计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
[doi: 10.15888/j.cnki.csa.009714] [CSTR: 32024.14.csa.009714]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于联邦学习的异常日志检测①

连宇瀚[1], 廖声扬[1], 张坤三[2], 邹维福[1], 林　楠[3]

[1](国网福建省电力有限公司 泉州供电公司, 泉州 362019)
[2](国网福建省电力有限公司 漳州供电公司, 漳州 363030)
[3](国网福建省电力有限公司 莆田供电公司, 莆田 351199)
通信作者: 廖声扬, E-mail: 1037802128@qq.com

摘　要: Hadoop 系统作为大数据存储的分布式架构被广泛使用, 运行时生成大量日志数据来记录设备的异常情况, 这为定位和分析问题提供重要线索. 然而, 传统的日志异常检测模型通常在中心服务器上收集日志数据, 导致数据收集过程中存在敏感信息泄露的风险. 联邦学习作为一种新的机器学习范式, 通过在本地服务器上训练模型并仅在中心服务器上聚合模型参数, 有效解决了数据隐私问题. 本文提出了一种基于联邦学习的日志异常检测架构, 结合本地服务器和中心服务器进行检测任务, 避免了敏感信息在网络传输过程中的泄露风险. 此外, 本文采用树解析器实现日志模板标准化. 为了有效地捕获日志数据中的复杂模式和异常行为, 建立基于自注意力机制的 BiLSTM 模型作为本地服务器模型. 为了验证所提出方法的有效性, 本文使用公开的分布式系统架构数据集进行仿真实验. 结果表明, 该模型的综合评价指标稳定, 准确率保持在 93% 以上, 具有较高的适用性.

关键词: 异常日志检测; 联邦学习; 树解析器; BiLSTM; 自注意力机制

引用格式: 连宇瀚,廖声扬,张坤三,邹维福,林楠.基于联邦学习的异常日志检测.计算机系统应用. http://www.c-s-a.org.cn/1003-3254/9714.html

Abnormal Log Detection Based on Federated Learning

LIAN Yu-Han[1], LIAO Sheng-Yang[1], ZHANG Kun-San[2], ZOU Wei-Fu[1], LIN Nan[3]

[1](Quanzhou Power Supply Company, State Grid Fujian Electric Power Co. Ltd., Quanzhou 362019, China)
[2](Zhangzhou Power Supply Company, State Grid Fujian Electric Power Co. Ltd., Zhangzhou 363030, China)
[3](Putian Power Supply Company, State Grid Fujian Electric Power Co. Ltd., Putian 351199, China)

Abstract: The Hadoop system is widely used as a distributed architecture for big data storage. It generates a large amount of log data during runtime to record device anomalies, which provides important clues for locating and analyzing problems. However, traditional log anomaly detection models typically collect log data on a central server, which introduces the risk of sensitive information leakage during data collection. Federated learning, a novel machine learning paradigm, effectively protects data privacy by training models on local servers and aggregating model parameters only on a central server. This study proposes a log anomaly detection architecture based on federated learning, which combines local and central servers to perform detection tasks, avoiding the risk of leaking sensitive information during network transmission. Additionally, it employs a tree parser to standardize log templates. To effectively capture complex patterns and anomalous behaviors in log data, a BiLSTM model based on the self-attention mechanism is established as a local server model. To validate the effectiveness of the proposed method, simulation experiments are conducted using publicly available datasets of distributed systems. The results demonstrate that the model maintains stable comprehensive evaluation metrics, with an accuracy rate above 93%, indicating high applicability.

Key words: abnormal log detection; federated learning; tree parser; BiLSTM; self-attention mechanism

① 基金项目: 福建省电力有限公司科技项目 (52133023000C)
收稿时间: 2024-05-17; 修改时间: 2024-06-17; 采用时间: 2024-07-11; csa 在线出版时间: 2024-10-31

1 引言

如今, 大数据技术不断发展, Hadoop 作为大数据背景下的开源系统框架, 专门用于处理及管理海量数据, 它提供的分布式存储 (Hadoop distributed file system, HDFS) 已经成为大数据生态的存储标准. 日志在分布式系统中扮演着至关重要的角色, 它记录了系统运行过程中的各种活动和事件, 为故障诊断、性能优化和安全监控提供了关键信息. 异常日志是系统中潜在问题的先兆, 及早发现并处理这些异常能够有效避免系统崩溃、数据丢失和安全漏洞等问题, 因此异常日志检测在分布式系统管理中具有重要意义[1].

为了实现检测任务, 传统采用人工分析的方法判定日志性质, 通过人工调试代码、频繁打印日志线程状态等方法实现异常日志筛查. 近年来, 随着人工智能的迅速发展, 机器学习技术已经被引入异常日志检测领域, 基于异常日志检测工作主要表现为集中式异常日志检测. 集中式学习的优势在于其易于管理与维护, 所有的日志数据和计算资源集中在中心服务器, 学习日志信息的同时也保证了现有数据的安全性[2]. 然而, 集中式学习存在鲁棒性差的问题. 由于 HDFS 数据量众多, 频繁向中心服务上传日志数据会占用大量的网络资源, 并且单点故障问题不容忽视. 集中式学习高度依赖中心服务器, 如果中心服务器出现故障, 检测无法进行. 为了解决单点故障问题, 分布式学习的方法被提出[3]. 分布式学习的优势在于学习任务在各个分布式节点完成, 通过中心服务器整合各个节点的训练信息来构建检测模型. 分布式学习虽然减少了对中心服务器的依赖, 但在收集日志的过程中, 可能造成数据泄露[4]. 从应用角度分析, 集中式学习更适用于日志数量较少且系统自治的检测任务, 例如内网域名系统, 而无法用于大数据条件下的分布式存储系统. 基于此, 本文重点关注 HDFS 下的异常日志检测.

鉴于集中式学习必然会给检测任务带来计算开销问题和单点故障问题, 进而影响模型的准确性, 已有相关方法大都致力于解决在日志数据存在于中心服务器的前提下, 尽可能地提高检测精度. 迄今为止, 虽然已经取得了一些较好的成果[5−10], 但是同样也存在隐私与模型鲁棒性的挑战, 具体表现为:

(1) 日志检测存在隐私泄露问题. 集中式学习方案在一定程度上保护了个人数据, 但无法适用于大数据情况. 分布式学习方案将原始数据中的一部分或全部的数据集中在中心服务器, 这可能会增加数据泄露的风险. 即使通过加密等手段保护数据在传输过程中的安全性, 但数据在传输过程中仍可能暴露部分信息.

(2) 日志检测存在效率低问题. 分布式检测的中心服务器负担更多计算任务, 而另一些服务器处于空闲状态, 这会导致计算资源利用不均衡. 同时, 每次模型更新都需要所有数据参与, 导致模型更新效率低.

针对上述挑战, 本文提出一种基于联邦学习的日志异常检测方法. 本文分析了联邦学习在基于深度学习的日志数据异常检测中的适用性. 与现有的人工智能模型不同, 联邦学习通过仅传输在每个本地服务器上训练的权重来学习全局模型到中心服务器, 不直接采集数据, 可以防止包含敏感信息的日志数据被泄漏. 联邦学习的训练任务被分发给各个节点, 并且原始日志数据没有集中在同一服务器训练, 提高了数据处理和模型更新的效率. 首先, 为了实现日志文件解析, 设计了日志解析树, 对未被结构化处理过的日志进行分析. 每个联邦节点建立日志解析树能够提取出日志数据中关键信息, 为后续特征训练提供带有日志有效信息的数据, 并在节点内完成模型构建. 其次, 每个联邦节点的模型参数被传输给中心服务器, 中心服务器进行参数整合, 最终得到完整的异常日志检测模型.

为验证本文方法, 首先对基于联邦学习的日志检测流程进行了详细设计, 描述了检测步骤以及具体工作流程. 我们利用树解析器对日志模板进行标准化处理, 在本地服务器上使用自注意力机制的 BiLSTM (bidirectional long short term memory,) 模型进行模型训练, 捕获日志数据中的复杂模式和异常行为. 在中心服务器上, 通过参数聚合更新全局模型, 确保一致性和有效性. 为验证该方法的有效性, 我们选取 HDFS 日志数据进行实验, 并与其他现有方法进行对比. 实验结果表明, 本文方法设计的日志解析器效率显著, 提取的特征信息通过模型训练后, 日志异常检测的准确率稳定在 93% 以上. 通过联邦架构设计避免数据隐私泄露的同时, 本文方法仍然保持了较高的检测指标, 表明所设计的模型具有较好的稳定性和理想的检测效果. 进一步的实验验证显示, 换用其他数据集进行测试, 同样验证了本文提出方法的有效性和适用性, 避免了实验结果的偶然性. 因此本文具有的实际应用价值主要在于实现了联邦学习架构下多方信息聚合的学习, 以达到检测异常日志的目的.

本文第 2 节介绍相关工作和研究动机; 第 3 节介绍本文的方法所涉及的日志异常检测架构; 第 4 节对检测性能进行评估; 第 5 节总结全文.

2 相关工作

由于系统数据规模逐渐庞大引发了一系列安全管理问题, 因此, 异常日志检测技术受到广泛关注. 集中式异常日志检测是一种在单一中心服务器上进行的异常检测方法, 所有的日志数据都被收集到中心服务器上进行学习训练. 在中心服务器上使用有监督学习分类是检测问题中常用的方法, 他们挖掘大量实践中历史数据的行为规律, 将掌握的带有标签的历史数据都交给有监督学习模型训练, 最终实现异常日志检测.

文献[5]从控制台日志文本中挖掘不变属性特征, 通过匹配日志事件在系统执行过程中的行为信息, 当日志运行生命周期中打破了某个不变量行为, 则检测到系统异常, 这种方法存在大量的不确定性, 能够检测出的异常必须带有明显特征信息, 对于信息量较少的日志文本来说并不是有效的方法. 文献[6]通过提取日志事件的语义信息解决日志数据的不稳定性, 包括日志语句的演变和日志数据中的噪声, 建立注意力机制模型学习不同日志事件的重要性, 进而实现对异常日志的检测. 然而, 现有的基于日志语义的方法未能充分考虑单词、日志和日志序列之间的语义信息, 并且使用日志解析器可能会对检测精度产生负面影响并导致语义丢失. LayerLog[7]考虑到日志、单词和序列之间的语义关系, 设计了“词-日志-日志序列”的结构层次, 并提出了一种基于日志数据层次语义的日志序列异常检测框架. 它不需要进行日志解析, 可以有效从每一层提取语义信息, 实现端到端同时检测执行顺序异常、操作异常以及日志序列不完整异常.

完全监督的日志异常检测方法承受着注释大量未标记日志数据的沉重负担. 研究者提出了许多半监督方法来借助解析模板来降低注释成本. 文献[8]其核心思想是通过概率标签估计结合历史异常的知识, 将监督方法的优越性引入到机器学习中, 带有标签的小样本数据作为数据源, 使用聚类基于已知正常的日志序列信息来估计未标记日志序列的标签. 将所有日志数据整合, 在日志处理过程中使用语义嵌入的方法避免不稳定日志数据的噪声影响, 最终实现基于门控循环单元网络的异常检测. 然而, 这基于语义分析的方法独立地考虑每个关键字, 忽略了关键字之间的相关性以及日志序列之间的上下文关系. LOGLG[9]提出了一种新颖的弱监督日志异常检测框架, 以探索序列中关键字之间的语义连接. 首先提取未标记日志的关键字以构建日志事件图, 然后构建一个子图注释器来为未标记的日志序列生成伪标签. 为了提高注释质量, 它采用自监督任务来预训练子图注释器, 使用生成的伪标签训练检测模型. 根据分类结果从日志序列中重新提取关键字, 并更新下一次迭代的日志事件图. 然而, 现有半监督检测方法通常仅利用单一类型的监控数据 (通常是日志或指标), 无法有效利用不同类型数据之间的联合信息, 因此会出现许多错误的预测. 考虑到日志和指标可以协同、互补地体现系统异常情况, 但仅靠其中任何一个都是不够的. 文献[10]提出了 Hades, 这是第 1 个基于异构数据有效识别系统异常的端到端半监督方法. 它采用分层架构, 通过融合日志语义和度量模式来学习系统状态的全局表示. 它通过以半监督方式训练的跨模式注意力模块从异构数据中捕获区分特征和有意义的交互.

综上所述, 现有基于集中式学习的异常日志检测方案主要研究方向为日志文本处理与模型预测精度问题, 未考虑日志数据隐私相关和系统鲁棒性问题. 在本文分析中, 当分布式存储的日志被集中分析时, 庞大的数据量使得关键信息提取非常困难, 日志数据泄露造成的隐私威胁也是不可忽略的, 这些问题是否解决对于分布式系统运行的稳定安全起决定性的因素.

3 流程设计

3.1 方法总体设计

本文提出了一种基于联邦学习的异常日志检测框架, 如图 1 所示. 该架构分为模型训练层和模型聚合层. 模型训练层基于 HDFS 结构建立, 保持了分布式数据的存储与管理特点. 各个节点独立地进行数据处理和模型训练. 其中, 每个 DataNode 负责管理本地存储空间, 执行块存储功能. 它们独立存储和处理日志文件, 从而避免了集中数据存储带来的隐私泄露风险. NameNode 负责块管理, 确保数据块的位置和存取路径的记录. 它包含预写日志系统, 所有数据的修改在提交之前都会记录到日志文件中. 这些日志文件作为研究数据, 通过联邦学习的方法进行本地模型训练. 模型聚合层由中心服务器组成, 负责对各个本地模型进行聚合和更新.

其中, 中心服务器接收来自各个 DataNode 的本地模型参数, 通过联邦平均 (federated averaging, Fed-Avg) 算法进行全局模型的更新. 聚合后的模型参数再分发回各个 DataNode, 以便下一轮训练. 该框架通过在分布式节点上独立训练模型并在中心服务器上进行参数聚合, 实现了以下优点: (1) 隐私保护: 数据仅在本地处理, 避免了敏感信息在网络传输过程中的泄露风险; (2) 分布式计算: 充分利用各个节点的计算资源, 分散计算负担, 提高了模型训练的效率; (3) 全局一致性: 通过中心服务器的参数聚合, 确保全局模型的一致性和有效性.

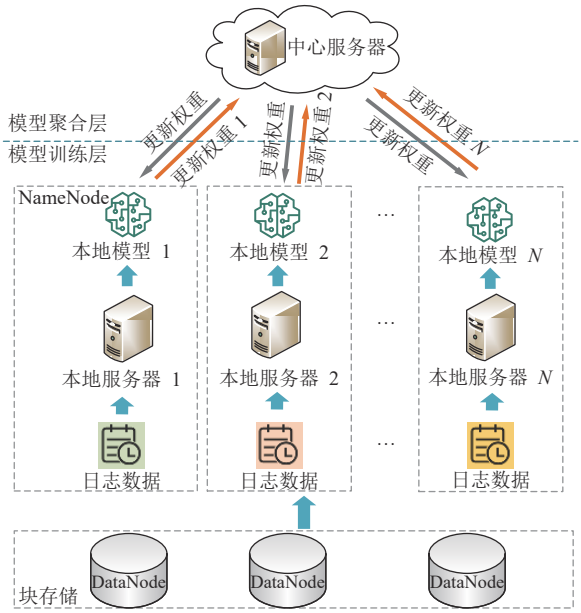


图 1 基于联邦学习的异常日志检测架构

该框架详述了基于联邦的异常日志检测流程. 为每个 NameNode 建立一个学习本地数据集的本地服务器, 并选择一个汇总从本地服务器发送的模型权重的全局服务器, 并且将相同的深度学习模型应用于每个本地服务器. 每个本地服务器根据从本地数据中提取的日志文本特征和设置的迭代次数进行学习. 然后, 根据本地模型重复相应的学习过程, 本地服务器将学习生成的权重参数发送到全局服务器, 本地服务器的原始数据不直接传输到全局服务器, 这样避免了数据的直接传输, 保证数据在信道传输过程中不被泄露. 联邦学习被认为是一个迭代过程, 每次迭代都会改进中心机器学习模型. 服务器端在全局模型收敛后, 会将训练好的全局模型发送给所有参与者. 全局服务器根据 Fed-Avg 算法更新所有本地服务器的聚合权重, 更新后的权重被发送回每个本地模型.

3.2 基于解析树的日志解析

原始日志消息是非结构化的数据, 包含可变的日志参数和系统异常行为. 识别这些信息是困难的, 需要把日志文件进行结构化处理. 为了解决日志检测效率低的问题, 从日志数据源头分析出发, 对于每一个联邦节点来说, 构建日志解析模板是至关重要的, 这将直接影响每一个节点的模型输入. 文献[11]中的树解析器一种高效构建解析模板的方法. 针对大量日志数据聚合仅使用一颗模板树解析, 但对于 HDFS 日志数据背景下, 各类系统的日志结构不同, 不能应用同一种正则表达式进行筛选, 这种方法不适用于不同 NameNode 中的日志数据. 因此, 本文基于解析树构建了日志解析器, 每个节点拥有自己的单棵树.

以单个解析树为例, 当一条新的原始日志信息到达时, 解析器通过正则表达式对其进行预处理. 按照日志信息在树内进行相关搜索, 树内部节点设定特有规则搜索日志组, 每个叶子节点都代表一个文本规则, 如果找到合适的日志组, 日志信息将与存储在该日志组中的日志事件相匹配, 不再创建新的日志组. 否则, 将根据日志信息创建一个新的日志组, 即在树中添加新节点. 解析树中的每条路径都以存储日志组列表的叶子节点为结尾. 每个日志组有两个部分即日志事件和日志 ID. 日志事件是由日志信息模板记录, 包含日志信息常量数据. 为了避免树枝存储爆炸, 使用了 maxChild 参数来限制节点的最大子节点数, 如图 2 所示表示构建的整体树解析器. 具体步骤如下: (1) 预处理: 删除原始日志中与正则表达式匹配的标记; (2) 长度搜索: 根据预处理后的日志信息长度选择路径, 例如日志语句 “Invalid user webmaster from 173.234.31.186”以空格分界显示长度为 5, 解析器遍历到树中的内部节点; (3) 标记搜索: 遍历第 1 层节点, 假设日志信息开头位置的标记是常量, 使用 maxChild 参数限制节点的最大子节点数, 当到达解析树设定的深度时, 生成日志组代替叶子节点; (4) 基于 token 相似性搜索: 解析器已遍历到一个叶节点, 包含日志组列表; (5) 更新解析树: 如果找到合适的日志组, 解析器将当前日志信息的日志 ID 添加到该日志组, 并更新日志事件. 如果未找到合适的日志组, 则创建新的日志组, 并用新的日志组更新解析树.

为了提取日志事件的语义信息, 本文将每个日志文本视为包含非字符的标记 (如数字、分隔符等) 及单词串联的复合标记. 本文通过拆分日志文本中的单词

做日志文件的预处理, 然后采用 Word2Vec[12]做分词处理, 建立词向量可以有效捕捉日志文本中词之间的内在关系, 从处理的日志事件中的每个词提取语义信息. 同时, 将处理后的日志事件中的词语转换为d维向量. 基于词频-逆向文本频率 (term frequency-inverse document frequency, TF-IDF)[13]的聚合是通过词嵌入将词转换为d维向量后, 聚合日志事件中的所有词向量, 进一步将日志事件转换为语义向量. 本文采用信息检索中广泛使用的 TF-IDF 方法进行聚合, 考虑日志中每个单词的重要性, 通过 TF-IDF 可以有效地衡量每个单词的重要性. TF (词频) 衡量单词w在日志事件中出现的频率表示为:

$$TF(w)=\frac{count(w)}{N} \tag{1}$$

其中, $count(w)$是单词在日志事件中出现的次数, N是日志事件中的总字数. IDF (逆文档频率) 衡量单词w在所有日志事件中的常见程度计算为:

$$IDF(w)=\log\frac{count(L)}{count(L_w)} \tag{2}$$

其中, $count(L)$是日志事件总数, $count(L_w)$是包含w的日志事件的数量. 一个单词的权重ω, 可以通过$TF\times IDF$计算出来. 最后, 日志事件的语义向量表示为v, 可以通过将日志事件中的所有词向量与 TF-IDF 权重求和来表示为:

$$V=\frac{1}{N}\sum_{i=1}^{N}\omega_i\cdot v_i \tag{3}$$

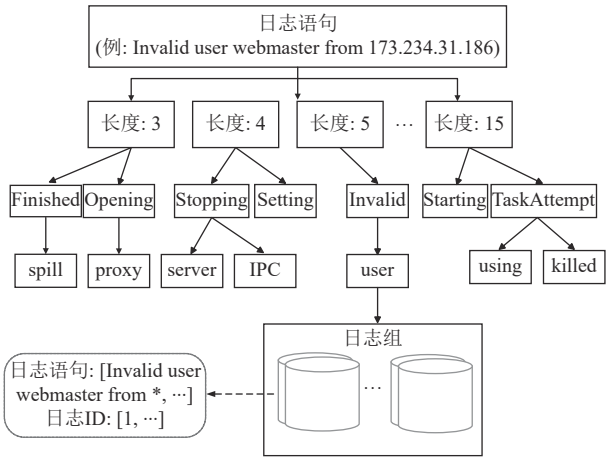


图 2 日志解析树

采用 Word2Vec 与 TF-IDF 方法结合将日志事件表示为语义向量, 有效地融合了它们的语义信息, 有助于识别语义相似的日志事件并区分不同的日志事件, 两种方法结合可以计算词语之间的相似度和语义关系, 还可以捕捉日志中上下文语境中的词语关系, 更准确地表示出语义的相似性.

3.3 基于联邦学习的 BiLSTM 检测模型

异常日志检测模型训练层主要负责接收分布式存储系统节点的模型训练任务, 并进行检测模型的迭代训练. 其功能主要包括模型分发、本地模型训练和全局模型更新. 模型分发阶段指在确定了 HDFS 系统中加入联邦的节点后, 有效地将模型参数传输到各个设备上进行联邦训练, 以提升模型精度和泛化性能. 本地模型训练阶段是指联邦学习中参与模型训练的节点, 节点利用本地数据进行本地模型的训练, 并将本地模型参数上传到联邦学习服务器上, 用于全局模型的更新. 全局模型更新阶段是指通过联邦学习服务器对各个设备上传的本地模型参数进行更新, 使用 Fed-Avg 算法以获得新的全局模型. Fed-Avg 算法作为联邦学习中的一种模型参数聚合方法使得原始数据始终留存在本地设备上, 不会传输到中心服务器上, 从而保护了用户的隐私数据. 只有本地模型参数的平均值被传输到中心服务器, 因此不会泄露用户的个人信息. 假设中心服务器初始化全局模型参数, 每个本地服务器进行模型训练后得到的本地参数为θ_k, 中心服务器收集所有本地服务器的模型参数进行加权凭据, 得到新的全局模型参数θ公式为:

$$\theta=\frac{1}{K}\sum_{k=1}^{K}\theta_k \tag{4}$$

其中, K是参与训练的本地服务器数量. 中心服务器将更新后的全局模型参数θ分发给各个客户端, 客户端使用新的全局模型参数进行下一轮本地训练, 直至模型收敛. 这种模型参数更新策略只需要传输本地模型参数的平均值到中心服务器, 而不需要传输所有的本地模型参数, 减少通信开销, 并且将多个模型参数平均有助于提高模型的泛化能力减少过拟合风险. 具体表现为, 每个设备在本地模型训练完成后, 将本地模型参数上传到联邦学习服务器上. 然后, 服务器将所有设备上传的本地模型参数进行聚合, 计算出新的全局模型参数, 并将其广播给所有设备, 供下一轮本地模型训练使用. 训练层的主要功能可以概括为以下 3 点.

(1) 模型分发: 中心服务器接收到异常日志检测任

务后, 向设备下发初始模型.

(2) 本地模型训练: 设备接收到中心服务器的初始模型后, 利用本地数据进行检测模型训练.

(3) 全局模型更新: 设备在进行一定次数的本地训练后, 将训练好的模型上传至中心服务器, 中心服务器更新全局模型, 并进行下一轮迭代.

以上步骤重复进行, 直至全局模型收敛.

在异常日志检测任务中, 时间序列是一个至关重要的信息源, 它能够揭示系统行为的变化趋势和异常事件的发生时机, 每个联邦节点的本地服务器都需要对日志文本进行处理. 长短期记忆网络 (long short-term memory, LSTM) 是循环神经网络的一种, 其适用于处理和预测时间序列中间隔较长的重要事件, 本文选择 LSTM 作为初始下发模型. 选择 LSTM 的原因有 3 个: (1) LSTM 模型更适用于处理长时间依赖问题, 模型中遗忘门的作用是控制记忆细胞中保存的信息长期存在; (2) LSTM 模型在多个领域[14–16]被证明是有效且高效的检测方法; (3) LSTM 的记忆细胞和门控制功能相比于普通循环神经网络来说解决了长时间依赖会出现的梯度爆炸问题, 避免无关信息干扰和梯度消失问题. 在异常日志检测中, LSTM 被广泛应用于序列数据的建模和分析. 由于异常日志通常具有时序性和序列性的特点, LSTM 能够有效地捕捉日志数据中的时序模式和规律, 从而提高了异常事件的检测性能. 尽管 LSTM 在捕获时间序列方面表现出色, 但它仍存在局限性. 其中最主要的问题是难以捕捉上下文的依赖关系, 在处理长序列数据时会产生信息遗忘现象. 为了解决这一问题, 本文引入了双向长短期记忆网络 (BiLSTM), 其具有更好的上下文信息考虑能力. BiLSTM 通过在模型中引入双向信息传递, 能够更好地捕获句子中的上下文信息, 提高了模型对句子中关键信息的把握能力. 同时引入注意力机制应对输入日志信息的不均匀分布, 使模型关注日志文本中的重要信息. 中心服务器下发给本地服务器的异常日志检测模型如图 3 所示.

输入层预处理后的日志数据作为序列数据输入到 BiLSTM 层, 将原始日志数据转换为适合 BiLSTM 处理的标准化输入. BiLSTM 层捕捉日志数据中的时间依赖关系, 生成隐藏状态序列, 用于进一步的注意力权重计算. 注意力机制层生成的上下文向量作为输入传递给输出层. 它通过计算注意力权重, 聚焦于重要的隐藏状态, 生成上下文向量, 输出日志是否异常的概率, 用于最终的分类决策. 模型设计的主要思想是利用 BiLSTM 模型来捕获日志序列中的时间相关性和上下文信息, 并通过自注意力机制来进一步强调序列中重要的信息. 以解析后的日志语句“Invalid user webmaster from *”为例, 假设向量归一化后的语句表示为x_t. 语句分别从前向和后向出发, 按照时间戳传播, 将两种状态输出拼接起来就能够得到 BiLSTM 的输出向量. 将单元状态连接起来, 可以控制信息传递给下一时刻. LSTM 模型的输入序列定义为$X=\{x_1,x_2,\cdots,x_n\}$, 特征向量定义为$H=\{h_1,h_2,\cdots,h_n\}$, 其表示为隐藏层节点向量序列, 对应输出的向量序列定义为$Y=\{y_1,y_2,\cdots,y_n\}$. BiLSTM 的前向传播公式为:

$$h_t^{\text{forward}}=\text{LSTM}_{\text{forward}}(x_t,h_{t-1}^{\text{forward}}) \tag{5}$$

$$h_t^{\text{backward}}=\text{LSTM}_{\text{backward}}(x_t,h_{t+1}^{\text{backward}}) \tag{6}$$

其中, $\text{LSTM}_{\text{forward}}$和$\text{LSTM}_{\text{backward}}$分别表示前向和后向 LSTM 模型, 将两种隐藏状态拼接起来, 得到时间戳t的 BiLSTM 输出向量$BiVector_t$计算为:

$$BiVector_t=[h_t^{\text{forward}},h_t^{\text{backward}}] \tag{7}$$

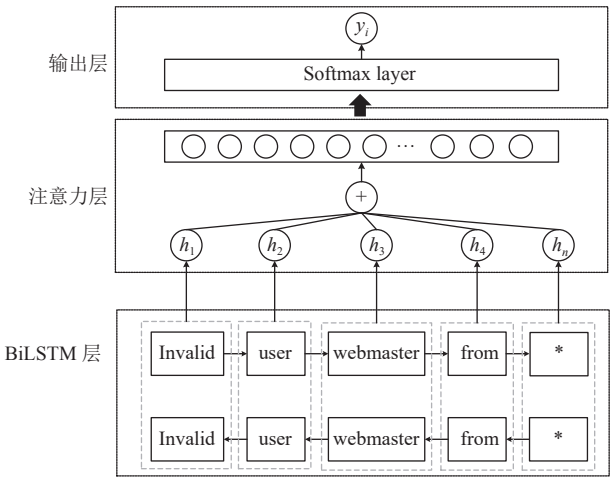


图 3 本地服务器的日志异常检测模型结构图

3.4 注意力机制

BiLSTM 在一定程度上改善了模型对句子中关键信息的捕获能力, 但仍然容易忽略一些重要信息, 主要表现为无法有效区分句子中不同部分的重要程度. 为了解决这一问题, 本文进一步引入了注意力机制, 通过给予句子中不同部分不同的注意力权重, 实现对关键信息的更好捕获. 目前被广泛应用的注意力机制主要涉及两种: 自注意力机制和多头注意力机制. 多头注意

力机制的主要思想是在自注意力机制的基础上, 将查询、键、值进行多次线性变换, 得道多组矩阵, 分别进行自注意力机制计算, 最终将结果拼接, 这种机制更注重模型在不同语义空间中进行学习, 提高模型的表达能力. 本文涉及的训练数据属于日志数据, 都表现为文本结构, 具有较强的同质性, 多头注意力机制不适用于本文背景. 自注意力机制的主要思想是, 在一个句子或序列中, 将每个单词都与其他位置的单词进行交互, 从而计算出每个位置的表示. 注意力机制中输入的序列经过 3 个线性变换得到查询 (Query)、键 (Key)、值 (Value) 这 3 个矩阵, 并通过计算得到全新的融合了注意力机制的文本矩阵. 这种自注意力机制可以学习到日志序列文本中单词之间的依赖关系和上下文信息关系. 本文中仅涉及的日志数据语义空间固定, 不需要分析日志数据在不同语义空间中的行为. 仅针对日志序列文本间的依赖关系及上下文关系进行分析, 专注于自注意力机制.

自注意力机制可以帮助模型关注输入数据中不同位置之间的相互作用, 本文将其引入 LSTM 模型中的每个时间戳中, 计算当前时间戳与之前时间戳的相似度, 进而得到加权表示的日志序列文本向量, 这一向量可以同当前时间戳的输入进行加和, 再得到新的输入向量[17]. 使用自注意力机制的方法应用到 LSTM 模型中可以更加灵活地利用历史信息生成当前状态下的输出. 对于本文的日志文本矩阵来说, 利用自注意力机制实现文本内各单词相互注意, 即建立词与词之间的联系, 词与词之间产生权重矩阵, 对值加权求和产生融合自注意力机制的新文本矩阵. 将注意力机制融入文本的具体步骤如下.

(1) 定义 BiLSTM 的 t 时刻的输出向量表示为 $BiVector_t$, 即日志文本矩阵, Query、Key、Value 的 3 个变换矩阵即卷积核表示为 W_q, W_k, W_v;

(2) 文本矩阵与 3 个权重矩阵做线性变换映射为 Q_{Bi}, K_{Bi}, V_{Bi}, 其中 $Q_{\mathrm{Bi}} = BiVector_t W_q$, $K_{\mathrm{Bi}} = BiVector_t W_k$, $V_{\mathrm{Bi}} = BiVector_t W_v$;

(3) 点积运算实现相似度度量表示为 $G = Q_{\mathrm{Bi}} K_{\mathrm{Bi}}^{\mathrm{T}} / \sqrt{d}$, 当两个向量越相似时, 它们的点积结果也会越大, 也就是说 Q 被搜索时更应该搜索到 K. $\sqrt{d}$ 表示为 K_{Bi} 的维度, 防止内积数值过大影响后续的训练, G 的每行表示某一单词在各个词上的得分;

(4) 归一化的权重矩阵 $W = Softmax(Q_{\mathrm{Bi}} K_{\mathrm{Bi}}^{\mathrm{T}} / \sqrt{d})$, 应用 $Softmax$ 激活函数实现权重矩阵构建;

(5) 最终得到加权求和矩阵表示为 $Attention(Q_{\mathrm{Bi}}, K_{\mathrm{Bi}}, V_{\mathrm{Bi}}) = W \cdot V_{\mathrm{Bi}}$, 结果表示为全新的融合了注意力机制的文本矩阵.

将自注意力机制融入基于 BiLSTM 的异常日志检测模型中, 这样可以将处理过的日志序列文本通过加权求和的方式进行处理, 对日志文本添加注意力权重, 来突出特征文本的影响因子. 通过后续实验证明, 添加注意力机制可以更好地挖掘日志数据的特征以及发生异常事件后日志变化规律信息, 提高检测模型性能, 为日志异常做出预警.

4 实验分析

4.1 数据集和评价指标

本文采用 HDFS 公开数据集[18]进行实验, 并在对比实验中增加 Blue Gene/L (BGL) 数据集[19]用于验证本文方法的性能, 数据集详细信息如表 1 所示. HDFS 数据集是由 203 个 Amazon EC2 节点运行基于 Hadoop 的 MapReduce 作业 38.7 h 生成的日志, 根据区块 ID 被分割成日志序列, 并通过创建的规则来进行正常或异常的标注. 数据集中共有 57 万多个日志块, 其中约 1.7 万个块被 Hadoop 领域专家标注为异常. BGL 数据集由加利福尼亚州劳伦斯利弗莫尔国家实验室 (LLNL) 的 BlueGene/L 超级计算机系统生成, 该系统拥有 131 072 个处理器和 32 768 GB 的内存. 日志包含由警报类别标签标识的警报和非警报信息, 其中的警报信息被视为异常日志信息, 在日志的第 1 列由“-”标识符来区分. 本文基于一台配置为 12th Gen Intel(R) Core(TM) i7-12800HX 2.4G CPU 和 64 GB 内存的 Linux 系统上进行实验, 在 Python 3.7 的环境下实现了该日志检测工具. 最终划分训练集与测试集比例为 6:4.

表 1 数据集

数据集	时间跨度 (h)	数据量 (个)	数据大小 (MB)	异常占比 (%)
HDFS	38.7	11 175 629	1 505.28	2.9
BGL	5 152.8	4 747 963	708.76	7.3

为了评价所设计的异常日志检测模型, 建立混淆矩阵, 更直观地感受到测试集的分类结果. 根据混淆矩阵结果计算准确率 (*Accuracy*)、召回率 (*Recall*)、精确率 (*Precision*)、*F*1 值 (*F*1-score) 等 4 个基本评价指标

作为实验结果. 当进行分类训练时, 二分类混淆矩阵将会出现 4 种分类情况, 如表 2 所示. 其中, *TP* (true positives) 表示为正类判定为正类; *FP* (false positive) 表示为负类判定为正类; *FN* (false positive) 表示为正类判定为负类; *TN* (true positive) 表示为负类判定为负类.

表 2　二分类混淆矩阵

日志标签	判定为0 (正常日志)	判定为1 (异常日志)
真实为0 (正常日志)	*TP*	*FP*
真实为1 (异常日志)	*FN*	*TN*

根据以上二分类混淆矩阵内实现各类计算, 表现出以下 4 种评价指标, 其公式如表 3 所示. 准确率 (*Accuracy*) 是正确预测的样本数量与所有样本数量的比值, 准确率在一定条件下可以作为评价分类器或检测结果有效性指标, 但是并不能作为判定实验结果有效的唯一标准. 精确率 (*Precision*) 是所有被判定为正类的样本数据中, 真实正类所占比例. 召回率 (*Recall*) 是所有真实为正类的样本数据中, 被判定为正类所占比例. 精确率与召回率两个指标通常是此消彼长的情况, 难以兼得, 在数据集合中相互制约, 因此需要综合考虑两个指标因素, 又引进了 *F*1 值 (*F*1-score), 该值作为精确率与召回率的加权调和平均, 综合考虑了精确率和召回率的结果, 进而验证实验结果的有效性.

表 3　模型评价指标

评价指标	公式
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
*F*1-score	$\frac{2\times Precision\times Recall}{Precision+Recall}$

4.2　基于解析树的日志文本解析

由于检测框架的松耦合性, 基于解析树的日志文本解析由本地服务器独立完成. 各本地服务器基于自身日志数据构建解析树, 无需依赖中心服务器进行数据处理. 它不仅减少了数据传输负担, 还有效保护数据隐私. 本文为方便后续训练, 将两种类型数据均进行数据预处理, 将日志文件进行解析并做特征提取. 根据树解析器建立标准化的日志解析模板, 如图 4 表示了日志文件解析过程及解析结果.

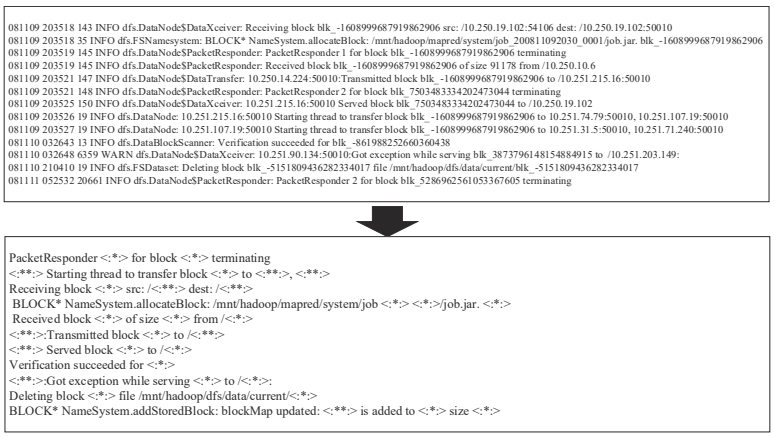


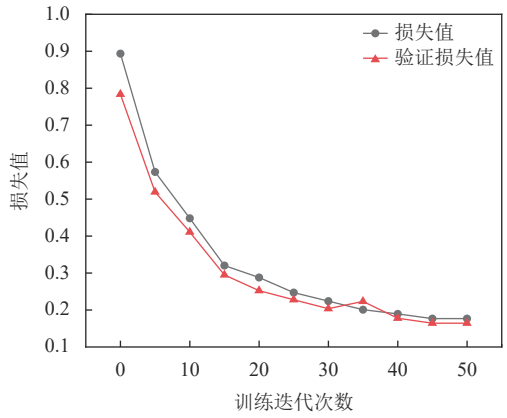
图 4　基于 HDFS 数据集的日志解析过

将语句前半部分的多余赘述删除, 例如进程号、DataNode 名称等单词, 只留下后半部分记录的行为信息, 再依据上述方法中提到的将连续的数字信息及数字内掺杂的符号信息 (例如“.”“_”等) 都替换为“*”表示, 将替换后的语句直接存为模板, 每条被解析后的语句都匹配模板库, 一旦发现解析后的语句已经存在与模板文件中, 既可以不再重复记录. 以某一日志文本记录为例, 显示日志解析的详细过程, 如图 4 所示. 图中显示的日志文本中包含 13 条语句, 解析后仅包含 11 条, 以源日志文本的第一行为例, 读取前两个单词后表示为“Receiving block”, 识别到的下一文本存在大量数字表示, 因此解析为“<: *: >”, 以此类推继续读取文本, 最终生成该句的模板表示为“Receiving block <: *: > src: / <: *: > dest: / <: *: >”. 第 8 行和第 9 行解析后的结果相同, 都表现为“<: **> Starting thread to transfer block<: *: >to<: *: >, <: *: >”. 解析后的日志将不再添加多余的

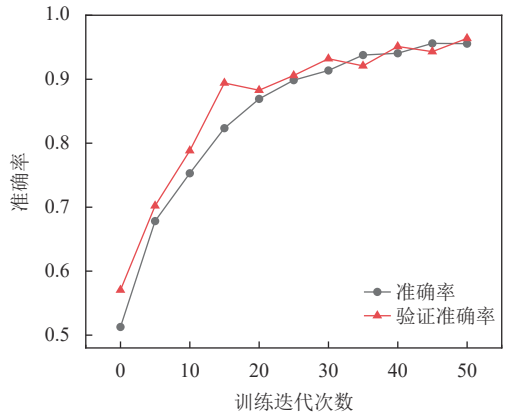
日志语句, 避免为后续学习日志文本带来多余开销.

4.3 异常日志检测模型实验评估

为了验证异常日志检测模型的准确性, 本文结合训练迭代次数进行实验分析, 结果如图 5.



(a) 模型训练损失值变化



(b) 模型训练准确率变化

图 5 模型性能评估

实验记录了模型训练的损失值, 如图 5(a) 所示. 随着实验训练轮次的增加可以明显看出损失值与预测损失值在明显降低, 损失值是训练数据的成本函数值, 而验证损失值是交叉验证数据的成本函数值. 本文使用损失函数来表明预测值与真实值之间的差距, 损失值逐渐减小表明检测模型的训练效果较好, 同时验证损失值逐渐减小表明模型的分类效果逐渐符合预期, 损失函数值越小得到的监测模型的检测效果越好. 随着训练轮次的迭代增加, 验证损失值和损失值一直在平稳下降, 这表示训练正常也是模型最优状态. 本文在训练过程中同样记录了模型精度与验证精度的变化过程, 如图 5(b) 所示, 随着实验训练轮次的增加可以明显看出精度逐渐攀升, 模型精度用模型训练的准确率来衡量, 而验证精度是使用交叉验证数据的准确率值. 通过调整网络层数等超参数, 使模型逐渐处于相对最优的状态, 从而获得各项评价指标. 从图 5 中可以发现在迭代轮次未到达 20 次时, 检测模型持续学习. 通过两张图对比观察发现, 当横坐标的训练轮次处于 30–50 之间时, 各类损失和准确率分别保持逐渐减小和增加的趋势, 曲线趋于平缓, 此时模型逐渐收敛并稳定, 说明异常日志检测模型训练效果符合预期.

为了更直观地说明模型的训练效果, 通过生成混淆矩阵来观察实验结果, 图 6 表示为模型训练后产生的混淆矩阵. 对比表 2 与图 6 的格式一致, 当模型完成训练后, 使用测试集的数据进行检测以验证模型的有效性, 横坐标表示预测标签类别, 纵坐标表示真实标签类别. 深色部分表示为判定正确的准确率, 可从图 6 中看到判定正常日志和异常日志准确率分别为 96% 和 92%, 两种准确率均达到 90% 以上.

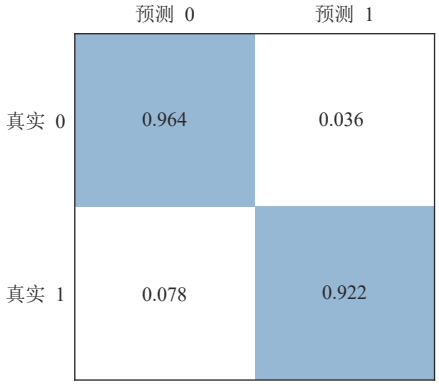


图 6 混淆矩阵

为了体现检测模型的准确性, 使用 HDFS 日志数据进行实验验证, 最终检测结果如表 4 所示. 表格中显示的是训练后的检测指标, 设定的评价指标都保持在 93% 以上, 其中 $F1$ 值稳定在 95%. 模型的检测结果与真实值误差较小, 表现出良好性能.

表 4 模型评价指标

指标	准确率	精确率	召回率	$F1$值
结果	0.9363	0.9513	0.9594	0.9553

4.4 对比实验

为了进一步验证检测框架的优势, 遵循实验公平原则, 对比实验采用相同的数据集, 训练集、测试集的比例保持一致. 为了验证异常日志检测模型的性能, 选取了 CNN、Transformer、Autoencoder 和 LSTM 这 4 种常用的深度学习算法进行对比实验, 其中 $F1$-score 作为综合指标是本文进行比较的重要指标, 该值表现

良好的分类器被视为分类效果较好的分类器. 图 7 表示为在 HDFS 数据条件下, 5 种分类器的检测表现情况, 为了比对效果更直观将各类结果绘制成柱状图, 其中 BiLSTM 表示为本文构建的异常日志检测模型. 从图中可以看出在 HDFS 数据下, 4 种评价指标均表现良好, 预测的指标都能保持在 90% 以上, 其中召回率明显高于其他分类器. 虽然 LSTM 在准确率上略高于 BiLSTM, 但是精确率和召回率都表现不佳, 这种现象表明这些对比模型对正常日志的检测效果良好, 但容易将部分异常日志划分为正常日志, 这将为系统安全带来隐患. 其余分类器在联邦学习的限制条件下, 各类评价指标仍低于本文模型. 为了进一步验证架构的有效性, 本文选择了一组与原始数据不同的数据集进行验证, 以避免偶然性对实验结果的影响.

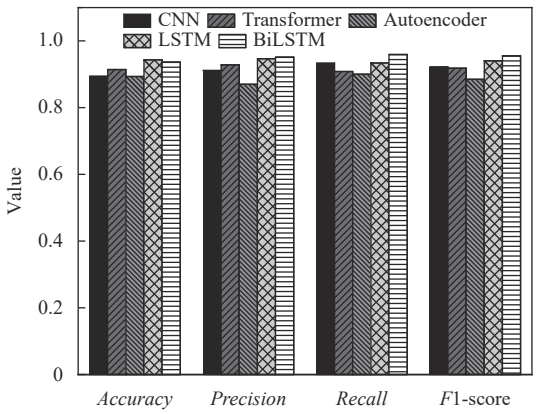


图 7 基于不同分类器的检测结果对比

图 8 表示为在 BGL 新数据集条件下, 5 种分类器的检测表现情况. 在 BGL 数据集下, LSTM 的召回率略高于本文模型架构 1%, LSTM 可能更倾向于将日志样本判定为正. Autoencoder 的评价指标明显低于其他分类模型, 在本文的联邦学习背景下, 这种无监督学习算法不适用于判定日志性质. BiLSTM 在 4 种评价指标下仍稳定保持在 94% 以上, 在精确率上表现良好. 新日志数据具有不同的特征和分布, 通过对该数据集进行实验验证, 可以更全面地评估本文提出架构在不同场景下的适用性和性能表现. 这一步骤是必要的, 可以增强对架构有效性的验证, 并为实际应用提供可靠支持.

本阶段对比实验采用了联邦学习和集中式学习两种方法进行实验对比. 在联邦学习方案中, 各个本地服务器独立训练 BiLSTM 模型, 并利用 Fed-Avg 算法在中心服务器上聚合参数. 而集中式学习则将所有日志数据汇集到中心服务器, 在此进行统一的 BiLSTM 模型训练. 由于所有数据集中在一处, 训练过程中能够充分利用所有数据的信息, 模型参数通过统一的数据进行更新, 直至模型收敛. 实验设置中, 保证了相同的硬件环境, 使用相同的数据集, 以确保实验的可比性.

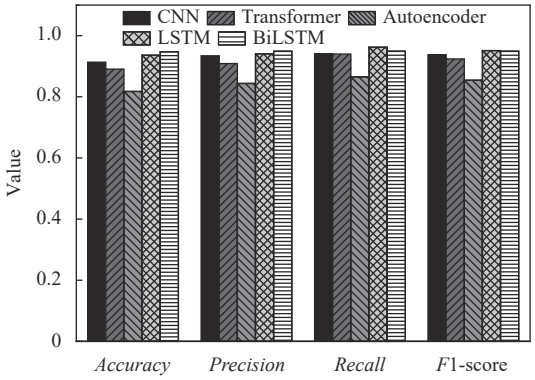


图 8 基于 BGL 数据集的不同分类器检测结果对比

为了验证本文的联邦学习建模结果与集中式学习之间的差异, 在相同的环境下进行实验, 表 5 表现为与集中式学习方法进行性能比较. 从表 5 可以看出本文的方法虽然略低于集中式的检测方法, 但仍然保持在 2% 以内的评价指标波动范围内. 这表明, 尽管联邦学习的方法在一定程度上牺牲了一些性能, 但在避免数据隐私泄露的前提下, 仍然能够保持较高水平的异常检测指标. 这进一步验证了本文提出的基于联邦学习的日志异常检测架构的稳定性和可行性.

表 5 集中式模型与联邦学习模型评估

指标	准确率	精确率	召回率	F1值
集中式BiLSTM	0.9490	0.9637	0.9730	0.9683
联邦学习BiLSTM	0.9363	0.9513	0.9594	0.9553

图 9 是各类集中式训练的分类器检测结果. 在集中式训练条件下, BiLSTM 和其他分类器模型比较仍能表现良好, 说明了选择 BiLSTM 作为异常日志检测的初始模型的正确性. CNN 在召回率指标上逼近 BiLSTM 的结果, 但在准确率和精确率上仍存在一定差距. 通过观察综合考虑了精确率和召回率的 $F1$ 值可以发现, BiLSTM 比 CNN 高 1%, 调和两种评价指标并综合考虑模型的性能后, BiLSTM 模型的检测结果更稳定.

5 结论与展望

本文提出了一种基于联邦学习的异常日志检测架构. 首先, 本文设计了联邦学习架构解决分布式数据系

统下的异常日志检测问题, 这种方法有效解决了集中式方法存在的日志数据隐私泄露问题, 仅通过传输模型参数就能够实现多方日志检测模型聚合. 其次, 本文设计了树解析器模板实现日志文件结构化, 并且利用 TF-IDF 实现文本信息的特征选择, 这种方法能够有效处理日志文本并保留日志信息. 最终, 本文选取基于自注意力机制的 BiLSTM 作为初始模型下发给联邦节点, 不仅考虑到日志文本的上下文关系, 而且实现对文本的关键信息捕捉, 进而提高模型训练的准确性. 利用仿真实验进行模型评估, 并与其他深度学习模型、集中式检测方法进行对比实验. 实验结果表明, 本文提出的检测模型能够得到较好的效果, 不仅保证了日志数据的隐私安全而且验证了模型的有效性.

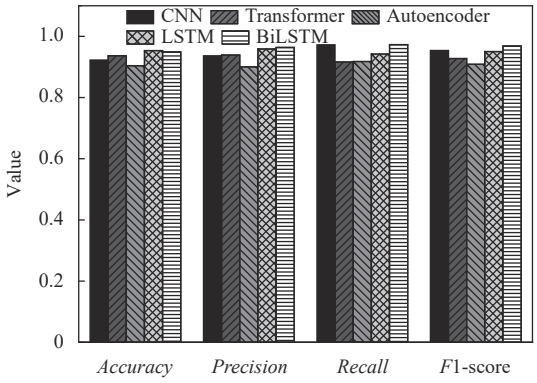


图 9 基于集中式的检测结果对比

参考文献

1 周德, 杨成慧, 罗佃斌. 基于 Hadoop 的分布式日志分析系统设计与实现. 现代信息科技, 2023, 7(23): 57–60. [doi: 10.19850/j.cnki.2096-4706.2023.23.012]

2 Le VH, Zhang HY. Log-based anomaly detection with deep learning: How far are we? Proceedings of the 44th International Conference on Software Engineering. Pittsburgh: ACM, 2022. 1356–1367. [doi: 10.1145/3510003.3510155]

3 Verbraeken J, Wolting M, Katzy J, *et al*. A survey on distributed machine learning. ACM Computing Surveys (CSUR), 2020, 53(2): 30. [doi: 10.1145/3377454]

4 Landauer M, Onder S, Skopik F, *et al*. Deep learning for anomaly detection in log data: A survey. Machine Learning with Applications, 2023, 12: 100470. [doi: 10.1016/j.mlwa.2023.100470]

5 Lou JG, Fu Q, Yang SQ, *et al*. Mining invariants from console logs for system problem detection. Proceedings of the 2010 USENIX Annual Technical Conference. Boston: USENIX Association, 2010. 24.

6 Zhang X, Xu Y, Lin QW, *et al*. Robust log-based anomaly detection on unstable log data. Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Tallinn: ACM, 2019. 807–817. [doi: 10.1145/3338906.3338931]

7 Zhang CK, Wang XY, Zhang HY, *et al*. LayerLog: Log sequence anomaly detection based on hierarchical semantics. Applied Soft Computing, 2023, 132: 109860. [doi: 10.1016/j.asoc.2022.109860]

8 Fenza G, Gallo M, Loia V. Drift-aware methodology for anomaly detection in smart grid. IEEE Access, 2019, 7: 9645–9657. [doi: 10.1109/ACCESS.2019.2891315]

9 Guo HC, Guo YH, Yang J, *et al*. LOGLG: Weakly supervised log anomaly detection via log-event graph construction. Proceedings of the 28th International Conference on Database Systems for Advanced Applications. Tianjin: Springer, 2023. 490–501. [doi: 10.1007/978-3-031-30678-5_36]

10 Lee C, Yang TY, Chen ZB, *et al*. Heterogeneous anomaly detection for software systems via semi-supervised cross-modal attention. Proceedings of the 45th IEEE/ACM International Conference on Software Engineering. Melbourne: IEEE, 2023. 1724–1736. [doi: 10.1109/ICSE48619.2023.00148]

11 He PJ, Zhu JM, Zheng ZB, *et al*. Drain: An online log parsing approach with fixed depth tree. Proceedings of the 2017 IEEE International Conference on Web Services. Honolulu: IEEE, 2017. 33–40. [doi: 10.1109/ICWS.2017.13]

12 Johnson SJ, Murty MR, Navakanth I. A detailed review on word embedding techniques with emphasis on Word2Vec. Multimedia Tools and Applications, 2024, 83(13): 37979–38007. [doi: 10.1007/s11042-023-17007-z]

13 Abid MA, Mushtaq MF, Akram U, *et al*. Comparative analysis of TF-IDF and loglikelihood method for keywords extraction of twitter data. Mehran University Research Journal of Engineering and Technology, 2023, 42(1): 88–94. [doi: 10.22581/muet1982.2301.09]

14 Abou Houran M, Bukhari SMS, Zafar MH, *et al*. COA-CNN-LSTM: Coati optimization algorithm-based hybrid deep learning model for PV/wind power forecasting in smart grid applications. Applied Energy, 2023, 349: 121638. [doi: 10.1016/j.apenergy.2023.121638]

15 Alaca Y, Celık Y, Goel S. Anomaly detection in cyber

security with graph-based LSTM in log analysis. Chaos Theory and Applications, 2023, 5(3): 188–197. [doi: 10.51537/chaos.1348302]

16 Han PF, Li HK, Xue G, *et al*. Distributed system anomaly detection using deep learning-based log analysis. Computational Intelligence, 2023, 39(3): 433–455. [doi: 10.1111/coin.12573]

17 Fu YY, Liang K, Xu J. MLog: Mogrifier LSTM-based log anomaly detection approach using semantic representation. IEEE Transactions on Services Computing, 2023, 16(5): 3537–3549. [doi: 10.1109/TSC.2023.3289488]

18 Xu W, Huang L, Fox A, *et al*. Online system problem detection by mining patterns of console logs. Proceedings of the 9th IEEE International Conference on Data Mining. Miami Beach: IEEE, 2009. 588–597. [doi: 10.1109/ICDM.2009.19]

19 Liu X, Liu WY, Di XQ, *et al*. LogNADS: Network anomaly detection scheme based on log semantics representation. Future Generation Computer Systems, 2021, 124: 390–405. [doi: 10.1016/j.future.2021.05.024]

(校对责编: 张重毅)