日志异常检测方法及其应用研究

作者姓名 赵晓庆

学校导师姓名、职称 蒋忠元 副教授

企业导师姓名、职称 赵刚 高工

申请学位类别 工程硕士

学校代码 10701　　学　号 20151213585
分 类 号 TP39　　密　级 公开

西安电子科技大学

硕士学位论文

日志异常检测方法及其应用研究

作者姓名：赵晓庆

领　　域：网络与信息安全

学位类别：工程硕士

学校导师姓名、职称：蒋忠元 副教授

企业导师姓名、职称：赵刚 高工

学　　院：网络与信息安全学院

提交日期：2023 年 6 月

Log Anomaly Detection Method and Application Research

A thesis submitted to

XIDIAN UNIVERSITY

in partial fulfillment of the requirements

for the degree of Master

in Engineering

By

Zhao Xiaoqing

Supervisor: Jiang Zhongyuan Title: Associate Professor

Supervisor: Zhao Gang Title: Senior Engineer

June 2023

摘　要

随着大数据、云计算、5G 通讯和区块链等业务的快速发展，现代大型服务系统的规模迅速增加。大型服务系统的稳定可靠运行尤为重要，任何异常事件的发生都有可能造成不可估量的损失，因此，系统运行状态的监控十分关键。格式化或非格式化的海量日志文件记录了现代复杂系统的运行状态与操作信息，是系统异常溯源与分析的基础。基于日志数据的异常检测是系统状态监控与异常响应的关键手段，可以帮助系统运维人员及时发现异常行为和系统故障，并有效解决问题。

本文对日志异常检测框架进行了梳理，并对各技术点进行了较为详细的综述。旨在较为全面的介绍日志异常检测的现有技术内容，为相关科研工作者提供参考和借鉴。目前各类数据挖掘、机器学习和深度学习算法已经被广泛的应用于日志异常检测领域，但是日志异常检测依然存在诸多问题，例如，非结构化日志解析困难、异常定义困难和检测流程复杂等。为解决上述问题，本文对日志异常的高效检测和应用进行了相关的探索和研究，主要内容包括以下几项工作：

(1) 提出了一种基于启发式规则的流式日志解析方法 HRTree。在 Drain 方法解析结构树的基础上，本文提出了使用启发式规则对日志进行拆分构造，并优化了解析结构树的部分构造方式；从而解决其日志参数过拟合、不同系统日志解析结果不稳定问题。实现了不仅解析结果分类准确，而且参数内容识别准确的解析效果。HRTree 解析方法在不同的系统日志上均展现了 90% 以上的解析准确率。

(2) 提出了一种基于纵向逻辑回归的日志异常检测方案。面对日志数据异常定义困难和数据不平衡问题，本文提出使用纵向联邦学习技术，不同参与方共享数据标签和特征，实现联合模型训练。本文设计了基于 RSA 哈希盲签的隐私集合求交对齐技术，解决纵向联邦中的日志样本对齐问题。同时利用泰勒展开对逻辑回归的损失函数进行改造，通过 Paillier 同态加密进行联合训练中间结果的交换，在不泄露本地原始数据的前提下，实现联合模型的训练。实验证明纵向联邦的逻辑回归方案在日志异常检测任务中，检测结果明显优于仅使用单方特征的检测效果。

(3) 实现了一个可视化日志异常检测应用系统的开发。针对日志异常检测流程复杂和检测应用困难等问题，本文对多种解析方法和检测模型进行了复现。全面构建了日志异常检测的可视化流程，将检测步骤中复杂的控制参数转化为简洁的前端控制，实现对异常检测的可视化自动控制。通过算法复现和检测流程简化，以降低日志异常检测在不同系统日志条件下的应用难度。

关键词：异常检测，日志分析，日志解析，数据挖掘，联邦学习

ABSTRACT

With the rapid development of big data, cloud computing, 5G communication and blockchain, the scale of modern large-scale service systems is rapidly increasing. The stable and reliable operation of large service systems is particularly important, and any abnormal event may cause incalculable losses, so system operation status monitoring is critical. Formatted or unformatted massive log files record the operation status and information of modern complex systems, which is the basis of system anomaly tracing and analysis. Anomaly detection based on log data is a key means of system status monitoring and abnormal response, which can help system maintenance personnel to discover abnormal behaviors and system failures in time and solve problems effectively.

In this paper, the framework of log anomaly detection is reviewed and the technical points are reviewed in detail. This paper aims to comprehensively introduce the existing technical content of log anomaly detection, and provide reference for related researcher. At present, various data mining, machine learning and deep learning algorithms have been widely used in the field of log anomaly detection. However, there are still many problems, such as the difficulty of unstructured log parsing, the difficulty of anomaly definition and the complexity of detection process. In order to solve the above problems, this paper explores and researches the efficient detection and application of log anomaly, including the following main works:

(1) A heuristic rules-based streaming log parsing method HRTree is proposed. Based on the Drain method for constructing parsing trees, this paper proposes using heuristic rules to split and construct logs, and optimizes the partial construction mode of analytic structure tree. In this way, the problems of overfitting log parameters and unstable log parsing results of different systems can be solved. The implementation achieves not only accurate classification of parsing results, but also accurate recognition of parameter content. The HRTree parsing method shows an accuracy more than 90% on different system logs.

(2) A log anomaly detection scheme based on vertical logistic regression is proposed. In the face of the difficulty in defining log data anomalies and data imbalance, this paper proposes the use of vertical federation learning technology, where different participants share data labels and features to achieve joint model training. In this paper, the privacy set inter-

section technology based on RSA hash blind signatures is designed to solve the log sample alignment problem of vertical federation. Meanwhile, the loss function of logistic regression is modified through Taylor expansion, and Paillier homomorphic encryption is used to exchange intermediate results in joint training, achieving joint model training without leaking local original data. Experiments demonstrate that the logistic regression scheme of vertical federation significantly outperforms the detection results in log anomaly detection tasks than those using only unilateral features.

(3) A visual log anomaly detection application system is developed. In response to the complexity of the log anomaly detection process and the difficulty of detecting applications, this paper has reproduced multiple parsing methods and detection models. The visual flow of log anomaly detection is constructed comprehensively, and the complex control parameters in the detection steps are transformed into concise front-end controls to achieve visualized automatic control of anomaly detection. The algorithm is reproduced and the detection process is simplified in order to reduce the difficulty of applying log anomaly detection under different system log conditions.

Keywords：Anomaly Detection, Log Analysis, Log Parsing, Data Mining, Federated Learning

插图索引

表格索引

符号对照表

符号	符号名称
ΔT	日志采样窗口的长度
Δt	日志采样窗口的滑动步长
Seq	日志索引消息序列
X_n	第 n 个行为事件索引
Vet_i	日志计数向量
C_n	第 n 个行为事件发生次数
Mat	日志事件计数矩阵
$word_i$	日志单词词向量
Sem_i	日志模板语义向量
u_i	日志单词语义权重
S_α	异常空间
S_d	正常空间
y_a	日志向量与正常空间的投影
Q_a	误差阈值
h	检测序列滑动窗口长度
k_i	第 i 类日志行为事件
K	所有日志行为事件集合
g	行为模式候选阈值
$seq(i)$	当前序列的第 i 个 token
$symb$	字符 token 列表
t_i	日志消息序列中的第 i 个 token
m	日志序列中的 token 数量
d	解析树的最大深度
c	当前叶子节点的候选日志组数量
e	公钥

d	私钥
$H(x)$	对 x 求哈希
r_i	随机数掩码
θ	逻辑回归参数值
η	学习率
λ	正则化参数

缩略语对照表

缩略语	英文全称	中文对照
HRTree	Heuristic Regex Tree	启发式规则树形解析方法
RAS	Reliability Availability Serviceability Log	可靠性可用性和可维护性日志
LSTM	Long Short Term Memory Network	长短时记忆神经网络
CNN	Convolutional Neural Network	卷积神经网络
GAN	Generative Adversarial Network	生成对抗网络
TF-IDF	Term Frequency-Inverse Document Frequency	词频-逆向文档频率
LR	Logistic Regression	逻辑回归算法
SVM	Support Vector Machine	支持向量机算法
PCA	Principal Component Analysis	主成分分析
IM	Invariants Mining	不变量挖掘算法
Bi-LSTM	Bidirectional-Long Short Term Memory Network	双向长短时记忆神经网络
TP	True Positive	真阳性
FP	False Positive	假阳性
TN	True Negative	真阴性
FN	False Negative	假阴性
PSI	Private Set Intersection	隐私集合求交
TPR	True Positive Rate	真正类率
FPR	False Positive Rate	假正类率
ROC	Receiver Operating Characteristic	受试者工作特征曲线
AUC	Area Under ROC Curve	ROC 曲线下面积

目 录

第一章　绪论

1.1 研究背景及意义

随着大数据时代的到来，现代服务系统的规模和复杂性迅速增加。例如，云计算、电子商务、5G 通讯和区块链等业务的快速发展，导致大型服务系统的构建越来越复杂。一方面，完整的大型服务系统通常由数以千计的计算机同时运行维护；另一方面，大型服务系统可能为数以百万计的用户提供服务。这些服务系统的任何异常行为，都有可能会影响系统可靠性和可用性，轻者影响用户体验，重者造成巨大经济损失[1]。因此，大型系统的可靠性和稳定性至关重要。

系统异常可能由各种原因引起，例如，软件错误、恶意攻击、内存泄漏、磁盘故障或错误配置。因此，维护人员需要仔细监控这些软件系统，以便快速发现异常并及时缓解异常。如何快速而准确地发现和定位系统中的异常行为和故障，已成为服务系统设计和运维的重要挑战。

日志作为是一种存在于系统的通用性资源，由运行在系统上的程序打印产生，是记录系统或者业务运行时的重要数据源。日志数据包含了各种系统活动中的事件和行为，例如用户访问、错误信息、资源利用、操作记录等。这些数据可以被用来监控系统运行时发生的行为事件，例如监控系统的网络情况、反映系统的硬件故障、保护软件安全等。

异常检测作为一种重要的数据挖掘技术，用于发现数据中的异常行为。随着大数据时代的到来，各种系统和应用产生的数据量不断增加，异常检测技术也变得越来越重要，适合应用于故障分析[2]、性能诊断[3]、入侵和欺诈检测[4]等系统安全维护方面。

通过对日志数据进行分析和处理，可以发现系统中的异常情况，检测出系统运行过程中不符合期望的异常行为数据。特别是在大型的分布式系统中，基于日志的异常检测，作为一种主动的防御性技术，可以帮助系统维护人员发现和解决问题，保证系统的稳定可靠运行，避免系统异常造成巨大损失。因此，基于日志的异常检测已成为了一项重要的研究内容。

日志异常检测研究对于保障系统的安全、稳定和高效运行，具有重要意义。一方面，随着网络攻击和黑客行为的不断增加，日志异常检测成为了保障计算机系统安全的重要手段。通过监控日志信息并及时发现异常行为，及时预警并采取措施，加强安全防护和风险控制，可以避免黑客入侵和数据泄露等安全问题。另一方面，日志异常检测可以帮助系统管理员及时发现系统故障和异常情况，减少故障修复时间和成本，提高系统的可靠性和可用性，保障服务质量和用户体验。这对于企业和机构来说，可

以降低系统运维的复杂度和风险，提高运维效率和稳定性。

1.2 研究挑战

对于传统的简单独立系统，维护人员可以通过检阅日志文件中的关键字或者创建一些自定义规则来定位和匹配异常[5-7]。但随着系统规模的扩展，系统日志变的越来越复杂，异常频率增高、种类多样且未知，人工检测耗时且低效，不再适用于这样的数据规模和条件[8]。

随着大数据和人工智能技术的发展，各类自动化算法已经在日志异常检测领域得到了广泛的研究和应用。当前已经出现了很多基于机器学习和深度学习的日志异常检测方法，这些方法通过对日志特征进行模式学习，设定阈值或者部分参数，生成一个检测模型。使用该模型对特征数据进行检测，从而对系统行为做出正常或者异常的判断。这些方法可以自动识别日志中的异常行为，有效地提高了日志异常检测的效率和准确性。但是随着系统规模的扩展和数据量的增大，日志异常检测同样面临新的挑战，主要可以总结为以下几个方面：

(1) 数据规模复杂庞大，日志解析困难；大型系统会产生大量的日志信息，甚至达到每小时吉比特字节的数据[9]。因此如何对系统产生的日志信息进行自动化高速有效的信息提取存在较大困难，特别是面对非结构化的日志数据，高效准确的日志解析技术是有效异常检测的前提。而当前已有解析方法，在面对日志数据多样、类型复杂的系统日志时，无法展现稳定的解析效果。

(2) 异常定义困难；在大型系统中，异常往往是频发且多样的，很难人工的对异常状态提前做出描述，因此需要标签信息的有监督检测算法很难实施，所以更倾向于需要使用半监督和无监督的检测算法自动的分析识别异常。但是在无监督算法的准确性方面，依然存在巨大的挑战。

(3) 日志数据通讯成本高，存在隐私泄露风险；随着分布式系统的发展，传统的数据汇总到主服务器统一训练模式不再适用。一方面，原始数据传输造成较高的通讯成本；另一方面，因为日志数据含有各种敏感信息，如用户身份、操作记录、访问记录等，存在隐私泄露的风险。

(4) 检测流程混乱，算法参数复杂，应用困难；现有的日志异常检测算法大多是基于机器学习和深度学习等方法的，算法的实现需要进行数据预处理、特征提取、模型训练等多个流程，每个流程都需要一定的专业知识和技能才能进行操作，技术门槛高，实现起来较为困难。并且不同算法之间差异较大，需要针对具体场景和数据内容进行算法优化和调整，在实际应用方面依然存在巨大鸿沟。

1.3 研究内容和主要贡献

根据对相关研究背景的介绍，可以发现日志异常任务在系统监控和系统安全维护方面有着重要的意义。然而，在大数据和大规模系统的背景下，目前日志异常检测任务依然存在众多挑战，有许多地方有待进一步完善。

本文对日志异常检测的相关研究内容做了较为全面的调研，根据所涉及技术内容，分别对不同流程技术进行了较为详细的综述，包括其中重要的日志解析技术、特征提取技术和各种算法模型的检测技术。能够帮助非领域专家熟悉日志异常检测相关研究内容及进展，为相关科研工作者提供参考和借鉴。此外，本文还对日志异常检测做了三点创新性的探索和研究，总结如下：

1）针对当前日志解析方法解析结果不稳定、解析准确率有待提高的问题，本文提出了一种基于启发式规则的流式在线日志解析方法 HRTree（Heuristic Regex Tree）。该方法以 Drain 的流式树形解析结构为基础，基于启发式规则对日志中的部分 token 进行有效拆分，实现固定文本和参数的分离；解决特定规则下的日志参数识别不准确问题，避免参数过度拟合模糊化处理。同时优化了解析结构树匹配搜索和节点更新的细节，进一步提高了解析文本的准确性。流式的高效日志解析模式，在不同的系统日志上均展现了较高的解析效果。不仅达到了解析日志事件的分类准确，而且实现了参数内容识别准确。

2）针对日志异常定义困难、数据特征不平衡和部分场景下日志存在数据隐私壁垒等问题，本文设计了一种纵向联邦场景下的日志异常检测方案。以日志异常标签共享为出发点，提出了基于纵向逻辑回归的联邦日志异常检测方案，能够解决当前日志检测异常标签短缺，数据特征不平衡等问题。设计了基于 RSA 哈希盲签的隐私集合求交对齐方案，实现纵向联邦模型日志序列的对齐操作。利用泰勒公式对传统逻辑回归的指数运算进行多项式展开，使用 Paillier 同态加密算法进行中间结果的加密交换计算。在不泄露本地原始数据的基础上，打破数据壁垒，实现特征共享，联合模型训练，从而提高模型异常检测的能力。

3）针对日志异常检测流程复杂、检测模型复现困难等问题，本文实现了一套全流程日志异常检测可视化应用系统的开发。通过对可视化系统的功能分析和模块设计，不仅集成了多种日志解析算法和检测模型的复现成果，而且对日志异常检测的相关流程进行了全面简化构建。将部分流程步骤中复杂的控制参数抽象到前端程序，以可视化的方式进行调控和配置。支持根据系统日志场景的不同，在多种解析算法和检测模型之间进行选择和调配，满足非专业人员的使用和维护，降低日志异常检测的应用难度，提高已有技术的实用性。

1.4 本文组织结构

本文的组织架构总共分为六个章节，每个章节的具体内容概述如下:

第一章：绪论。本章节首先介绍了日志异常检测在大数据时代下的研究背景，以及对于大型服务系统运行和维护的研究意义。然后分析了当前日志异常检测依然存在的相关问题和挑战。之后对本文的主要研究内容和贡献进行了总结。最后介绍了本文的组织结构。

第二章：相关研究现状综述。该章节主要对当前日志异常检测的相关研究现状进行了综述，首先总结了日志检测任务的流程框架，对各个技术内容进行了简介。之后分别对各流程涉及的相关技术内容进行了现状分析，其中主要分析了日志结构化解析方法的研究现状，介绍了多种不同模式的日志解析方法。然后对日志异常检测经常使用的几种特征类型和提取方式进行了介绍。最后重点对日志异常检测的方法模型进行了全面的综述分析，从传统的有监督和无监督机器学习检测方法，到近年来逐步兴起的深度学习方法下的各类神经网络模型，还特别分析了联邦学习方法目前在日志异常检测领域的现状。

第三章：基于启发式规则的流式在线日志解析方法。该章节提出了一种启发式的日志解析方法 HRTree，在介绍 HRTree 之前，首先对日志解析问题的定义和解析目的进行了阐述，解释了日志解析工作在异常检测任务中的重要性。然后介绍了目前日志解析方法存在以及本文解决的几个重要问题。之后对本文提出的解析方法进行了算法和流程的详细介绍，并对相关创新和改进内容进行了解释说明。最后本章对 HRTree 的解析效果，在多个指标性能上进行了全面的评估，通过不同方法在不同数据集上的实验结果对比，证明了 HRTree 的高效率和高准确性的稳定解析效果。

第四章：基于纵向联邦的日志异常检测方案。该章节提出了一种基于纵向逻辑回归的日志异常检测方案。首先分析了纵向联邦学习的特点和在日志异常检测领域的应用场景。之后对纵向逻辑回归的日志异常检测方案进行了详细的介绍，包括其中提出的一种基于 RSA 哈希盲签的隐私集合求交对齐方案，实现检测序列数据对齐；通过改造的传统逻辑回归模型，在不泄露原始数据的基础上，通过同态加密算法实现联合模型训练。最后，在三个数据集上进行了实测，在多个指标上分析了纵向联邦方案日志异常检测的性能。

第五章：可视化应用系统开发及实现。该章节主要对已掌握的日志异常检测相关技术进行了应用开发，通过需求描述，对系统开发的功能需求进行了分析。之后通过检测流程和涉及的技术内容进行了不同功能模块的设计，实现可视化控制。最后对其进行了具体实现和展示。

第六章：总结与展望。该章节总结了全文的主要工作，并对未来下一步的研究方向进行展望。

第二章　相关研究现状综述

基于日志的异常检测已经被广泛应用于系统监控之中，本章节调研了日志异常检测的整个流程，对各个环节涉及的相关技术进行了阐述和分析，特别是对其中的日志解析算法和异常检测模型方法进行了全面的综述。

2.1 日志异常检测框架

通过分析近年来异常检测相关领域的研究现状，基于日志数据的异常检测的模式流程可以概括为以下四个步骤：日志收集与预处理、日志解析、日志特征提取和异常检测[10-11]。可抽象构建为如图 2.1所示的异常检测流程框架。

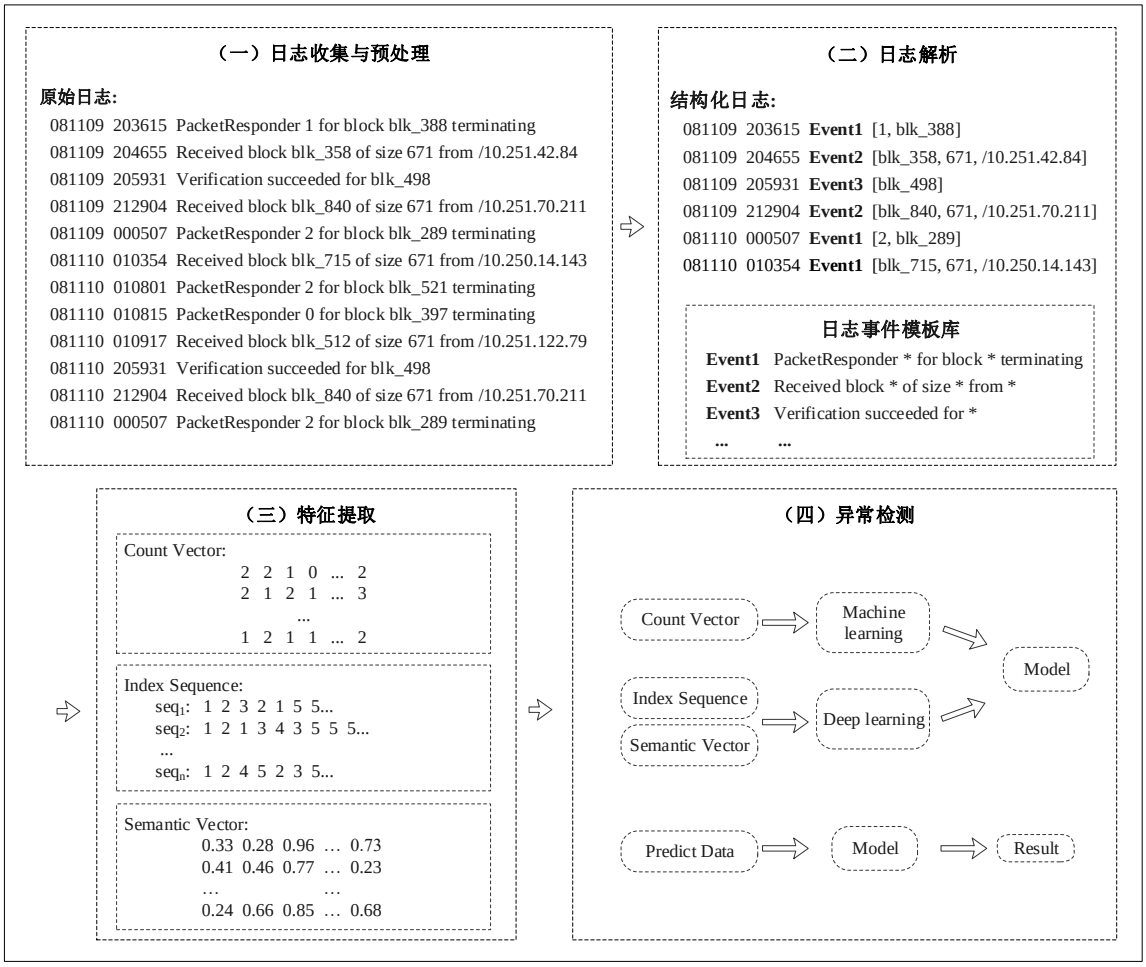


图 2.1　日志异常检测流程框架

日志收集与预处理：系统在运行过程中会产生记录系统运行状态详细信息的原

始日志数据，每条日志数据通常都包含有日志产生的时间、当前进程号、日志等级和描述具体行为的关键事件文本信息等。多数系统通常以文本形式直接保存在本地磁盘，少数系统将其存入数据库进行保存，异常检测通常将文本格式数据收集作为使用数据。在某些情况下，根据模型特征提取的需求，还需要对日志数据进行过滤和清洗，将一些重复和无用的干扰数据进行删除，只保留准确反映系统运行状态的有效数据。例如，Di 等人[12]采用基于加权相似度的时空消息过滤器，对 RAS 日志进行了重复过滤，避免了重复和无用的数据噪声对后续分析和特征提取的干扰，从而提高检测的准确性。

日志解析: 日志信息通常为非结构化的文本信息条目，该类非结构化日志普遍含有丰富的多重信息内容，例如，时间戳、进程号和描述语句等。时间戳和进程号等往往具有固定的格式，可以通过正则化匹配进行解析。而阐述日志行为的描述语句，通常为程序员编写的陈述性静态文本和相关动态参数信息的组合。即使在同一系统中，不同的行为类型和不同的业务模块下，该陈述性静态文本语言风格不尽相同，多数为非结构化的自由文本形式。特别是静态文本与动态参数信息的组合，会造成日志文本的数量类型在全局维度上的爆炸性增长。日志解析任务则通过频繁模式、聚类和启发式等数据挖掘方法，将非结构化日志文本转换为结构化数据，具体为从日志消息中分离参数信息，提取日志行为事件模板和参数。通过分离参数，使用日志行为事件模板可以表示系统的有限类行为，同时准确获取该行为特定参数信息，方便为后续的检测提供特征。

特征提取: 经过日志解析后，时间戳、进程号等标签信息被结构化，日志描述信息被提取为含有特定行为含义的行为事件模板和特定参数的组合。根据系统日志的具体情况，首先可以利用时间戳和进程号等信息进行日志数据的分组采样。例如，利用时间戳信息进行时间维度的数据分组，利用进程号等标签信息进行会话维度的分组等。分组完成后可进行日志描述文本的特征化数字编码，通过统计日志序列中每一类事件发生的次数可以生成日志事件计数向量特征，根据日志事件产生的顺序关系描述系统行为执行路径构建日志索引序列特征，通过解析日志语义可以构建日志语义信息向量特征表示日志事件语义内容。

异常检测: 异常检测主要是使用算法在提取的日志特征上进行模式构建，通过阈值或者部分参数的设定，对已知的训练数据进行模式学习，从而生成检测模型。使该模型可以对未知新数据进行检测，实现对系统行为做出正常或者异常的判定。主要的检测模型有基于传统机器学习的决策树和逻辑回归等有监督的检测模型，以及无监督的主成分分析、聚类、不变量挖掘等；也有基于深度学习的长短时记忆循环神经网络（Long Short Term Memory Network, LSTM）模型对日志序列进行路径学习，以及

与卷积神经网络（Convolutional Neural Network, CNN）和生成对抗网络（Generative Adversarial Network, GAN）等多网络结构结合的复杂模型。检测异常类型包括行为事件数量关系异常、行为序列顺序关系异常和参数值异常等。

2.2 日志结构化解析方法

在工业界，许多大型公司都为日志管理设置了专门的软件工具，他们可支持日志搜索解析和可视化分析等功能，例如 Splunk、Logz.io、LogEntries、Loggly 等，但是他们的日志解析功能只是对常见的日志类型和专业领域进行的应用，例如 Apache 和 Nginx 日志。并且工业的方法需要深度的领域知识介入，本文不对其进行详细讨论，只对学术领域相关研究内容进行介绍。

如上文所述，日志结构化解析主要将非结构化的自由文本内容解析为具有固定结构信息的静态文本和动态参数信息的组合，以有限状态的行为事件模板描述系统行为类型。传统的方法依赖于手工正则表达式提取，这种方法是一种耗时并且无法适应不同日志系统的低效方法。一方面人工编写规则无法适应大数据背景下的海量日志；另一方面手动编写无法适应不断变化更新的日志语句。在一些研究中探索了从源代码直接提取日志事件模板的分析方法[13-14]，通过对源代码输出语句进行解析来提取日志事件模板信息，这种方法有较高的准确性，但是访问源代码在多数情况下并不可行。所以近年来的研究主要聚焦在对已生成的日志文本进行自动化解析方向的探索，通过使用数据挖掘算法对日志信息进行自动解析，减少人为参与的同时提高解析的准确性。

日志信息中的时间戳、进程号和日志等级等是具有特定格式的规则化内容，通常具有相同的文本长度或者由间隔符区分。只需要对其定义一次统一的正则匹配即可对整个日志数据的该部分进行结构化划分。日志解析的关键对象是具有高度自由化文本的日志行为描述信息。通过对日志解析方法的调研研究，本文将目前的解法方法总结为以下几类：基于频繁模式挖掘的解析方法、基于聚类的解析方法、基于启发式的解析方法和其他的基于最长公共子序列等算法的解析方法。

2.2.1 基于频繁模式挖掘的解析方法

频繁模式是指数据集中频繁出现的一组项目。在日志数据中，日志事件模板可以被视为频繁出现的一组常量标记。因此，频繁模式挖掘是自动化日志解析的一种简单方法。

SLCT[15]是最早的基于频繁模式进行日志聚类的算法，SLCT 首次遍历数据集进行频繁数据的汇总，根据用户指定的支持阈值，挖掘出高于阈值的频繁单词，构建频繁项集。频繁项集由单词和其所在语句中的位置构成，包括（位置，单词）。然后再

次遍历数据集，选出每一行中在上述频繁项集中出现的单词集合，构成候选簇。最后从候选簇中选择支持值大于等于支持阈值的簇作为日志事件模板。

由于支持阈值的存在，SLCT 解析会忽略一些低于支持阈值的事件，这些事件重复了几次但是仍然小于阈值无法被识别，所以 SLCT 无法寻找到所有可能的模式。LFA[16]方法对每行日志消息都进行频率的遍历筛选，将每行中聚类相似的频率词抽象为事件类型，因此 LFA 可以解析所有的日志事件模式。

SLCT 和 LFA 由于都对其单词的位置进行了编码统计，所以对位置信息敏感，在有些日志系统中由于参数部分可能由一个或者多个单词构成的，在这种情况下，一些频繁单词的位置发生偏移，算法就会将该类型日志事件模板识别为多个簇。LogCluster[17]解析方法对单词位置不敏感，从日志事件中挖掘线性模式，可以很好地适应单词位置的变化。

Logram[18]使用 n 元语法（n-gram）对日志进行采样，利用字典来存储日志中的 n-gram 频率，Logram 认为频繁的 n-gram 更有可能是事件模板，而罕见的 n-gram 更可能是参数，实现高效的日志解析。Logram 相比于以上三种方法不同的是可以实现日志的在线解析，不需要提前遍历所有日志数据。

2.2.2 基于聚类的解析方法

基于聚类的解析方法主要是指将日志信息按照某种特质，使用传统的聚类算法进行相同类型日志事件模式的提取。

LKE[19]解析方法首先基于经验规则删除明显的参数值内容，例如 IP 地址、数字、Url 等。然后通过日志消息之间的加权编辑距离来度量原始日志消息的相似性，通过比较阈值将相似的原始日志聚集在一起。对同一位置的单词数量小于设定阈值的聚类进行分割，重复执行每个位置的分割过程，直到没有满足条件的组为止。

基于消息签名的 LogSig[20]解析算法将日志消息转化成单词对组成的二元组集合，首先进行随机分组，然后对其进行编码，为每一日志消息计算一个签名值。根据签名值与不同组的相似度，寻找到可能性最大的组划入该组。经过如此多轮迭代比较，日志签名划分稳定后，从每一组集群中提取日志事件模板。

LogMine[21]是一种层次聚类方法，首先对日志数据进行标记化和类型的检测清洗，之后通过计算日志之间单词的相似度距离，快速的对日志进行集群，然后通过模式识别从集群中生成一系列模式集合作为叶级。然后继续迭代在模式集合中合并相似模式构成父级，以从下到上的方式形成日志事件模板的层级结构。

SHISO[22]是一种在线的解析方法，通过统计单词中包含的字符类型（大小写字母、数字和符号等）来计算单词的向量信息，之后通过比较不同日志之间单词的欧氏距离来进行日志聚类。通过搜索和调整两个阶段进行流式解析，搜索阶段以一种树状

结构为新读入的日志消息查找合适的日志模式。调整阶段在出现新的日志模式时创建或更新已有的日志事件模板的格式。

与 SHISO 相似，LenMa[23]通过统计日志语句中每个单词的长度构建字长向量，每条语句中单词数量构建字数向量，以此特征进行日志聚类。新读入的日志通过计算与相同字数的每个簇之间的相似度来合并或者创建新的日志聚类，最后从每个簇中提取日志模板信息。LenMa 同样是一种在线的日志解析方式。

2.2.3 基于启发式的解析方法

对于日志信息往往具有一些鲜明的启发式特征，通过利用此类特征进行解析的方法称为基于启发式的解析算法。

AEL[24]是一种用于识别日志消息中参数的启发式方法。日志语句中往往存在这样的赋值对“word=value”和像“is[are|was|were] value”这样的短语，对上面的 value 内容进行匿名化处理，定义为参数值，使用通配符替代。然后统计每条日志中单词和通配符的数量，将具有相同数量单词和通配符的日志分为同一组。之后再对同一组内的日志消息进行调整合并处理。

IPLoM[25]是基于启发式的迭代分区聚类算法，IPLoM 基于以下两个启发式日志特点：I. 具有相同行为描述的日志消息，往往具有相同的日志长度。II. 基于日志事件的位置信息，不同单词数量最少的地方往往是静态部分。基于 I 特点，IPLoM 首先通过统计日志消息的单词数量，按照数量划分为不同的集群。然后基于启发式的 II 特点，统计相同集群中每个位置出现不同单词的数量，出现单词数量最少的位置最有可能为静态部分，然后根据该位置的唯一值进一步分割每个分区。这样每个分区的长度相同且有一位置具有唯一值。之后根据搜索日志不同位置单词的映射关系，进一步将其进行分区。最后对每个集群进行描述，标记位置唯一的被认为是常数值，含有多个单词的位置被认为变量值，使用通配符代替。IPLoM 并不只是寻找频繁的文本模式，而是寻找所有可能的模式。

Drain 算法[26]是一种基于固定深度解析树的高效在线日志解析方法。同样基于以上两种方法的启发式特征，首先基于领域知识，对日志数据中的特定变量使用正则表达式进行通配符的替代，例如 IP 地址和 Url 等，尽可能的将日志解析的干扰参数减少降低。之后同样基于 IPLoM 中相同日志事件类型往往具有相同的日志长度这一特征，统计日志的单词数量，基于日志长度建立不同的树节点。如图 2.2所示，之后在每个长度节点分支下根据日志语句中的单词，依次向下建立子节点。通过设定深度阈值，当节点深度超过阈值后不再继续创建，使其整个树结构保持相同的深度结构，此时叶子结点为日志事件类型的集合。当新的日志出现时，首先计算长度，然后根据单词依次找到叶子节点，当不存在分支节点时，可以创建新的分支。当到达叶子节点时，通

过计算相似度，寻找集合中最相似的日志事件，通过阈值的比较，满足相似度阈值的日志事件得到保存，同时对其中不一样的单词使用通配符替代，进行模糊化处理。不满足阈值则保留该模式进入集合作为新的日志模式。此方法可以保证所有的日志模式都可以识别到。但是由于 Drain 的单词切分过于粗糙，在部分系统日志场景下会出现静态文本被过度拟合成变量的问题。

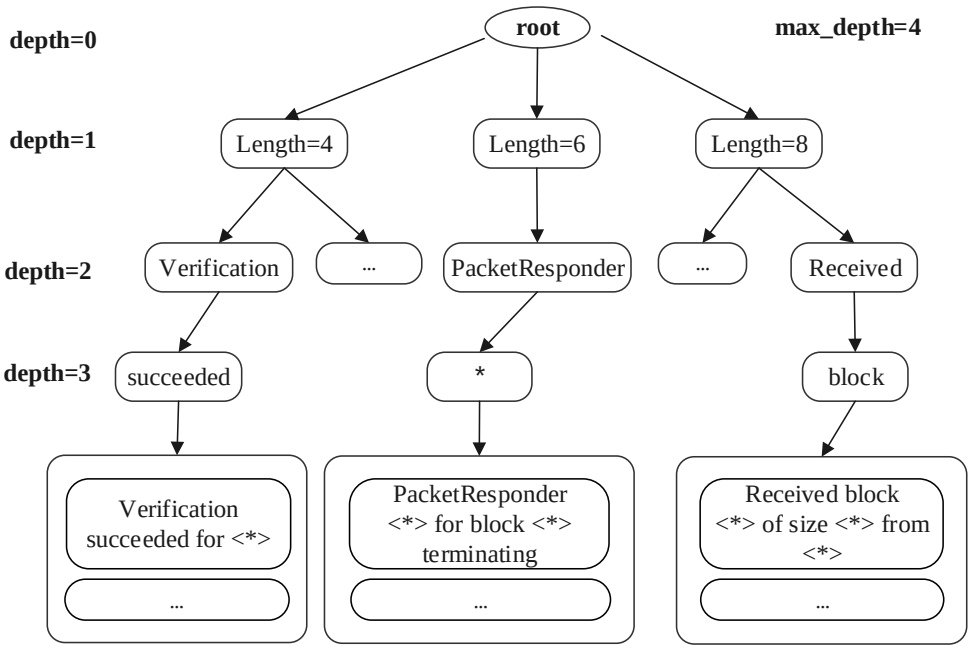


图 2.2 Drain 算法中解析树结构[26]

2.2.4 其他解析方法

Spell[27]是一种基于最长公共子序列来解析系统日志的方法，通过将日志信息拆分成单词序列，然后计算公共子序列来构建日志事件模式。该方法同样使用流式在线解析过程，每条日志输入后，通过解析成单词序列，计算该日志单词序列和已存在的事件的最长公共子序列。从已知事件中，选择公共子序列最大的作为候选模式，通过计算公共子序列与自身长度的比值作为相似度。之后与提前设定的阈值比较，一般情况下拥有自身信息一半以上相似的日志可以划分到同一个集合内。将新的日志和已存在日志归为一类，否则将该日志定义为新的日志模式。对于可以划分为同类的日志，使用通配符对不同部分模糊化，然后更新该事件。同时为减少新的日志序列在寻找最长公共子序列时与已知日志事件模式之间的频繁计算，Spell 设计了前缀树结构来减少计算的次数，从而提高了解析效率。

日志解析通过使用通配符替代参数部分，尽可能多的将参数信息进行模糊化处理，但是可能存在过度模糊的问题，将一些原本的描述性静态文本也进行了模糊化处

理，导致日志模板对行为事件的描述失去特定性，因此存在既要最大化的将参数模糊化又要最大化的保存行为事件特异性的冲突。Messaoudi 等人[28]将上述问题定义为生成具有高频率（匹配尽可能多的日志条目）和高特异性（每个日志事件都代表特定行为）日志事件模板的多目标优化问题，并基于 NSGA-II[29]提出了解决方法 MoLFI，一种基于多目标遗传算法和权衡分析的搜索方法。

LogParse[30]将日志模板提取转化为单词分类问题。首先从历史日志中提取模板，根据历史日志信息来对模板单词和可变词制定标签，使用单词的字符级计数向量特征，利用支持向量机算法构建单词分类器。一方面匹配现有模板，另一方面，通过分类器区分未知模板单词和变量单词，来学习新的日志模板，实时将日志增量更新到为模板集构建的前缀树中。

2.3 日志特征提取方法

日志特征提取主要指从解析得到的日志信息中提取有价值的数字特征。然后将该类特征馈送到相应的检测模型中。由于日志数量庞大，通常首先需要对日志数据进行分组采样，将日志数据以组的形式描述系统的一段行为。之后根据检测模型原理的不同，提取相应的以下几类日志特征：日志事件索引序列特征，日志事件计数矩阵特征，日志事件语义信息向量特征。

2.3.1 数据分组

数据分组将整个日志数据采样分割成块，每一块即系统在一段时间内行为的集合，日志检测即检测该连续行为集合是否发生异常。分组的类型主要有三种，分别是固定窗口、滑动窗口和会话窗口。

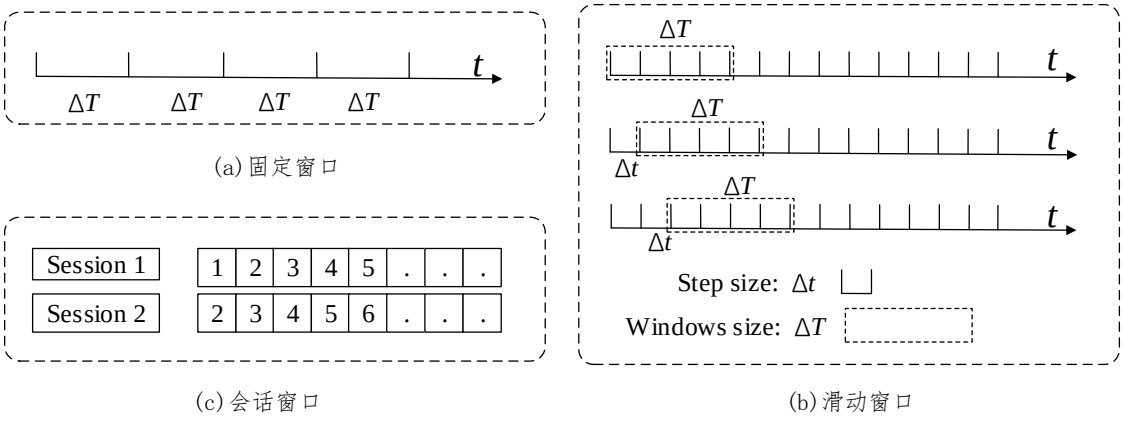


图 2.3　数据分组方式

固定窗口分组：固定窗口就是将整个日志数据，按照时间戳排序后，通过定义检测窗口的时间跨度，将日志切分成时间跨度相同的序列，如图 2.3(a) 所示，定义窗口大小为 ΔT 的固定值。一个 ΔT 时间段包含用户的一组行为序列，每组行为序列长度不定，具体由系统运行过程中产生日志的速率决定。对于产生日志较频繁的大型系统，可以设置拥有较短时间的窗口长度，具体可根据日志产生速率和模型检测结果对窗口长度做具体参数的调整。

滑动窗口分组：如图 2.3(b) 所示，滑动窗口在固定窗口的基础上，增加移动步长变量 Δt，将日志数据按照时间戳信息，通过滑动窗口和移动步长的组合来提取用户行为序列。当步长和滑动窗口长度相同时可理解为固定窗口，一般步长要比滑动窗口的长度要短。因此两个窗口的分割会有日志数据的重叠，以此避免对某些事件的遗漏，准确的掌握系统的连续行为。

会话窗口分组：会话窗口分组不同于上面两种基于时间进行分组的方式，如图 2.3(c) 所示，会话窗口基于日志的特定属性标签进行分组，例如：用户名、系统进程编号等。通过扫描一段时间内具有特定标签的所有行为，将其归为一组行为序列。每个组别代表一类标签下的行为。当然会话窗口也可以和上面两种方式结合，构建基于时间窗口下的会话窗口序列，即以某一段时间窗口为跨度，采集该时间内不同标签下的会话行为序列。

2.3.2 日志索引序列特征

当日志被解析成日志事件时，每一条日志语句代表了一类特定的行为。通过对日志事件进行索引编号，将每一组日志行为序列表示成由索引编号组成的日志序列特征。$Seq_i = [X_1, X_2, X_3, X_4, ..., X_n]$，$Seq_i$ 表示第 i 组日志行为序列；X_n 表示第 n 个日志行为事件的索引，序列的长度 n 为变量，表示每组序列日志行为的个数。X 按照序列的时间顺序依次排列，这样就可以构造成一条日志序列的索引特征。

该特征主要用到循环神经网络中，循环神经网络通过对历史事件的学习，对序列中可能出现的下一日志行为进行预测评估，从而做出异常与否的判断。

2.3.3 日志事件计数向量特征

同日志索引序列一样，日志事件计数特征的提取，在日志索引序列的基础上，通过统计方法，计算整条日志序列中每一种日志行为出现的次数，构建日志计数向量 $Vet_i = [C_1, C_2, C_3, C_4, ..., C_n]$，其中 Vet_i 表示第 i 组日志行为序列的计数向量；C_n 表示第 n 个行为在该组内出现的次数；此处 n 为整个日志中出现的所有日志事件的总数。因此该类特征维度相对固定，适合表示行为类型稳定的日志数据，对于经常更新的日志系统，出现新日志行为时需要对其进行维度的扩充，通常需要进行模型

的再训练。将所有的日志分组的向量特征进行合并后，可以组成日志事件计数矩阵 $Mat = [Vet_1, Vet_2, Vet_3, ..., Vet_m]^T$，*m* 表示日志组的总数目。

该类特征主要记录了不同事件行为之间发生次数的关联，主要在传统机器学习方法中使用，通过对日志行为出现次数的分析对系统行为进行检测。

2.3.4 日志语义信息向量特征

以上两种特征提取方法主要集中在日志事件层面，而语义信息特征提取则在日志信息的单词层面，通过自然语言处理技术，使用单词向量集合表示整个日志语句。通过构建日志单词的语料库，使用 Word2Vec[31]、FastText[32]等方法，将日志单词转换成词向量特征 $word_i = [v_1, v_2, v_3, v_4, ..., v_n]$，然后通过加权词向量构建日志模板信息的语义特征 $Sem_i = \sum(u_i * word_i)$，$Sem_i$ 表示一条日志语句的语义信息，$word_i$ 表示语义库训练的单词的词向量，可以通过词频逆文档频率 TF-IDF（Term Frequency-Inverse Document Frequency）等技术计算每个单词的重要程度权值 u_i。

该类特征主要在一些为语义特征设计的检测模型中，例如 LogAnomaly 检测模型[33]、LogRobust 检测模型[34]和 HitAnomaly 检测模型[35]等。

2.4 日志异常检测模型

目前基于日志事件模式的异常检测技术主要聚焦在一些传统的机器学习方法和深度学习方法的应用，其中包括监督学习、半监督学习和无监督学习。深度学习方法又涉及各种神经网络，包括长短时记忆循环神经网络 LSTM，卷积神经网络 CNN，生成对抗神经网络 GAN 和语义模型 Transformer 等。同时最近部分研究也对联邦学习在日志异常检测领域进行了探索。

2.4.1 机器学习模型

多种机器学习算法首先被应用于日志异常检测中，如决策树、逻辑回归、支持向量机、不变量挖掘等。He 等人整理的一份实验报告[36]对部分机器学习方法构建的日志异常检测模型进行了综述研究。根据训练数据是否被标记，它可以分为有监督检测和无监督检测。

监督学习方法是对有标签的训练数据进行行为模式学习的方法，这需要人工或者基于一定规则对分组后的日志序列打上对应的标签，多数为正常或者异常的二分类标签，数据越可靠，模型就越精准。检测方法也多为分类方法，将含有日志事件计数的矩阵信息和对应的标签读入模型进行训练学习。

逻辑回归模型 (Logistic Regression, LR) 利用 *logistic* 函数计算可能状态（正常或异常）的概率值，通过使用每个日志分组构建的事件计数向量和对应的标签训练 *logistic*

回归模型。在测试时，对新输入的分组计数向量，回归函数给出异常的可能值 $p(Vet_i)$，当 $p(Vet_i) \geq 0.5$ 时，那么这段分组序列行为就被检测为异常，否则即为正常。

决策树模型 (DecisionTree)[37]基于树形结构的归纳学习方法，在无序的训练样本中，通过计算最佳属性，提炼出树形的分类模型。使用分支结构来说明每一组日志的预测状态，每个节点表示某一日志事件行为出现的次数特征，从根节点到叶子节点形成所有事件类型的分类路径，最后每个叶子节点代表了一种异常或者正常的状态。决策树模型的构建复杂度与层数相关，数据处理效率很高，是一种高效的检测模型，同时决策树模型的分支结构特点，具有可以对异常定位分析的优点。

支持向量机 (Support Vector Machine, SVM)[38]也被应用到日志异常检测中，主要使用线性支持向量机，通过构造超平面来分离高维空间中的各类实例，来分离异常和正常数据，如果实例位于超平面之上，则标记为异常，否则为正常数据。

监督学习需要对数据进行标记，学习已知异常和正常数据的特征。但是对于异常检测任务，往往是没有标记的发掘类任务，检测人员无法对异常提前做出定义，因此需要检测算法通过自动分析，挖掘出偏离正常的离群异常行为。因此相较于有监督学习，异常检测任务更需要无监督或者半监督的检测算法。

主成分分析模型 (Principal Component Analysis, PCA) 是 Xu 等人[13]提出的一种无监督的检测算法，Xu 等人首先基于源代码分析恢复日志的固有结构进行日志解析；然后构建日志事件计数向量，同时增加了不同状态比率计数向量这一特征，该特征聚焦于信息类型中出现的术语频率。能够捕捉系统在一个时间窗口内的聚合行为。PCA 算法通过选择一组坐标（主要成分）来捕捉高维数据中的模式，以相对低维的 k 个主成分来保留高维数据中的主要特征。由于日志消息之间的强相关性，所以特征空间的维度之间也是高度相关的，Xu 等人构建的 PCA 模型旨在识别偏离这种相关模式的异常向量。通过 PCA 将日志事件计数向量计算生成两个子空间，异常空间 S_α 和正常空间 S_d。通过计算要检测的日志向量与正常空间 S_d 的投影 y_a 来检测是否异常 $y_a = (1 - PP^T)y$，其中 $P = [v_1, v_2, ..., v_k]$ 是由 PCA 算法选择的前 k 个主成分向量。通过比较平方计算误差 SPE 与阈值 Q_a 的大小判定异常与否。如果 $SPE = ||y_a||^2 > Q_a$，则被检测为异常，否则为正常。

聚类检测模型 (Clustering) 是 Lin 等人[8]提出的一种对日志进行聚类以简化基于日志的异常识别的方法。Clustering 的实施可以分为两个阶段，构建阶段和生产阶段。在构建阶段，使用从训练数据中收集的日志序列进行聚类以构建初始知识库。使用矢量化的日志计数向量，在此基础上利用逆文档频率和归一化处理计数向量。然后使用层次聚类对正常数据和异常数据进行聚类，生成两组向量聚类即正常聚类和异常聚类作为知识库。然后从每个聚类中选取一个代表向量。生产阶段，计算日志计数向量和代表向量之间的距离，如果距离达到阈值，则被添加到最近的集群中，这个集群的

代表向量被更新。否则，创建一个新的集群。向量日志聚类通过初始知识库来检查日志序列以前是否发生过，只需要从日志聚类中提取少量的以前从未见过的代表性日志序列来识别问题，从而减少了应检查日志的数量。检测时通过计算被检测向量与代表向量的距离，如果最小距离大于一个阈值则直接报警为异常，若小于阈值，则根据最近的集群的类别判定为异常或者正常。

不变量挖掘检测模型 (Invariants Mining, IM) 是 Lou 等人[39]基于不变量挖掘算法提出的异常检测方法。不变量挖掘可以揭示代表系统正常执行行为的多个日志事件之间的线性关系，例如，对于 "$open\ \ file$" 和 "$close\ \ file$" 通常是成对出现的。不变量挖掘就是通过找出日志计数向量特征中这种线性关系，构建不变量模式。首先通过奇异值分解估计不变量空间，确定之后要挖掘的不变量的数量。之后通过蛮力搜索找出不变量。最后通过支持度阈值验证挖掘的不变量。当新的日志计数向量特征输入到模型时，检测各行为之间的线性关系是否满足模型定义的不变量，若不满足则检测为异常，由于该模型需要挖掘各行为特征之间的关系，所以相较于前面所述机器学习方法，要花费更多的训练时间。

2.4.2 深度学习模型

深度学习方法主要集中在使用各种神经网络构建日志异常检测的模型，由于日志数据的时序性特点，目前在日志异常检测的任务中循环神经网络使用最多。同时为提高检测的效率，部分检测方法引入卷积神经网络进行模型加速，从而提高模型训练和预测速度。在解决日志异常标签短缺问题上，部分模型方法提出使用生成对抗网络训练模型，解决异常数据和正常数据不平衡问题，但其训练难度较大。

（1）长短时记忆循环神经网络

DeepLog： Du 等人[40]使用长短时记忆循环神经网络 LSTM，将系统日志建模为自然语言序列，提出了 DeepLog 检测模型。长短时记忆循环神经网络能够记住序列的长期依赖关系，在机器翻译[41]、情感分析[42]、医疗自我诊断[43]中取得了有效的成果。DeepLog 基于执行路径的检测，为每个任务构建工作流模型，从正常执行序列中自动学习日志模型，捕获日志条目之间潜在的非线性和高维的依赖关系，并通过该模型对偏离正常执行路径的日志数据进行异常检测。

日志信息通过解析和特征提取技术处理后构成一条索引序列 $Seq_i = [X_1, X_2, X_3, X_4, ..., X_n]$，显然 X_i 与之前的出现的日志序列存在依赖关系，DeepLog 通过建立模型挖掘 X_i 与历史行为的依赖关系，将异常检测建模为多分类问题，通过设定检测窗口的长度 h，对日志序列进行截取划分，一条日志序列被划分成形如 $[X_1, ..., X_{h-1}, X_h \rightarrow X_{h+1}]...[X_{i-h}, ..., X_{i-2}, X_{i-1} \rightarrow X_i]$ 这样的日志窗口集合。将历

史日志序列作为模型的输入，之后的下一行为作为模型判定结果。

训练时输入序列为 $[X_{i-h},\ldots,X_{i-2},X_{i-1}]$，输出为对应行为索引的独热编码 one-hot 向量，通过梯度下降使交叉熵损失最小化来学习网络中的权值参数。在检测时，模型同样对待检测的序列进行行为与历史行为序列的截取划分，输入历史序列进入模型，输出为可能的所有日志事件行为的概率 $Pr[X_i=k_i|w]$，其中 $k_i\in K$，K 为所有日志事件的集合。然后对概率进行排序，因为有些日志行为序列对应的行为模式可能不止一种，所以通过设置候选阈值 g，选取前 g 个概率最高的行为作为模型检测可接受的下一行为的预测。然后比较待检测的行为是否在该检测结果阈值之内，若在阈值之内则判定为正常，继续滑动检测下一窗口，直到检测结束。倘若在检测过程中，某一行为不在历史行为检测结果的可接受阈值内，这直接判定整条日志序列为异常。

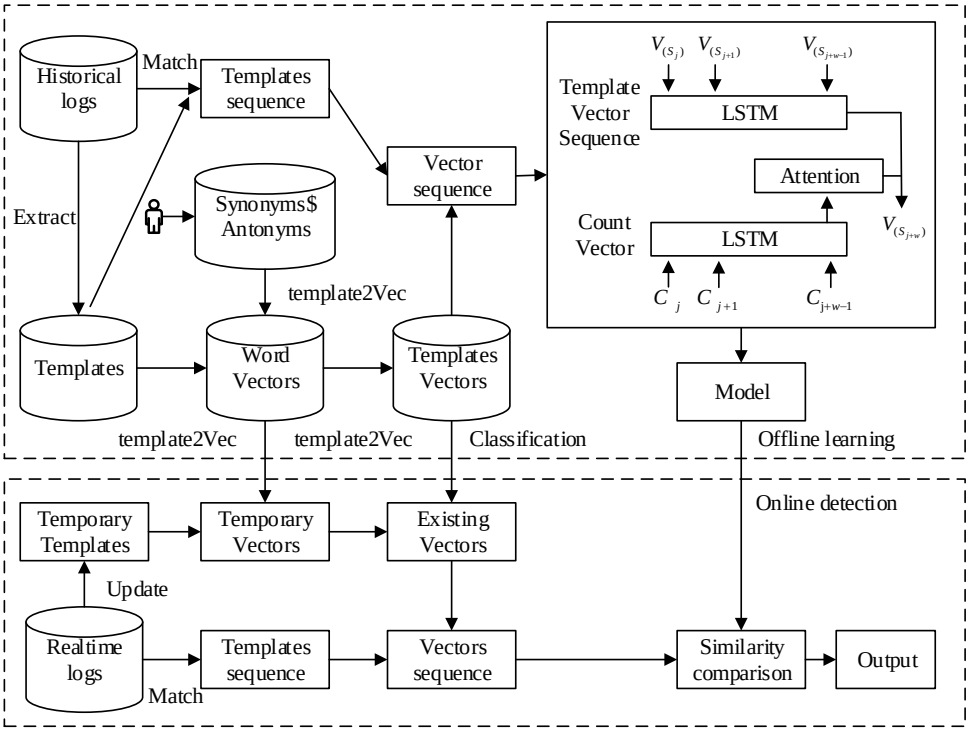


图 2.4 LogAnomaly 框架[33]

LogAnomaly：受到词嵌入 Word2Vec[31]的启发，Meng 等人[33]提出了 LogAnomaly 检测模型，从日志事件模板中提取语义信息向量，这是第一个考虑日志语义信息的日志异常检测模型。

Word2Vec 是一种常用的分布式单词表示方法，能够捕获单词的上下文特征，但是不能捕获同义词和反义词等语义信息。而日志模板信息通常表现为相同的语义信息或者相反的语义信息，这种特征在日志文本中较为显著。Meng 等人提出 Template2Vec，一种基于同义词和反义词的单词表示方法，通过构建同义词集和反义词集，将日志单词转换成语义向量，准确的表示不同日志模板之间的相同语义或者相反语义。

LogAnomaly 模型同时使用语义向量的序列特征和日志计数向量特征，是一个将两种特征同时应用到异常检测的模型，框架结构如图 2.4，既可以通过序列的上下文语义关系捕捉异常，又可以通过行为之间数量关系捕获异常。

LogRobust：随着时间的推移，日志系统中经常包含以前未见的日志事件或日志序列，其中日志语句的演化和日志数据中的噪声导致了日志数据的不稳定性。Zhang 等人[34]提出了一种新的基于日志的异常检测方法 LogRobust。将语义向量作为模型输入，然后利用基于注意力的双向 Bi-LSTM(Bidirectional-LSTM) 模型检测异常，该模型能够捕获日志序列中的上下文信息，并自动学习不同日志事件的重要性。通过这种方式，LogRobust 能够识别和处理不稳定的日志事件和序列。

LogRobust 采用 Drain 方法解析日志，提取日志中的参数和日志事件信息构建日志事件模板，然后将每一个日志事件转换为固定维度的语义向量。经过三个步骤进行矢量化：(1) 日志事件预处理；提取日志事件的语义单词。(2) 单词向量化；通过使用 FastText 语料库[32]对每个单词进行向量的映射。(3) 基于 TF-IDF 的聚合；通过词频逆文档频率加权每个单词的向量，获得固定维度的语义信息的特征。通过语义矢量化，每个日志事件就被转化为语义矢量，这样语义向量就可以识别语义相似的向量，同时区分不同的日志向量。双向的 Bi-LSTM 网络获取两个方向的日志预测结果，LogRobust 引入注意力机制，为日志事件分配不同的权重，同时降低日志数据中噪声的影响。

（2）卷积神经网络

LogCNN： Lu 等人[44]提出了一种使用卷积神经网络 CNN 建模日志异常检测的方法。日志事件索引使用一种 Logkey2vec 的编码方式，将索引映射编码为向量特征，日志索引序列被编码为序列向量矩阵。首先使用带有三个不同大小卷积核的一维卷积层并行提取特征，然后经过最大池化层过滤比较弱的相关特征，将较强的特征输入到全连接层，最后通过 softmax 函数以产生概率分布结果。由于 CNN 的特点，该模型相比于 LSTM 的模型具有更快的检测效率。Lu 等人仅对 HDFS 数据集进行了测试，在该稳定数据集上展现了比较不错的有监督检测效果。

CausalConvLSTM： Yen 等人[45]提出一种采用和 DeepLog 相同的序列建模方式 CausalConvLSTM 模型，基于历史行为序列对下一行为事件进行预测。在 DeepLog 的基础上设计了一种卷积神经网络 CNN 和循环神经网络 LSTM 相结合的混合网络架构 CNN-LSTM，通过使用卷积神经网络来提取空间特征，然后输出到循环神经网络学习序列中的顺序关系，最后输出到全连接层进行预测。

CNN 具有以并行的方式高效提取空间特征的能力，CausalConvLSTM 使用大小不同的卷积核来提取序列不同跨度上的特征。因为卷积核只沿序列一个方向滑动，所

以使用一维卷积。将卷积之后的结果依然按照顺序排列，保存日志序列的时间顺序关系。从 CNN 接收到特征后，LSTM 网络生成一个内部隐藏状态，传送到一个全连接层，通过 softmax 函数生成所有日志事件的概率分布。因此，CausalConvLSTM 的训练和检测思路同 DeepLog 一样，同样对日志序列进行滑动窗口训练。该模型达到了与仅使用 LSTM 模型相当的精度，但由于只使用了一层 LSTM，所以减少了训练的时间，提高了训练效率。

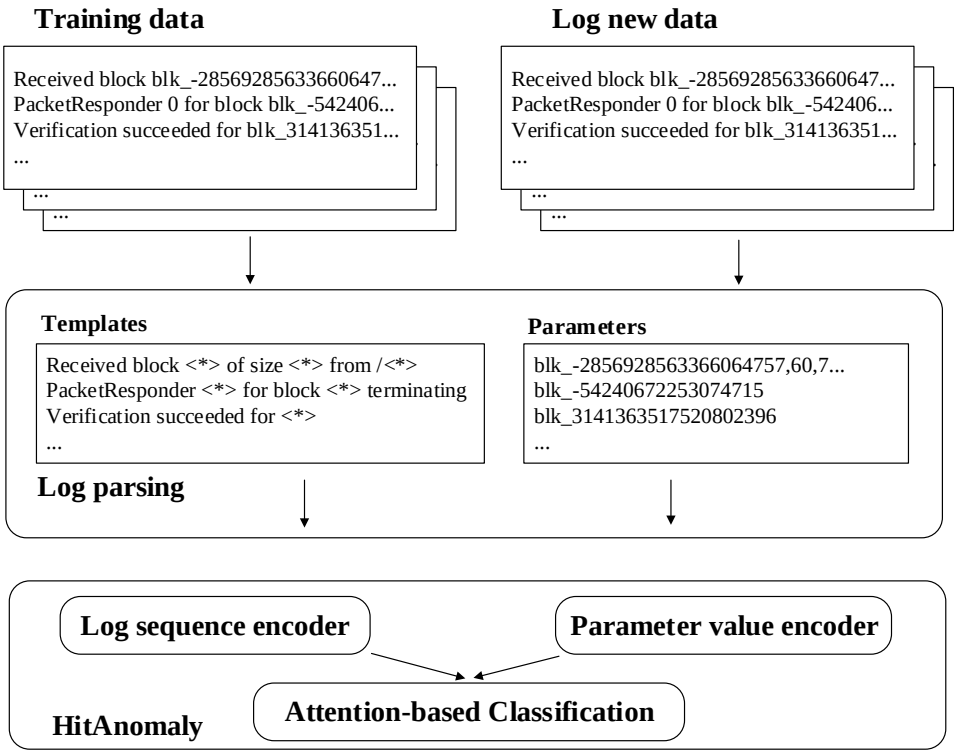


图 2.5　HitAnomaly 概述[35]

（3）Transformer

HitAnomaly：在最近的研究中，Vaswani 等人[46]提出一种自然语言处理的 Transformer 模型结构，拥有 Encoder 和 Decoder 两个部件。通过比较，Transformer 相对于循环神经网络 LSTM 在一些自然语言处理任务中拥有更好的表现。所以 Huang 等人[35]将 Transformer 引入到日志异常检测任务中。

由于之前的研究集中在日志事件模板上，有些将其进行了日志索引编码，有些对其进行了语义编码，但忽略的了其中参数值的信息，研究发现有些异常表现为参数值的异常。Huang 等人关注到了这一点，提出了一个分层转换器结构的异常检测模型 HitAnomaly，设计了一个日志序列编码器和一个参数值编码器，同时对日志语义序列和参数值编码，来获得不同特征对应的表示。如图 2.5，HitAnomaly 分别进行日志序

列信息和日志参数的编码，由于日志模板序列和参数值对结果有不同的影响，还通过注意力机制学习不同的权重参数，以此构建最终的分类模型。

（4）图神经网络

Log2vec： Liu 等人提出了 log2vec[47]，这是一种基于异构图嵌入的检测方法。日志条目根据它们之间不同关系的启发式规则被构造为异构图，然后通过图嵌入将异构图中的节点转换为低维向量，最后，通过阈值判断识别向量的恶意聚类。与图相关的内容在日志异常检测中使用不多，但在其他领域有很多研究，如欺诈检测和社会事件检测。这是解决日志复杂关系的一个值得研究的方向。

（5）生成对抗神经网络

生成对抗网络将机器学习问题视为两个模型生成器 (Generator) 和鉴别器 (Discriminator) 之间的博弈。生成器捕获真实样本的分布，并生成在特征表示方面与真实样本相似的样本。鉴别器则识别将要到来的样本是真实的还是生成的，以提高生成器生成样本的质量。通过不断博弈训练，提高生成器的性能。

LogGAN：在实际的系统应用中，异常是复杂且多样化的，除了预期的恶意攻击之外，未知的异常和错误是人工很难掌握的，因此异常检测具有严峻的异常定义挑战。Xia 等人[48]提出了一种基于序列的生成对抗网络 LogGAN，通过生成对抗式学习产生“真实”异常，以缓解正常和异常实例之间的不平衡问题，同时提高异常检测的性能。

GAN-EDC： Duan 等人[49]还设计了一种称为 GAN-EDC 的生成对抗模型。与 LogGAN 生成器和鉴别器都采用 LSTM 不同，GAN-EDC 的生成器使用 LSTM 网络结构编码，而鉴别器使用 CNN 网络结构识别生成的日志模板和实际模板之间的差异。在异常检测阶段通过计算欧几里得距离来计算异常的概率。

2.4.3 联邦学习模型

联邦学习于 2016 年首先被谷歌在解决安卓手机用户本地更新模型的问题中提出[50]。与机器学习和深度学习不同，联邦学习概念多是指一种分布式学习框架，主要解决模型训练过程中的数据隐私问题。其核心思想是多个拥有数据源的参与方，在不需要交换本地原始数据的基础上，通过交换中间结果或参数的方式，实现联合模型的构建，从而达到隐私保护和数据共享的计算平衡。根据参与各方数据源分布的情况不同，联邦学习可以分为三类：横向联邦学习、纵向联邦学习和联邦迁移学习[51]。横向联邦是指参与方的数据集特征是相同的，但是数据来源于不同的设备或者用户。纵向联邦是指参与方的数据集特征不同，但是数据来源于相同的设备或者用户。迁移

学习则是数据特征和数据来源均不同，但是模型任务和目标相似。

联邦学习的以上特点可以在实际应用中解决诸多问题，但是本文通过调研发现，当前联邦学习在日志异常检测方向的实际应用并不多，仅有个别研究探索了横向联邦和迁移学习在日志异常检测的实施方案，并且纵向联邦的日志异常检测暂处空白，因此，本文第四章节将讨论一种基于纵向联邦的日志异常检测方案。

FLOGCNN：分布式系统通常将日志收集到主服务器进行分析。然而原始数据传输不仅占用了巨大的带宽，而且存在数据泄漏的可能性。Guo 等人[52]引入了一个横向联合学习框架 FLOGCNN。由于联邦学习中的梯度传输证明可能受到攻击[53]，FLOGCNN 使用同态加密对梯度进行加密。由于参数加密和解密耗费大量时间，FLOGCNN 设计了一个轻量级的 CNN 模型，以提高模型的检测效率。FLOGCNN 验证了联合学习技术在日志异常检测中的可行性。

LogTransfer：现有方法需要为每种类型的系统训练特定的异常检测模型。由于日志的多样性，在某些系统上，无监督学习方法的准确性很低。监督学习方法需要大量的数据标签，这是不实用的。Chen 等人[54] 提出了一种迁移学习异常检测方法 LogTransfer，该方法从具有足够标签的系统中学习，并迁移到具有不足标签的另一系统中进行异常检测。不同类型系统的异常日志通常具有相似的模式；它们在语法上不同，但在语义上相似。LogTransfer 利用这种相似性将源系统转移到目标系统，以提高异常检测的性能。

2.5 本章小结

本章主要对日志异常检测的相关研究现状进行了详细的综述介绍。首先，通过对当前研究现状的全面分析，总结了日志异常检测的一般流程框架，其中包括日志收集与预处理、日志解析、特征提取和异常检测模型，阐述了各流程技术内容；其次，详细介绍了各个流程的相关研究技术现状，特别是重点介绍了日志结构化解析方法，按照不同解析原理，分别对各类解析算法进行了详述；然后，介绍了几种日志异常检测使用的特征类型和相关提取的方法；最后，本章详细介绍了目前检测模型的相关研究现状，按照传统的机器学习模型到深度学习模型、再到联邦学习的路线，详细分析了各大类模型方法下，相关检测模型的检测原理和所涉及技术内容。本章旨在较为全面的综述目前日志异常检测的研究现状。

第三章　基于启发式规则的流式在线日志解析方法

本章对日志解析的基本问题定义给予解释说明，并阐述目前日志解析的难点问题。重点介绍基于启发式规则的流式在线解析方法 HRTree 的解析原理和流程，给出解析步骤。最后通过实验分析对 HRTree 方法进行多方面的测试评估。

3.1 日志解析问题定义

日志作为存在于任何系统的通用资源，记录了系统某一时刻发生的特定事件，包含的信息不限于：时间、端口号、IP 地址、日志等级和具体描述信息等内容。图 3.1展示了 HDFS 数据集日志解析的示例过程，将含有日期、时间、进程号、日志等级、组件和文本内容的原始日志，解析成了完全结构化的信息文本。其中日期、时间、进程号等前置信息往往具有统一的格式，通过一些正则化方式即可准确提取。

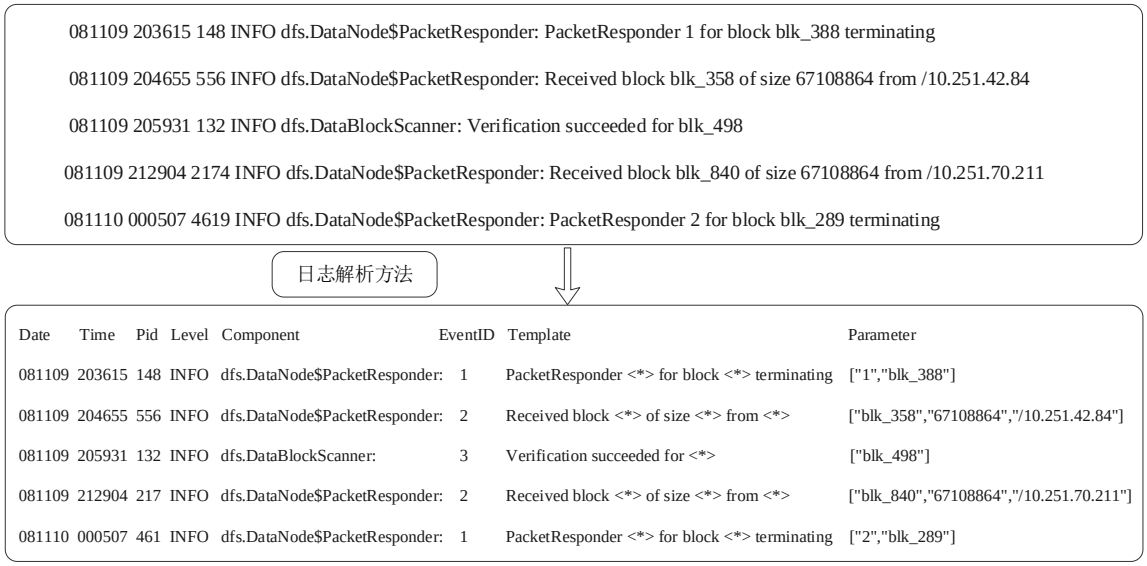


图 3.1　日志解析

日志解析的难点在于日志事件自由描述文本的结构化解析。自由文本通常是由系统内核中源代码的一段打印语句生成，该打印语句往往为程序员编写的一段描述性文本。一小部分为纯静态文本，其余大部分由静态文本和动态参数组合而成，例如：

printf(″exception syndrome register: %s″, reg);

运行过程中可能打印产生如下真实语句：

exception syndrome register: 0x00800000

其中*″exception syndrome register:″* 为静态事件文本，描述了当前日志记录的事件，即寄

存器发生异常；而*"0x00800000"* 则为具体的寄存器标志，以实时动态参数的方式添加到静态语句中。日志解析的目的就是从该类日志语句中准确识别原始的静态文本和参数信息，实现描述事件和具体参数的分离，即解析成事件模板 Template: *"exception syndrome register: <*>"* 和参数 Parameter：*"0x00800000"*。如 2.2 节内容所述，该任务即为目前所有日志解析方法的研究重点。

为何需要将日志事件描述文本进行参数分离？主要原因分析如下，在一个系统环境中，日志事件行为种类通常是有限的，它们通过固定的描述信息阐述了系统在运行过程中发生的具体事件。但当结合参数信息时，就会造成日志事件的爆炸性增长，特别是对于后续的日志异常检测任务来说，会造成特征维度的诅咒，不利于日志异常检测任务对有限特征的挖掘。而通过分离参数提取日志事件模板表示日志事件的具体行为类型，虽然缺少了参数信息，但是不影响该文本对整体事件行为含义的理解。参数信息则更多被当作事件的辅助信息，例如上文所述寄存器异常，只需要事件模板即可获知系统异常的类型，之后结合参数信息便可锁定具体异常位置，综上所述，日志解析进行参数分离是非常有必要的。

传统的日志解析依靠手工编写正则表达式进行匹配，是一种十分低效的方式，已经无法满足现代大型系统的规模增长；并且现代系统更新频繁，日志信息内容也会随着系统更新不断变换，因此手工日志正则维护起来十分困难。虽然基于源代码分析是准确率最高的一种方式，但是源代码在多数情况下又是无法完全获取的，存在一定的操作难度；并且该方式需要当前系统领域专家的介入，无法实现不同系统的迁移使用。所以如前文所述，目前的日志结构化解析，主要集中在通过使用数据挖掘算法，实现对已生成日志文本的解析，高效准确的解析方法是本文的一个研究重点。

3.2 日志解析难点分析

虽然目前已经存在多种日志解析方法，但是各类解析方法在面对不同系统日志时解析效果参差不齐。部分解析方法只能实现对特定日志的有效解析，当面对复杂系统日志时，解析效果明显下降。面对不同系统条件时，稳定可靠的有效解析方法依然是目前日志解析的一大挑战，通过分析已有解析方法，主要存在以下几点问题：

全覆盖问题　日志解析应该尽可能保证对所有日志模式的提取，即有效的日志解析需要覆盖到所有的日志行为模式模板，避免行为模式遗漏。例如，现有的 SLCT 和 LogCluster 解析算法由于频率支持阈值的设置，会造成低频行为模式的遗漏。而这种遗漏对于之后进行的日志检测并不友好，因为某些系统中存在一些具有报警信息的低频日志行为，该类行为往往不常出现，但是一旦出现就表现为系统的异常报警，对于该类低频行为的遗漏，会极大的影响异常检测算法对于真实异常的识别。

表 3.1　日志解析方法概要

方法	时间	覆盖	模式	技术
SLCT	2003	部分	离线	频繁模式挖掘
AEL	2008	全部	离线	启发式
LKE	2009	全部	离线	聚类
LFA	2010	全部	离线	频繁模式挖掘
LogSig	2011	全部	离线	聚类
IPLoM	2012	全部	离线	启发式 + 迭代分区
SHISO	2013	全部	在线	聚类
LogCluster	2015	部分	离线	频繁模式挖掘
LenMa	2016	全部	在线	聚类
LogMine	2016	全部	离线	聚类
Spell	2016	全部	在线	最长公共子序列
Drain	2017	全部	在线	启发式 + 解析树
MoLFI	2018	全部	离线	进化算法
LogParse	2020	全部	在线	解析树 + 词分类
Logram	2020	全部	在线	频繁模式挖掘
HPTree	2023	全部	在线	启发式 + 正则式 + 解析树

离线在线问题　根据解析过程是否能随时产生解析结果，可以将解析方法分为在线和离线两种模式。表 3.1统计了各类解析方法的不同解析模式。离线模式需要在解析时对所有日志数据进行读入分析，即遍历扫描之后才能进行日志事件模板的划分提取。日志事件模板一般固定后不可扩展更新，无法对发生偏移的日志数据进行有效解析。同时离线解析方法通常需要将待处理数据全部读入内存，对于大规模日志数据处理是一种挑战。在线模式则不需要提前获取全部的日志数据，随时读入日志数据，便可随时输出解析结果，内存压力较小。对于新出现的日志模板可以合并到已知相似事件或者创建新的事件模板，这种方法更适用于经常更新的大型系统。

准确性问题　高效准确的日志解析是异常检测任务的基础，不同的日志解析方法必然在解析准确性上存在一定的噪声和误差。尽管有些方法已经实现了 95% 以上的解析准确性，但是就像 He 等人[55]所发现的，日志挖掘可能对某些关键事件敏感。在关键日志上 4% 的解析错误，有可能在日志检测阶段导致极大性能下降表现。通过分析现有日志解析方法，日志解析的准确性表现在以下几点：（1）健壮性问题；例如，Drain 和 Spell 目前在 HDFS 和 Apache 几类稳定数据集上能够实现 95% 以上较高的准确性，但是在 Linux 和 Proxifier 几类复杂数据集上准确率降到了 50% 左右甚至更低，在不同类型系统日志的解析上现有各类解析方法不稳定。（2）参数过度拟合；部分方法虽然对于日志解析的分类准确，但是实际解析的结果却不如人意，比如 Drain 方法特别容易出现参数的过度拟合，将本不是参数的单词识别为参数，影响了

日志模板对日志事件的表达。特别是在部分提取语义特征和参数特征的检测模型中，参数提取不准确，对于检测模型效果的影响是非常大的。

解析效率问题 效率是日志解析方法的一个重要指标，当面对大型系统，特别是大数据时代下信息量爆炸增长，某些系统服务下一天的数据量能够达到千万条级别。倘若日志产生之后，无法做到高效解析，及时的异常检测无法正常运行，导致异常检测服务的滞后性，将会严重影响异常检测任务的效率，进而不能及时发现问题、解决问题。目前部分解析方法（例如 LKE、MoLFI）无法做到对大规模日志数据的解析，并且部分解析方法解析速度过慢，严重影响日志解析的流程，导致异常检测滞后性问题出现，所以解析效率是异常检测任务中解析方法选择时着重关注的问题。

3.3 启发式规则解析方法

如上所述，日志解析依然存在许多挑战，本文从两个方面出发，一方面，从各类解析方法的原理和技术内容出发，吸取目前已有解析方法的优点，同时发现已知的问题和不足，表 3.1展示了各解析方法的特点概要；另一方面，从数据层面入手，总结解析难度较大的几个数据集的特征，剖析解析方法面临的数据挑战。

通过以上两个方面的分析比较，本文提出一种基于启发式规则的解析方法 HRTree，该方法以目前较为高效的解析结构树方法 Drain 为基础，在其解析原理上进行了创新改进。启发式规则源于对系统日志的观察分析，发现日志信息中往往具有以下规则：1）同类日志描述信息往往具有相同的日志长度；2）日志数据中常含有一些具有特定格式的参数内容，例如，IP 地址，数字、url 等。3) 部分参数信息常跟在具有特定格式的文本后面，例如等号，冒号和括号内。HRTree 基于以上规则内容进行启发式的特定数据处理，同样采用树形结构存储解析结果，实现流式在线高效解析。。下面首先陈述 HRTree 方法的解析流程，同时附上相关主要步骤的伪代码内容，如算法 3.1、算法 3.2 和算法 3.3 所示。最后对相关改进和创新内容进行详细说明。

HRTree 解析流程：

第一步：基于领域知识进行预处理清洗；通过自定义正则表达式，对部分常用变量进行清洗；例如，IP 地址、HDFS 中的块地址 blk_id 等。这种正则表达式的定义较为简单，只需要对日志数据进行总览，然后制定简单的正则匹配规则即可。之后进入第二步。

第二步：基于启发式规则进行部分 token 拆分；如 AEL 方法中发现的一些启发式规则，日志数据中常含有 $''word = value''$、$''word : value''$ 和 $''(word, value)''$，其中 $word$ 是有具体含义的实词，而 $value$ 则是真实的参数内容。由于这些组合没有空格拆分，在 Drain 中被识别为一个 token，因此无法准确分离 $value$ 参数部分，导致参

数过度解析问题。

在对日志语句进行拆分时，加入′=′、′:′、′(′、′)′、′,′等符号的判断，在其符号两端加入空格符强行隔断。由于在分词之后还需要进行合并，因此为区分原本语句空格和新加入空格，在此之前先将原语句中空格使用特殊符号替换，之后再转换回去。具体内容演示如图 3.2 和图 3.3 所示，之后进入第三步。

```
//原始日志；
ciod: Received signal 15, code=0, errno=0, address=0x000001b0

//使用空格进行拆分；生成含有7个token的列表；
[“ciod:”,“Received”,“signal”,“15,”,“code=0,”,“errno=0,”,“address=0x000001b0”]

//使用Drain解析，通过比较，部分内容被模糊化成通配符，其中包括code/errno/address等内容；
[“ciod:”,“Received”,“signal”,“<*>”,“<*>”,“<*>”,“<*>”]

//将原始日志进行拼接，得到日志模板，出现过度模糊化的问题；
coid: Received signal <*> <*> <*> <*>
```

图 3.2　Drain 解析方法中 token 拆分示例

```
//原始日志；
ciod: Received signal 15, code=0, errno=0, address=0x000001b0

//先使‘$’替换空格符；
ciod:$Received$signal$15,$code=0,$errno=0,$address=0x000001b0

//在‘=’‘$’‘,’两侧加入空格；
ciod : $ Received $ signal $ 15 , $ code = 0 , $ errno = 0 , $ address = 0x000001b0

//再次使用空格符进行拆分；生成含有23个token的列表，其中6个原始空格‘$’；
[“ciod”,“:”,“$”,“Received”,“$ ”,“signal”,“$”,“15”,“,”,“$”,“code”,“= ”,“0”,
“, ”,“$”,“errno”,“=”,“0”,“, ”,“$”,“address”,“=”,“0x000001b0”]

//使用HRTree解析，通过与其他日志比较，参数部分被准确模糊化成通配符；
[“ciod”,“:”,“$”,“Received ”,“$ ”,“signal”,“$”,“<*>”,“,”,“$”,“code”,“= ”,“<*>”,
“, ”,“$”,“errno”,“=”,“<*>”,“, ”,“$”,“address”,“=”,“<*>”]

//将原始日志进行拼接;
ciod:$Received$signal$<*>,$code=<*>,$errno=<*>,$address=<*>

//使用空格替换‘$’,得到解析结果；
ciod: Received signal <*>, code=<*>, errno=<*>, address=<*>
```

图 3.3　HTPree 解析方法中 token 拆分示例

第三步：搜寻解析树第一层；根据启发式规则，相同日志种类的日志消息往往含有相同的长度，统计当前日志数据的 token 数量 (此处不再统计第二步中使用的各类分割符号)。搜寻根节点下第一层子节点，若搜寻到当前长度的子节点，则更新到当

算法 3.1 HRTree 主体算法

输入：所有日志集合:log; 相似度阈值:st；最大深度:$depth$; 简单正则表达式:reg;
输出：树形存储的日志结构:$Root$;

```
Initialize 树形日志存储结构：Root;
function HRTree(log,st)
    for i = 1 to length(log) do
        //首先基于正则进行常用变量清洗;
        log_i ← regex(reg)                                          //第一步
        //对日志数据进行部分符号替换，拆分成 token 列表;
        log_i ← preprocess(log_i)                                   //第二步
        //根据当前日志 token 列表在 Root 中寻找相似模板;
        if treeSearch(log_i,Root,st,depth) then                     //第三-五步
            //如果搜索到符合相似度日志模板，则进行日志信息合并;
            MatchLog ← 搜索到的相似模板日志
            for j=1 to length(log_i) do                             //第六步
                if log_i[j] ≠ MatchLog[j] then
                    //合并日志模板时，若出现相同位置 token 值不同，则认定其为变量，使用通配符“<*>”替换;
                    MatchLog[j] = "<*>"
        else
            //如果未搜索到符合相似度日志模板，则将当前日志信息更新到 Root 树形结构中;
            updateTree(log_i, Root, depth)                          //第七步
    return Root
```

前长度子节点上，转入第四步；若无该长度节点，则返回空值，跳转到创建新节点的第七步，更新创建新节点。

第四步：依次搜寻 token 路径，直到叶子节点；在进行完第三步长度节点的搜寻后，对 token 内容进行搜寻 (跳过第二步中的各类分割符号 token)。若当前深度的子节点含有当前 token 内容，则更新到新的节点中，反复进行该步搜寻流程（循环第四步），直到搜寻到叶子节点（执行第五步）。若无 token 节点搜寻匹配成功，则返回无匹配结果，跳转到创建新节点的第七步。

第五步：通过相似度搜寻匹配日志模板；到达此步说明已经搜寻到叶子节点处，该叶子节点为一组相同路径信息的日志组合；之后在该组日志中寻找和当前日志信息最相似的一组。通过阈值判断，若最相似日志组相似度大于当前设定阈值。则返回当前找到的最相似组日志信息内容（之后执行第六步）。否则返回未搜寻到结果，跳入第七步。本文此处重新定义了相似度的计算方式如下：

$$simSeq = \frac{\sum_{i=1}^{m} equ(seq_1(i), seq_2(i))}{m - count(symb)} \tag{3-1}$$

$$equ(t_1, t_2) = \begin{cases} 1, & \text{if } t_1 = t_2 \text{ and } t_1 \notin symb \\ 0, & \text{otherwise} \end{cases} \tag{3-2}$$

其中 seq_1 和 seq_2 分别表示当前日志消息和匹配日志事件消息；$seq(i)$ 表示当前序列的第 i 个 token；m 表示当前序列的 token 数量；$symb$ 表示第二步构建的启发式符号列表；$count()$ 统计序列中所有符号 token 的数量；$equ()$ 统计相同位置 token 相同并且非符号 token 的数量；t_1 和 t_2 表示两个 token。

算法 3.2　HRTree 搜索算法

输入： 当前搜索日志:log_i; 解析结构树:$Root$; 相似度阈值:st; 最大深度:$depth$;
输出： 若搜索到符合相似度的叶节点，则返回该日志模板信息；否则返回 None;

```
function treeSearch(log_i, Root, st, depth)
    curdepth = 1
    fatherNode ← Root[length(log_i)]
    //搜寻第一层长度节点;
    if fatherNode then                                          //第三步
        curdepth ← curdepth + 1
        //依次搜寻 token 路径;
        while log_i[curdepth] ∈ fatherNode  and  curdepth<depth  do   //第四步
            fatherNode ← fatherNode[log_i[curdepth]]
            curdepth ← curdepth + 1
        //在叶子节点列表中搜寻相似度最高的日志模板;
        for logTemple  in  fatherNode.logClust  do              //第五步
            MaxSim ← simSeq(log_i, logTemple)
        //进行相似度阈值比较;
        if MaxSim ≥ st then
            return logTemple
        else
            return None

    else
        return None
```

第六步：若搜寻到当前结果，则对当前结果和当前日志消息进行比对，若相同位置的 token 相同则认定该 token 为常量，否则认定为变量信息，对其进行通配符 <*> 的模糊化处理。

$$merge(t_1, t_2) = \begin{cases} t_1, & \text{if } t_1 = t_2 \\ <*>, & \text{if } t_1 \neq t_2 \end{cases} \tag{3-3}$$

第七步：创建更新解析结构树节点；若未找到当前日志相似叶子节点，则在原解析结构树中创建新的节点；若存在当前日志 token 长度节点，则更新到当前长度节点，若不存在则创建该长度节点；之后创建 token 节点，若存在于当前节点的子节点列表中，则更新到下一深度，否则创建该层节点。在创建过程中若超过了最大深度节点，则将最大深度一层定义为叶子节点，不再继续创建。将当前日志信息更新到叶子节点中。由此创建了一条由根节点到叶子结点的 token 路径，为日志序列的前缀信息。

算法 3.3 HRTree 更新算法

输入：当前搜索日志:log_i; 解析结构树:$Root$; 最大深度:$depth$;
输出：返回更新后的树形存储日志结构:$Root$;

```
function updateTree(log_i, Root, depth)                    // 第七步
    curdepth = 1
    seqLen ← length(log_i)
    if seqLen ∈ Root then
        //如果根节点下存在当前长度子节点，则更新到该节点下;
        fatherNode = Root[length(log_i)]
    else
        //如果根节点下未存在当前长度子节点，则创建该长度子节点，并更新到该节点下;
        Root[length(log_i)] ← Node(length(log_i))
        fatherNode = Root[length(log_i)]
    curdepth ← curdepth + 1
    for token in log_i do
        if token ∈ fatherNode then
            //如果 token 为当前父节点的子节点，则更新到该节点下;
            fatherNode = fatherNode[token]
        else
            //如果 token 不是当前父节点的子节点，则创建该长度子节点，并更新到该节点下;
            fatherNode[token] ← Node(token)
            fatherNode = fatherNode[token]
        curdepth ← curdepth + 1
        if curdepth > depth then
            break
    //将当前日志所有信息更新到该叶子节点上;
    fatherNode.logClust add log_i
```

在测试中，本文发现会出现日志长度小于树的最大深度的日志信息，Drain 算法

未对该情况进行讨论，HPTree 将该情况日志 token 提前创建叶子节点，打破了 Drain 固定深度的规则，仅对最大深度进行控制，实现在较小深度也可创建叶子结点。同时在创建 token 节点的过程中，对于第一步正则表达式解析的常用变量，Drain 方法虽然也对其进行了识别，但是在模板创建过程中并未更新到日志模板信息中，本文将其实时更新到日志模板库中。

算法复杂度分析　结合算法流程步骤，可以推理出 HRTree 方法与 Drain 方法具有相同的线性时间复杂度。具体地，HRTree 时间复杂度可以计算为 $O((d+cm)n)$；解析方法以流式依次对每条日志信息进行解析，n 是待解析日志信息的数量；d 为解析树的最大深度，在搜索和更新过程中，最多 d 次比较，寻找到叶子节点；c 为当前叶子节点的候选日志组数量，m 为日志信息含有的 token 数量，当寻找最大相似度的日志组时，进行 cm 次比较；其中 d 显然为常数，而 m 和 c 也可以视为一个波动的常量，因为日志信息的 token 数量和每一个叶子节点的日志组数量十分有限，即使是它们的乘积，其值也远远小于日志信息的数据量 n，因此，HRTree 的时间复杂度为 $O(n)$。

结合以上解析步骤内容，HPTree 是一种基于 Drain 的启发式规则改进解析方法，具有 Drain 全覆盖、流式在线解析的特点，同时又克服了 Drain 存在的参数过度解析、部分叶子结点无法创建和常用变量无法及时更新等问题，能够实现更高效的解析效果。最后总结 HPTree 具有以下特点：

1）同样采用树形解析结构方式，通过日志长度计算，缩短日志搜索的范围，提高解析速度。

2）同样采用最大深度和最大叶子节点数控制，控制日志结构树的规模，避免爆炸性增长。

3）创新提出使用启发式方法对部分 token 进行拆分，将 Drain 以空格为标志的 token 拆分方式转换为规则化的启发式拆分，对组合 token 进行有效拆分解决日志参数识别不准确问题。

4）优化相似度计算方式，解决 Drain 方法中相似度要求较低，容易过度拟合的问题。

5）解决了日志更新步骤中，序列长度小于最大深度的节点无法创建为叶子节点问题。

6）解决了明显参数类节点，将其模糊化处理实时更新到解析树的问题。

3.4 实验对比评估

本文在该部分从日志解析方法的典型指标、准确率指标和运行效率指标多个方面，对 HPTree 与其他解析方法在不同数据集上进行全面的评估测试，评估 HPTree 的真实解析效果和大规模日志数据的解析效率。

3.4.1 实验准备

实验环境 本节的实验环境为一台 64 位 Intel Core i9-12900KF 3.20GHz 16 核 24 线程 CPU，64GB 内存，NVIDIA GeForce RTX3090Ti GPU 的 Ubuntu 20.4 系统服务器。下文所有实验均在该服务器上进行，为避免其他应用程序占用对运行结果的干扰，特别是时间效率评估方面，本文所展示的每组数据均为三次及以上测试结果的平均值。

数据集 真实的日志数据具有一定的机密性，能公开获取的数据资源十分有限。本文通过调研，发现 Zhu 等人[56]公开了一个日志数据仓库-loghub，其中包含了 16 种不同系统下的日志数据，涵盖了分布式系统、超级计算机、操作系统、移动系统、服务器应用程序和独立软件。该公开的日志数据仓库为日志分析技术的研究和开发贡献了数据资源。下文部分实验内容使用了该数据仓库提供的实验数据，其中具体使用情况于相关小节做具体描述。

对比方法 如前文 2.2 节所综述内容，日志解析涉及各种原理类型方法，本文通过使用 Zhu 等人[56]提供的日志解析工具 logparser，复现了其中的 12 种解析方法，包括频繁模式挖掘方法 LFA 和 LogCluster，聚类方法 LKE、LogSig、LogMine、SHISO 和 LenMa，启发式方法 AEL、IPLoM 和 Drain，其他解析方法 Spell 和 MoLFI。使用这 12 种解析方法与本文 HRTree 解析方法进行不同指标的实验对比（共 13 种方法）。

3.4.2 典型指标评估

本文首先使用分类算法的典型评价指标精确率 $Precision$、召回率 $Recall$ 和 F_1-$measure$ 指数来评估 HPTree 的解析性能，之前的多个研究中也使用了这个评估度量[26,55]，该组指标主要评估日志消息在日志行为种类上分类是否准确，测试解析日志行为事件分类的效果。

指标定义如下：

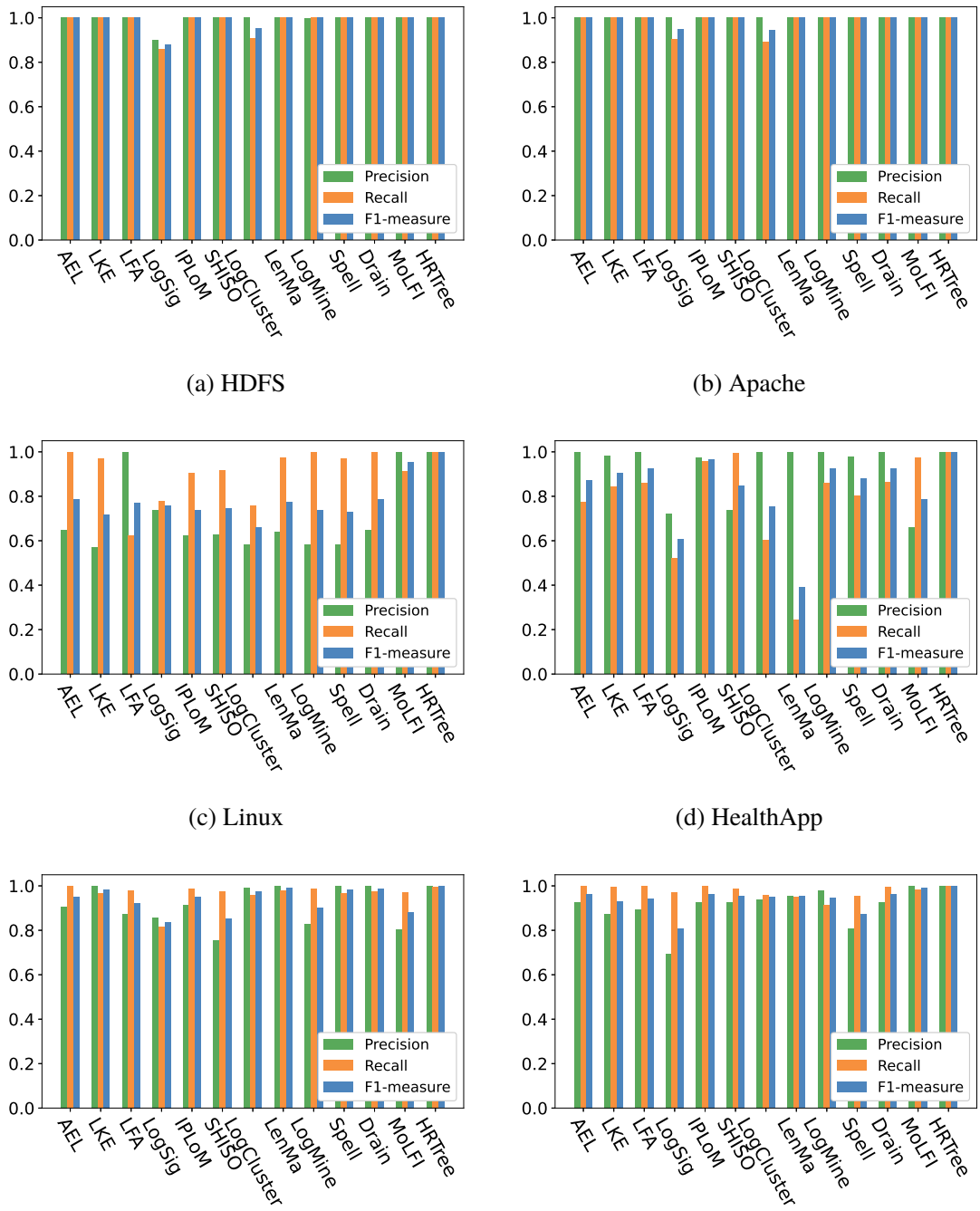
$$Precision = \frac{TP}{TP + FP} \tag{3-4}$$

$$Recall = \frac{TP}{TP + FN} \tag{3-5}$$

$$F_1\text{-}measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3-6}$$

其中真阳性 TP(True Positive) 表示将两个具有相同日志事件的日志消息分配到同一个日志组；假阳性 FP(False Positive) 表示将两个具有不同日志事件的日志消息分配到同一个日志组中；假阴性 FN(False Negative) 表示将两个具有相同日志事件的日志消息分配给不同的日志组。

每个系统的日志规模均不相同，有的日志条目达到了千万级别，日志种类达到了



(a) HDFS　(b) Apache

(c) Linux　(d) HealthApp

(e) Android　(f) Proxifier

图 3.4　多种解析方法在不同数据集上典型指标评估结果

上百种。为了方便测试，本节实验使用 Zhu 等人提供的一个小型标注数据集，该数据集从每种日志数据集中随机抽取了 2000 条日志消息，并且手动将其进行了日志事件模板的标注。虽然 2000 条日志数据为随机采样所得，但是保留了全体数据的关键属性，如日志事件的冗余和事件的多样性。

由于本小节包含 13 种方法和 16 个数据集，并且包含三个指标，测试结果较多。

因此本文在正文部分挑选了部分具有代表性的结果进行展示分析，其余测试结果可查看附录 A。

典型指标解析结果分析 如图 3.4所示，HPTree 在所有数据集上均取得了较好的典型指标表现。图 3.4a和图 3.4b展示了所有解析方法在 HDFS 和 Apache 数据集上的解析结果，除 LogSig 和 LogCluster 两种检测方法效果略微较低外，其余解析方法均对其在典型指标上具有较高的解析评价。分析原因为该类数据集较为稳定，日志种类数量较少，解析任务难度不高，所以多数解析方法在该类数据集上均能取得较好的典型指标结果。

如图 3.4c-f 所示，除 HPTree 外，其余解析方法在 Linux、HealthApp、Android 和 Proxifier 数据集上，均出现了解析效果的较大浮动，有些方法 F_1-$measure$ 指标甚至低于了 80%。解析效果评价较高的 Drain 方法也在 Linux 和 HealthApp 上出现了解析效果的下降，分析原因为该类数据集日志行为事件种类较多，例如，Linux 数据集仅在 2000 条日志数据中就包含了 115 种日志事件类型，相较于 HDFS 的 14 种事件类型，解析任务难度显著增加。对于该类复杂日志数据集，其他方法出现了将多类行为合并为了一类行为的解析不准确现象。而 HRTree 由于更加严苛的相似度计算和 token 拆分处理，避免了相似行为的过度合并，在分类的典型指标上效果更好，证明了 HPTree 解析效果面对不同数据集的稳定性。

3.4.3 准确率指标评估

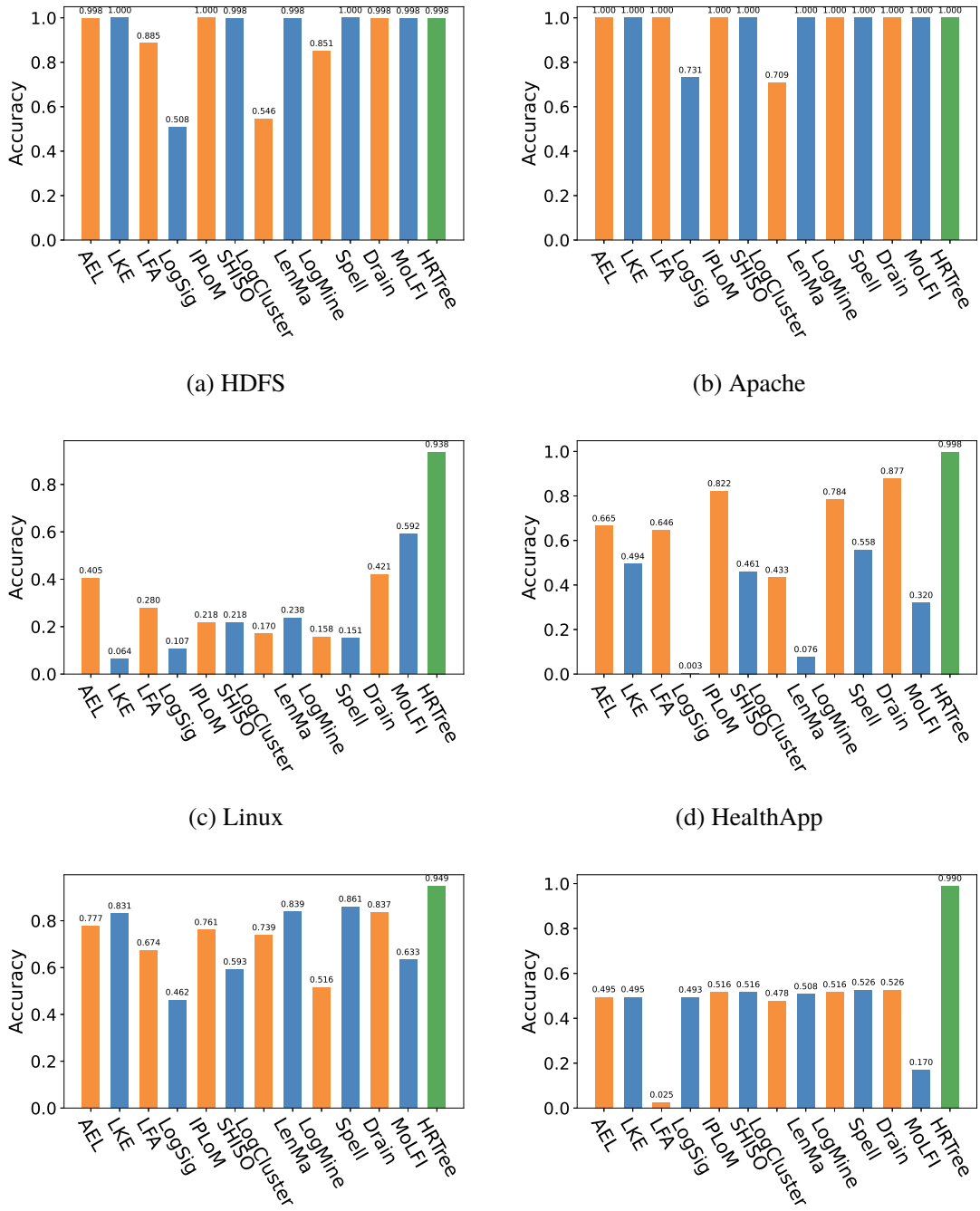
在以上典型评估指标的基础上，本文还进行了准确率指标 $Accuracy$ 的评估，在日志解析任务中准确率指标相对于上面三个指标更加严格。主要对日志解析的日志内容进行了评估，检验日志消息的参数识别是否准确。该指标统计在所有解析结果中，日志事件内容解析正确的日志数量占所有日志总数的比例。

其定义如下：

$$Accuracy = \frac{\text{解析正确日志数量}}{Total(\text{总日志数量})} \tag{3-7}$$

同样和上一小节一样，本小节使用该指标对所有数据和方法进行了全面的评估，由于测试结果较多，此处仅挑选了部分具有代表性的结果进行展示分析，如图 3.5，其余测试结果于附录 A 中进行全面展示。

准确率指标解析结果分析 在部分数据集上，虽然典型指标评估效果可以接受，但是当使用更加严苛的准确率指标时，有些方法出现了大范围的性能下降。这是因为虽然解析方法对于日志事件的分类是正确的，比如都准确解析成了同一类行为，但是对于具体的解析事件模板的内容，部分解析方法并没有对其中的部分参数进行准确识别，出现了参数的误识别和漏识别。导致了在该性能指标上评估效果的下降。而准



(a) HDFS　(b) Apache

(c) Linux　(d) HealthApp

(e) Android　(f) Proxifier

图 3.5　多种解析方法在不同数据集上准确率指标评估结果

确识别日志参数，可能成为影响后续日志异常检测的关键。因此其他解析方法未能在精准识别参数和模板上展现很好的任务能力。

对于本文 HPTree 解析方法，由于加入了启发式的规则，特别是对于“$word = value$”、“$word : value$”和“$(word, value)$”这类日志信息，能够在字符层面准确识别到具体的参数，避免了误识别。同时加入的部分规则化条件，可以对常用变量提前

处理，避免了漏识别，因此展现了较高的解析结果。同时可以帮助后续的日志检测方法，特别是语义层面的特征提取上，实现更加准确的特征提取。

3.4.4 时间效率评估

为了处理大规模的日志数据，效率是日志解析方法需要考虑的一个重要方面。部分方法虽然解析效果相近，但是由于其算法原理不同，出现无法胜任大规模日志数据的解析问题，因此一个好的日志解析方法，能否在大数据条件下高效解析十分重要。为了衡量日志解析方法的效率，本文对日志解析方法在不同数量级日志数据上进行了解析运行时间的测试，对每种方法从读取、解析、到最后数据结果的输出，进行了全流程运行时间的统计。

由于 Zhu 等人公开的日志数据仓库提供的每个数据集 2000 条的数据规模较小，各类解析方法均能在较短时间内完成，不能检验其在大规模日志数据上的解析效果。通过查看公开的全量数据集，部分数据集整体规模并不大，例如 Linux 和 Proxifier 数据集仅有 2 万条。为满足百万级别以上实验条件，本文选取了 HDFS、BGL 和 Android 三个日志数据集，分别为分布式系统日志、超级计算机系统日志和移动系统日志，其中 HDFS 数据集最大达到了千万条级别，BGL 数据集达到了 400 万条，Android 数据集 100 万条。

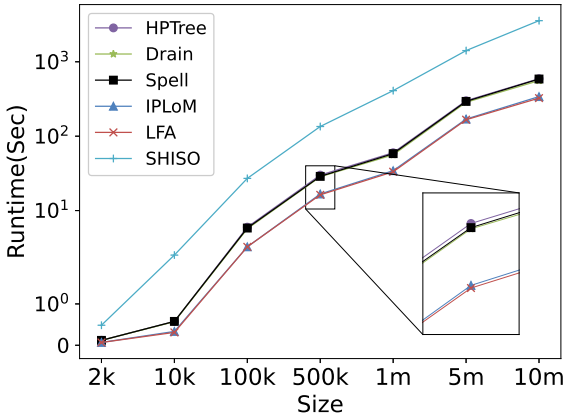
本文在相同数据集上进行了不同数据规模的拆分，构建了不同数量条目的测试数据。例如，HDFS 有一千万条日志，于是测试了 2 千条、1 万条、10 万条、50 万条、100 万条、500 万条和 1000 万条七个数据规模，分别对应 2k、10k、100k、500k、1m、5m 和 10m。而对于 BGL 数据集和 Android 数据集，由于数据仅在百万规模，所以测试了 2 千条、1 万条、10 万条、50 万条和 100 万条五个数据规模，分别对应 2k、10k、100k、500k 和 1m。

由于方法众多，本文仅从相同原理方法中选择了一至两种解析方法对比，频繁模式挖掘中选择了 LFA 方法，聚类解析方法中选择了 SHISO 解析方法，启发式方法中选择了分层聚类的 IPLoM 和 Drain，以及其他方法中的 Spell。其中 LFA 和 IPLoM 为离线解析方法，其余为在线解析方法。所以，本节对六种方法在三个数据上进行了测试。

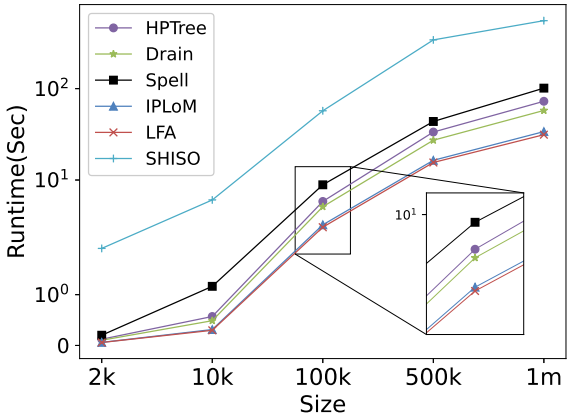
时间效率指标解析结果分析　通过分析如图 3.6结果，可以看到几种方法在不同的数据规模上时间效率基本呈线性趋势，随着数据规模的增大，解析时间随之增长。其中 SHISO 较其他解析方法，在三个数据集上均表现为最差的解析速率，在百万级别的日志规模上，相差了近十倍。其中 IPLoM 和 LFA 解析方法，在三个数据集上展现了相近的解析效率，成为解析速度最快的解析方法，主要原因为其使用了分层聚类的思想，通过日志长度的启发式思想，分层解析大大提高了解析的速度。但是两种方

法均为离线解析，需要全部将日志数据读入内存，对计算资源占用较大，当测试数据超过处理器内存时，无法进行有效解析，所以对于更大规模的日志数据集，无法胜任解析工作。

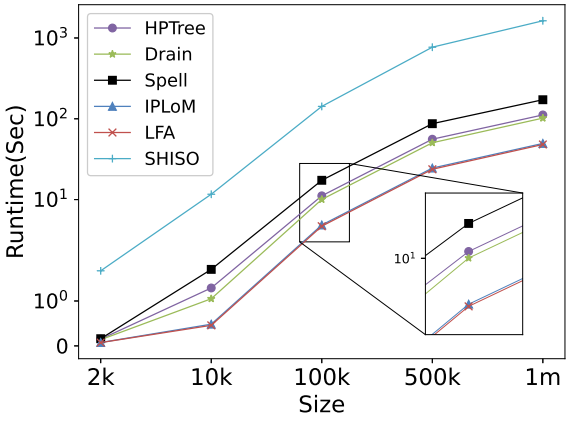
本文的 Drain、HPTree 和 Spell 解析速度相差不大，其中 Drain 的解析速度三者之中相对较快，HPTree 居中。HPTree 由于和 Drain 具有类似的解析原理，加入了 token 拆分和规则化过程，解析速度略慢于 Drain，但两者相差不大。Spell 虽然采用了最长公共子序列算法，但是其结构也使用前缀树，在解析效率上也有不错的表现。整体分析 HRTree 解析方法，具有和 Drain 相当的解析效率，并且为在线流式解析方法，能够胜任大规模日志数据的实时解析，不受计算机内存问题的限制，是一种高效率的解析方法。



(a) HDFS



(b) BGL



(c) Android

图 3.6　不同解析方法时间效率对比评估

3.5 本章小结

本章主要提出了一种基于启发式规则的流式日志解析方法，在此之前分析了目前日志解析问题的详细定义，并且对几大难点问题进行了分析。之后对 HRTree 的解析流程结合算法步骤进行了详细的介绍，阐述了 HRTree 解决的几大问题。最后在典型指标、准确率指标和运行时间效率指标几个方面进行了详细的评估，无论是在典型指标还是准确率指标方面，HRTree 均表现了较为理想的解析效果，特别是在面对不同的日志数据集时，当其他解析方法出现解析效果的下降时，HRTree 展现了稳定的解析效果。在运行效率方面，HRTree 作为一种在线的解析方式，可以不受内存的限制，并且展现出了和 Drain 相近的解析速度。通过以上几个指标的评估实验，证明了本文提出的 HRTree 方法是一种可以覆盖全部日志信息，实现在线高效率和高准确率解析的一种有效方法。

第四章　基于纵向逻辑回归的日志异常检测方案

本章主要提出了一种基于纵向逻辑回归的日志异常检测方案，并对其进行了全流程检测步骤的方案设计。包括使用隐私集合求交技术实现日志检测序列的数据求交和对齐，改造传统逻辑回归模型损失函数和梯度的计算方式，在保证不泄露原始数据的前提下，使用同态加密算法进行联合检测模型的参数更新。最后，本章通过各类指标对纵向联邦学习的日志异常检测方案进行了综合评估。

4.1 纵向联邦日志异常检测问题定义

在日志异常检测领域，目前已经涌现出了各类检测模型。如第二章综述内容的介绍，目前的检测模型从机器学习方法的逻辑回归、决策树到主成分分析、聚类和不变量挖掘，从深度学习的长短时记忆循环神经网络、卷积神经网络再到 Transformer 语义模型，各种模型算法已经在日志异常检测领域进行了广泛应用。

虽然各种检测方法模型被提出，但是目前在大数据时代，特别是分布式系统的大规模部署下，日志异常检测依然存在多种问题，其中异常定义困难和数据隐私问题尤为突出。例如部分模型算法需要依赖有标签的异常数据进行监督学习，对数据标签依赖程度较高；分布式系统下的日志收集，存在数据泄露的风险；单一系统的异常检测，泛化能力较差，无法应对不同的场景和数据分布等问题。

联邦学习作为一种解决数据隐私问题的模型训练方法，近年来在众多领域已经有了广泛应用，而在日志异常检测领域却应用甚少。根据本文调研，目前仅有一种基于横向联邦的日志异常检测方法 FLOGCNN[52]和基于迁移学习的日志异常检测方法 LogTransfer[54]被提出。目前纵向联邦的日志异常检测方案尚处于研究空白，但基于纵向联邦学习的日志异常检测却可以解决以下实际问题。

数据隐私问题：系统日志通常包含敏感信息，如 IP 地址、用户名、密码等。使用纵向联邦学习可以在不暴露这些信息的情况下，让多个参与者联合建模，提高模型的准确性，并保护数据隐私。

数据不平衡问题：在大型分布式系统中，可能有多个日志生成者，它们记录的日志包含相互关联的信息，如同一个用户的登录信息、操作记录等。由于每个参与者所拥有的数据量和质量可能不同，使用纵向联邦学习可以在保护数据隐私的前提下，让多个参与者共同训练模型，从而减少数据不平衡问题。

模型泛化问题：日志异常检测需要具有一定的泛化能力和鲁棒性，以应对不同的场景和数据分布。使用纵向联邦学习可以让多个参与者共同训练模型，从而增强模型

的泛化能力，使模型能够更好地适应不同的数据分布。

总之，纵向联邦学习可以在日志异常检测领域解决数据隐私、数据不平衡、模型泛化和鲁棒性等实际问题。但是日志异常检测过程由于涉及日志解析、特征提取等重要前置流程，部分解析方法和特征提取方法无法适用于联邦场景，本文通过对检测流程和数据样本内容的分析，提出一种纵向联邦在日志异常检测领域的应用方案。主要解决数据标签定义困难和短缺问题，假设一方数据是有标签的异常标记，通过纵向联邦学习，可以帮助另一方实现异常检测，发现异常。

4.2 基于纵向逻辑回归的日志异常检测方案

为解决日志异常检测存在的以上问题，本文创新性的提出了基于纵向逻辑回归模型的日志异常检测方案，在传统逻辑回归模型的计算方式上进行了训练过程的改造，双方通过加密交换中间计算结果，实现联合模型的训练。方案框架如图 4.1所示，其中主要涉及本地数据预处理、隐私集合求交数据对齐和联合模型训练几个重要方面，本文对其进行了详细的推理论证。具体内容陈述如下。

4.2.1 数据对齐

在纵向联邦学习中，数据对齐是实现模型训练的前提条件，如何获知哪些数据为双方同时存在的数据并对其进行对齐成为一个重点。训练双方需要在不泄露各自真实数据的基础上，实现数据集合的求交并对齐操作。在其他的纵向联邦学习应用中，这种对齐通常是以“用户 ID”为基础的，主要通过隐私集合求交技术 PSI（Private Set Intersection）实现对相同“用户 ID”的交集计算。

隐私集合求交技术是隐私保护领域的一项重点研究内容，目前已经有多种解决方案，其中包括基于公钥加密的隐私集合求交[57-58]、基于不经意传输的隐私集合求交[59]和基于混淆电路的隐私集合求交[60]。

而对于日志异常检测工作存在一个重要问题，即没有其他场景下的“用户 ID”这一概念，从而缺少了真实的 ID 标签。本文通过对日志异常检测特征提取技术的分析，提出了几种适用于日志异常检测应用领域的 ID 构建方案。如前文 2.3 节所述，本文综述了目前日志异常检测的粒度，分为固定窗口、滑动窗口和会话窗口三种分组采样技术，分别适用于不同的日志异常检测条件。基于三种分组条件，提出适用于不同系统日志数据的对齐 ID 构建方案。

1）若系统日志检测特征分组为会话窗口，日志数据中含有明显的标签类信息，例如，用户名和进程编号等，则自动使用该标签信息为特征 ID，此方法类似其他领域的“用户 ID”概念。

2）对于没有具体标签信息进行标识的特征，多数采用了固定窗口和滑动窗口方

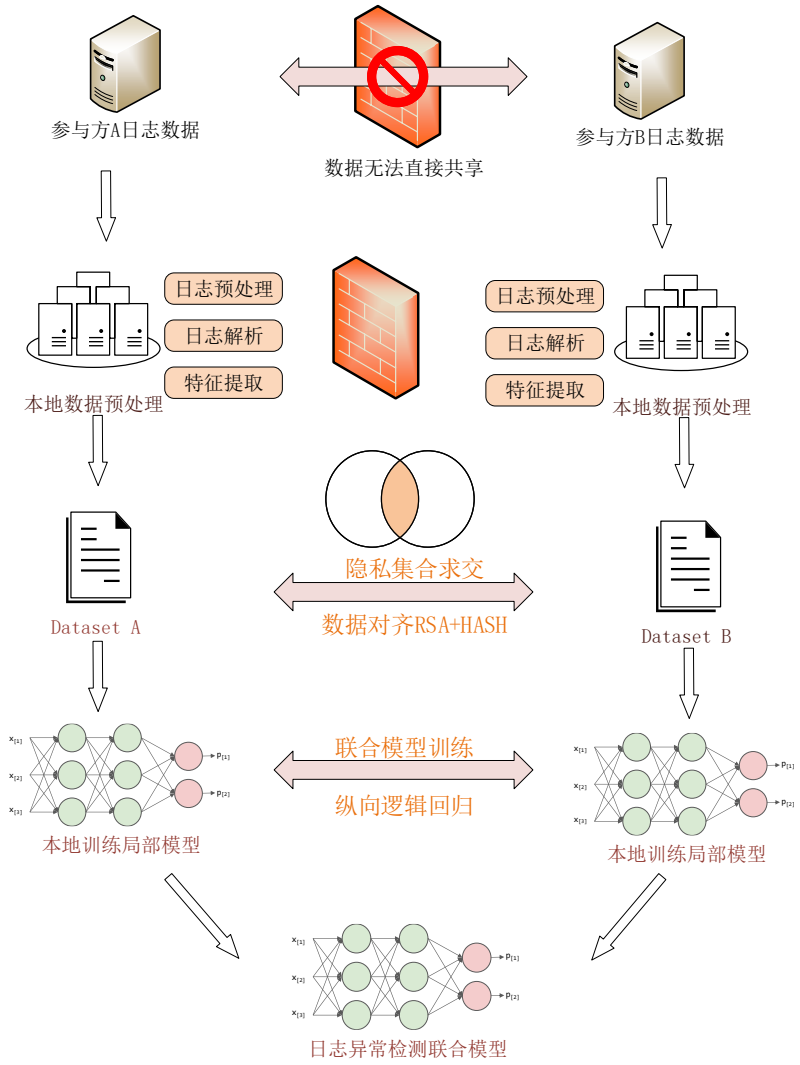


图 4.1　基于纵向联邦的日志异常检测方案框架

式进行了时间维度数据的采集，此时本文提出通过获取固定窗口和滑动窗口的时间戳信息，将每个序列的起始时间定义为特殊的 ID，基于此“特殊 ID”实现数据特征的对齐操作。使用该 ID 定义方法，在进行数据采集前需要进行起始时间和窗口大小的一致性协商，从而保证检测时间维度的一致性。

3）对于部分系统日志，已存在以用户 ID 为标签的检测维度数据，但是由于检测频率较高。在较短时间内，可能会出现对同一个用户的不同行为序列进行检测，因此本文提出使用“用户 ID+ 时间戳”的标识方法，保证序列的唯一性。

在本文纵向联邦日志异常检测的数据对齐过程中，具体使用基于 RSA 公钥加密体系的隐私集合求交技术，如图 4.2所示，双方通过 RSA 加密和散列哈希（本文使用 SHA-1）等隐私保护技术，在不泄露原始数据的基础上，进行相同集合的求解对齐。

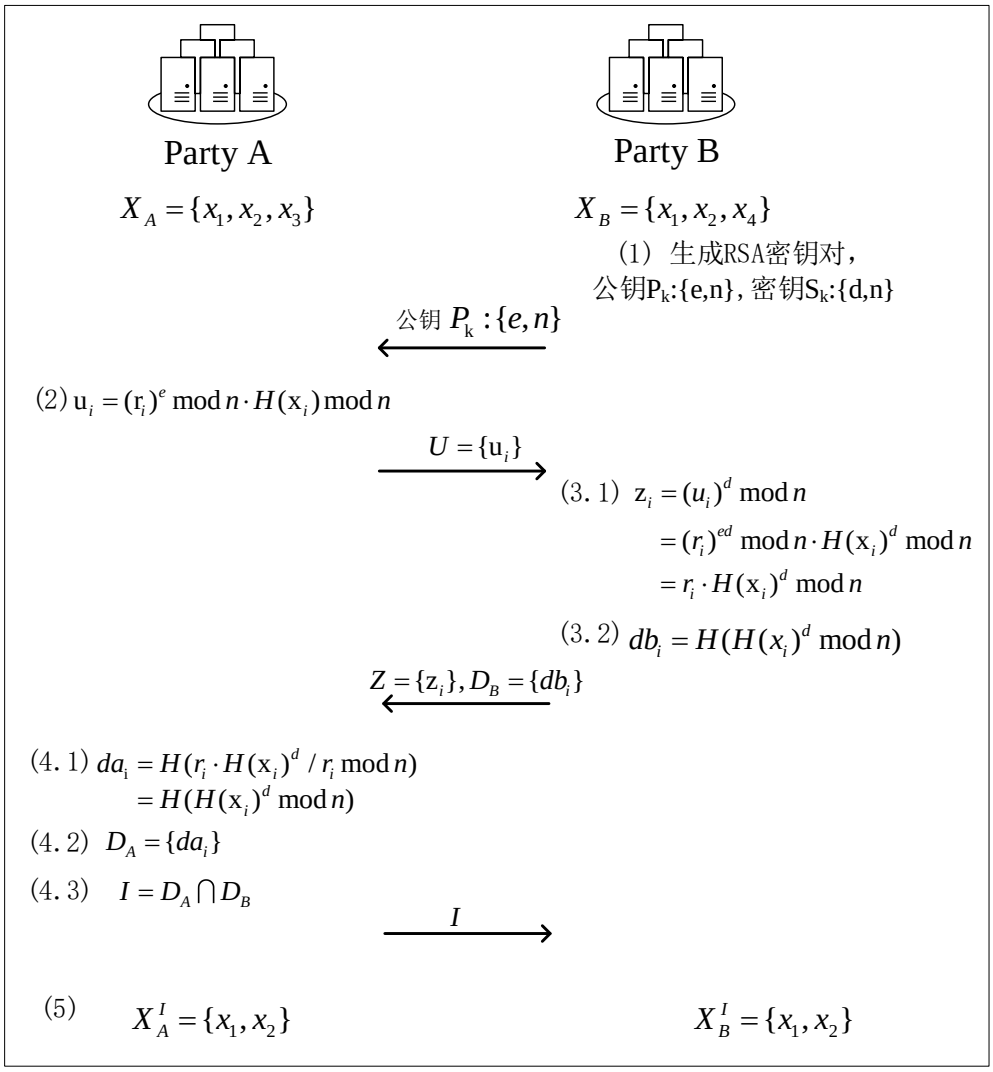


图 4.2 基于 RSA 公钥加密体系的日志数据对齐方案

首先数据双方准备待对齐数据列表，一般情况下该数据列表需保持一定的顺序性，在求解过程中保持索引位置固定。此处定义 A、B 双方两个实体，拥有数据 X_A 和 X_B。具体步骤如下：

（1）第一步：由任意一方通过 RSA 生成加密公钥和私钥密钥对，将公钥发送至对方，私钥保留本地。此处需要注意，当双方样本数据差距较大的情况下（差几个数量级），数据多的一方拥有私钥，可以显著节省计算和通讯的成本。图 4.2为 B 方生成密钥，私钥保留本地，公钥发送至 A 方。

（2）第二步：A 方通过哈希散列函数 $H(x)$ 将每一个 ID 值进行哈希加密处理；然后引入一个随机数掩码 r_i，通过公钥 e 进行加密，之后与哈希值进行乘积得到 u_i，对 x_i 的哈希值进行扰动。将所有 x_i 进行加密后，得到 u_i 的集合 U，发送至 B 方。

（3）第三步：在 B 方得到 u_i 后，使用私钥对其进行解密。首先拆开看，其中由于 r_i 在 A 方经过了公钥 e 加密，此处再次使用私钥 d 解密得到 r_i；而 x_i 的哈希值 $H(x_i)$

则又通过私钥进行了一次加密。所以由私钥 d 解密后的值为 $r_i \cdot H(x_i)^d mod\, n$ 的整体 z_i，此时 B 方无法从中获得 A 方的 ID 信息。之后 B 方将本方 ID 值依次进行加密，加密步骤为先进行一次哈希处理 $H(x_i)$，之后使用私钥 d 加密一次 $H(x_i)^d mod\, n$，然后再进行一次哈希 $H(H(x_i)^d mod\, n)$ 得到 db_i。最后将两个值的集合 Z 和 D_B 发送给 A 方。

（4）第四步：由于 z_i 为 $r_i \cdot H(x_i)^d mod\, n$，通过移除 A 方第二步加入的噪声 r_i 后，再通过一次哈希加密，则得到了与 db_i 具有相同处理步骤的加密值。而对于隐藏有 B 方 ID 的 db_i，因为经过了私钥 d 加密后又进行了哈希，所以 A 方无法获知 B 方 ID 信息，最后通过对 D_A 和 D_B 求交运算，得到经过两次哈希和一次私钥加密数据值的相同集合 I。A 方最后将 I 发送给 B 方。

（5）双方得到 I，与各自 db 对应后，即可得到相应位置的索引，最后通过索引即可找到集合交集。此处为保证后续模型训练工作数据的 ID 索引关系一一对应，各自双方应按照 I 顺序，依次构建原始 ID 顺序，最终实现数据对齐。

以上加密过程，由于加入了随机数掩码同时引入 RSA 和哈希散列函数，双方在任何一步均无法获知对方真实 ID 数据，实现了在数据安全下的隐私集合求交对齐过程。

4.2.2 传统逻辑回归模型

为方便下文纵向逻辑回归方案各处细节介绍，此处对传统逻辑回归模型进行简单回顾。逻辑回归算法是机器学习中常见的二分类算法，与线性回归不同的是逻辑回归引入了阶跃函数 Sigmoid 函数，值域为 (0,1)，任意变量经 Sigmoid 函数映射后的结果可以看成一个概率值，通常将大于 0.5 的数据归为 1 类，小于 0.5 的数据归为 0 类，实现二分类的计算效果。具体解析式如下：

$$f(z) = \frac{1}{1+e^{-z}} \tag{4-1}$$

线性回归表达式为：

$$z_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{4-2}$$

$$= \sum_{i=0}^{n} \theta_i x_i = \theta^{\mathbf{T}}\mathbf{x} \tag{4-3}$$

将式 (4-3) 经 Sigmoid 函数式 (4-1) 处理后得到逻辑回归函数的表达式：

$$h_\theta(x) = f(z_\theta(x)) = \frac{1}{1+e^{-\theta^{\mathbf{T}}\mathbf{x}}} \tag{4-4}$$

逻辑回归任务与线性回归相同，寻找最佳的参数 θ 来获取最佳分类。其中逻辑回

归作为二分类问题，拥有标签 y=0 或 y=1，因此利用最大似然估计可以作出式 (4-5) 和式 (4-6) 假设：

$$p(y=1|x;\theta)=h_\theta(x) \tag{4-5}$$

$$p(y=0|x;\theta)=1-h_\theta(x) \tag{4-6}$$

通过对 n 组数据的极大似然估计（为方便简化对其取对数）得到：

$$log(L(\theta))=log\prod_{i=1}^{n}p(y_i|x_i;\theta) \tag{4-7}$$

$$=log\sum_{i=1}^{n}\left[(h_\theta(x_i)^{y_i}+(1-h_\theta(x_i))^{1-y_i}\right] \tag{4-8}$$

$$=\sum_{i=1}^{n}\Big[y_ilog(h_\theta(x_i))+(1-y_i)log(1-h_\theta(x_i))\Big] \tag{4-9}$$

最大似然估计法求 $logL(\theta)$ 的最大值，因此在其前面加上负号，即为损失函数求其最小值，正好与损失函数特性吻合。对 n 个样本取平均得到损失函数如式 (4-11)：

$$J(\theta)=-\frac{1}{n}\sum_{i=1}^{n}[y_ilog(h_\theta(x_i))+(1-y_i)log(1-h_\theta(x_i))] \tag{4-10}$$

$$=-\frac{1}{n}\sum_{i=1}^{n}\left[y_ilog(\frac{1}{1+e^{-\theta^{\mathbf{T}}\mathbf{x_i}}})+(1-y_i)log(1-\frac{1}{1+e^{-\theta^{\mathbf{T}}\mathbf{x_i}}})\right] \tag{4-11}$$

对于逻辑回归模型的训练，可通过使用损失函数 $J(\theta)$ 对 θ 进行下降方向的计算，之后不断更新其权值，如式（4-12），其中 η 为学习率。

$$\theta_j=\theta_{j-1}-\eta\frac{\partial J(\theta)}{\partial\theta} \tag{4-12}$$

4.2.3 纵向逻辑回归模型

在传统逻辑回归中，如式（4-11）和（4-12），损失函数和梯度的计算离不开特征数据 x 和标签数据 y。显然双方不能通过将己方独有明文数据直接发送给对方，由对方计算出梯度，然后返回梯度值进行更新。在联邦学习场景中，双方数据存在机密性，需要使用隐私保护技术防止原始数据泄漏。本文采用一种非对称加密方式，通过密文形式进行双方数据交换。

具体的，为满足当得到对方密文数据后可以进行计算的需求，本文选用基于 Paillier 同态加密[61-62]的方法实现纵向联邦学习。同态加密作为一种特殊的加密算法，对密文进行处理计算后得到仍然是加密的结果，满足对明文运算后结果加密和直接对密文进行运算得到的结果相同。Paillier 方案由于效率较高、安全性证明完备的特点，

在各大顶会和实际应用中被广泛使用，是隐私计算场景中最常用的一种同态加密方法。

Paillier 同态加密算法作为一种部分同态加密算法，支持加法和常数的乘法运算，无法进行指数运算，而逻辑回归损失函数和梯度计算公式中存在指数运算，因此使用 Paillier 同态加密需要对原公式进行改造，本文使用泰勒展开进行近似，将指数运算转换为多项式乘法，同时为尽量少的交付己方数据，对损失函数进行了特征分解。

首先，对式（4-11）所示标准损失函数进行了化简，得到下式（4-13），其中具体的化简推导细节，在附录 B 中进行了详细展示。

$$J(\theta)=\frac{1}{n}\sum_{i=1}^{n}\left[log(1+e^{-\theta^{T}x_{i}})+(1-y_{i})(\theta^{T}x_{i})\right] \tag{4-13}$$

使用式（4-14）对式（4-13）中的对数指数部分进行二阶泰勒展开，结果如式（4-15）：

$$log(1+e^{-z})=log2-\frac{1}{2}z+\frac{1}{8}z^{2}+o(z^{2}) \tag{4-14}$$

$$log(1+e^{-\theta^{T}x_{i}})\approx log2-\frac{1}{2}\theta^{T}x_{i}+\frac{1}{8}(\theta^{T}x_{i})^{2} \tag{4-15}$$

得到展开后损失函数：

$$J(\theta)=\frac{1}{n}\sum_{i=1}^{n}\left[log2-\frac{1}{2}\theta^{T}x_{i}+\frac{1}{8}(\theta^{T}x_{i})^{2}+(1-y_{i})(\theta^{T}x_{i})\right] \tag{4-16}$$

逻辑回归模型训练过程中，为避免参数过度拟合，通常引入正则化方法，在原损失函数的后面加上正则项来惩罚特征权重，使模型更简单。正则化处理又分为 L1 正则和 L2 正则，L1 正则化增加了所有权重 θ 参数的绝对值之和，使更多的 θ 参数为零，从而使其特征变的稀疏，实现特征选择。而 L2 正则化增加所有权重 θ 参数的平方和，使 θ 尽可能接近零但不等于零，其逼近速度远低于 L1 正则化。未加入正则化的拟合函数，会出现在很小的区间中，函数值变化范围较大，从而某些参数权值较大，L2 正则化的加入惩罚了其权重变大的趋势。对于绝对值较大的权重进行很重的惩罚，而绝对值很小的权重给予非常小的惩罚。所以，一般来说，L1 正则化导致特征稀疏，可以用于特征选择。L2 正则化防止模型过拟合。本文在模型训练中使用 L2 正则化避免模型过度拟合。

下式（4-17）为加入 L2 正则化的损失函数，其中正则参数为 λ。

$$J(\theta)=\frac{1}{n}\sum_{i=1}^{n}\left[log2-\frac{1}{2}\theta^{T}x_{i}+\frac{1}{8}(\theta^{T}x_{i})^{2}+(1-y_{i})(\theta^{T}x_{i})\right]+\frac{\lambda\theta^{2}}{2n} \tag{4-17}$$

纵向联邦学习中，由于数据分属两方所有，所以引入对特征值的分解，X 拆分为

X_A 和 X_B。

$$X = [X_A|X_B] \tag{4-18}$$

$$\theta^T X = \theta_A^T X_A + \theta_B^T X_B \tag{4-19}$$

进一步得到特征分解后损失函数，为简化表达，去掉求和符号，使用 X_A 表示参与方 A 所有特征，使用 X_B 表示参与方 B 所有特征。

$$J(\theta) = \frac{1}{n}\left[log2 - \frac{1}{2}(\theta_A^T X_A + \theta_B^T X_B) + \frac{1}{8}(\theta_A^T X_A + \theta_B^T X_B)^2 + (1-y)(\theta_A^T X_A + \theta_B^T X_B) + \frac{\lambda(\theta_A^2 + \theta_B^2)}{2}\right] \tag{4-20}$$

之后对双方各自参数求偏导计算梯度：

$$\frac{\partial J(\theta)}{\partial \theta_A} = \frac{1}{n}\left[X_A\left(\frac{1}{4}\theta_A^T X_A + \frac{1}{4}\theta_B^T X_B - y + \frac{1}{2}\right) + \lambda\theta_A\right] \tag{4-21}$$

$$\frac{\partial J(\theta)}{\partial \theta_B} = \frac{1}{n}\left[X_B(\frac{1}{4}\theta_A^T X_A + \frac{1}{4}\theta_B^T X_B - y + \frac{1}{2}) + \lambda\theta_B\right] \tag{4-22}$$

最终，如式（4-21）和式（4-22）所示，参与方 A 计算梯度，需要从参与方 B 获取中间值 $\frac{1}{4}\theta_B^T X_B - y + \frac{1}{2}$，定义为 U_B；参与方 B 计算梯度需要从参会方 A 获取中间值 $\frac{1}{4}\theta_A^T X_A$，定义为 U_A。

经过以上公式推导准备，双方数据即可通过纵向逻辑回归方案进行模型训练，在本文方案中，除提供特征数据的参与方 A 和参与方 B 以外，引入一个受信赖的第三方 C，默认其是诚实的，不提供中间数据，仅协助建模过程中间数据的计算。而对于参与方 A 与参与方 B 为诚实但好奇的。

结合方案流程图 4.3所示，具体的异常检测联合模型训练步骤如下：

第一步：在双方数据对齐的基础上，参与方 A 和参与方 B 各自准备其参与异常检测的特征数据，并初始化各自权值参数 θ。

第二步：第三方 C 生成 Paillier 密钥对，将其公钥分发给双方，私钥保留在本地。

第三步：双方各自同时计算中间值。其中 A 方作为无标签方，计算中间值 U_A 和 Z_A^2 用作梯度和损失值计算，并使用 Paillier 加密算法对其进行加密，[[x]] 表示使用 Paillier 加密结果。其中 B 方作为有标签方，计算中间值 U_B。双方将其中间结果发送至对方。由于对明文进行了加密，双方仅持有公钥，所以无法获得对方明文信息。

第四步：双方根据对方中间结果，进行各自梯度计算。其中由于中间结果为加密状态值，最终梯度信息依然为 Paillier 的加密内容，需要发送至第三方进行解密。为避免拥有私钥的第三方获取梯度信息，双方在各自梯度加密结果上又增加了一个随

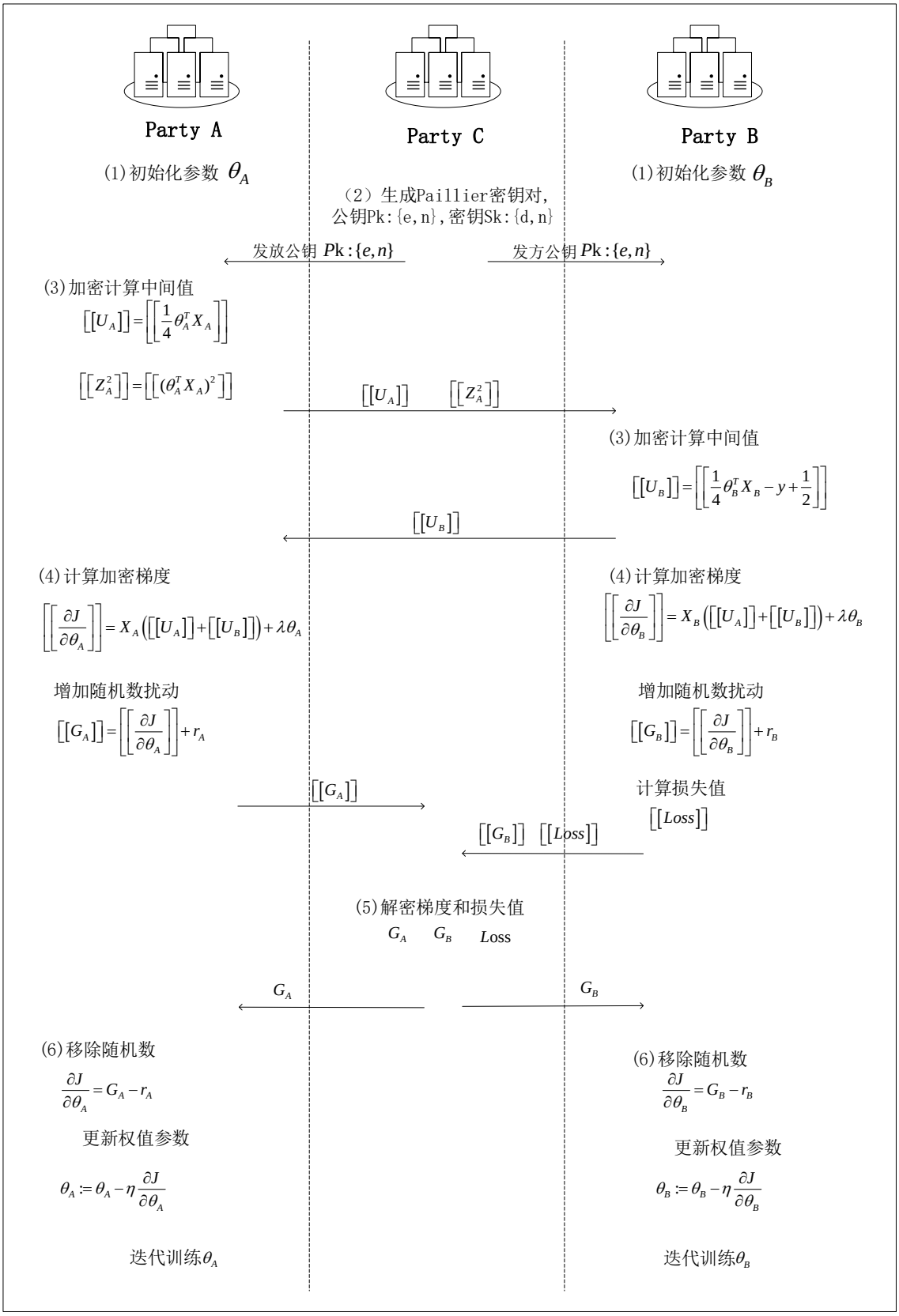


图 4.3　基于 Paillier 同态加密的纵向逻辑回归方案

机数掩码，进行信息扰动。其中，由参与方 B 进行损失值 [[Loss]] 计算，于本文附录 B 进行了详细的计算过程公式推导。

第五步：第三方 C 对两方梯度值信息使用私钥进行解密，由于加入了随机数掩码，即使是解密后的密文，除已知随机数掩码的双方自己外，其他任何一方仍然无法获取有效信息。同时该过程中，第三方会进行损失值的判断，来控制是否继续训练。

第六步：当参与方 A 与参与方 B 得到解密的梯度明文中间结果后，移除各自的随机数掩码信息。从而得到各个的梯度值。然后通过梯度更新函数进行各自梯度的更新。至此，一次训练过程结束，双方利用对方数据进行了各自参数的训练。依次执行第三至六步，直到满足停止训练要求。

最终，在参与双方未泄露原始数据的基础上，双方利用对方数据信息内容进行了各自参数的训练。在之后的检测过程中，只需向对方提供各自特征和参数计算后的概率值，双发进行概率值的求和即可进行异常的检测。

4.2.4 安全性分析

本节对纵向逻辑回归日志异常检测方案进行安全分析，在模型训练过程中假设各参与方为诚实但好奇的，它们遵循联邦学习的安全协议，不会以任何方式篡改协议；但它们试图从模型训练过程中得到的任何中间结果中获取其他参与方的真实数据信息。诚实但好奇的假设在本文的方案中是合理的，因为参与方都有训练能够实现高效日志异常检测模型的动机。

表 4.1 各参与方可获得数据信息

不同阶段	参与方 A	参与方 B	参与方 C
数据对齐阶段	Pk, D_B, I	Pk, Sk, u_i, I	
模型训练阶段	Pk , $[[U_B]]$	Pk, $[[U_A]]$, $[[Z_A^2]]$	Sk, $\frac{\partial J}{\partial \theta_A} + r_A$ $\frac{\partial J}{\partial \theta_B} + r_B$

在数据对齐阶段，采用 RAS 加密和哈希盲签的方式进行数据标签的求交，RSA 算法作为一种非对称性加密算法，其安全性基于分解大整数困难性的假定。RSA 密钥长度介于 1024~2048 比特之间是安全的，是迄今为止理论上最为成熟完善的公钥密码体制。

如表 4.1可见，对齐阶段参与方 A 可以获得的 D_B，为 B 方标签 ID 经过一次 RSA 加密和两次哈希盲签之后的数据；I 为 D_A 和 D_B 的交集，除了可以映射己方标签进行真实数据对齐外，无法从 D_B 中获取其他标签信息。而 B 方虽然持有私钥，但是对于加入随机数掩码计算结果的 u_i，依然无法获取任何有效信息。

面对外部攻击，即使传输过程中通讯数据泄露，由于各种中间结果均为加密数据，任何敌方无法从中获得有效信息。通过分析该方案，唯一存在的可能漏洞问题源于诚实但好奇的假设，假设 A 方不诚实，当 A 方计算获得交集后，可能会出现隐瞒交集的情况，造成 A、B 双方数据不对称的风险。在本方案基于诚实但好奇的假设前提下，可以忽略该类风险。

在模型训练阶段，本方案使用 Paillier 同态加密算法进行中间结果的加密，Paillier 密码系统已经被证明符合加密方案的标准安全定义[63]，其具有语义安全性，即在选择明文攻击下的密文具有不可区分性。其安全性基于合数幂剩余假设[61]，即给定一个合数 n 和整数 z，判定 z 在 n^2 下是否是 n 次剩余是困难的。所以，只要私钥没有泄露，其他任何一方无法从密文中获得任何明文信息。

在训练过程中，参与方 A、B 从第三方获取加密公钥 Pk，无法获取私钥 Sk。假设 A、B 双方的数据都是私密的，仅在第三步进行加密中间结果 $[[U_B]]$、$[[U_A]]$ 和 $[[Z_A^2]]$ 的交换，并且双方通过安全的通道进行通信。即使一方是好奇的，密文状态下的中间结果不会泄露任何信息。

第三方实体拥有私钥 Sk，在对 A、B 双方第四步提交的加密梯度进行解密后，由于参与训练双方引入了随机数掩码 r 的扰动，第三方无法获得真实的梯度信息。因此，在训练过程中第三方无法从训练双方获得任何信息。

外部攻击；在秘钥安全的前提下，假设外部攻击劫取中间传输信息，但是由于第三、四步中间结果是加密信息，且第五步的梯度为加入随机数掩码的噪声信息，因此，外部攻击者无法从中间过程获取任何有效信息。

4.3 实验评估

以上为本文提出的纵向联邦日志异常检测的具体数据对齐和模型训练的理论方案，该部分则通过实际数据对纵向逻辑回归的实施流程和检测效果进行评估，具体内容包括实验数据集处理及其特征提取详细内容的方案实施细节。

4.3.1 数据集

如上文所述，真实的系统日志数据具有较高的机密性，获取大量的真实日志数据存在一定的困难。并且对于纵向联邦场景，需要不同数据特征由多方同时持有。目前所有日志数据中，未能找到能很好的完全满足当前场景的适用数据集。本文通过对已知的公开数据集进行改造处理，来构造适用于纵向联邦场景的日志数据集，从而进行当前方案的验证。具体通过将不同日志事件信息分为不同组别，来模拟日志信息由不同参与方产生的纵向联邦场景。对数据集的构造在下一小节数据预处理部分进行详细说明，以下为选取的三个可用数据集信息，表 4.2统计了三个数据集的基本情况。

表 4.2 数据集详情

数据集	数据大小	数据总量	时间跨度	异常情况	异常占比
HDFS	1.46GB	11,175629	38.7 小时	16838(块异常)	2.93%
BGL	708MB	4,747,963	215 天	348460(日志异常）	7.34%
HW	332MB	524,053	287 天	5267(序列异常)	9.21%

HDFS 数据集：该数据集是目前日志和异常检测使用最多的一组数据。该数据集收集于 200 多个亚马逊的 EC2 节点上运行的 Hadoop 实例产生的日志文件。共有 11,175,629 条日志，每一个日志条目都可以由唯一的块地址 blk_id 进行区别，领域专家对其进行了异常的标记，大约 2.93% 是异常的，所以可以使用会话分组对其进行特征分组。该数据集有 29 种日志事件，行为事件相对稳定，所以被多数检测方法用来进行模型的评估。

BGL 数据集：该数据集包含 4747,963 条日志信息，这些信息是由劳伦斯利弗莫尔国家实验室的 BlueGene/L 超级计算机系统记录的。与 HDFS 数据不同，BGL 日志没有为每个作业执行记录标识符。因此，需要使用固定窗口或滑动窗口将日志切片为日志序列，在 BGL 数据中，有 348,460 条日志消息被标记为失败。该数据集含有的日志事件种类有 300 多种，并且时间跨度 215 天，多被用来进行日志偏移挑战方面的检测模型所使用。

HW 数据集：该数据集为在参与某公司实际项目中使用的脱敏日志数据，是某公司在真实系统运行环境下采集到的用户行为日志数据。因其机密性质，本文仅对其进行实验测试，不对数据详细内容进行公开。该数据集有 524,053 条日志记录，时间跨度 287 天，包含 26 种日志行为。由于数据集拥有用户名标签，所以采用了对其进行会话 + 滑动窗口的分组方式，将其某个用户在某段时间内的行为进行了序列分组。该数据集异常数据由手动标注和构造生成，异常数据占比 9.21%，可用作进行系统用户异常行为的检测研究。

4.3.2 数据预处理

（1）数据准备

为满足纵向联邦实验条件，本文对三个数据集进行了拆分构造，将同一数据集的不同日志信息根据事件类型分为两组，分别由两方单独持有，数据标签交由其中一方持有。以此模拟纵向联邦学习场景中，不同日志事件特征分属不同参与方产生和独有条件。

具体的，三个数据集均为含有异常 label 信息的数据集，本文通过基于数据集的前置信息，折半普通日志事件行为特征，将数据交由两方持有，而标签数据交由其中

一方（本文统一定义 B 方持有标签数据）。真实场景下双方无法获知对方具备特征详情和特征数量。本文默认双方持有数据特征均参与模型训练，但在其他场景或真实条件下，可能会出现参与方持有的部分特征无法适用于模型训练的情况，需要进行特征选择，本文此处不再对其进行讨论，默认所有特征均参与模型训练。

表 4.3　数据集拆分构造详情

数据集	特征总数	A 方持有特征	B 方持有特征
HDFS	29	15	14+label
BGL	373	187	186+label
HW	26	13	13+label

（2）日志解析

对于上述三个数据集，本文均采用 HRTree 解析方法进行日志解析。具体的，参与双方分别于本地对各自拥有数据使用 HRTree 方法进行解析，如表 4.3所示。对于 HDFS 数据集，参与方 A 解析出 15 类日志行为事件，参与方 B 解析出 14 类日志行为事件；对于 BGL 数据集，参与方 A 解析出 187 类日志行为事件，参与方 B 解析出 186 类日志行为事件；在 HW 数据集上，参与方 A 解析出 13 类日志行为事件，参与方 B 也解析出 13 类日志行为事件。

（3）数据分组及 ID 制定

日志事件解析之后，日志条目被解析成含有具体行为信息的日志事件和参数信息。异常检测并非对单条日志信息进行内容检测，而是对系统内一段时间或者某个节点的一组日志序列集合进行检测。因此采用数据分组进行异常检测的粒度划分，在纵向联邦检测中，需要保证分组粒度的一致性，使双发在同一粒度下进行异常检测，不同的分组粒度引入了不同的数据集合标识，即 ID 信息，同时需要进行数据对齐。

此处，本文对三个数据集采用了不同的数据分组和序列标识方案。如 4.2.1 节所述，对于 HDFS 数据集，采用了 blk_id 的方式进行日志序列信息的采集，那么双方即可根据各自的 blk_id 进行特征标识，这样直接对 blk_id 通过隐私集合求交即可实现数据对齐，此情况为最理想的日志异常检测分组粒度。而对于 BGL 数据集，由于没有具体的标签信息标识，只能进行以时间窗口为条件的数据分组。本文对其进行了窗口长度为 6 小时和步长为 1 小时的滑动窗口分组序列采集，通过协商起始时间点，进行相同粒度的滑动窗口数据采集。对于 HW 数据集，由于其中含有用户名 ID 信息，但是其时间跨度为 200 多天，仅使用用户名标识，会导致一个用户行为序列时间跨度较长，并不满足真实的异常检测情况。本文将相同用户采用滑动窗口的分组方式，对同一用户不同时间点的日志行为序列进行了分组采集。因此采用的标识方案为“用

户名 ID+ 时间戳”的组合方式。由于 HW 数据集较为稀疏，滑动窗口长度设置为 12 小时，步长为 6 小时。

在纵向联邦方案中，需要保证参与双方分组方案的一致性，参与双方可通过提前协商进行分组方案的确定。

（4）日志特征提取

在对日志数据进行分组后，需要进行具体模型训练特征的提取。在 2.3 小节中，介绍了日志异常检测常用的三种特征表征方式，分别为日志索引序列特征、日志事件计数向量特征和日志语义信息向量特征。其中，日志索引序列对日志事件进行索引编码，并按照时间顺序进行排序，具有明显的事件上下文关系。日志事件计数向量特征在索引编码日志事件之后，对每种日志事件发生的次数进行统计，没有时间顺序关系的约束。日志语义信息向量特征，对每一个日志事件进行了语义向量的编码，并按照时间顺序关系进行了排序，通过语义上下文的理解检测异常内容。

在纵向联邦学习方案中，由于双方无法获取对方的具体信息内容，因此对于具有明显顺序关系的两类特征很难实现，并且采用索引序列和语义信息的模型方法多为神经网络模型。目前纵向联邦方案中，对于深度神经网络模型的纵向联合训练还未有较为合适的解决方案。基于本文的纵向逻辑回归模型，此处异常检测参与双方采用提供日志事件计数向量特征参与模型训练的方案。

参与异常检测模型训练双方，统计一个检测分组粒度内每种事件发生的次数，构成行为事件计数向量。将每一类行为事件发生的次数，作为一类特征信息。

4.3.3 模型训练

本文通过单机模拟，进行了两方纵向联邦学习模型的训练过程。具体的，本文通过内存直接进行了数据交换，未对纵向联邦中参与方数据通信进行单独构建。除此之外，其余流程均模拟纵向联邦训练过程，参与方仅进行中间结果的交换，对于双方真实数据通过 Paillier 同态加密算法实现交换，完全模拟纵向逻辑回归模型的参数训练步骤。本章节所有实验测试使用设备环境同 3.4.1 节一样。

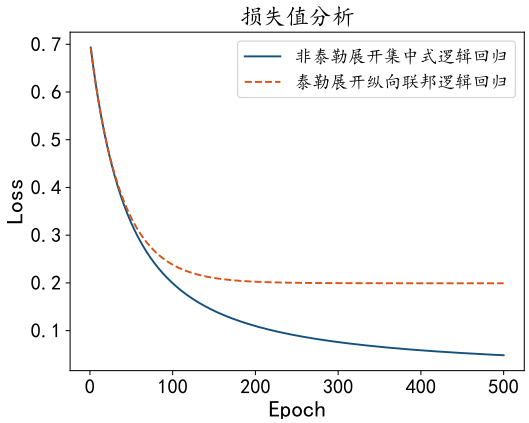
参与训练双方，各自使用具有 ID 标识的行为事件计数向量特征，通过纵向逻辑回归模型，训练可以对其进行异常检测判断的回归模型参数。本文对三个数据集分别进行了模拟训练。数据使用情况如表 4.4所示，其中 HDFS 数据集由于数据量较大，并且数据较为稳定，因此只使用了 10% 的数据参与训练。

为衡量泰勒展开对纵向联邦学习方案的影响程度，本文对纵向联邦学习和非联邦学习的集中式逻辑回归模型的收敛情况进行了测试。如图 4.4所示，两种方案在三个数据集上损失值的收敛情况对比。可以看出异常检测纵向联邦学习方案收敛速度

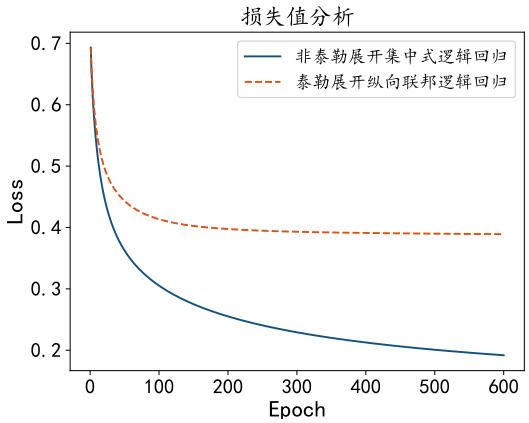
表 4.4　训练测试数据使用情况

数据集	序列数	训练数据占比	测试数据占比
HDFS	575061	10%	90%
BGL	5150	50%	50%
HW	57210	50%	50%

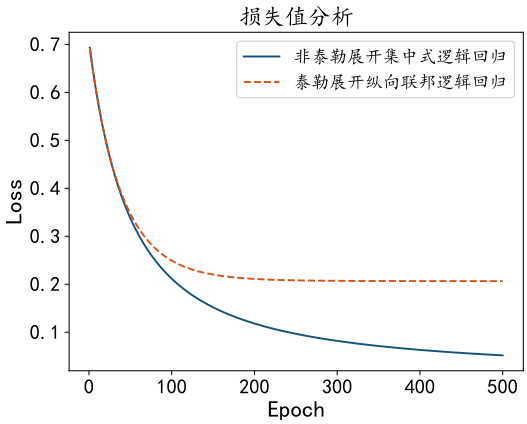
快于非纵向联邦学习，且其损失值略微高于非纵向联邦方案。符合纵向联邦学习进行各自局部参数更新并且损失函数近似到二项式的前提条件[11]。其中 HDFS 数据集和 HW 数据集特征数量相近，因此纵向联邦方案的损失曲线走势相同、差值相近。而 BGL 数据由于 300 多种的特征数量，并且其异常类型并不稳定，所以两种方案损失差值略大于其他两个数据集。



(a) HDFS



(b) BGL



(c) HW

图 4.4　集中式与纵向联邦模型训练损失值对比

4.3.4 检测结果对比分析

异常检测作为一种二分类任务，本文主要使用二分类模型的评估指标进行部分性能的测试，由于暂无其他纵向方法进行对比，所以本文仅对纵向联邦和非联邦的逻辑回归进行对比。主要进行如下指标的评估：准确率、召回率、F_1-$measure$ 指数、ROC 和 AUC。此处借助表 4.5分类结果的混淆矩阵，进行部分指标的解释。此处用正类表示异常，负类表示正常。

表 4.5 分类结果混淆矩阵

		预测结果	
		正类	负类
真实结果	正类	TP (真正类)	FN (假负类)
	负类	FP (假正类)	TN (真负类)

真正类（True Positive, TP）：实际为正类，且被模型预测为正类的样本。真实为异常被模型准确检测为异常的样本。

假正类（False Positive, FP）：实际为负类，但被模型预测为正类的样本。真实为正常被模型错误检测为异常的样本。

真负类（True Negative, TN）：实际为负类，且被模型预测为负类的样本。真实为正常被模型正确检测为正常的样本。

假负类（False Negative, FN）：实际为正类，但被模型预测为负类的样本。真实为异常被模型错误检测为正常的样本。

（1）精确率和召回率指标评估

通过混淆矩阵，可以计算二分类任务的精确率和召回率，以及二者的调和指标 F_1-$measure$ 指数。其中，精确率又叫查准率，表示检测为异常的样本中，真实的异常所占的比例。召回率又称查全率，表示检测正确的异常在全部异常中的占比。为了同时调和精确率和召回率，使用 F_1-$measure$ 指数调和两个指标，较高的 F_1-$measure$ 指数能够保证精确率和召回率都比较高。

定义如下：

$$Precision = \frac{TP}{TP + FP} \tag{4-23}$$

$$Recall = \frac{TP}{TP + FN} \tag{4-24}$$

$$F_1\text{-}measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4-25}$$

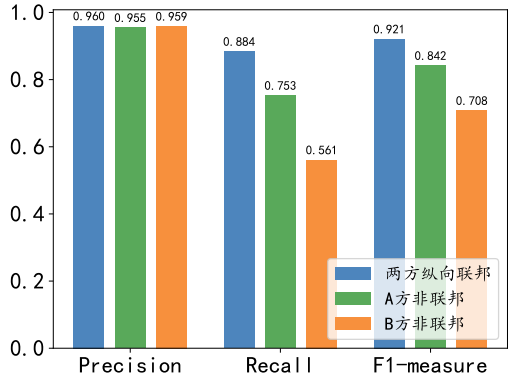
表 4.6　联邦和集中式模型测试结果对比

数据集	模型	Precision	Recall	F_1-measure
HDFS	集中式	0.960	0.886	0.922
	联邦	0.960	0.884	0.921
BGL	集中式	0.979	0.784	0.871
	联邦	0.984	0.712	0.826
HW	集中式	0.992	0.977	0.985
	联邦	0.998	0.971	0.984

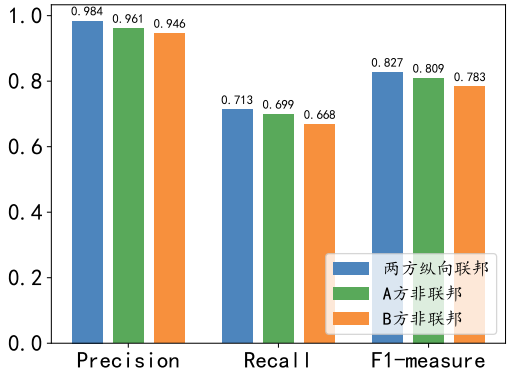
首先，本文对经过泰勒展开的联邦模型和非联邦的集中式逻辑回归模型，在三个数据集上进行了检测效果的对比测试。如表 4.6所示，纵向联邦逻辑回归和非联邦集中式逻辑回归在三个数据集上均展现了较为理想的检测效果，由于泰勒展开式的引入，纵向联邦相较与集中式在相同数据集上，各类指标仅有微小降低，未明显影响其实际检测效果，其中 HDFS 数据集和 HW 数据集由于其数据特征数量较少，并且异常种类较为稳定，所以在联邦方案中仅有 0.001 的 F_1-$measure$ 损失，而 BGL 由于其数据特征众多，模型参数较多，在 F_1-$measure$ 指数上有 0.045 的损失。

通过表 4.6可以看出，HW 数据集在三个指标上均展现了较为突出的检测结果，这是因为 HW 数据集异常类型较为稳定，且多为用户行为事件数量关系较为明显的异常，所以在使用日志事件计数向量特征时就能够实现较为理想的检测效果。而对于 HDFS 和 BGL 数据集，在逻辑回归原理的检测方法中，无论联邦和集中式模型均有略低的召回率指标，通过分析异常检测的结果，发现对于部分异常未能实现检测，主要原因在于异常内容在仅使用日志事件计数向量特征的时候无法进行检测，例如部分异常表现为前后文的顺序关系异常和参数值异常，这也暴露了当前检测方案的局限性，由于纵向联邦逻辑回归的限制，对于其他部分异常特征无法使用，导致部分异常类型无法进行准确识别。

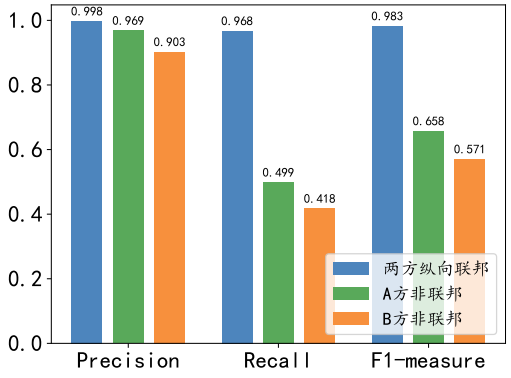
除以上结果对比外，本文还模拟了数据特征分属不同参与方，各参与方仅使用本地数据特征进行模型训练检测效果与纵向联邦模型的检测效果对比，此时默认双方均含有数据 label 标签辅助训练。如图 4.5所示，在三个数据集上，纵向联邦的日志异常检测方案，各项指标均优于单独使用一方特征进行异常检测的模型，充分证明了纵向联邦模型，可以利用多方数据特征，可以增强模型鲁棒性的特点。同时通过观察 A 方非联邦和 B 方非联邦两方仅使用各自特征进行模型训练的检测效果，可以看出两个参与方虽然使用了相同数量的特征，但是训练模型的检测效果却有较大的不平衡性，异常检测效果差异明显。而通过联合模型的训练，可以充分解决双方数据的不平衡问题，从而实现明显优于双方更高的检测效果。



(a) HDFS



(b) BGL



(c) HW

图 4.5　两方联邦训练和非联邦单独训练指标对比

（2）ROC/AUC 指标评估

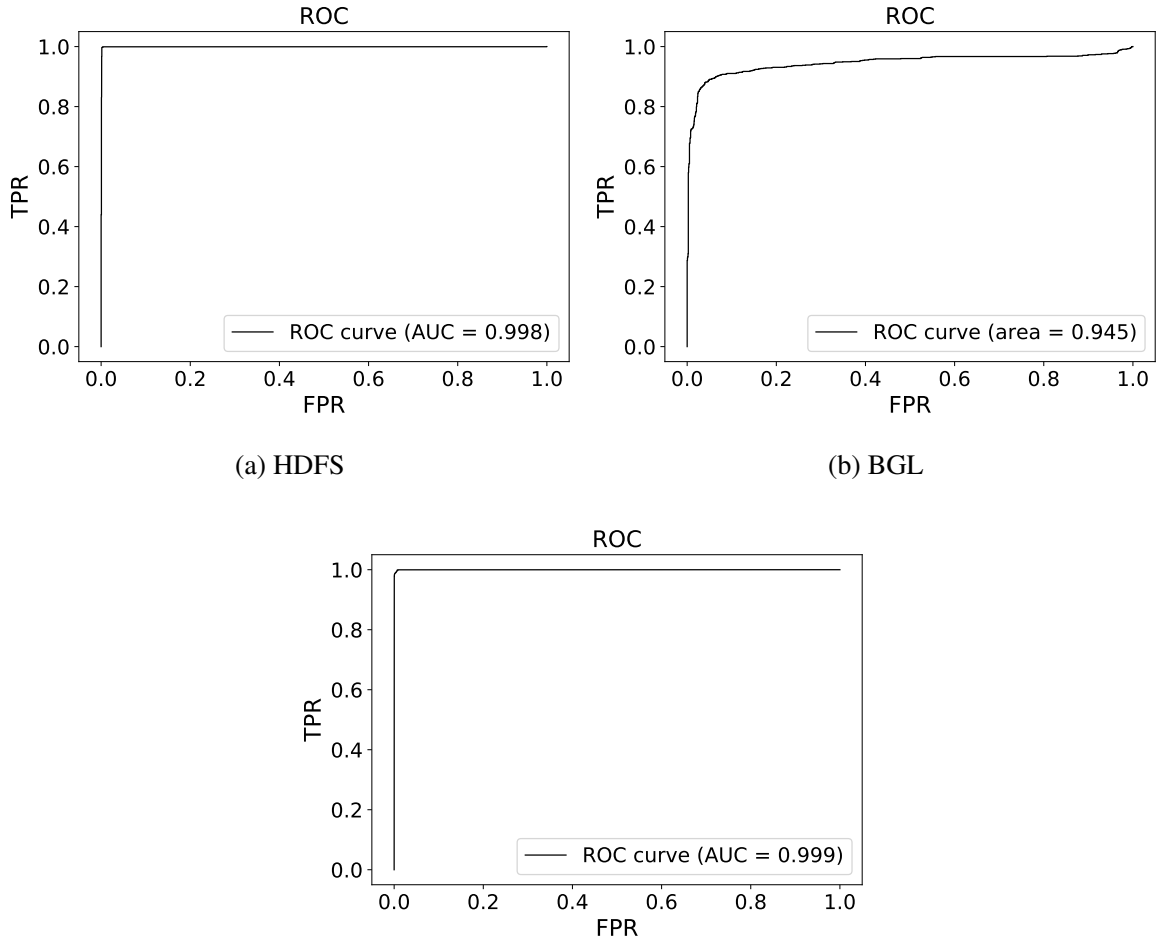
为更全面评估该模型在日志异常检测领域的性能，本文还引入了 ROC“受试者工作特征”（Receiver Operating Characteristic）曲线，评估分类模型性能的指标。由于逻辑回归模型由 Sigmoid 函数输出为一个概率值，只是在通常的使用中，习惯将小于 0.5 的概率分为 0 类，大于等于 0.5 的分为 1 类，所以此处存在一个阈值。为了更好的评估模型的分类效果，ROC 曲线刻画不同阈值下的真正类率 (True Positive Rate, TPR) 和假正类率 (False Positive Rate, FPR) 的关系。ROC 曲线是以假正类率（FPR）为横坐标，真正类率（TPR）为纵坐标的曲线。AUC(Area Under ROC Curve) 为 ROC 曲线下的面积，用来评估分类器的性能，AUC 越大，分类器性能越好。

定义如下：

$$TPR = \frac{TP}{TP + FN} \tag{4-26}$$

$$FPR = \frac{FP}{TN + FP} \tag{4-27}$$

本文对纵向逻辑回归联邦学习在三个日志上进行了 ROC 曲线和 AUC 的计算，结果如图 4.6所示。此处纵向联邦逻辑回归模型的 ROC 曲线在 HDFS 数据集和 HW 数据集上展现了近乎完美 ROC 曲线效果，AUC 甚至达到了 0.99 以上。在 BGL 数据集上，ROC 曲线也较为理想，AUC 达到了 0.945。这意味着在该类任务中，逻辑回归模型在三个数据集上能够正确分类绝大多数的正常和异常。



(a) HDFS

(b) BGL

(c) HW

图 4.6　纵向联邦模型在三个数据集上的 ROC 曲线

4.3.5 时间效率评估

由于引入了各种加密算法进行数据隐私保护，因此该方案在实施过程中具有一定的时间成本损耗。该部分对本方案进行时间效率评估，通过分析纵向联邦逻辑回归的日志异常检测方案，时间消耗主要在数据对齐和模型训练阶段。下面分别进行时间

效率的实验测试和评估。

数据对齐阶段 其中数据对齐引入了 RSA 加密和哈希处理过程，为模拟数据标签的不一致，参与两方各自随机从构建标签数据中丢掉了 10% 的数据，构建真实的求交对齐过程。HDFS 数据集标签为“blk_id”，拥有 21∼24 个字符长度；BGL 数据集使用 13 位时间戳作为标签，由窗口开始时间戳和结束时间戳共 26 位组成；HW 数据集使用用户名和时间戳组合方式构成标签，由长度为 5∼8 个字符的用户名字符和 26 位窗口时间戳组成。

通过在三个数据集上进行实测，结果如表 4.7，整个数据耗时可以拆解为两个主要步骤，其中基于 RSA 的求交过程使用 1024 位秘钥进行加密，结果表明求交过程为线性耗时，三个数据集上求交的平均耗时差距较小；而对齐过程，需要对每组数据与原始列表进行依次对比，从而确定具体对齐位置，时间复杂度为 $O(n^2)$。三个测试数据集的数据量不同，具有较多数据的 HDFS 数据集在平均对齐操作中拥有更长的耗时。

表 4.7 标签求交对齐耗时对比

数据集	标签字符数	数据量	总用时	平均用时	求交耗时	求交平均耗时	对齐耗时	对齐平均耗时
HDFS	21∼24	517554	8558.905s	16.537ms	264.735s	0.512ms	8294.136s	16.026ms
BGL	26	4040	2.624s	0.650ms	2.127s	0.526ms	0.497s	0.123ms
HW	31∼34	51485	109.231s	2.121ms	27.001s	0.524ms	82.227s	1.597ms

模型训练阶段 由于在双方参与的训练过程引入了 Paillier 同态加密算法，此处同样采用 1024 位密钥对中间结果进行加密交换计算，相比于传统的逻辑回归方案，引入了更多的时间成本。对三个数据集的训练耗时统计如表 4.8，通过对比可以看出，两者训练时长差距在 10^3 数量级左右。纵向逻辑回归的联邦日志异常检测方案将传统逻辑回归的计算方式进行分步计算，增加了计算的步骤。但是其中需要注意的是，传统的逻辑回归在模型训练之前需要数据汇总，存在较大的通讯成本，而联邦方案仅需在训练过程进行参数和梯度的交换。由于本文未对通讯过程进行模拟，所以未能展示通讯成本方面的时间效率，但是该方面也是分布式联邦场景部署时需要考虑的一个重要问题。

表 4.8 联邦和非联邦模型训练时间对比

数据集	数据量	数据维度	联邦耗时/epoch	非联邦耗时/epoch
HDFS	57506	30	275.335s	216.167ms
BGL	2575	374	88.397s	8.010ms
HW	28605	27	136.799s	71.857ms

总体来说，纵向联邦引入数据对齐和加密的中间结果训练过程，在模型训练时间

上有所损耗。但是纵向联邦的异常检测方案省去了大量原始日志数据的传输通讯成本，在该层面是有所提高的，同时加密方案解决了数据泄露的问题，保证了数据隐私安全。可以在更多场景下进行日志异常检测模型的部署，更好的帮助解决日志异常检测的数据不平衡问题。

4.4 本章小结

本章首先对纵向联邦日志异常检测问题进行了阐述，分析了纵向联邦学习的特点以及在日志异常检测领域的研究前景。之后，本章重点提出了一种基于纵向逻辑回归的日志异常检测方案，并对其进行了全流程的方案设计。包括提出了一种基于RSA 哈希加密的隐私集合求交对齐技术，实现待检测数据序列对齐；再通过改造传统逻辑回归模型，使用 Paillier 同态加密算法仅进行中间结果的交换，在不泄露原始数据的基础上，实现联合检测模型的训练。

最后，本章通过使用三个数据集，通过详细的数据预处理、日志解析、数据分组、ID 标签制定、特征提取和模型训练等流程，对日志异常检测的纵向逻辑回归方案进行了具体实施测试。并通过各类指标对其进行了评估，在精确率、召回率和 ROC 曲线等指标上，相比于单方的异常检测，本文的纵向逻辑回归异常检测方案均取得了较为理想的检测结果；在时间效率上，联邦方案需要付出更多的时间成本，但是可以做到数据的隐私保护。以上多个方面证明了纵向联邦学习的日志异常检测方案的可行性。

第五章　可视化应用系统开发及实现

本文前面章节详细介绍了日志异常检测的具体流程框架，包括多种日志解析方法、特征提取方式和基于机器学习、深度学习的多种检测模型。本文对以上内容进行了复现，并且对整个日志异常检测流程进行了规范化设计。本章将结合实际应用，整合各种解析方法和检测模型，实现一个可视化日志异常检测应用系统的开发。

5.1 需求分析

目前已涌现的多种数据解析和检测模型，对于非专业人员的实际应用，依然存在一定的技术障碍。一方面部分算法内容具有一定的技术门槛，实现起来存在较大的挑战；另一方面，目前日志异常检测的相关文献研究仅局限于其中的某个环节，对于全流程日志异常检测的实现，缺失关键逻辑步骤。因此，对于非专业人员使用目前先进的检测技术存在较大的技术挑战。本章着眼于实际应用，通过对算法步骤的可视化开发，能够逻辑清晰地使用已有部分检测模型算法。并且实现当面对不同数据集时可以进行模型的选择和调整，使其具有较高的适用性。同时通过可视化的开发，将关键数据进行展示，获取和了解关键数据有利于对异常检测情况的掌握。

现对具体系统需求进行以下分析：

(1) 日志采集与预处理功能需求

该系统需要能够采集不同来源的日志数据，支持从多个数据源和日志格式中收集数据，并将其转换为可供后续处理的统一格式。并对其进行预处理，包括去除重复日志、缺失数据的填充等操作。并且需要提供一个数据仓库来存储预处理的数据，以便在后续的处理和分析中使用。

(2) 日志解析功能需求

该系统需要能够对采集到的日志数据进行解析，并且能够支持解析多种数据格式，并将其转换为结构化的日志记录内容和日志行为事件参数组合。这些记录应该包括时间戳、源地址、目标地址、请求类型等信息，以便后续的分析和建模。该系统应该支持自定义解析规则和正则表达式，以适应不同的日志格式和数据源。

(3) 日志特征提取功能需求

该系统应该能够从解析后的日志事件记录中提取特征，可以实现不同分组技术和不同的特征转换方式，支持日志行为事件计数向量特征提取、日志索引序列提取和日志语义向量提取等多种特征提取方式，以便不同检测模型使用。

(4) 异常检测模型训练和测试功能需求

该系统应该支持多种异常检测算法和模型，如基于规则的方法、聚类、分类和深度学习等。支持模型的训练和测试，以及模型的调参和优化。支持实时模型推断和预测，以便及时发现和处理异常事件。

(5) 异常行为报警分析处理功能需求

应用系统需要能够根据异常检测模型的结果，实时报警异常行为，并提供详细的异常信息内容，以便管理人员及时分析和解决异常。

(6) 日志可视化功能需求

应用系统需要能够将采集到的日志数据以可视化的形式展示出来，方便用户进行观察和分析。并且对其中的部分流程的控制，以可视化的方式进行操作，例如，各种参数和阈值的设置。

(7) 其他功能需求

该系统需要能够对采集到的日志数据进行加密和权限控制，确保数据的安全性和隐私性。该系统需要具有良好的用户界面和易于使用的功能，以便用户能够方便地使用和管理系统。应用系统需要具有易于维护和管理的结构和代码，以方便系统管理员进行维护和升级。

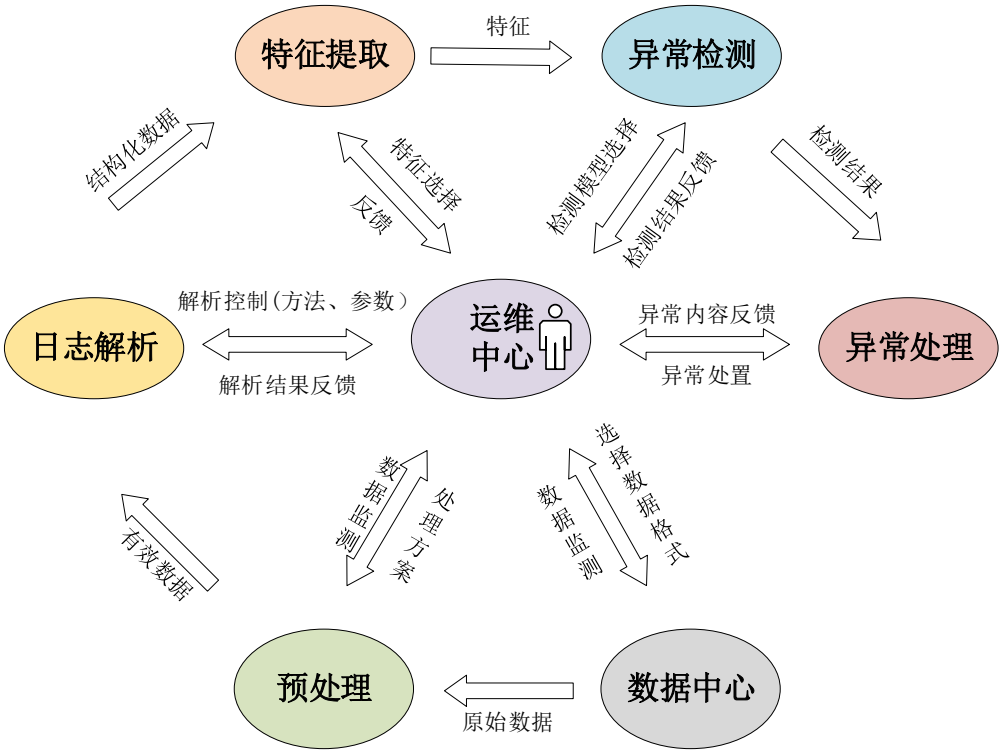


图 5.1　日志异常检测可视化系统功能关系

5.2 可视化应用系统设计

经过以上需求分析，结合日志异常检测流程，构建如图 5.1所示系统应用功能关系。各功能模块由运维中心进行整体功能控制，其中按照日志异常检测的流程，分别构建各流程模块功能。通过需求功能分析，将日志异常检测的可视化系统抽象为六层结构，如图 5.2所示，各层模块负责不同功能任务。

(1) 日志采集层

该层主要负责日志数据的采集，可支持多种数据源，如本地文本文件、数据库、API 接口数据等。对于该层主要进行接口的抽象，提供可供运维人员选择的数据接入方式，例如，本地数据直接输入本地待处理日志文件路径即可。对于数据库日志，则通过简单的数据库配置，通过底层驱动程序，实现多种数据库的连接，实现从数据库直接读取日志数据。

(2) 日志预处理层

该层主要负责原始日志数据的预处理工作，支持对数据进行过滤、清洗、格式化等预处理操作。并且通过配置文件等相关设置，能够实现数据实时或定期获取的功能。

(3) 日志解析层

该层主要部署目前已掌握的多种日志解析方法，将解析算法各类参数抽象为前端可视化控件，通过前端可视化配置，实现对不同算法的复现调用。并且对于不同日志类型格式，设置不同解析模式，同时支持用户自定义日志格式，以应对特殊情况。对于日志解析结果，能够以可视化内容展示在前端，由运维人员实时查看解析结果，并对当前已解析得到的同一系统下的日志事件模板类型进行存储。通过对历史日志事件模板信息的读取，支持在线更新解析。同时为提高解析的准确度，引入人工矫正过程，运维人员可通过前端页面对部分解析结果进行调整，对于部分解析存在偏差的日志事件进行矫正。同时多种日志解析算法的提供，可以满足对不同日志类型的解析需求，运维人员可对比分析选择最合适的解析方式。

(4) 特征提取层

该层主要负责日志异常检测序列的提取，在系统层面，可以制定相关的检测粒度，比如异常检测的周期、时间跨度等，进行不同粒度日志序列的提取。同时为满足不同检测模型的需要，设置多种特征提取方式，可以对序列进行索引序列提取、计数向量计数和语义向量映射，并且对于特征提取后数据可进行数据拆分处理，选择具体数据量大小的特征内容参与后续的训练过程。

(5) 检测分析层

该层为日志和异常检测系统的核心，其中主要提供了丰富的日志异常检测模型，包括各类传统的机器学习模型，例如，逻辑回归、主成分分析、决策树等；还涵盖了

目前多种深度学习模型，包括 DeepLog、LogAnomaly、LogRobust 等，本文对上述检测模型进行了详细的复现。可以涵盖多种检测模型，同时由于检测模型参数复杂，为方便运维人员控制，将此类参数展示到前端，可由运维人员可视化调参训练模型。对于检测的异常信息，可以及时的进行告警展示，同时为满足异常及时处理的需求，将部分可利用数据进行展示，以方便运维人员进行异常的定位和处理。

(6) 异常处理层

异常检测最重要的部分为在发现异常时及时的进行处置，该层主要对异常模型的检测点进行告警，然后为运维人员提供有效的信息内容，方便运维人员及时定位问题。同时为提高检测模型的效果，还提供了日志异常的部分处理，比如若发现为误报异常，则可将该异常进行标识，这样在后续的检测中该类误报不会继续提醒，减少告警量。另一方面，对于已经确定的异常，构建异常库，可以保存已知异常的数据特征，可以在后续的模型再训练等流程中，加强模型的检测能力。

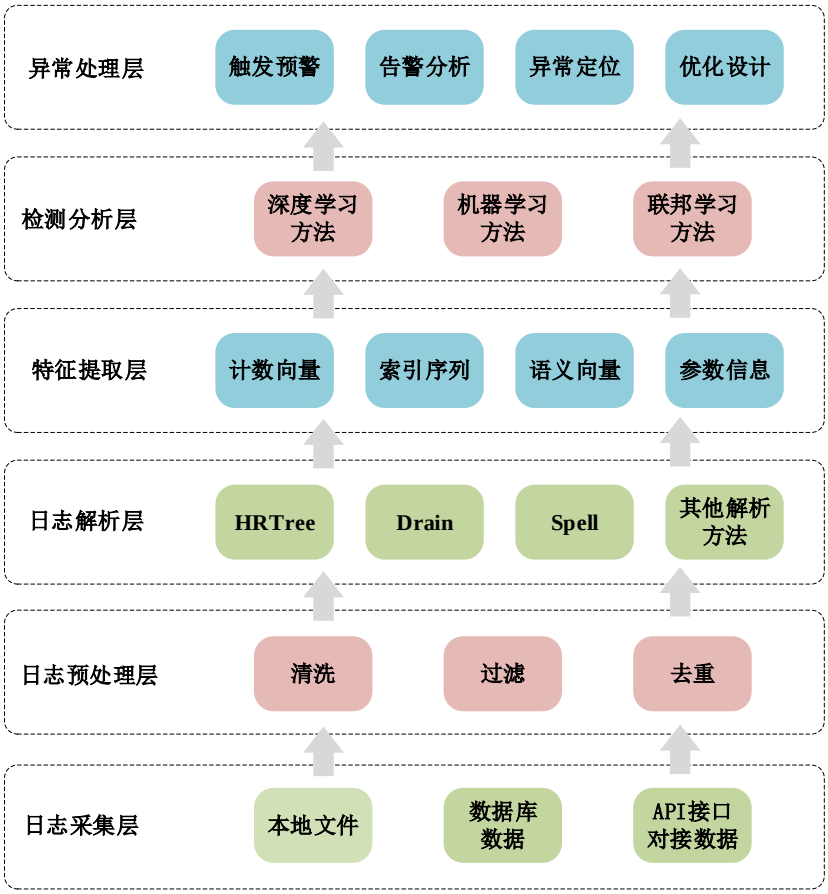


图 5.2　日志异常检测可视化系统层次结构

5.3 可视化应用系统实现

由于部分算法内容为 Python 编程语言开发，为方便各类算法工具的调用，本文采用开源的 Python Web 框架 Django 进行开发，它使用了 Model-View-Controller (MVC) 的设计模式，简化了 Web 应用程序的开发过程，同时提高了开发效率和代码可读性。

Django 具有以下特点：

（1）自带管理后台：Django 提供了一个可定制的管理后台，它可以方便地进行数据的管理和查询，可以大大提高开发效率。

（2）安全性：Django 提供了一系列的安全机制，如防止 CSRF 攻击、XSS 攻击、SQL 注入攻击等。

（3）易于扩展：Django 的模块化设计使得它非常易于扩展，可以方便地添加新的应用程序、模板、中间件等。

（4）超快的开发：Django 通过提供一系列的通用应用程序、模板和中间件等，可以快速地构建出一个完整的 Web 应用程序。

开发的主要思路为将抽象的算法步骤逻辑设计为可执行的页面逻辑，通过页面逻辑控制算法流的实施过程。对于算法需要的具体参数，通过页面进行读取，然后以接口的方式将页面数据传入要执行的算法程序，使程序执行。同时将程序执行过程进行展示，在页面内能够掌握程序执行的进度。对于有数据展示需求的页面，能够在页面对程序执行后得到的数据进行展示。

图 5.3　系统首页展示

目前系统设置有用户登录页面、系统介绍页面和内部检测流程处理页面几个主要前端页面，使用 HTML 和 CSS 开发设计。登录页面授权指定用户进行系统的登录，与后端数据库进行校验授权登录。系统介绍页面进行当前系统的功能介绍，如图 5.3所示。内部检测流程页面集合在了一个主界面中，通过功能选项进行不同流程

步骤的切换，内容包括日志解析、日志处理和日志检测几个大类模块选项，各模块下设置各类检测流程步骤。

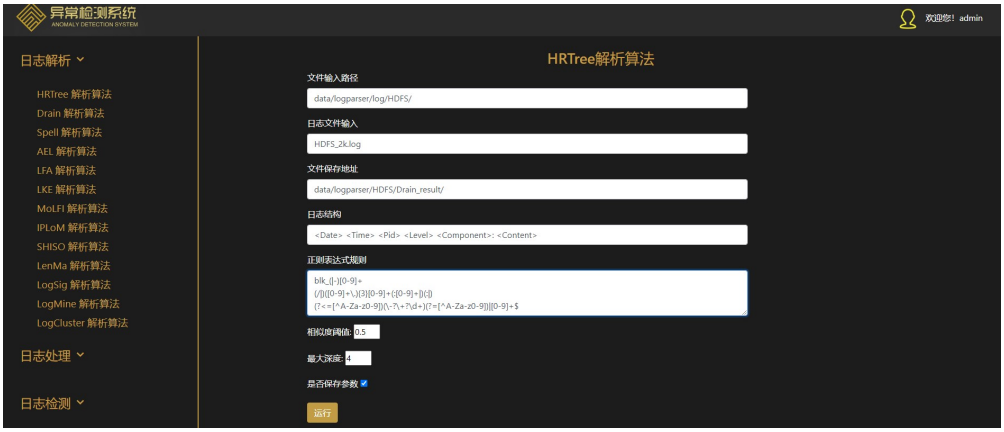


图 5.4　日志解析模块展示

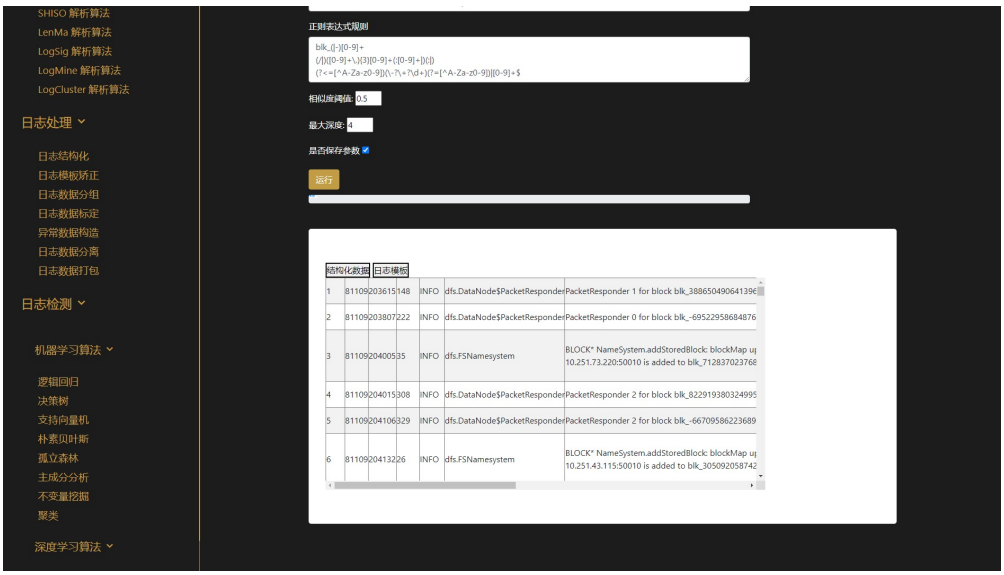


图 5.5　日志解析结果展示

日志解析模块，如图 5.4所示，本文复现了 13 种日志解析方法，支持各种类型的日志解析，并且对于各类解析方法涉及的参数内容，进行了前端参数控制的设计，用户仅需根据提示进行可视化配置即可。解析结果产生两类数据，包括日志事件模板和结构化的日志信息，前端程序可以读取部分解析结果数据，并将其展示在前端。对于解析过程，加入进度条提示，可实时查看解析进度，且支持解析程序后台运行和中断。

日志处理模块，如图 5.6所示，当前系统将日志结构化、日志数据分组、数据打包等多种日志异常检测中涉及的技术内容进行了集成，可以实现对日志检测流程中各类数据的处理，包含了日志采集层、日志预处理层和特征提取层的主体工作。

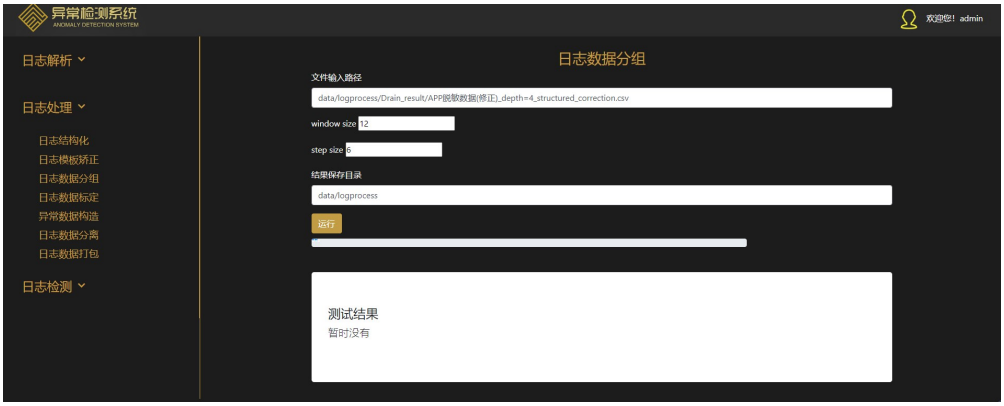


图 5.6　数据处理模块展示

异常检测模块，如图 5.7所示，当前系统集成了机器学习算法和深度学习算法中的各类检测模型，接入了 8 种传统机器学习算法；复现了 DeepLog、LogAnomaly 和 LogRobust 三种深度学习检测模型，支持模型的训练和检测。

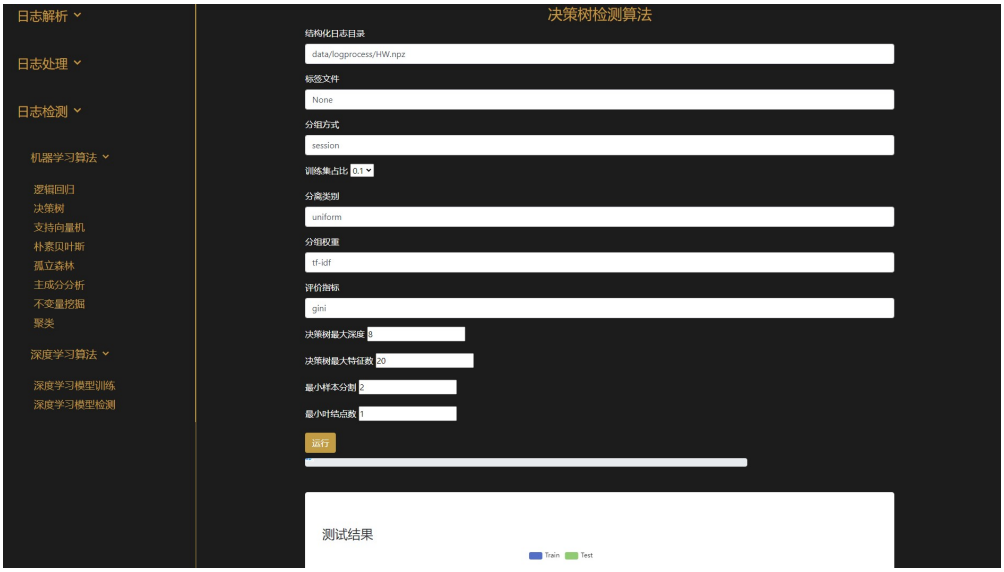


图 5.7　异常检测模块展示

5.4 本章小结

本章主要对日志异常检测进行了可视化系统的开发，通过需求分析、系统设计和具体系统实现三部分内容，进行了整个系统开发的内容介绍。通过应用系统开发，将部分算法内容及检测数据的处理流程展示在前端，以可视化的控制方式实现异常检测，方便了运维人员进行相关工作的开展。需要说明的是，目前本部分内容仍处在系统的初步开发阶段，部分功能模块依然有待进一步开发和完善，且未对其进行性能优化、安全验证等具体的系统开发校验。但该系统雏形的开发，为日志异常检测的应用

提供了可能。

第六章　总结与展望

本章先对全文内容进行总结，然后对日志异常检测下一步的研究方向提出几点展望。

6.1 总结

随着大数据时代的到来，现代服务系统的规模越来越复杂。微小的异常情况都有可能影响系统的正常运行，如何及时发现异常、保证大型系统的稳定可靠运行是目前服务系统设计和运行维护的重点。日志作为一种通用资源，记录了系统运行时的各类关键事件和行为信息，例如，访问记录、操作记录、网络情况等。因此，基于日志信息的异常检测技术被应用于系统监控领域，涌现出了多种检测模型算法。但当前基于日志的异常检测依然存在诸多问题，异常检测流程复杂、解析算法不稳定、日志行为异常数据短缺、实际应用困难等。

本文聚焦于解决以上问题，对日志异常检测的相关内容进行了探索研究，本文梳理了日志异常检测的全流程技术内容，总结了一套日志异常检测的流程框架。框架包括日志收集与预处理、日志解析、特征提取和异常检测四个步骤，旨在较详细解析系统日志异常检测流程与较全面综述现有的技术方法，为相关科研工作者提供参考与借鉴。

此外，本文主要创新性研究成果如下：

(1) 提出了一种基于启发式规则的流式日志解析方法 HRTree。日志解析作为异常检测的重要步骤，目前即使最高效的解析方法 Drain，面对不同系统日志数据时，依然存在参数过度拟合、解析结果不稳定等问题。本文基于启发式规则，在 Drain 基础上对部分日志参数进行拆分，使用解析树结构对日志进行在线的流式解析。实验证明 HRTree 在不同数据集上均展现了较为理想的解析效果，为日志异常检测准确的特征提取提供了保障。

(2) 重点设计了一种基于纵向逻辑回归的日志异常检测方案。日志数据不平衡、异常定义困难是当前异常检测模型面临的难点，本文通过使用纵向联邦方案，在不泄露原始数据的基础上，使多个参与方通过特征和异常标签共享，实现联合模型训练。具体的，本文设计了基于 RSA 哈希的隐私集合求交对齐技术实现检测序列的标签数据对齐；提出使用日志计数向量特征参与模型检测，对传统逻辑回归进行二阶泰勒展开近似，通过 Paillier 同态加密算法实现中间结果的秘密交换，最终实现梯度计算和参数更新训练。

(3) 最后实现了一套可视化的日志异常检测系统开发。面对复杂的日志异常检测流程和各种类型的解析方法和检测模型，本文通过需求分析和功能设计，对其进行了全面的复现和梳理，将检测流程和检测方法转为为可视化的前端操作，方便更多非专业人员的上手使用，同时减轻了维护人员的工作负担，对其进行了应用开发。

6.2 展望

通过对整个日志异常检测流程的分析研究，发现依然存在有待进一步研究的内容，本文对日志异常检测未来的研究做出以下几点展望：

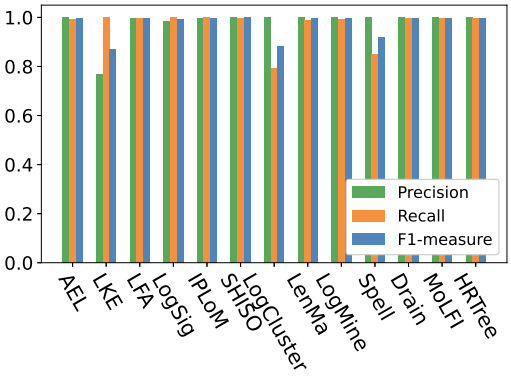
(1) 联邦学习方向；联邦学习作为一种解决数据隐私问题的联合模型训练方式，可以解决数据隐私、数据不平衡等问题，进而提高模型的鲁棒性和泛化能力。特别是纵向联邦学习方案，可以通过“数据共享”解决部分业务系统异常标识短缺问题。但是目前日志异常检测在联邦学习方面探索相对较少，本文仅探索了使用日志计数向量特征进行纵向逻辑回归建模的一个方向，其他特征的使用和其他模型的构建依然存在较大的研究探索的空间。

(2) 异常原因分析；目前的检测模型主要以报警为主，并未对具体的异常内容进行分析和说明，但与之同等重要的是对异常进行原因分析，从而发现导致异常的真实问题。所以未来更需要对异常进行可解释方向的研究，通过模型结果分析，准确定位异常发生的类型，同时能够对异常发生的原因给予解释。方便运维人员对报警异常进行正确处置。

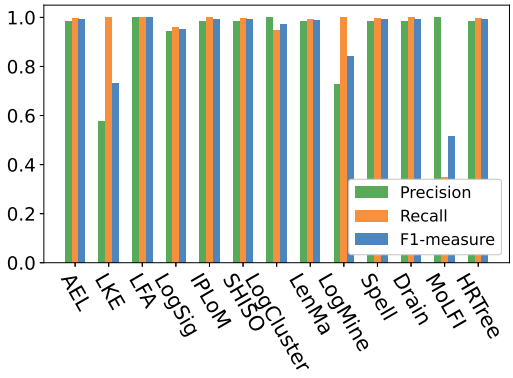
(3) 大型语言检测模型；目前日志异常检测还存在一个日志偏移的问题，即日志信息会随着系统的更新发生文本的更新。特别是目前较多的检测模型使用了日志计数向量特征和日志索引序列特征，日志信息的微小偏移更新，就会造成日志索引的变化，增加新的维度特征，进而导致模型的重新训练等问题。而日志语义信息的多维特征属性可以较好的解决该问题，但是目前在日志异常检测方向的语义模型对日志信息的表达准确性有待提高。目前大语言模型的发展正十分火热，未来借助于大语言模型进行日志行为的分析将会是一个值得探索的方向。

附录 A　日志解析结果附录

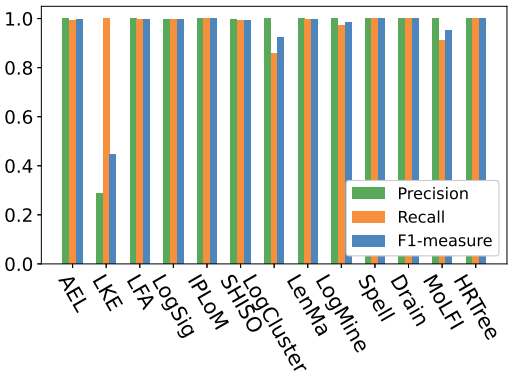
不同解析方法典型指标评估结果



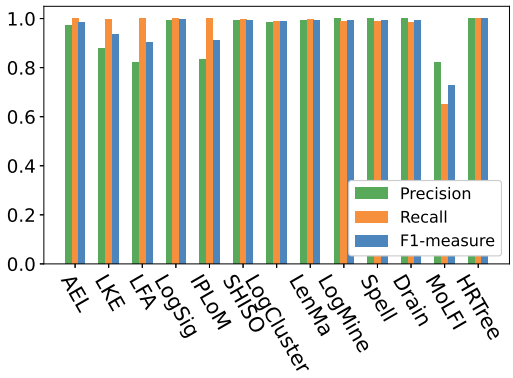
(a) Hadoop



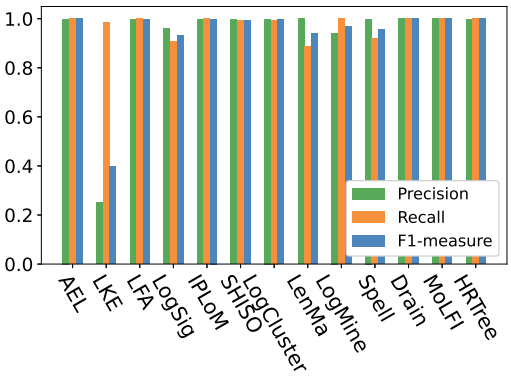
(b) Spark



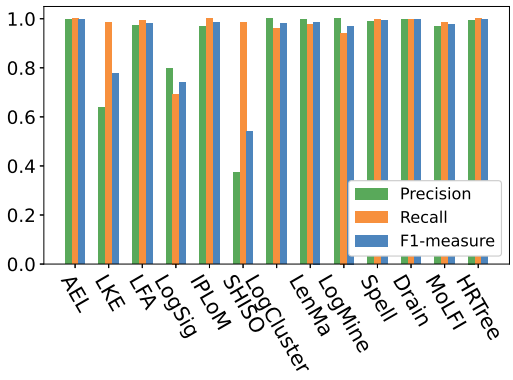
(c) Zookeeper



(d) OpenStack

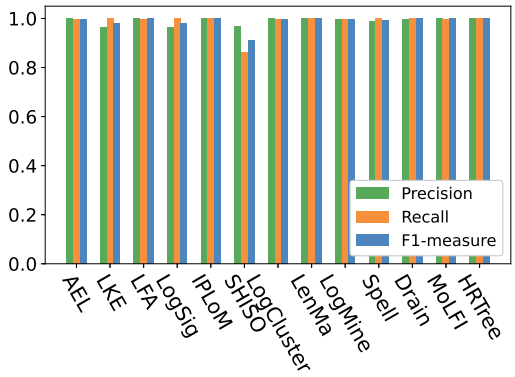


(e) BGL

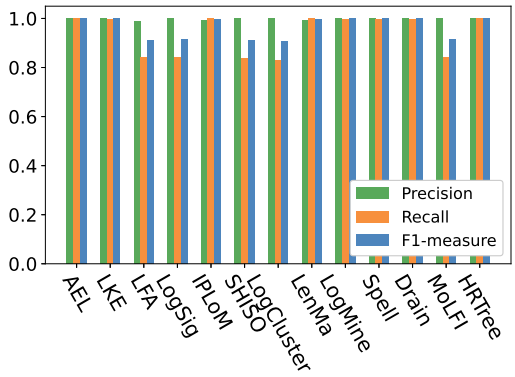


(f) HPC

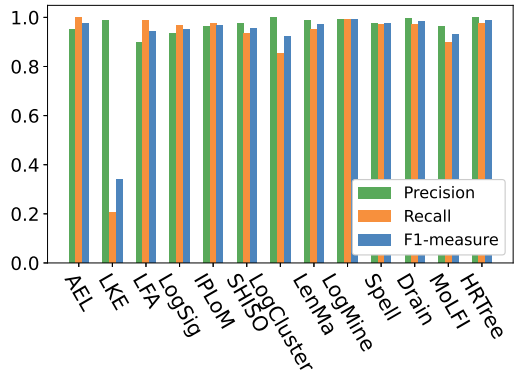
图 A1　多种解析方法在不同数据集上典型指标评估结果-附 (1)



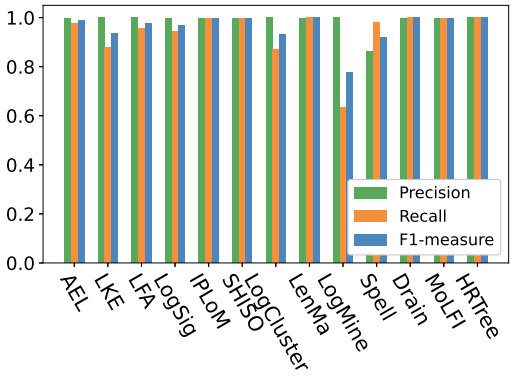
(a) Thunderbird



(b) Windows



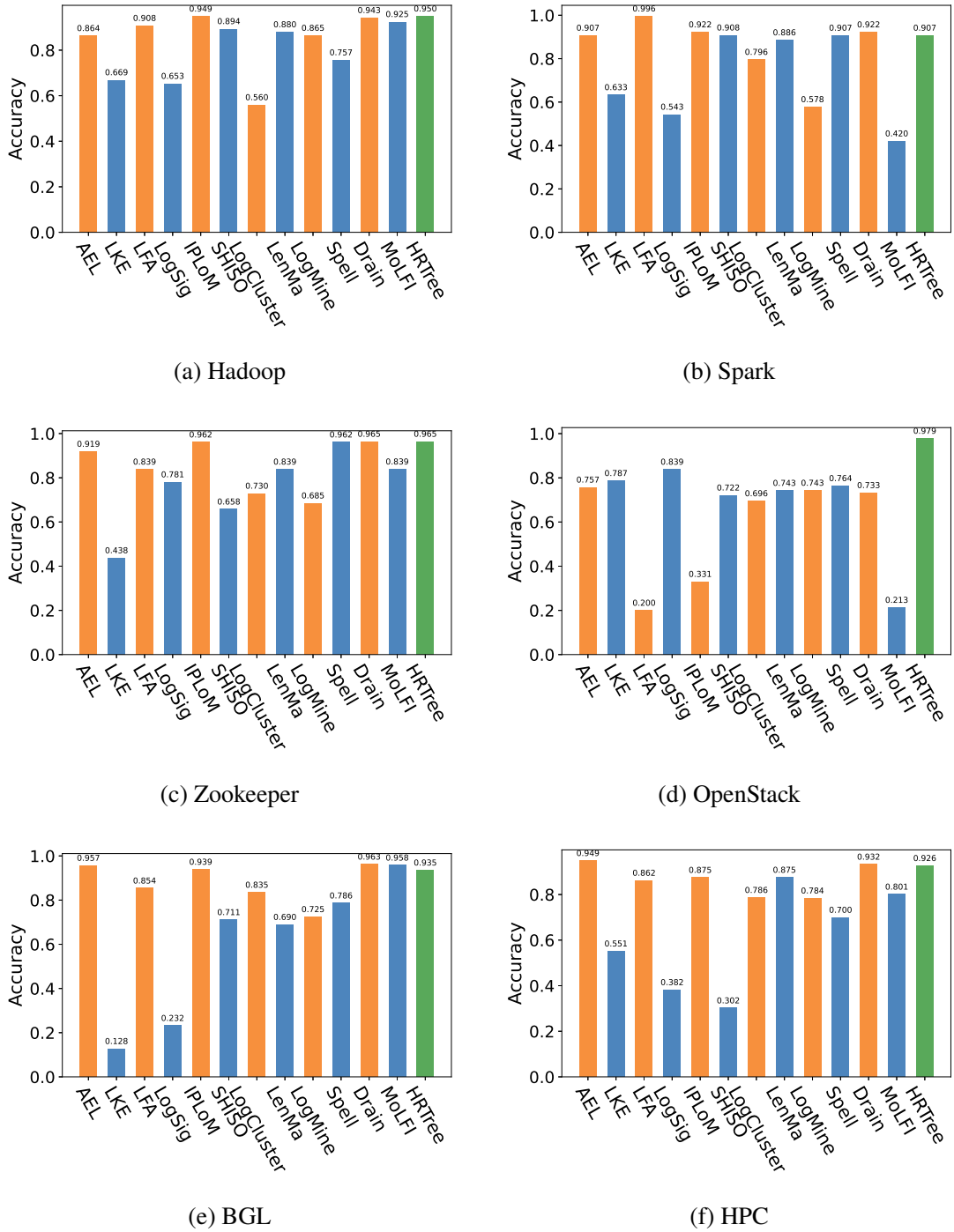
(c) Mac



(d) OpenSSH

图 A2　多种解析方法在不同数据集上典型指标评估结果-附 (2)

不同解析方法准确率指标评估结果附录

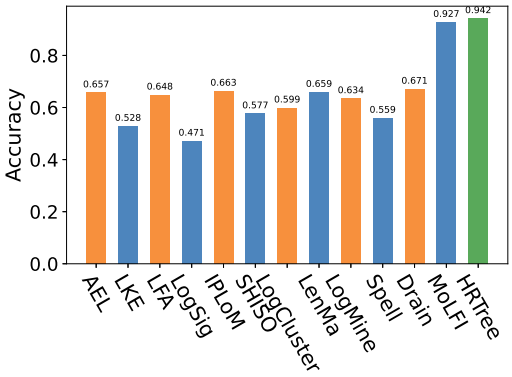


(a) Hadoop　(b) Spark

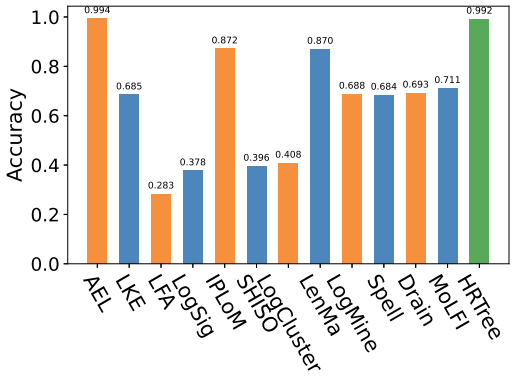
(c) Zookeeper　(d) OpenStack

(e) BGL　(f) HPC

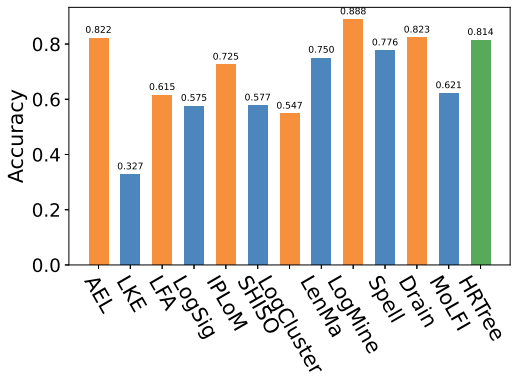
图 A3　多种解析方法在不同数据集上准确率指标评估结果-附 (1)



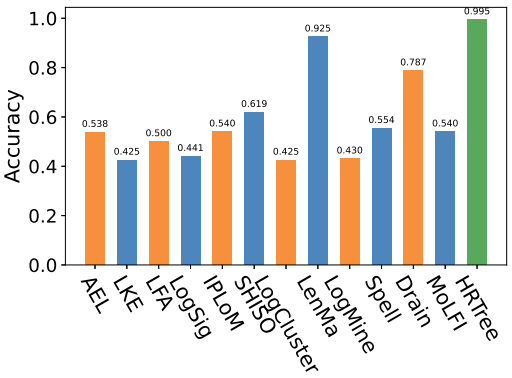
(a) Thunderbird



(b) Windows



(c) Mac



(d) OpenSSH

图 A4　多种解析方法在不同数据集上准确率指标评估结果-附 (2)

附录 B　纵向逻辑回归公式推导附录

逻辑回归损失函数简化推导细节

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i log(\frac{1}{1+e^{-\theta^T x_i}}) + (1-y_i)log(1-\frac{1}{1+e^{-\theta^T x_i}})\right] \tag{B-1}$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[-y_i log(1+e^{-\theta^T x_i}) + (1-y_i)log(\frac{e^{-\theta^T x_i}}{1+e^{-\theta^T x_i}})\right] \tag{B-2}$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[-y_i log(1+e^{-\theta^T x_i}) + (1-y_i)(log(e^{-\theta^T x_i}) - log(1+e^{-\theta^T x_i}))\right] \tag{B-3}$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\left[-y_i log(1+e^{-\theta^T x_i}) + (1-y_i)(-\theta^T x_i) - log(1+e^{-\theta^T x_i}))\right] \tag{B-4}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left[y_i log(1+e^{-\theta^T x_i}) + (1-y_i)(\theta^T x_i) + log(1+e^{-\theta^T x_i}))\right] \tag{B-5}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left[log(1+e^{-\theta^T x_i}) + (1-y_i)(\theta^T x_i)\right] \tag{B-6}$$

纵向逻辑回归中参与方 B 损失值 [[Loss]] 的计算推导

由图 4.3中可知，在 B 方进行损失值计算时拥有 B 方所有数据，以及参与方 A 发送的 $[[U_A]]$ 和 $[[Z_A^2]]$，具体如下：

$$[[U_A]] = \left[\left[\frac{1}{4}\theta_A^T X_A\right]\right] \tag{B-7}$$

$$\left[\left[Z_A^2\right]\right] = \left[\left[(\theta_A^T X_A)^2\right]\right] \tag{B-8}$$

由第四章式（4-20）有:

$$J(\theta) = \frac{1}{n}\left[log2 - \frac{1}{2}(\theta_A^T X_A + \theta_B^T X_B) + \frac{1}{8}(\theta_A^T X_A + \theta_B^T X_B)^2 + (1-y)(\theta_A^T X_A + \theta_B^T X_B) + \frac{\lambda(\theta_A^2 + \theta_B^2)}{2}\right] \tag{B-9}$$

通过拆分化简可以得到：

$$J(\theta) = \frac{1}{n}\left[log2 + (\frac{1}{2} - y)(\theta_A^T X_A + \theta_B^T X_B) + \frac{1}{8}(\theta_A^T X_A)^2 + \frac{1}{8}(\theta_B^T X_B)(2(\theta_A^T X_A) + \theta_B^T X_B) + \frac{\lambda(\theta_A^2 + \theta_B^2)}{2}\right] \tag{B-10}$$

将式（B-7) 和式（B-8）代入，可得缺少惩罚项的 Loss：

$$Loss = \frac{1}{n}\left[[[log2]] + (\frac{1}{2} - y)\left[4([[U_A]] + \left[\left[\theta_B^T X_B\right]\right])\right] + \frac{1}{8}[[Z_A^2]] + \frac{1}{8}(\theta_B^T X_B)(8\,[[U_A]] + \left[\left[\theta_B^T X_B\right]\right])\right] \tag{B-11}$$

参考文献

[1] 廖湘科, 李姗姗, 董威, 等. 大规模软件系统日志研究综述[J]. 软件学报, 2016(8): 14.

[2] QIANG F, LOU J G, LIN Q, et al. Contextual analysis of program logs for understanding system behaviors[C]//Working Conference on Mining Software Repositories. 2013.

[3] CHOW M, MEISNER D, FLINN J N, et al. The mystery machine: End-to-end performance analysis of large-scale internet services[J]. USENIX Association, 2014.

[4] AIT EL HADJ M, KHOUMSI A, BENKAOUZ Y, et al. Efficient security policy management using suspicious rules through access log analysis[C]//Networked Systems: 7th International Conference. 2019: 250-266.

[5] CINQUE M, COTRONEO D, PECCHIA A. Event logs for the analysis of software failures: A rule-based approach[J]. IEEE Transactions on Software Engineering, 2013, 39(6): 806-821.

[6] PREWETT J E. Analyzing cluster log files using Logsurfer[J]. Proceedings of the 4th Annual Conference on Linux Clusters, 2003.

[7] ROUILLARD J P. Real-time log file analysis using the simple event correlator (SEC)[C]//Conference on Systems Administration. 2004: 133-150.

[8] LIN Q, ZHANG H, LOU J G, et al. Log clustering based problem identification for online service systems[C]//2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C). 2016: 102-111.

[9] MI H, WANG H, ZHOU Y, et al. Toward fine-grained, unsupervised, scalable performance diagnosis for production cloud computing systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2013.

[10] ZHAO X, JIANG Z, MA J. A survey of deep anomaly detection for system logs[C]//2022 International Joint Conference on Neural Networks (IJCNN). 2022: 1-8.

[11] HARDY S, HENECKA W, IVEY-LAW H, et al. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption[J]. arXiv preprint arXiv:1711.10677, 2017.

[12] DI S, GUO H, PERSHEY E, et al. Characterizing and understanding HPC job failures over the 2k-day life of IBM BlueGene/Q system[C]//2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). 2019: 473-484.

[13] XU W, HUANG L, FOX A, et al. Detecting large-scale system problems by mining console logs [C]//Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles. 2009: 117-132.

[14] NAGAPPAN M, WU K, VOUK M A. Efficiently extracting operational profiles from execution logs using suffix arrays[C]//2009 20th International Symposium on Software Reliability Engineering. 2009: 41-50.

[15] VAARANDI R. A data clustering algorithm for mining patterns from event logs[C]//Proceedings of the 3rd IEEE Workshop on IP Operations Management (IPOM). 2003: 119-126.

[16] NAGAPPAN M, VOUK M A. Abstracting log lines to log event types for mining software system logs[C]//2010 7th IEEE Working Conference on Mining Software Repositories (MSR). 2010: 114-117.

[17] VAARANDI R, PIHELGAS M. LogCluster - A data clustering and pattern mining algorithm for event logs[C]//2015 11th International Conference on Network and Service Management (CNSM). 2015: 1-7.

[18] DAI H, LI H, CHEN C S, et al. Logram: Efficient log parsing using n-Gram dictionaries[J]. IEEE Transactions on Software Engineering, 2020: 1-1.

[19] FU Q, LOU J G, WANG Y, et al. Execution anomaly detection in distributed systems through unstructured log analysis[C]//2009 Ninth IEEE International Conference on Data Mining. 2009: 149-158.

[20] TANG L, LI T, PERNG C S. LogSig: Generating system events from raw textual logs[C]// Proceedings of the 20th ACM International Conference on Information and Knowledge Management. 2011: 785-794.

[21] HAMOONI H, DEBNATH B, XU J, et al. Logmine: Fast pattern recognition for log analytics[C]// Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. 2016: 1573-1582.

[22] MIZUTANI M. Incremental mining of system log format[C]//2013 IEEE International Conference on Services Computing. 2013: 595-602.

[23] SHIMA K. Length matters: Clustering system log messages using length of words[J]. arXiv preprint arXiv:1611.03213, 2016.

[24] JIANG Z M, HASSAN A E, FLORA P, et al. Abstracting execution logs to execution events for enterprise applications (Short Paper)[C]//2008 The Eighth International Conference on Quality Software. 2008: 181-186.

[25] MAKANJU A, ZINCIR-HEYWOOD A N, MILIOS E E. A lightweight algorithm for message type extraction in system application logs[J]. IEEE Transactions on Knowledge and Data Engineering, 2012, 24(11): 1921-1936.

[26] HE P, ZHU J, ZHENG Z, et al. Drain: An online log parsing approach with fixed depth tree[C]// 2017 IEEE International Conference on Web Services (ICWS). 2017: 33-40.

[27] DU M, LI F. Spell: Streaming parsing of system event logs[C]//2016 IEEE 16th International Conference on Data Mining (ICDM). 2016: 859-864.

[28] MESSAOUDI S, PANICHELLA A, BIANCULLI D, et al. A search-based approach for accurate identification of log message formats[C]//Proceedings of the 26th Conference on Program Comprehension. 2018: 167-177.

[29] DEB K, PRATAP A, AGARWAL S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.

[30] MENG W, LIU Y, ZAITER F, et al. LogParse: Making log parsing adaptive through word classification[C]//2020 29th International Conference on Computer Communications and Networks (ICCCN). 2020: 1-9.

[31] MIKOLOV T, CHEN K, CORRADO G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.

[32] JOULIN A, GRAVE E, BOJANOWSKI P, et al. Bag of tricks for efficient text classification[J]. arXiv preprint arXiv:1607.01759, 2016.

[33] MENG W, LIU Y, ZHU Y, et al. LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs.[C]//Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. 2019: 4739-4745.

[34] ZHANG X, XU Y, LIN Q, et al. Robust log-based anomaly detection on unstable log data[C]// Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2019: 807-817.

[35] HUANG S, LIU Y, FUNG C, et al. HitAnomaly: Hierarchical transformers for anomaly detection in system log[J]. IEEE Transactions on Network and Service Management, 2020, 17(4): 2064-2076.

[36] HE S, ZHU J, HE P, et al. Experience report: system log analysis for anomaly detection[C]//2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE). 2016: 207-218.

[37] CHEN M, ZHENG A X, LLOYD J, et al. Failure diagnosis using decision trees[C]//International Conference on Autonomic Computing. 2004: 36-43.

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

[39] LOU J G, FU Q, YANG S, et al. Mining invariants from console logs for system problem detection. [C]//USENIX Annual Technical Conference. 2010: 1-14.

[40] DU M, LI F, ZHENG G, et al. Deeplog: Anomaly detection and diagnosis from system logs through deep learning[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 1285-1298.

[41] SUTSKEVER I, VINYALS O, LE Q V. Sequence to sequence learning with neural networks[J].

Advances in Neural Information Processing Systems, 2014, 27.

[42] DAI A M, LE Q V. Semi-supervised sequence learning[J]. Advances in Neural Information Processing Systems, 2015, 28.

[43] LIU C, SUN H, NAN D, et al. Augmented LSTM framework to construct medical self-diagnosis android[C]//2016 IEEE 16th International Conference on Data Mining (ICDM). 2016.

[44] LU S, WEI X, LI Y, et al. Detecting anomaly in big data system logs using convolutional neural network[C]//4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech). 2018: 151-158.

[45] YEN S, MOH M, MOH T S. Causalconvlstm: Semi-supervised log anomaly detection through sequence modeling[C]//2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA). 2019: 1334-1341.

[46] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[J]. Advances in Neural Information Processing Systems, 2017, 30.

[47] LIU F, WEN Y, ZHANG D, et al. Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019: 1777-1794.

[48] XIA B, BAI Y, YIN J, et al. LogGAN: A log-level generative adversarial network for anomaly detection using permutation event modeling[J]. Information Systems Frontiers, 2021, 23(2): 285-298.

[49] DUAN X, YING S, YUAN W, et al. A generative adversarial networks for log anomaly detection[J]. Computer Systems Science and Engingeering, 2021, 37(1): 135-148.

[50] KONEČNỲ J, MCMAHAN H B, RAMAGE D, et al. Federated optimization: Distributed machine learning for on-device intelligence[J]. arXiv preprint arXiv:1610.02527, 2016.

[51] YANG Q, LIU Y, CHEN T, et al. Federated machine learning: Concept and applications[J]. ACM Transactions on Intelligent Systems and Technology, 2019, 10(2): 1-19.

[52] GUO Y, WU Y, ZHU Y, et al. Anomaly detection using distributed log data: A lightweight federated learning approach[C]//2021 International Joint Conference on Neural Networks (IJCNN). 2021: 1-8.

[53] ZHU L, LIU Z, HAN S. Deep leakage from gradients[J]. Advances in Neural Information Processing Systems, 2019, 32.

[54] CHEN R, ZHANG S, LI D, et al. LogTransfer: Cross-system log anomaly detection for software systems with transfer learning[C]//2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE). 2020: 37-47.

[55] HE P, ZHU J, HE S, et al. An evaluation study on log parsing and its use in log mining[C]//2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). 2016: 654-661.

[56] ZHU J, HE S, LIU J, et al. Tools and benchmarks for automated log parsing[C]//2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). 2019: 121-130.

[57] MEADOWS C. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party[C]//1986 IEEE Symposium on Security and Privacy. 1986: 134-134.

[58] DE CRISTOFARO E, TSUDIK G. Experimenting with fast private set intersection[C]//Trust and Trustworthy Computing: 5th International Conference, TRUST 2012, Vienna, Austria, June 13-15, 2012. Proceedings 5. 2012: 55-73.

[59] FREEDMAN M J, NISSIM K, PINKAS B. Efficient private matching and set intersection[C]// Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques. 2004: 1-19.

[60] YAO A C. Protocols for secure computations[C]//23rd Annual Symposium on Foundations of Computer Science (sfcs 1982). 1982: 160-164.

[61] PAILLIER P. Public-key cryptosystems based on composite degree residuosity classes[C]//Advances in Cryptology—EUROCRYPT' 99: International Conference on the Theory and Application of Cryptographic Techniques Prague. 1999: 223-238.

[62] GENTRY C. Fully homomorphic encryption using ideal lattices[C]//Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing. Association for Computing Machinery, 2009: 169-178.

[63] BRESSON E, CATALANO D, POINTCHEVAL D. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications[C]//Advances in Cryptology-ASIACRYPT 2003: 9th International Conference on the Theory and Application of Cryptology and Information Security. 2003: 37-54.