1.日志类型：

Loghub包含从6类系统收集的19个日志数据集。

1.1分布式系统：HDFS、Hadoop、Spark、Zookeeper、OpenStack

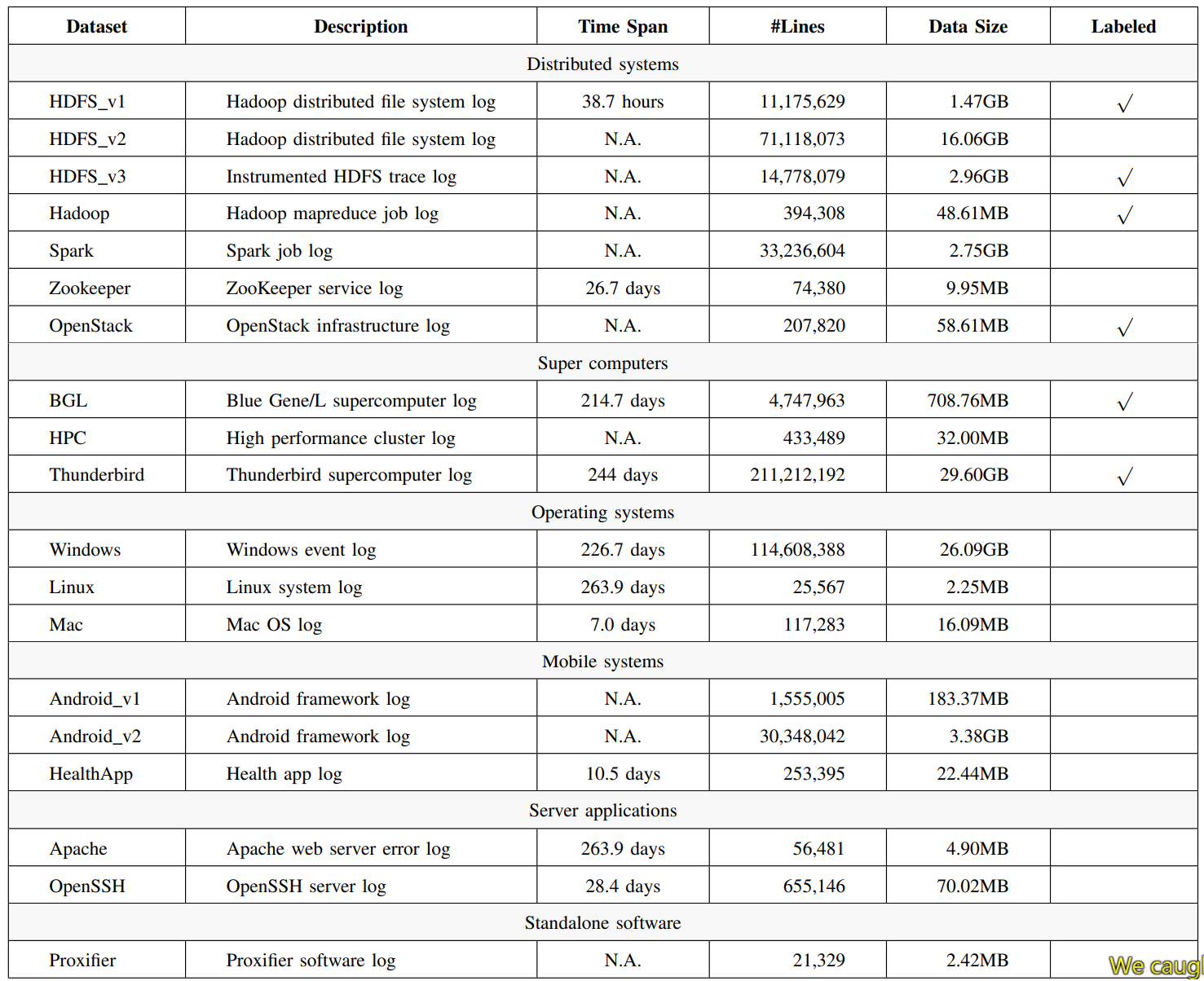
1.2超算：BGL、HPC、Thunderbird

1.3操作系统：windows、Linux、Mac

1.4移动应用：Android、HealthApp

1.5服务器应用：Apache、OpenSSH

1.6独立软件：Proxifier网络应用



2.数据集的应用场景

2.1日志解析：将非结构化日志消息（含调试信息和自然文本）转换为结构化系统事件，例如带有事件 ID 的系统事件列表或矩阵。

2.2日志压缩：系统维护往往需要存储较长时间段的日志信息，归档系统日志会消耗大量存储空间，导致高昂的成本。

2.3异常检测：

基于日志的异常检测方法，通过分析系统运行时日志来报告潜在的异常系统行为，进而增强系统的可靠性。

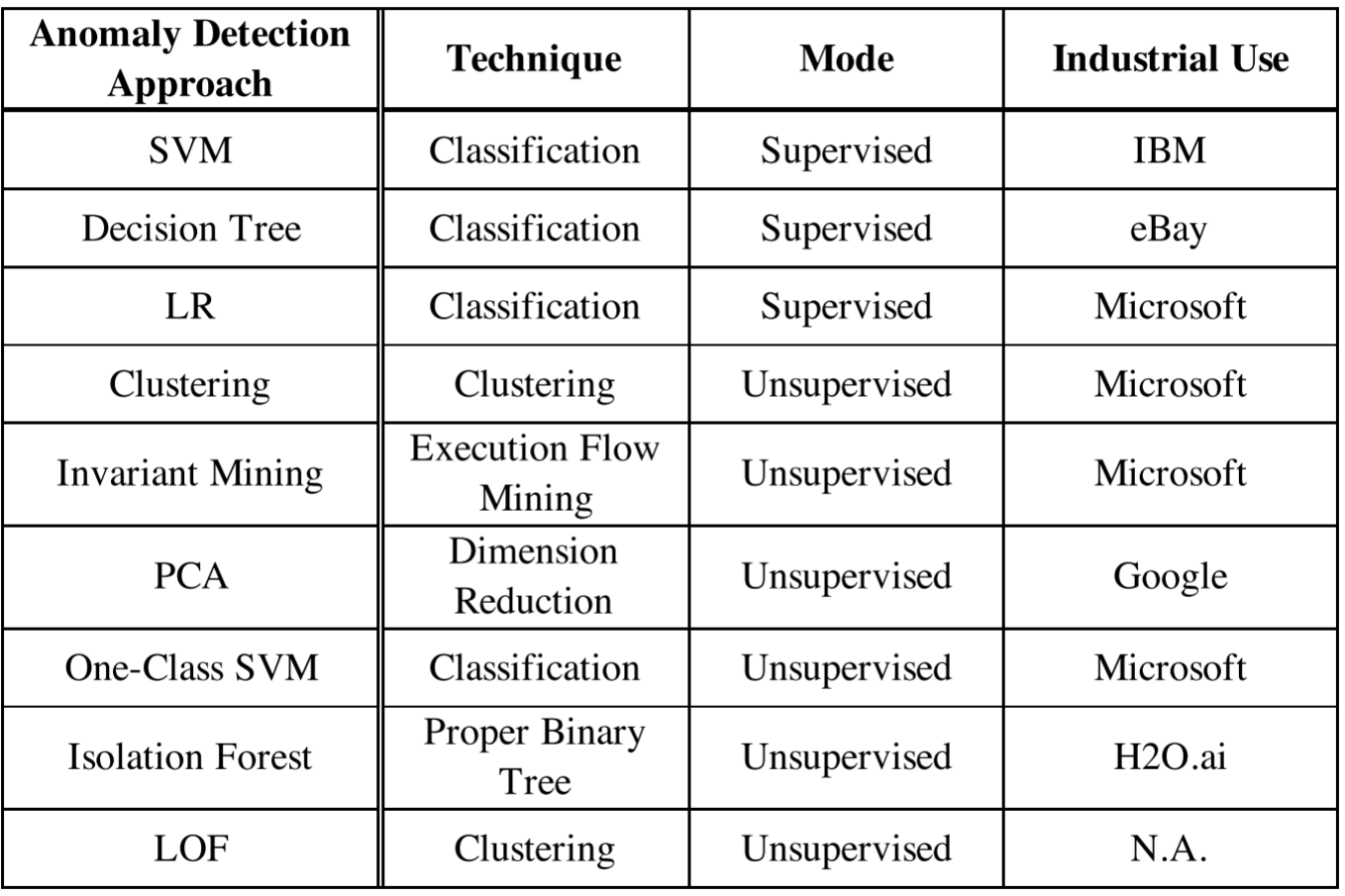
异常检测常归类为二元分类问题： 输入结构化系统事件列表或矩阵，输出指示实例（例如事件或时间段）是否异常的标签列表。

日志异常检测方法主要有两类：无监督和监督。采用了分类算法常用的评估指标F1 分数。

研究过程使用有标注的数据集，包含实例异常标签的日志数据。

2.4重复问题识别：识别重复问题有助于减少开发人员手工投入。建模为聚类问题，两个问题的日志消息表现出相似的模式（例如出现频率、顺序），则这两个问题将聚集到同一组中。

3. 异常检测应用案例（基准测试）



1. LogLLM：利用大模型进行基于日志的异常检测

基于日志的异常检测已成为一个重点研究领域，旨在通过日志数据识别系统问题，最终提高软件系统的可靠性。传统的深度学习方法通​​常很难捕获嵌入在日志数据中的语义信息，而日志数据通常以自然语言组织。

4.1 LogLLM 采用 BERT（Transformer编码器）从日志消息中提取语义向量，同时利用基于Transformer解码器的模型 Llama 对日志序列进行分类。

LLM因为参数很大，如何将整个日志序列输入到llama会导致内存不足的问题，且难以定位异常所在。LogLLM采用Bert汇总每条日志消息，有效缓解该问题；

BERT 的使用确保了日志消息之间的清晰边界，因为每条消息都由不同的嵌入向量表示，从而提高了分类性能。

4.2 引入了一个projector来对齐 BERT 和 Llama 的向量表示空间，确保对日志语义的一致理解。

4.3 与需要日志解析器提取模板的传统方法不同，LogLLM 使用**正则表达式预处理**日志消息，从而简化了整个过程；通过一种新颖的三阶段程序进行训练，旨在增强性能和适应性。

四个公共数据集的实验结果表明 LogLLM 优于最先进的方法。 即使在处理不稳定日志时，也能有效捕获日志消息的语义并准确检测异常。

5.改进与创新点

5.1 LogLLM同时采用基于Transformer编码器和基于解码器的LLM，特别是BERT和Llama，进行基于日志的异常检测。

5.2 提出了一种新颖的三阶段程序来优化深度模型中不同组件的训练和协调，从而提高性能和适应性。

6.基于深度学习的传统方法，分为两类：基于重建的方法和基于二元分类的方法。

7.基于LLM的日志异常检测：

7.1 基于Transformer编码器的模型，如BERT、RoBERTa、SpanBERT

7.2 基于Transformer解码器的模型，如GPT-4、Llama3、ChatGLM等

7.3 利用LLM的两种流行策略是提示词工程和微调prompt engineering and finetuning。

（1）基于提示词工程的方法

依靠LLM内部的知识检测异常，通常采用基于Transformer解码器模型。但基于提示词工程的方法通常很难为特定数据集定制解决方案，因此检测性能不佳。

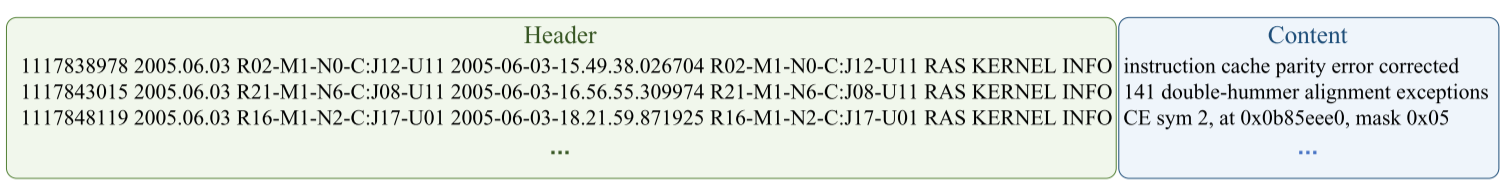
（2）基于微调的方法

将LLM合并到深度神经网络中，并根据用户自己的数据集对其进行定制。

8.实现方法

日志序列使用正则表达式进行预处理，然后输入深度神经网络，该网络集成了 BERT [43]、投影仪和 Llama [26] 以进行日志序列分类。

8.1 预处理

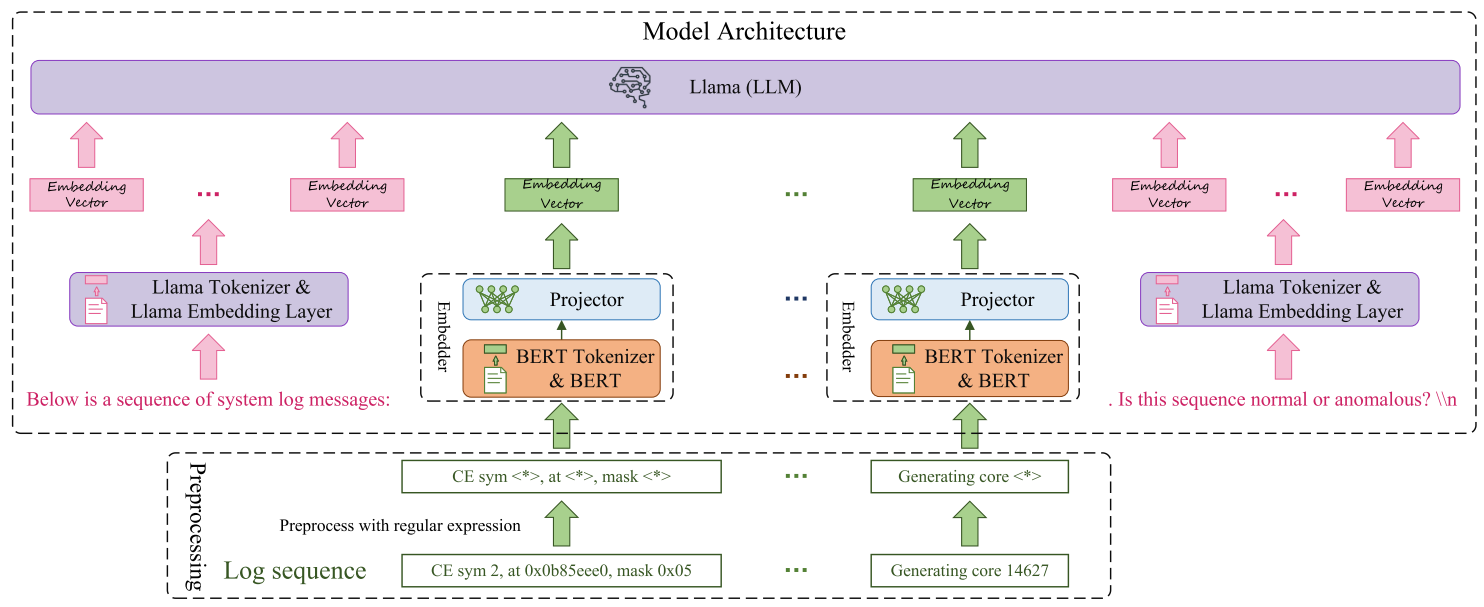


日志常包含动态runtime信息的可变参数，这些参数始终与异常无关，并使深度模型训练复杂化，因此需要识别这些参数并用常量token替换它们。

但现有的日志解析器并不是在所有的日志信息中表现良好，难以处理log信息中词汇外的（OOV）单词，从而导致语义信息丢失。日志信息不稳定也会导致解析器的作用越来越弱，不能支持后续的异常检测。

使用简易的正则表达式可以识别特定对象的参数文本格式。将可变参数（例如帐户、目录路径和 IP 地址）替换为“<\*>”。优势：快速高效且不需要训练。

8.2 模型架构



深度模型组成如下，期中，BERT和Llama都是经过预训练的LLM：

（1）BERT 用于提取日志消息的向量表示。

* 每条日志消息经预处理后，使用 BERT 分词器和 BERT 模型编码为语义向量。
* BERT 的输出是语义向量序列 C = (c1, c2, ..., cN) ∈ RN×dBERT ，其中 N 表示日志序列的长度（日志消息数量），dBERT 是每个语义向量的维度（即隐藏大小）。

（2）Llama 用于对日志序列进行分类。

根据嵌入的日志序列生成相应的文本查询，每个查询由三个部分组成：

a.第一部分介绍日志序列，例如“以下是系统日志消息的序列：”

b.第二部分包括投影仪输出的令牌嵌入 E。

c.第三部分询问该序列是否异常，例如询问“该序列是正常的还是异常的？”。

第一和第三部分分别输入Llama tokenizer分词器和Llama embedding layer嵌入层。然后将三个部分的token embeddings连接起来，输入给Llama。

（3）projector充当桥梁，对齐 BERT 和 Llama 的向量表示空间。

projector为线性层，将语义向量 C ∈ RN×dBERT 映射到 Llama 接受的 token 嵌入向量，表示为 E = (e1, e2, ..., eN) ∈ RN×dLlama ，其中 dLlama 是 Llama 的隐藏大小。

8.3 训练

（1）少数类过采样

LogLLM 是一种有监督的异常检测方法，检测结果仅有正常和异常两种。为了处理数据不平衡的情况，对样本较少的类进行过采样。

（2）目标：训练深度模型来预测给定的日志序列是正常还是异常

（3）训练过程分为三个阶段：

a. **微调 Llama 以捕获答案模板**：

第一阶段涉及微调 Llama 以捕获答案模板。训练 Llama 用“序列是异常/正常”来响应“这个序列是正常还是异常？”的prompt。 此阶段仅需要少量数据样本。

b. **训练日志消息嵌入器**

第二阶段涉及训练日志消息embedder，特别是 BERT 和projector，目标是将每条日志消息投影到 Llama 中最合适的 token 的嵌入中，使 Llama 能够辨别给定的日志序列是正常还是异常。

c. 微调整个模型

微调整个模型，以确保所有组件的一致和准确的性能。

（4）LLM 的高效微调

利用 QLoRA来最大限度地减少内存使用。QLoRA 通过将梯度反向传播到冻结的 4 位量化模型中来实现这一目标，同时保持在完整 16 位微调过程中实现的性能水平。

QLoRA的核心思想是：不直接微调整个庞大的模型，而是通过一个微小的“适配器”来间接影响模型，并且在整个过程中，大模型本身始终以一种高度压缩的格式（4-bit）存在。

9.实验

9.1 实验设置

使用**BERT-BASE**模型和**LLAMA-3-8B**模型作为主干。

9.2 评价指标

P、R、F1

P = TP/(TP+FP)、R=TP/(TP+TN)、F1=2\*PR/(P+R)

9.3 数据集

划分比例，训练集:测试集=8:2

（1）会话窗口

HDFS，基于每个日志消息中的block\_id，将志消息分组为序列。

（2）滑动窗口

BGL、Thunderbird和Liberty，使用滑动窗口策略对日志消息进行分组，窗口大小为100条消息，步长为100条消息。如果日志序列包含至少一条异常日志消息，则该日志序列被视为异常。