体来说, 我们有

$$p(t) = \text{softmax}(h^{(t-1)}W(p) + h^b(t+1)W^b(p) + b(p)) \quad (2.8)$$

与普通的单向 LSTM 相比, 很明显, 反向 LSTM 的隐藏值矩阵 W 被添加到预测函数中, 使我们能够同时通过正负方向预测某个位置的人物。

最后, 交叉熵损失定义为

$$\frac{1}{K} \sum_{t=1}^K H(x(t), p(t)) \quad (2.9)$$

来更新权重。我们使用随机小批量(非截断)时间反向传播来训练这个模型。

3.相关工作

最相关的工作是[5, 6]。其中, [5]构建了 Knet2016 数据集的双层 LSTM 模型。其下层由 LSTM 网络组成。在此基础上, 将每个令牌的隐藏状态作为该行的特征, 得到该行的类型并构建 LSTM 模型, 完成对下一行类型的预测。然而, 实验结果表明, 双层结构并没有提高预测性能, 其度量低于简单的单层模型。由于计算能力的限制, 我们不得不使用更少的数据, 所以我们没有建立双层 LSTM 模型, 而是使用单层 BiLSTM 模型, 通过该模型可以同时双向预测可能出现的字符。这一方面在一定程度上保证了检测效果, 另一方面直接降低了对计算能力的需求。[6]详细讨论了许多日志的价值和潜在问题, 并给出了一个新的、全面的、真实的网络安全数据概述。本文列举了几种日志, 统计了词频等信息, 但没有给出详细可行的处理方法。每种日志都有不同的格式和保留字。即使我们能够进行详细的分析, 也很难找到一种通用的、快速的方法来对不同类型的日志进行异常检测。因此, 我们有必要进行特征层面的分析和预测, 这可以直接避免不同测井曲线之间的巨大差异, 而且由于正常测井曲线和异常测井曲线在特征上有明显的不同, 所以我们有必要进行特征层面的分析和预测。

4.实验结果

4.1 数据

我们使用了洛斯阿拉莫斯国家实验室(LALL)网络安全数据集(Kent 2016),该数据集连续 58 天收集了 LALL 内部计算机网络的事件日志,超过 10 亿条日志线,包括认证、网络流量和其他记录。涉及隐私的领域已经匿名。除了正常的网络活动,还记录了 30 天的红队攻击。[5]给出了一些统计描述,我们使用的部分,即身份验证事件日志,如表 1 所示。我们在 20 天内使用了 3028187 个日志,前 14 天用于培训,其余 6 天用于测试,有 51 个异常日志和 26 个异常用户日。标记化后,所有日志行的长度都填充为 112。我们还使用了内部线程测试数据集(R6.2),它是一个合成内部线程测试数据集的集合,提供了背景和恶意参与者合成数据。它包含了设备交互、电子邮件、文件系统等方面的日志内容。实验中我们选择了设备访问日志和网络访问日志。表 2a 总结了网络访问日志的字段和统计信息,表 2b 显示了设备访问日志的字段和统计信息。

Field	Example	#unique labels
time	1	5,011,198
source user	C625@DOM1	80,553
dest. User	U147@DOM1	98,563
source pc	C625	16,230
dest. pc	C625	15,895
auth.type	Negotiate	29
logon type	Batch	10
auth. Orient	LogOn	7
succes	Succes	2

表 1 knet 2016 数据集统计

我们在 21 天内使用了 1676485 个日志,前 14 天用于培训,其余 7 天用于测试,有 530 个异常日志和 530 个异常用户日。

在标记化过程中,发现网络访问数据集的日志长度高达 2500 个字符。如果所有的行都填充到这么长的序列中,无疑会花费非常大的计算量。所以 URL 栏只提取域名,页面的简介栏通过 rake 算法提取其关键短语[8]。最后,序列的长度被确定为 830。

时间表 对于这项工作,我们考虑以下时间表。

首先，日志的每一行都记录了一个相对独立的动作，比如一个用户做了一个身份认证，结果是成功还是失败，所以每一行都被视为一组独立的输入。我们称这个时间刻度为对数线水平。顾名思义，对数线水平分析将计算每条线的异常分数。

其次，为了与基线实验进行比较，我们将每个用户每天的日志记录的所有动作进行整合，将一个用户每天的每个日志线的异常分值进行汇总，计算出该用户当天的异常分值。因为我们使用用户的最大异常分数，所以它被称为用户日最大级别。此级别表示用户在特定日期出现异常的可能性。

此外，我们使用规范化策略。对于每一行的异常分值，先减去对应用户当天的平均异常分值，再进行正常计算，实现原始分值的归一化。这种归一化的计算方法被称为差异，所以它被称为用户日差异级别，标记用户在特定一天的最大异常分数和平均分数之间的差异。

4.2 度量

我们使用 AUC 作为评估度量，它表示接收器操作员特征曲线下的区域。它描述了模型检测性能在真阳性和假阳性之间的权衡。

AUC 值不会大于 1，值越高，模型性能越好。

4.3 基线

根据[5]，对于我们使用的数据，我们为每个用户定义了一个多维聚合特征向量，作为对比实验的基础。

我们考虑三种基线模型，即一类 SVM 模型、GMM 模型和主成分分析模型。

数据处理 在数据方面，我们先过滤掉数据集中的异常日志，只保留正常日志，取 70%作为训练集；然后将原始数据集中剩余的 30%作为测试集。

具体来说，字段值出现的可能性分为常见和不常见，对象分为单个用户和所有用户。源 PC 名称、目标 PC 名称、目标用户、进程名称和进程的 PC 名称作为统计信息的关键字段。时间分为一整天，0-6、6-12、12-18 和 18-24。

Field	Example	#unique labels
date	01/02/2010 6:21:31	726,611
user	ANC1950	3867
pc	PC-4921	3867
url	http://icio.us/John.asp	132,352
activity	WWW Visit	6
content	"Further consultation with post-production team"	54,946
Field	Example	#unique labels
date	01/02/2010 07:17:18	62,420
user	SDH2394	756
pc	PC-5849	1287
file_tree	R:\R\JKS2444	763
activity	Connect	2

表 2 R6.2 的数据集统计信息:(a)网络访问日志字段和统计信息；(b)设备访问日志字段和统计信息。

在概率上做笛卡尔乘积，面向对象，关键字段和时间得到 100 个特征统计信息，然后将日志中经常出现的字段(如登录结果)作为特征，最终得到 134 维特征向量。

每个用户的每日日志都会被汇总。如果某个字段中只有几个不同的值，如登录结果，则这些值的频率将被计算在内。如果某个字段中有大量不同的值，如 PC 名、用户名等。这些价值观会被分为普通的或不普通的。

对于一个用户来说，如果一个值的出现频率小于 5%，就会被归类为不常见；否则，它将被归类为普通。对于所有用户来说，如果一个值的出现频率小于其所在领域的平均值，则归类为不常见；否则，它将被归类为普通。每天统计特征向量的各个维度的信息，得到每个用户的特征向量。一个例子如图 2 所示。

A.主成分分析

对提取的特征向量进行主成分分析，将原始数据从原始空间投影到主成分空间，然后将投影重建到原始空间。如果只使用第一主成分进行投影和重建，对于大多数数据，重建后误差较小；但是对于异常值，重构后的误差仍然相对较大。

B.一类 SVM

在日志的检测中，只有两类:正常和异常，正常的数据要比另一类多得多。因此，可以用一类 SVM 对提取的特征进行分类，完成异常检测。

C.高斯混合模型

高斯混合模型(GMM)是最流行的统计方法之一，用于通过使用最大似然估计(MLE)方法来执行高斯分布的均值和方差估计来检测异常[9]。将几个高斯分布联合起来表示提取的特征向量，找出离群点。

Field	Example	# unique labels
time	1	5011198
source user	C625@DOM1	80553
dest. user	U147@DOM1	98563
source pc	C625	16230
dest. pc	C625	15895
auth. type	Negotiate	29
logon type	Batch	10
auth. orient	LogOn	7
success	Success	2

图 2 基线处理特征的例子。

5.结果和分析

表 3a 总结了 Knet2016 数据集的检测性能，而 ROC 曲线如图 3 所示。

在所有的方中，最好的是边界元法的对数水平检测，用户日水平的检测也是令人满意的。基线使用的特征向量相当于用户日水平检测，表明边界元模型的性能优于基线。

对于基线，使用的特征向量决定其性能。目前，向量侧重于统计用户在一天中不同时间段生成的日志[5]。如果这种统计方法不能直接反映异常日志的模式，用这些向量训练的基线将得不到满意的结果。

对于边界元法来说，它在对数线级比用户日级取得了更好的性能。日志级别的好性能很容易实现，因为正常日志的数量是异常日志的几十万倍，少量的误报不会使分数降低太多。这也是为什么[5]中 logline 级别的 AUC 得分是如此高的原因。

在用户日级别的计算过程中，我们将每个 logline 的异常得分聚合为一个用户在一天中的异常得分。正常和异常的比例在聚合后降低到几千倍，使得更难达到更高的 AUC 评分。此外，可能是异常分数的集合导致了这种现象。需要探索一种更好的用户异常评分计算方法。

Model or Level	Tokenization	AUC
PCA	Vector	0.693
One-Class SVM	Vector	0.684
GMM	Vector	0.500
BEM logline	Characters	0.913
BEM user-day-max	Characters	0.821
BEM user-day-diff	Characters	0.711
Model or Level	Tokenization	AUC
PCA	Vector	0.954
One-Class SVM	Vector	0.869
GMM	Vector	0.500
BEM logline	Characters	0.984
BEM user-day-max	Characters	0.984
BEM user-day-diff	Characters	0.502

表 3 检测性能:(a)在 Knet2016 上的性能和(b)在 R6.2 上的性能

Field	Example	#unique labels
date	01/02/2010 6:21:31	726611
user	ANC1950	3867
pc	PC-4921	3867
url	http://icio.us/John.asp	132352
activity	WWW Visit	6
content	"Further consultation with post-production team"	54946

(a)

Field	Example	#unique labels
date	01/02/2010 07:17:18	62420
user	SDH2394	756
pc	PC-5849	1287
file_tree	R:\R\JKS2444	763
activity	Connect	2

(b)

图 3 数据集 Knet2016 的 ROC 曲线。

表 5b 总结了在 R6.2 数据集上的检测性能，而 ROC 曲线如图 4 所示。几条曲线完全重合。所以我们认识到这个实验中使用的数据集太小了。

用户日差值法的结果不令人满意。原因之一可能是正常和不正常的比例太小。

与文献[5]的结果相比，我们还得出该模型在对数水平上的效果优于用户日水平的结论。然而，当使用相同的数据进行训练时，[5]没有发现对数线水平和用户日水平之间的显著差异。这可能是由于训练数据量差异大造成的。由于计算能力的限制，我们只使用数据集的一小部分来训练模型，所以只有对数据需求 1R6.21e 的方法才能保持最佳效果。

这两种日志也分别进行了分析，结果如图 5 所示，可以看出，用户日差方法的结果并不是由不同日志的结果组合而成的。

44,7,0,19,18,74,15,8,27,24,2,0,0,0,2,0,0,0,0,0,0,
 0,0,0,183,42,36,46,59,25,6,2,2,15,330,81,59,86,1
 04,14,2,2,2,8,16,4,2,2,8,317,97,67,67,86,287,89,5
 9,59,80,359,104,67,86,102,16,4,2,2,8,16,4,2,2,8,1
 78,62,31,40,45,336,98,65,84,89,31,23,8,0,0,2,2,0,
 0,0,0,0,0,0,0,0,0,3,13,324,0,114,0,0,7,0,0,0,0,0,
 265,0,0,0,0,0,0,12,0,57,21,0,0,156,121,6,361,0

图 4 数据集 R6.2 的 ROC 曲线

Model or Level	Tokenization	AUC
PCA	Vector	0.693
One-Class SVM	Vector	0.684
GMM	Vector	0.500
BEM logline	Characters	0.913
BEM user-day-max	Characters	0.821
BEM user-day-diff	Characters	0.711

(a)

Model or Level	Tokenization	AUC
PCA	Vector	0.954
One-Class SVM	Vector	0.869
GMM	Vector	0.500
BEM logline	Characters	0.984
BEM user-day-max	Characters	0.984
BEM user-day-diff	Characters	0.502

(b)

图 5 数据集 R6.2 的独立 ROC 曲线:(a)设备数据和(b) http 数据。

6. 结论

在分析数据集日志内容的基础上,构建了基于 LSTM 的异常检测模型。对 Knet2016 和 R6.2 数据集的日志进行处理,对提取的日志行文本进行训练和测试,结果表明能够正确检测异常。通过对比实验,可以说其效果优于其他模型,包括主成分分析、一级 SVM 和 GMM。

然而,该模型与其他经典模型比较仍需进一步检验。一方面,应在来源更广、数量更多的数据集上进行培训和测试,以获得更普遍的结果;另一方面,应该引入隔离森林作为比较实验的基线,因为在最近的 DARPA 内部线程检测计划中,它不是表现最好的异常测量算法[10]。

对于边界元法本身,数据集的使用仍需调整。对于一天中的每个用户,可以选择固定数量的日志进行训练,以避免任何单个用户的过大影响。目前,LSTM 层只有 64 个单元,下一步可以使用更多的单元。

需要探索一种更好的用户异常分数的计算方法。异常日志线包含异常用户的信息，因此更好的整合方法可以提高模型在用户日层面的检测效果。

通过理论分析，我们放弃了多层模型的构建。这使得该模型能够在较低的计算能力水平下工作。但是，这并不意味着多层模型没有意义。

[11]提出另一种对数线水平的训练和预测方法，该方法也使用 LSTM 模型。通过由[12]提出的 drain 方法，对数线进行解析以获得类别，该类别被依次用于预测经过 LSTM 训练后的下一个对数线类别的分布。

显然，这种异常检测模式工作在日志级别，与本文的模型不同。这两个层面的预测互不干扰，所以可以在一定程度上考虑。例如，直接对两者的结果进行加权，或者使用不同于依赖 BiLSTM 隐藏值的任何解析方法来提取日志线特征来解析日志。在对数线特征序列的训练中，这种训练也可以通过 BiLSTM 进行。

由于计算能力的限制，数据集没有得到充分利用，因此应该扩大训练数据的规模，以获得更好的结果。另外，在 R6.2 数据集中，除了使用的设备访问日志和网络访问日志外，还有登录日志、文件访问日志等。这些日志的综合使用将增强检测异常用户的能力，对于识别红队用户尤其有用。

然而，综合利用不同来源的日志需要进一步研究。显然，不同来源的日志具有不同的结构。直接混合这些日志并通过角色级模型进行训练可能不会产生好的结果。一个自然的思路是，分别检测这些日志，得到每个用户的异常分值后，再对这些异常分值进行整合；另一个想法是将日志解析为类别代码[12]，按时间对这些代码进行排序，然后使用几个模型对序列进行预测。显然，这种方法会丢失一部分信息。

从某种意义上说，这也是一种等级结构。无论是按日志类别计算用户的异常分值然后进行整合，还是按日志线解析然后进行检测，都是综合某一级别的已有信息，然后利用综合结果检测更高级别的异常。

总之，建立层次结构模型的方法值得进一步研究。

参考文献

- [1] Y. Zhao, X. Wang, H. Xiao and X. Chi, Improvement of the Log Pattern Extracting Algorithm Using Text Similarity, 2018 IEEE International Parallel And Distributed Processing Symposium Workshops (IPDPSW), Vancouver, BC, 2018, pp. 507–514.

-
- [2] Xu, K. Y., Gong, X. R., & Cheng, M. C. (2016). Audit log association rule mining based on improved Apriori algorithm. *Computer Application*, 36(7), 1847–1851.
 - [3] Y. Zhao and H. Xiao, Extracting Log Patterns from System Logs in LARGE, 2016 IEEE international parallel and distributed processing symposium workshops (IPDPSW), Chicago, IL, 2016, pp.1645–1652.
 - [4] Seker, S. E., Altun, O., Ayan, U., & Mert, C. (2014). A novel string distance function based on Most frequent K characters. *International Journal of Machine Learning & Computing*, 4(2), 177–183.
 - [5] Tuor A, Baerwolf R, Knowles N, et al. Recurrent neural network language models for open vocabulary event-level cyber anomaly detection. 2017.
 - [6] Kent and Alexander D. Cyber security data sources for dynamic network research, *Dynamic Networks and Cyber-Security*. 2016.
 - [7] Hochreiter S. and Schmidhuber J.. Long Short-Term Memory, *Neural computation* 9(8):1735–1780.
 - [8] Rose S, Engel D, Cramer N, et al. Automatic keyword extraction from individual documents, *Text Mining: Applications and Theory*. John Wiley & Sons, Ltd, 2010, Automatic Keyword Extraction from Individual Documents.
 - [9] W. Contributors. Maximum Likelihood Estimation
available:https://en.wikipedia.org/w/index.php?title=Maximum_likelihood_estimation&oldid=857905834, (2015).
 - [10] Gavai, G., Sricharan, K., Gunning, D., Hanley, J., Singhal, M., & Rolleston, R. (2015). Supervised and unsupervised methods to detect insider threat from enterprise social and online activity data. *Journal Of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 6(4), 47–63.
 - [11] Du M, Li F, Zheng G, et al. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning, *Acm Sigsac Conference on Computer & Communications Security ACM*, 2017.
 - [12] He P, Zhu J, Zheng Z, et al. Drain: An online log parsing approach with fixed depth tree, 2017 IEEE international conference on web services (ICWS). IEEE, 2017.

文献综述

1 引言

现代系统在运行过程中往往产生大量日志文件，日志文件包含有关系统中事件的信息。日志分析已经在控制台、网络应用、存储系统、并发系统以及一般系统等广泛应用，日志文件在监控网络情况、检查硬件故障、保护软件安全等方面起着重要作用[1]。通过分析系统日志记录程序运行时的动态信息，可以有效帮助维护人员分析重现错误，进而更正系统错误，提高系统运行时的可靠性[2]。然而，在系统运行中，每天产生的日志数量庞大，如何有效地分析仍然是一个巨大的挑战。研究人员主要通过日志解析获取日志相关事件模板，完成对日志的自动化分析。日志解析包括很多方法，如正则表达式、代码分析、机器学习和自然语言处理等相关技术，且已经具有较为成熟的工具。在日志解析基础上，日志分析应用也很广泛，主要包括异常检测、故障诊断、程序分析与验证、调查取证和性能诊断等。异常检测是指在数据中发现与预期行为不符的模式的问题，这些不合格的模式通常被称为不同应用领域中的异常情况、不一致的观察结果、意外情况等。一方面，系统越来越先进和复杂，如果希望系统安全有效，那么异常检测是一项必不可少的任务；另一方面，执行异常检测对于大规模的开发、维护和性能优化具有重要作用。

2 日志异常检测技术简介

日志的异常检测框架主要包括 4 个部分：日志收集与预处理、日志解析、特征提取和异常检测。

（1）日志收集与预处理

日志收集[3]主要指从设备中获取相关日志记录，包括系统的运行状态、时间信息等，这些信息用来记录系统运行过程中的重要信息，是日志分析的前提。在收集完日志数据后，进行预处理，主要是根据具体需求，将日志数据中的无效信息进行剔除，包括重复信息、无用信息等。

（2）日志解析

日志解析是从无结构的日志中提取相应的事件模板，每个模板由多个指定参数构成，作为后续特征提取的基础。由于日志数量庞大，当前主要采用固定时间窗口、滑

动时间窗口、会话窗口等为日志分割的方式[4]，将庞大的日志划分为多个部分进行检测，或者通过采样的方式进行检测。在数据采样过程中，采样方式、样本尺寸与检测正确率关系密切，因此，在该方向进一步探索，对提高所提方案的实用性有重要推动作用。

（3）特征提取

通过设定固定的时间范围作为窗口，对窗口内日志内容进行解析，提炼其中的事件模板，将日志切分成一组日志事件序列 $\text{Log}=\{e_1, e_2, \dots, e_n\}$ ，进而从提取信息中选择合适的变量和分组来表示相关内容，并进行数字化向量表示，构建成特征向量，方便后续进行机器学习。

（4）异常检测

异常检测过程主要是使用机器学习的方法（如聚类分析、决策树、深度学习等）对特征向量进行学习，产生一个异常检测模型，通过对输入特征矩阵机器学习模型进行训练，从而生成一个异常检测模型，并使用该模型对新的日志进行检测。

3 日志异常检测技术

3.1 监督学习

监督学习被定义为一种从标记的训练数据中得出模型的机器学习方法。带标签的训练数据（通过标签指示正常或异常状态）是监督异常检测的先决条件，训练数据标签越多，模型越精确。由于监督学习需要前期进行人工标记，工作量大，因此自动化程度相对不高，更多的研究集中到自动化程度更高的无监督学习。

3.2 无监督学习

无监督学习是指没有任何标记的训练数据情况下得到模型的机器学习方法。常见的无监督方法包括各种聚类方法[5]、关联规则挖掘、基于主成分分析等，具体如下。

（1）聚类方法

LogCluster 是一种将日志聚类来进行日志问题识别的方法。首先将每个日志序列通过一个向量表示，然后计算两个日志序列之间的相似度值，并应用聚集层次聚类技术将相似的日志序列分组为聚类。Log2vec 是一种基于异构图嵌入的威胁检测

方法。首先，根据日志条目之间的关系将其转换为异构图；然后使用一个改进的图嵌入方法，可以自动对每个日志描述为一个低维的向量；最后，通过检测算法，可以将恶意日志和良性日志进行聚类，识别出恶意行为。

（2）关联规则挖掘

首先，使用日志解析器将非结构化日志转换为结构化日志。然后，根据日志参数之间的关系，结构化日志消息被进一步分组以记录消息组。之后程序不变式将自动从日志消息组中挖掘，挖掘的不变量可以揭示内在的线性特征程序工作流程，自动检测日志中的异常。

（3）基于主成分分析（PCA）

首先通过系统执行时的特征构造特征矩阵，然后使用 PCA[6]与信息检索中的术语加权技术相结合，构造异常检测方法。DILAF 是一个大型系统日志的分布式分析异常检测方法。DILAF 采用了批量同步并行计算（BSP）模型来并行化机器学习过程。BSP 算法将工作负载分配到各个参与流程之间，每个流程均执行本地计算。当一个进程完成其任务并到达障碍时，它将等待直到所有其他进程达到相同的同步障碍。

3.3 深度学习

近年来，基于深度学习的异常检测算法变得越来越流行，并且在多种任务集中深度学习的应用超越了传统方法，因此这里单独对其进行介绍。

（1）CausalConvLSTM

CausalConvLSTM 是一种新型混合模型，它利用卷积神经网络（CNN）的优势，可以有效地提取并行方式中的空间特征，并通过长短期记忆（LSTM）网络来捕获顺序关系。该模型可以有效解决异常检测中概念漂移的问题。

（2）DeepLog

DeepLog[7]是主要使用长短期记忆的神经网络模型，将系统日志建模为自然语言序列。DeepLog 从正常执行中自动学习日志模型，并通过该模型，对正常执行下的日志数据进行异常检测。

（3）LogGAN

LogGAN 是一种基于 LSTM 的生成对抗网络，以基于时态系统日志区分即将到

来的异常事件。首先，利用自定义的日志解析器提取结构化信息（即时间戳、签名和参数），并将每个日志转换为事件。其次，事件的组合（即模式）使用滑动窗口从时态系统日志中收集相应的即将发生的事件。

（4）LogRobust

LogRobust 是一种基于不稳定日志数据的异常检测方法。LogRobust 提取日志事件的语义信息并将它们表示为语义向量。然后通过使用基于注意力的 Bi-LSTM 模型检测异常，该模型具有捕获日志序列中的上下文信息的能力，并且自动了解不同日志事件的重要性。

4 结束语

日志已经成为当前信息系统产生的重要信息资源。基于日志的异常检测技术可以有效发现系统中存在的安全问题，发掘潜在的安全威胁，成为当前研究的热点。随着人工智能技术的发展和普及，越来越多的相关研究成果已经应用于基于日志的异常检测。在基于日志的异常检测方法中，包括日志收集、日志解析、特征提取、异常检测等步骤。在基于日志的异常检测方法中，主要分为监督学习、无监督学习和深度学习等。异常检测方法大多针对特定场景和数据集进行离线分析，缺乏通用性和高准确性的实用方法。因此，在后续的研究中，不仅关注单一日志来源，还需要结合不同事件、不同设备进行日志解析，进而进行异常检测等；此外，利用机器学习的相关研究将进一步应用于在线检测，构建通用、有效的在线基于日志的异常检测方法，并应用到实际中。

参考文献

- [1] 廖湘科, 李姗姗. 大规模软件系统日志研究综述[J]. 软件学报, 2016, 27(8): 1934-1947.
- [2] OLINER A J, GANAPATHI A, XU W. Advances and challenges in log analysis[J]. Communications of the ACM, 2012, 55(2):55–61.
- [3] ARIEL R, RANDY K. Chukwa: a system for reliable large-scale log collection[C] //Unix Large Installation Systems Administration Conference. 2010: 1-15.

- [4] HE SL, ZHU J M, HE P J, et al. Experience report: system log analysis for anomaly detection[C]// 27th International Symposium on Software Reliability Engineering. 2016: 207-218.
- [5] DU M, LI F F. Spell: streaming parsing of system event logs[C]// ICDM 2016. 2016: 859-864.
- [6] DUNIA R, QIN J S. Multi-dimensional fault diagnosis using a subspace approach[C]//ACC. 1997:1-5
- [7] DU M, LI F F, VIVEK S. DeepLog: anomaly detection and diagnosis from system logs through deep learning[C]// CCS 2017: 1285-1298. [8] French W. Silences: A Voice from China [N]. Atlantic Weekly, 1987-8-15(33).

苏州大学本科生毕业设计（论文）中期进展情况检查表


学院（部）：计算机科学与技术学院

学生姓名	陈健	年级	2017	专业	计算机科学与技术（人工智能）	填表日期	2021/5/6
设计（论文）选题	日志异常检测算法研究与实现						
已完成的任务	学习日志异常检测算法的步骤，对相关日志进行解析。						
	是否符合任务书要求进度		是				
尚须完成的任务	掌握 DeepLog 算法的核心思想，基于 LSTM 对日志进行异常检测。						
	能否按期完成任务		是				
存在的问题	存在的问题	算法的代码尚需优化，论文参考资料较少。					
	拟采取的办法	进行代码优化，查询更多相关资源。					
指导教师意见	加快进度 签名：贾德金						
中期检查专家组意见	通过 组长签名：王宜						
学院（部）意见	同意 教学院长签名：王宜						

检查日期：

苏州大学本科生毕业设计（论文）答辩记录表

学院（部）：计算机科学与技术学院

学生姓名	陈健	年 级	2017	专 业	计算机科学与技术 (人工智能)
设计（论文）题目	日志异常检测算法研究与实现				
答辩时间	2021 年 5 月 13 日	答辩地点	理工楼 234		
答辩小组成员： 杨哲，樊建席，姚望舒，杨哲，陈伟					
答辩中提出的主要问题及学生回答问题的简要情况： 1. 日志异常检测有什么实际应用？ 系统会产生大量的日志，日志会记录着系统的状态信息，因此从日志中可以看出系统的正常或者异常状态。通常正常的日志序列为具有上下联系的较为规范的文本，如果出现某个日志出现错误，则说明此时日志会产生不同于之前的日志序列，正常的日志序列逻辑会被打断，说明此时日志发生了异常，也就意味着系统在此时发生了异常。日志异常检测如今已经具有许多相关技术，主要应用在日志解析和异常检测两个方面，但是由于各个系统的日志各有其模板，格式不统一，仍然缺少相对应的通用的检测方法。 2. 你的实验的创新点是什么？ 我一共进行了两个实验，分别是基于 LCS 的日志解析实验和基于 LSTM 的异常检测实验。对于基于 LCS 的日志解析实验，在不得到源代码的情况下，我们很难直接对单条日志的日志键和日志参数进行提取，因此可以考虑在对比中进行提取，基于最大公共子序列的方式，寻找两个日志的最大公共子序列，通过提取其相同的部分进行作为日志键，不同的部分作为日志键，取得了不错的效果。对于基于 LSTM 的异常检测实验，日志可以看作较为规范化的自然语言处理，LSTM 很适合自然语言处理，因此考虑使用 LSTM 进行异常检测实验，通过输入的部分序列来预测下一序列，对比与实际值的差异，以此达到异常检测的效果。 3. 数据集的具体情况和数据处理的情况？ 日志解析实验的数据采用了公开的 HDFS 数据集，它是亚马逊 EC2 平台收集到的 Hadoop 分布式系统日志，共包含了 11175629 条日志消息，在日志解析中选取 1000 条数据进行实验。异常检测的数据集同样为 HDFS 数据集，不同的是已经由相关领域的专家进行了特征编码，分为 hdfs_train、test_hdfs_normal 和 test_hdfs_abnormal 三个文件，每一个数字代表了一条日志键，实验中以长度 h 进行读取，依次读取 hdfs_train 的 h 条数据进行 reshape 和归一化之后作为输入，其后的 1 条数据作为实际值并编码为 one-hot 向量进行训练，训练之后的模型分别采用两个测试集进行测试，并通过计算得出相应的性能指标。  答辩小组组长签名：					

答辩小组成员签名：

樊建席, 陈伟, 招培

陈伟

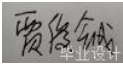
记录人签名：



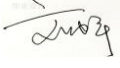
陈伟

2021 年 5 月 13 日

苏州大学本科生毕业设计（论文）成绩评定表

学院（部）：计算机科学与技术学院

设计（论文）题目：日志异常检测算法研究与实现			
姓 名	陈健	学 号	1727405145
年 级	2017	专 业	计算机科学与技术（人工智能）
指导教师评语	本文主要以日志异常检测流程作为研究对象，介绍现有的日志解析和异常检测算法，对比各种算法的优缺点。在分析各种日志异常算法的基础上，对日志异常检测算法的有效性和效率进行了系统评估，并进行了实验验证。方案设计、文献检索、总体工作进度负荷要求，质量达标，工作量充分，态度较好。体现了一定的分析问题和解决问题的能力。		
	评 价 内 容		得 分
	设计（论文）方案设计、文献检索、阅读及综述能力、进度等情况评价分（计 25 分）		20
	毕业设计（论文）质量和工作量评价分（计 50 分）		45
	科学素养、学习态度、纪律表现等情况评价分（计 25 分）		22
	成绩（满分 100 分）：87 签名：  2021 年 5 月 13 日		

评阅教师评语	该同学毕业设计学习了日志异常检测算法，并针对学习的日志异常检测算法进行了编码实现。从论文中可以看出，该学生学习基于最长公共子序列(Longest Common Subsequence, 简称 LCS) 和基于长短时记忆(Long short-term memory, 简称 LSTM) 两种日志解析算法，并设计了相关实验，完成了实验结果分析等。表明该学生具有一定的学习能力和问题分析能力。论文论述基本流畅，工作量符合要求，但在论述日志异常检测算法章节的结构不是很好，建议采用算法原理、算法流程、实验设计、实验结果分析的结构论述。	
	评 价 内 容	得 分
	毕业设计（论文）文字书写评价分（计 20 分）	15
	毕业设计（论文）质量评价分（计 40 分）	35
	工作量情况评价分（计 20 分）	16
	毕业设计（论文）创新及分析问题、解决问题能力评价分（计 20 分）	16
	成绩（满分 100 分）：82 签名：  2021 年 5 月 13 日	
答辩小组评语	陈健能很好地运用所学理论和专业知识，简明扼要地阐述论文的主要内容，思路清晰，语言表达准确，表现出对所从事的研究内容掌握得很透彻。回答问题有理论根据，思路清晰，论点正确，基本概念清楚，对主要问题回答深入正确，有一定创见。全面完成了任务书所规定的各项要求；论文书写完全符合规范化要求。	
	评 价 内 容	得 分
	毕业设计（论文）介绍表达情况评价分（计 20 分）	16
	回答问题表现评价分（计 40 分）	35
	毕业设计（论文）水平和工作量评价分（计 40 分）	33
	成绩（总分 100 分）：84 组长签名：  2021 年 5 月 13 日	
按权重折算总成绩		82
审定成绩：84 答辩委员会主任签名：  日 期：		

文本复制检测报告单

No: ADBD2021R_2018112714114520210511214258810825879539

检测文献: 日志异常检测算法研究与实现

作者: 陈健

检测范围: 中国学术期刊网络出版总库
中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库
中国重要会议论文全文数据库
中国重要报纸全文数据库
中国专利全文数据库
图书资源
优先出版文献库
大学生论文联合比对库
互联网资源(包含贴吧等论坛资源)
英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等)
港澳台学术文献库
互联网文档资源
CNKI大成编客-原创作品库

时间范围: 1900-01-01至2021-05-11

检测时间: 2021-05-11 21:42:58

总文字复制比: 7.1%

去除引用: 7.1%

去除本人: 7.1%

重合字数: 899

文献总字数: 12593

总段落数: [6] **疑似段落数:** [5] **疑似段落最大重合字数:** [329]

前部重合字数: [105] **后部重合字数:** [794] **疑似段落最小重合字数:** [34]

5.4% 中英文摘要等(总2855字)
33.5% 第一章绪论(总982字)
1.4% 第二章日志异常检测技术(总2401字)
0% 第三章基于LCS的日志解析算法研究(总2012字)
9.9% 第四章基于LSTM的日志异常检测算法研究(总2998字)
6.3% 第五章总结与展望(总1345字)