Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心


# 日志数据的深度学习异常检测方法

博士研究生　张毅飞

2021年09月21日

# 内容提要

- 背景简介
- 基本概念
  - 日志数据
  - 日志解析
  - 检测流程
- 算法原理
  - DeepLog
  - LogAnomaly
- 应用总结
- 参考文献

# 预期收获

- 预期收获
  - 1. 了解日志数据异常检测的问题和挑战
  - 2. 了解日志数据的分类、用途、解析等
  - 3. 理解日志数据深度学习检测方法
  - 4. 了解应用领域和发展方向等

# 背景简介

- 异常检测是构建安全可靠的计算机系统的一项基本任务
  - 学术界与工业界将日志数据广泛地应用于异常检测
- 各种系统日志是异常检测和在线监测的最优质的信息源
  - 日志数据在几乎所有的计算机系统中都是普遍存在和可用的
  - 系统日志记录不同关键点的系统状态和重要事件
  - 理解系统状态和性能问题

# 背景简介

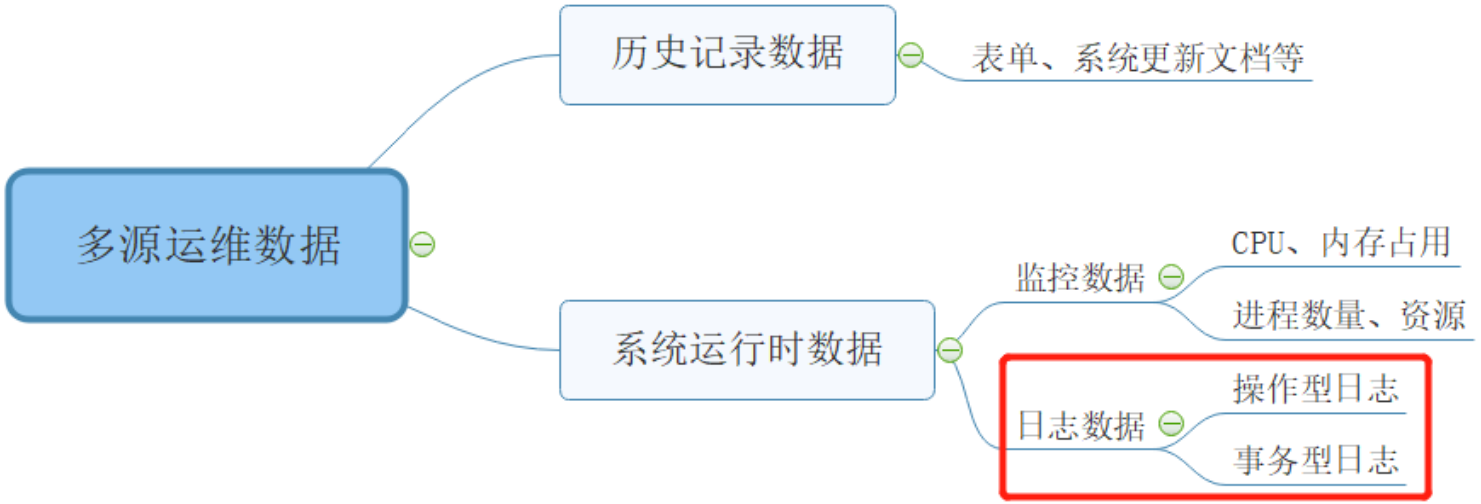
- 利用日志数据在大规模分布式系统(虚拟化云计算平台)进行异常检测面临挑战
  - 从海量日志数据中进行在线异常检测极具挑战性
    - 现代系统的日志正以大约50GB的速度增长（约每小时2亿行）
  - 日志数据是非结构化的，格式和语义因系统而异
    - 发生问题时使用非结构化日志数据检查异常、定位问题、分析根因极为困难
  - 现有方法成为一项劳动密集型且容易出错的任务
    - 例如主成分分析(PCA)、不变量挖掘(IM)、聚类等无法顺利解决系统迭代导致的日志格式变化、海量数据实时处理等问题

# 基本概念

- 数据对象
  - 多源运维数据
    - 历史记录数据：主要包含表单和系统更新文档等
    - 系统运行时数据：反映系统的动态特征及系统发生故障时的上下文信息，对故障和异常具有更好的探测和表达能力
  - 系统运行时数据
    - 监控数据：系统运行状态下的资源占用情况
    - 日志数据：程序开发人员为辅助调试在程序中嵌入的打印输出代码所产生的文本数据，用以记录程序运行时的变量信息、程序执行状态等。

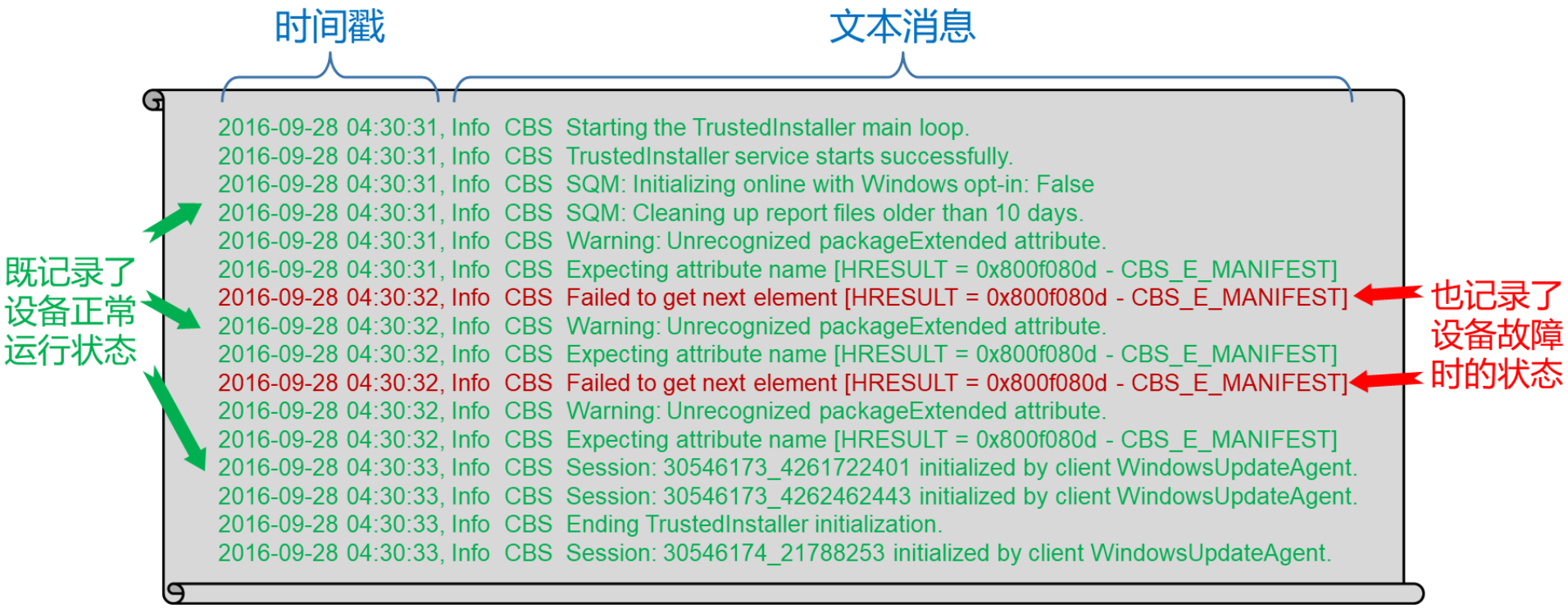
# 基本概念

- 日志数据
  - 关注细粒度的程序执行逻辑，能够定位到特定的日志及事件信息，更加适用于异常检测、故障诊断等任务
    - 操作型日志：每一条日志代表一个独立事件，相互之间一般是相互独立的
    - 事务型日志：表征请求或事务执行逻辑，一般存在明显因果关联关系



# 基本概念

- 日志数据实例
  - windows公开数据集中的部分日志样例
    - 日志是一种时序文本数据，由时间戳和文本消息组成，实时记录业务运行状态，通过收集并分析日志，可以发现或预知系统中已发生或潜在的异常、故障等



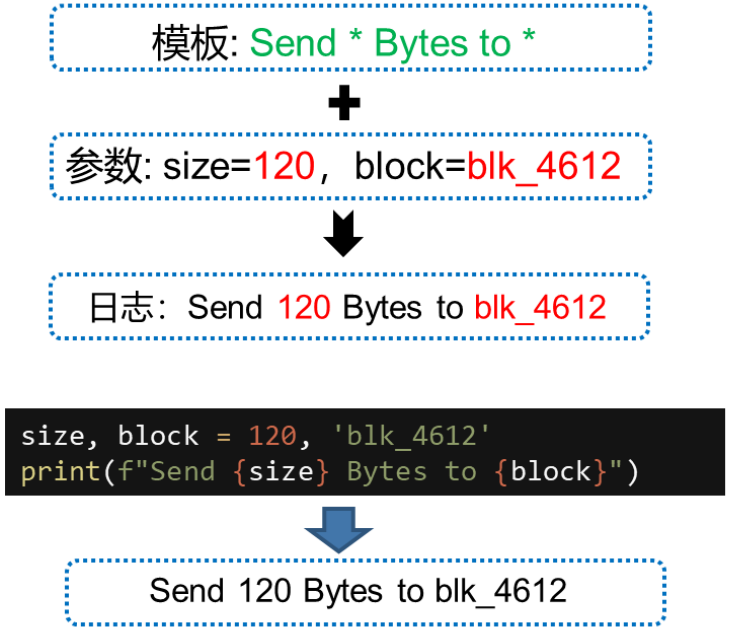
# 基本概念

- 日志数据实例
  - 分布式应用系统中各种不同日志
    - 格式不同，呈现非结构化特征，格式不统一、时间戳不统一、记录粒度不统一、专用词汇及缩略词不统一等

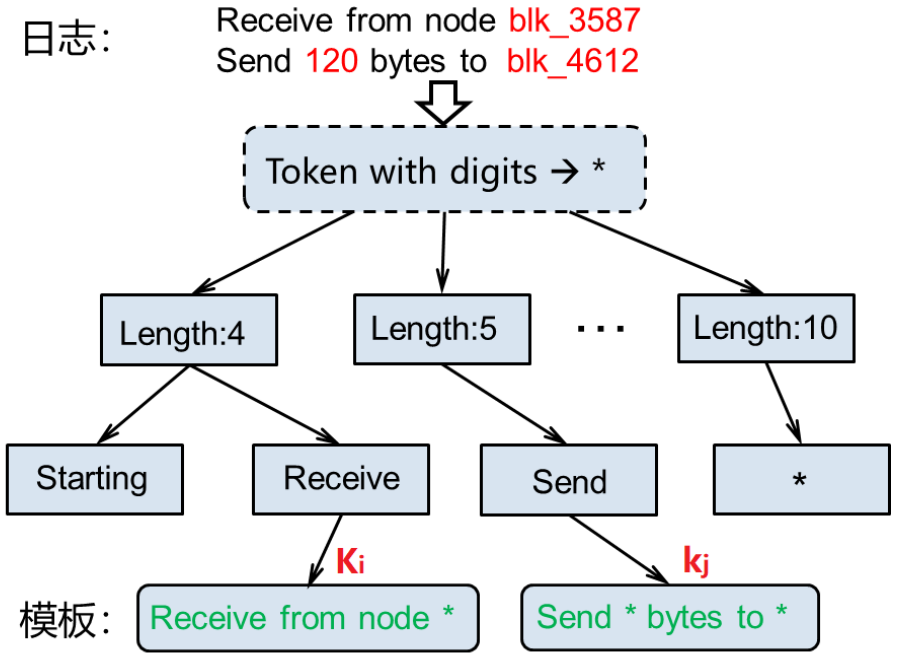
```
HDFS日志样例数据
081110 103850 32 INFO dfs.FSNamesystem: BLOCK* NameSystem.allocateBlock: /user/root/rand/_temporary/_task_200811101024_0001_m_000056
081110 103922 19 INFO dfs.FSDataset: Deleting block blk_7754477838724397551 file /mnt/hadoop/dfs/data/current/subdir46/blk_775447783
081110 104018 9085 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_-3452734573703603740 terminating
081110 104047 8882 INFO dfs.DataNode$PacketResponder: Received block blk_-4034227974316905164 of size 67108864 from /10.250.15.198
081110 104224 34 INFO dfs.FSNamesystem: BLOCK* NameSystem.allocateBlock: /user/root/rand/_temporary/_task_200811101024_0001_m_000346
081110 104231 8978 INFO dfs.DataNode$DataXceiver: Receiving block blk_2972699288862031202 src: /10.250.5.161:51077 dest: /10.250.5.1

Hadoop日志样例数据
2015-10-18 18:01:47,978 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: Created MRAppMaster for application appattempt_1
2015-10-18 18:01:48,963 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: Executing with tokens:
2015-10-18 18:01:48,963 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: Kind: YARN_AM_RM_TOKEN, Service: , Ident: (appAt
2015-10-18 18:01:49,228 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: Using mapred newApiCommitter.
2015-10-18 18:01:50,353 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: OutputCommitter set in config null
2015-10-18 18:01:50,509 INFO [main] org.apache.hadoop.mapreduce.v2.app.MRAppMaster: OutputCommitter is org.apache.hadoop.mapreduce.l

OpenStack日志样例数据
nova-compute.log.1.2017-05-16_13:55:31 2017-05-16 00:00:04.562 2931 INFO nova.compute.manager [req-3ea4052c-895d-4b64-9e2d-04d64c4d9
nova-compute.log.1.2017-05-16_13:55:31 2017-05-16 00:00:04.693 2931 INFO nova.compute.manager [req-3ea4052c-895d-4b64-9e2d-04d64c4d9
nova-api.log.1.2017-05-16_13:53:08 2017-05-16 00:00:04.789 25746 INFO nova.osapi_compute.wsgi.server [req-bbfc3fb8-7cb3-4ac8-801e-c8
nova-api.log.1.2017-05-16_13:53:08 2017-05-16 00:00:05.060 25746 INFO nova.osapi_compute.wsgi.server [req-31826992-8435-4e03-bc09-ba
nova-compute.log.1.2017-05-16_13:55:31 2017-05-16 00:00:05.185 2931 INFO nova.virt.libvirt.imagecache [req-addc1839-2ed5-4778-b57e-5
nova-compute.log.1.2017-05-16_13:55:31 2017-05-16 00:00:05.186 2931 INFO nova.virt.libvirt.imagecache [req-addc1839-2ed5-4778-b57e-5
nova-compute.log.1.2017-05-16_13:55:31 2017-05-16 00:00:05.367 2931 INFO nova.virt.libvirt.imagecache [req-addc1839-2ed5-4778-b57e-5
nova-api.log.1.2017-05-16_13:53:08_2017-05-16 00:00:06.321 25746 INFO nova.osapi_compute.wsgi.server [req-7160b3e7-676b-498f-b147-77

Apache日志样例数据
[Sun Dec 04 04:47:44 2005] [notice] workerEnv.init() ok /etc/httpd/conf/workers2.properties
[Sun Dec 04 04:47:44 2005] [error] mod_jk child workerEnv in error state 6
[Sun Dec 04 04:51:08 2005] [notice] jk2_init() Found child 6725 in scoreboard slot 10
[Sun Dec 04 04:51:09 2005] [notice] jk2_init() Found child 6726 in scoreboard slot 8
[Sun Dec 04 04:51:14 2005] [notice] workerEnv.init() ok /etc/httpd/conf/workers2.properties
```

# 基本概念

- 日志解析(log parse)
  - 非结构化的日志数据直接处理非常困难，通常的做法是通过日志解析得到日志的模板(log key/template)，然后再对log key序列进行异常检测



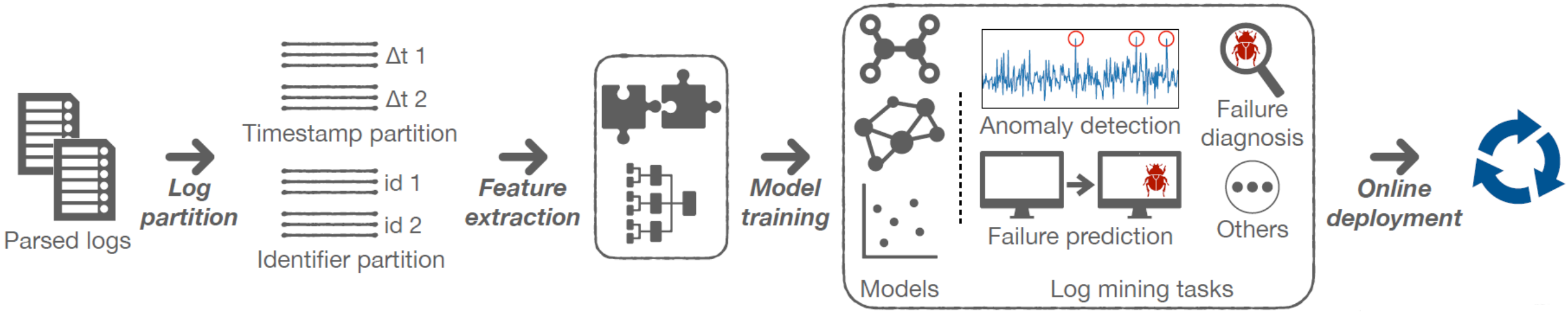

(a) 模板与日志的关系




(b) 日志解析原理图

# 基本概念

- 基本流程
  - 日志分割（Log partion）
  - 特征提取（Feature extraction）
  - 模型训练（Model training）
  - 在线部署（Online deployment）



# 算法原理

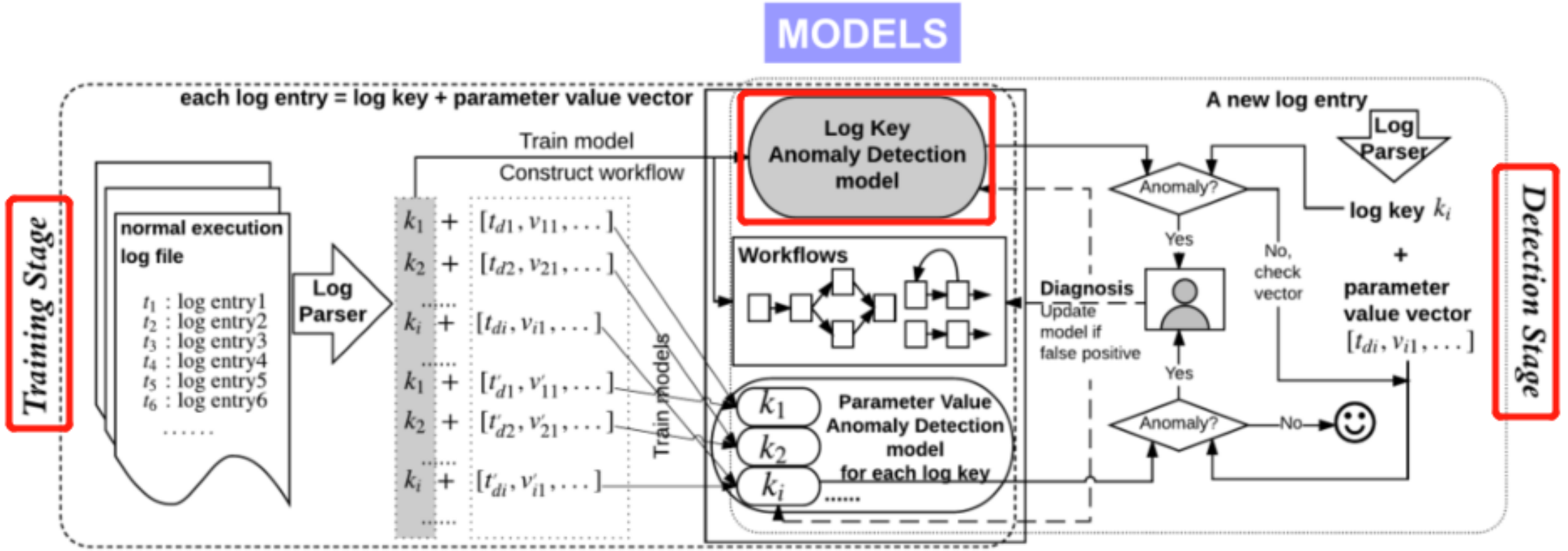
# 算法原理 DeepLog

| | |
|---|---|
| T | 检测日志异常 |
| I | 系统/应用运行产生的日志数据 |
| P | 1.解析日志并抽取log keys<br>2.按照不同任务、线程等将日志转换为log key序列<br>3.根据滑窗长度、步长等参数获取训练样本<br>4.训练多层堆叠LSTM模型<br>5.对待检测日志数据进行预处理、预测和判定 |
| O | 待检测日志数据是/否异常 |

| | |
|---|---|
| P | 海量非结构化日志数据的实时检测 |
| C | 多任务、多线程、可解析的未受攻击日志数据（篡改等） |
| D | 序列模型如何在检测阶段实现异常判别 |
| L | ACM CCS (CCF-A会议) |

# 算法原理 DeepLog

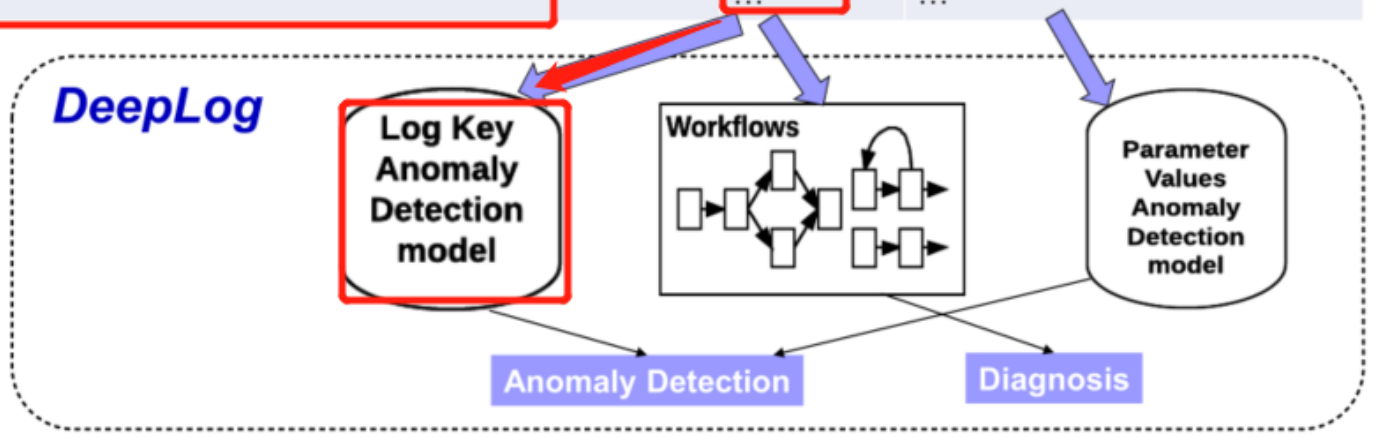
- 总体框架
  - 三个主要组件：日志键异常检测模型、参数值异常检测模型和对所检测到的异常进行诊断的工作流模型



# 算法原理 DeepLog

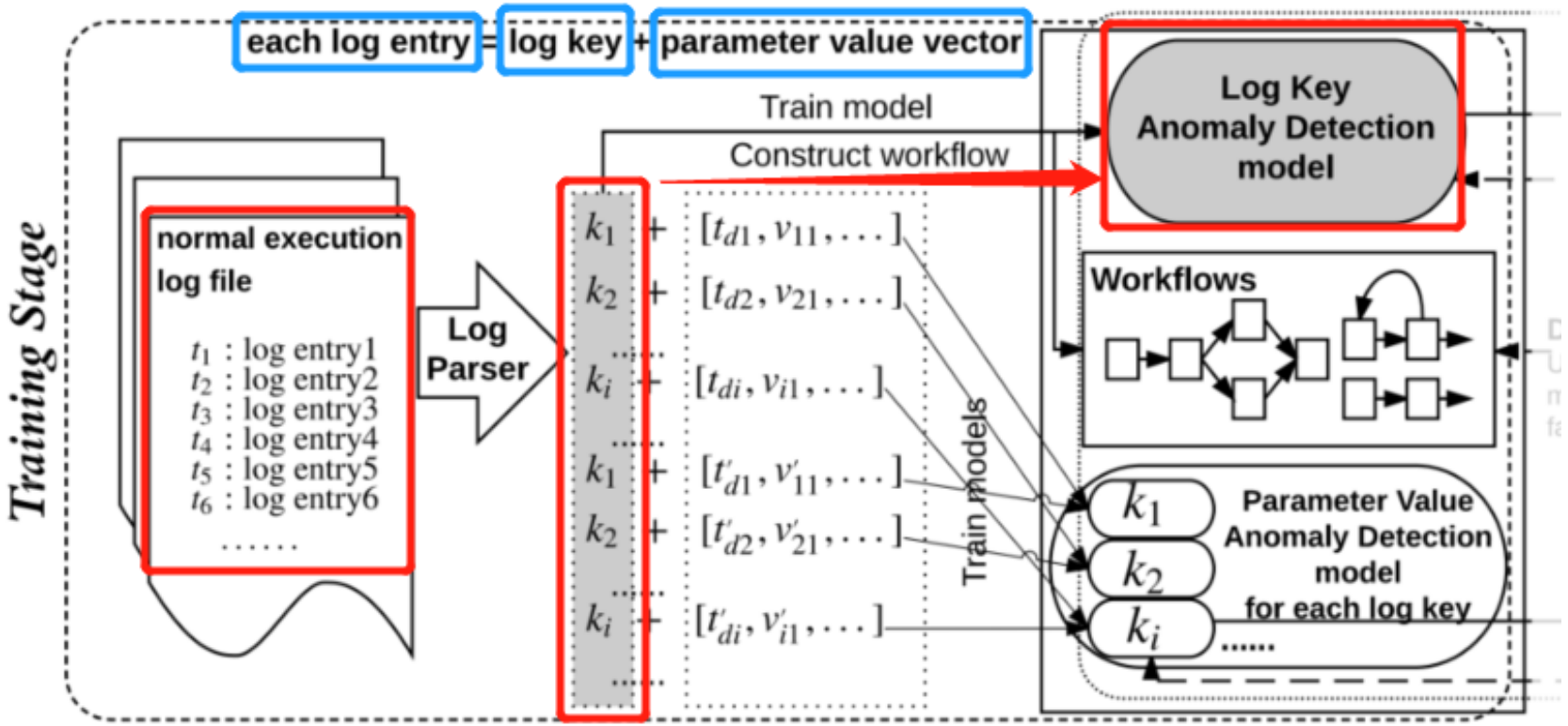
- 日志解析（log parser）
  - 将非结构化自由文本日志条目解析为结构化表示，通过学习过程构建顺序模型
  - 按线程号、任务号等区分和提取不同隶属的模板序列
  - 某特定系统中所有不同log key的总数n是恒定的，即$K = \{k_1, k_2, \ldots, k_n\}$
  - 日志条目被解析为log key序列，其反映了系统代码/事件的特定执行/触发顺序

| log message (log key underlined) | log key | parameter value vector |
|---|---|---|
| $t_1$ Deletion of file1 complete | $k_1$ | [$t_1$ - $t_0$, file1] |
| $t_2$ Took 0.61 seconds to deallocate network … | $k_2$ | [$t_2$ - $t_1$, 0.61] |
| $t_3$ VM Stopped (Lifecycle Event) | $k_3$ | [$t_3$ - $t_2$] |
| … | … | … |



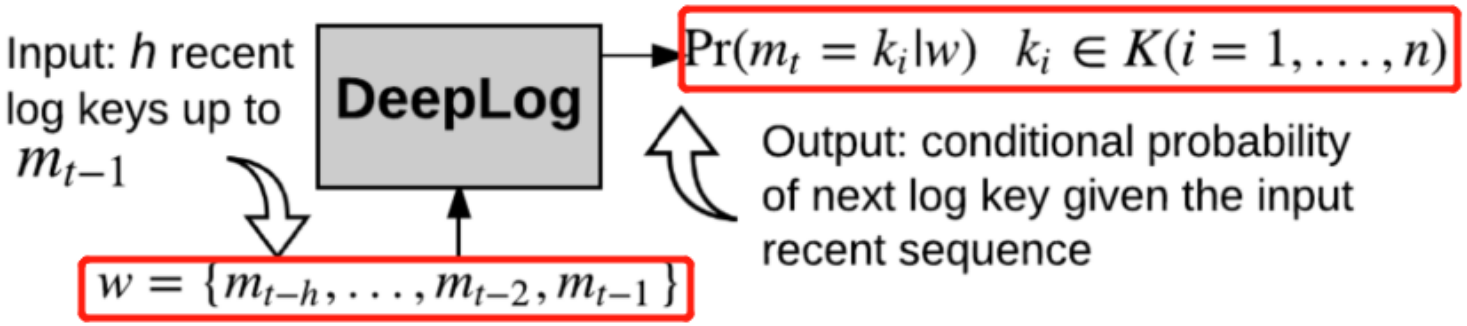
# 算法原理 DeepLog

- 训练阶段
  - 训练数据来自正常日志条目，每个日志条目被解析为“log key + 参数值向量”
  - 使用由日志解析转化而来的序列数据训练Log Key异常检测模型



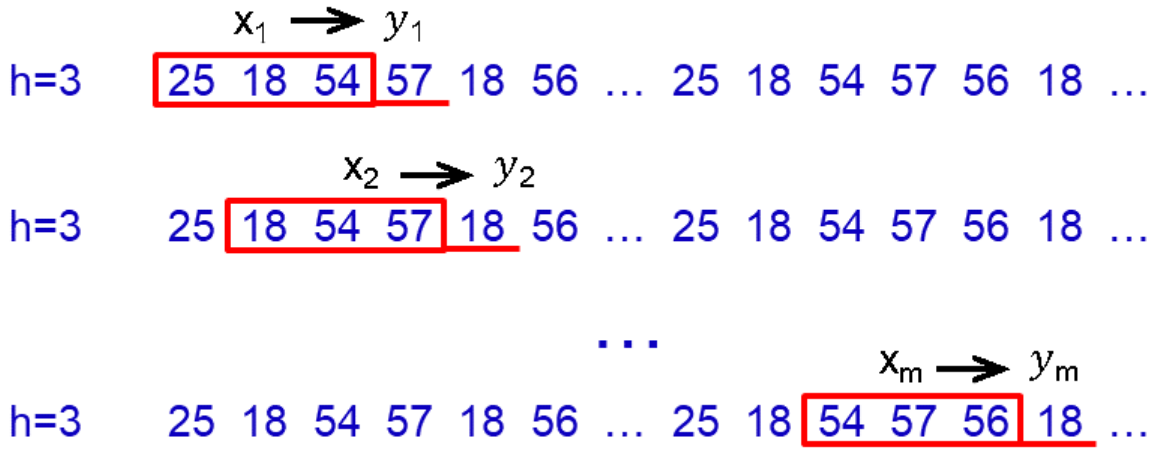
# 算法原理 DeepLog

- 训练阶段
  - 用$m_i$表示log key序列中位置i处的key值，即其可能为n个keys中的某一个，并且，它的出现强烈依赖于在$m_i$之前出现的最近的keys
  - 将log key序列中的异常检测建模为一个多分类问题，将每个不同的log key定义为一个特定的类
  - DeepLog模型是这样一个多分类器：它的输入是最近的历史log key，输出为$K$的n个log key的概率分布，表示历史序列的下一个log key是$k_i \in K$的概率



# 算法原理 DeepLog

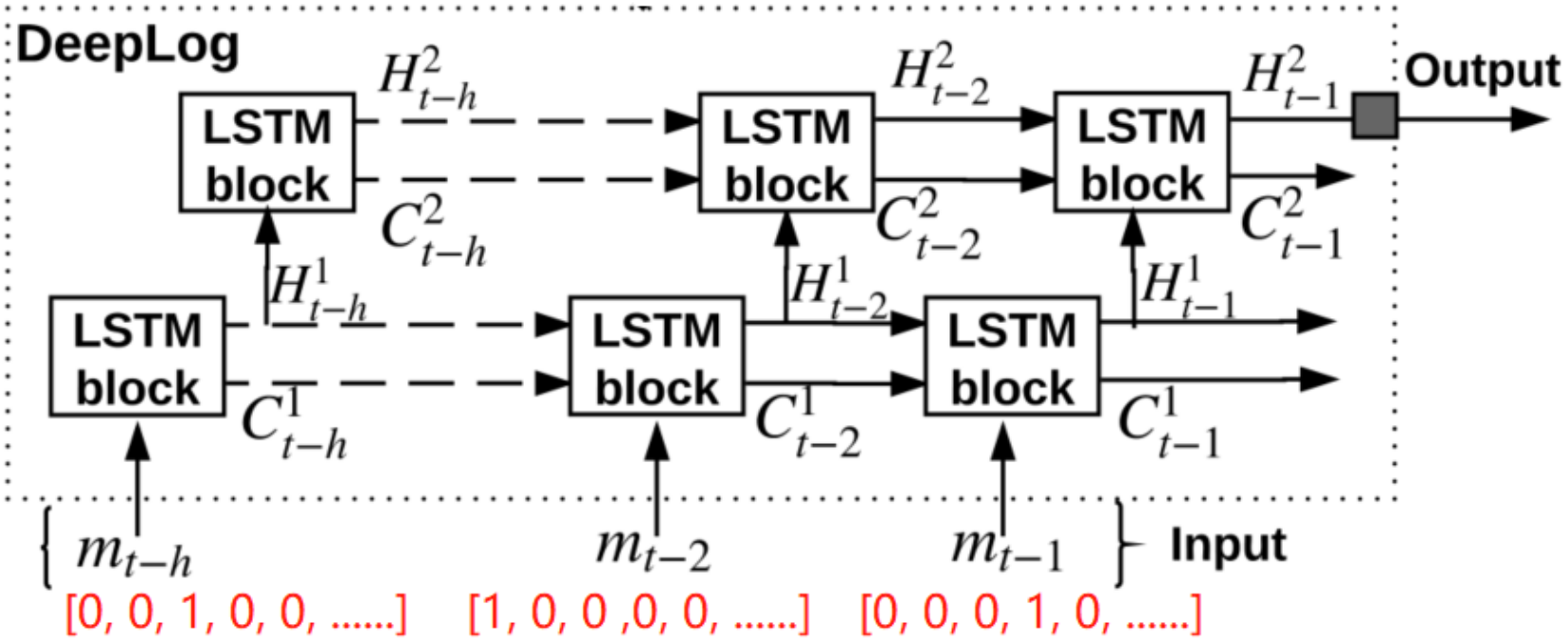
- 训练阶段
  - 假设模型的输入$w$的宽度为3
  - 即在log key序列上生成训练数据时使用长度为3的窗口进行步长为1的滑动操作




$$\mathbf{X}_{train} = \begin{bmatrix} 25 & 18 & 54 \\ 18 & 54 & 57 \\ \cdots & & \\ 54 & 57 & 56 \end{bmatrix} \qquad \mathbf{Y}_{train} = \begin{bmatrix} 57 \\ 18 \\ \cdots \\ 18 \end{bmatrix}$$

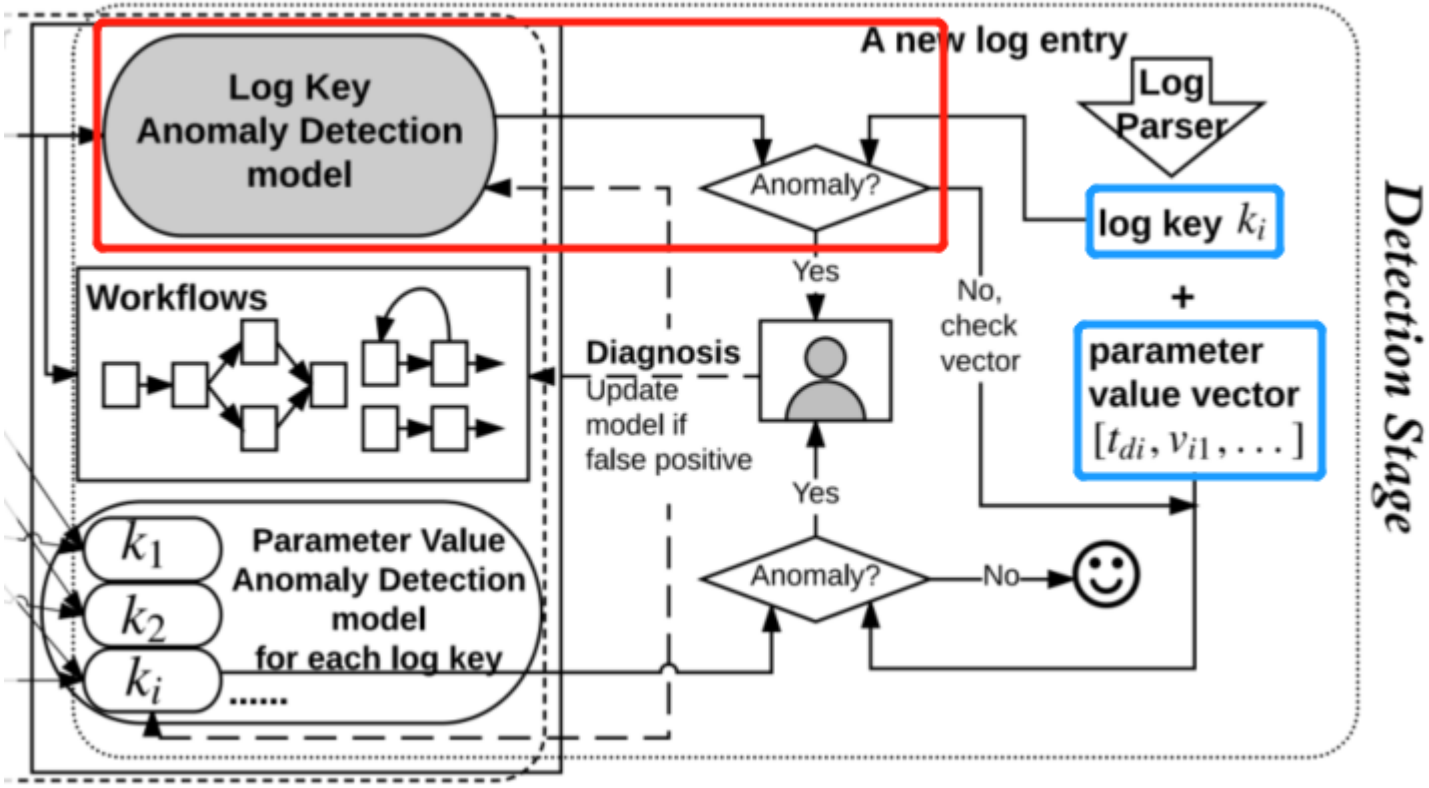
# 算法原理 DeepLog

- 训练阶段
  - DeepLog训练堆叠LSTM神经网络构建检测模型
  - 输入为$m_i$的one-hot编码, DeepLog采用两层LSTM，之后接全连接网络，经过softmax函数处理后，输出概率分布$\Pr[m_t|w] = \{k_1:p_1, k_2:p_2, \ldots, k_n:p_n\}$,它表示每个log key作为给定历史序列的下一个log key出现的概率



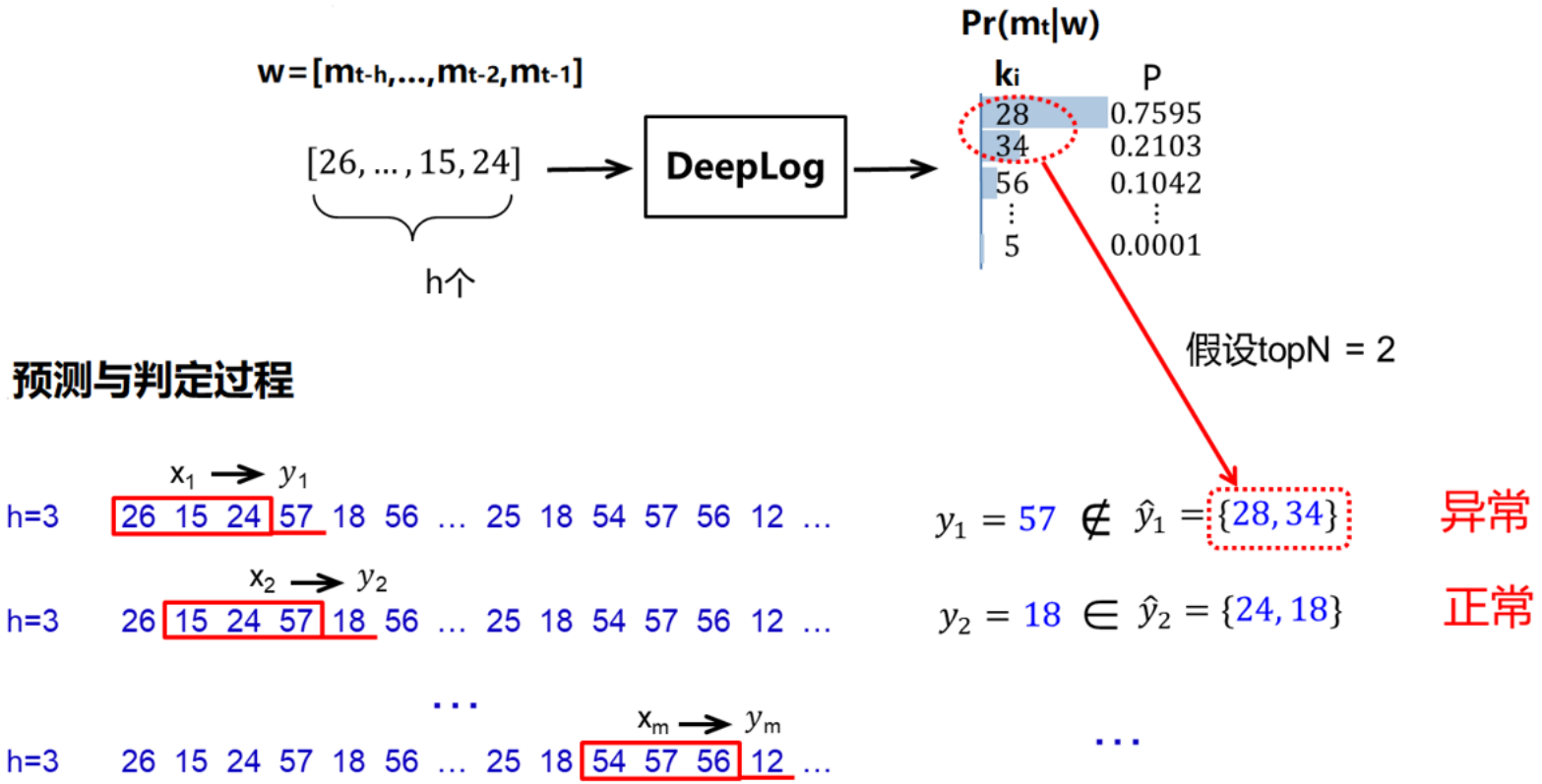
# 算法原理 DeepLog

- 检测阶段
  - 新到达的日志条目同样被解析为log key和参数值向量
  - 使用log key异常检测模型来检查传入的log key序列是否正常



# 算法原理 DeepLog

- 检测阶段
  - 为检测$m_t$是否异常，发送$w = \{m_{t-h}, \dots, m_{t-2}, m_{t-1}\}$作为已训练模型的输入
  - 输出概率分布$\Pr[m_t|w] = \{k_1 : p_1, k_2 : p_2, \dots, k_n : p_n\}$,它表示每个log key作为给定历史序列的下一个log key出现的概率，按照Top N原则判定异常/正常



- 数据集
  - HDFS日志数据集
    - 在超过200个Amazon的EC2节点上运行基于Hadoop的map-reduce作业生成的，并由Hadoop领域专家进行标记。
  - OpenStack日志数据集
    - 在CloudLab上部署了一个OpenStack实验系统(版本为Mitaka)，运行脚本持续执行虚拟机相关任务后收集日志数据。

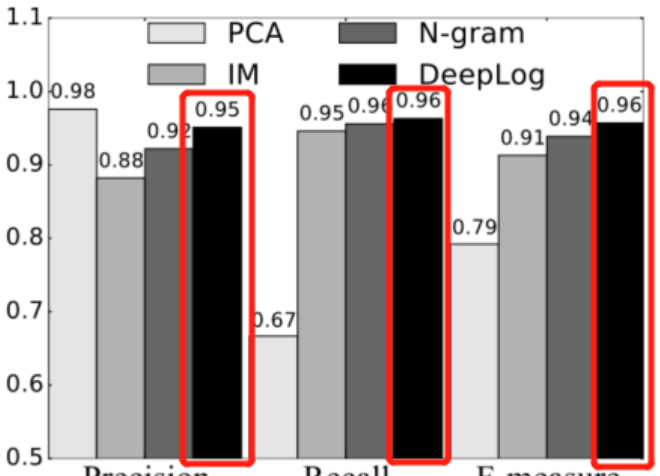
| Log data set | Number of sessions | | $n$: Number of log keys |
|---|---|---|---|
| | Training data (if needed) | Test data | |
| HDFS | 4,855 normal;<br>1,638 abnormal | 553,366 normal;<br>15,200 abnormal | 29 |
| OpenStack | 831 normal;<br>50 abnormal | 5,990 normal;<br>453 abnormal | 40 |

# 算法原理 实验结果

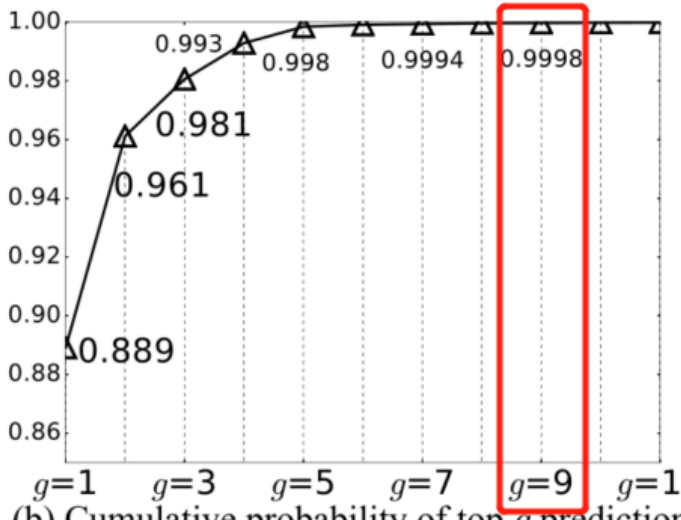
- 在HDFS数据集上不同方法的指标数据
  - PCA实现了最少的假阳性，但代价是更多的假阴性

| | PCA | IM | TFIDF | N-gram | DeepLog |
|---|---|---|---|---|---|
| false positive (FP) | 277 | 2122 | 95833 | 1360 | 833 |
| false negative (FN) | 5400 | 1217 | 1256 | 739 | 619 |

  - Precision、Recall、F1指标情况：DeepLog整体性能最好，F-measure为96%。当历史长度为1时，N-gram也能获得良好的性能。但是，随着历史窗口变长，其性能急剧下降。相比之下，基于lstm的方法更加稳定



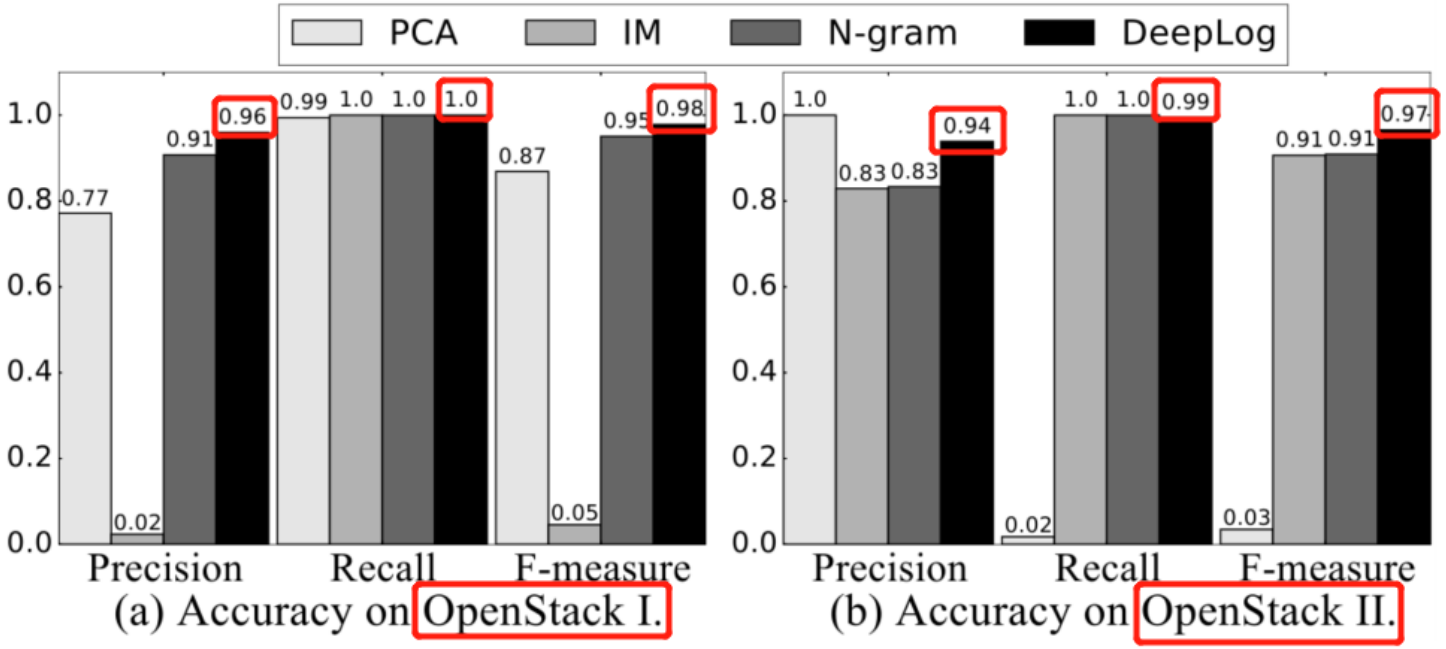
(a) Accuracy on HDFS.



(b) Cumulative probability of top $g$ predictions.

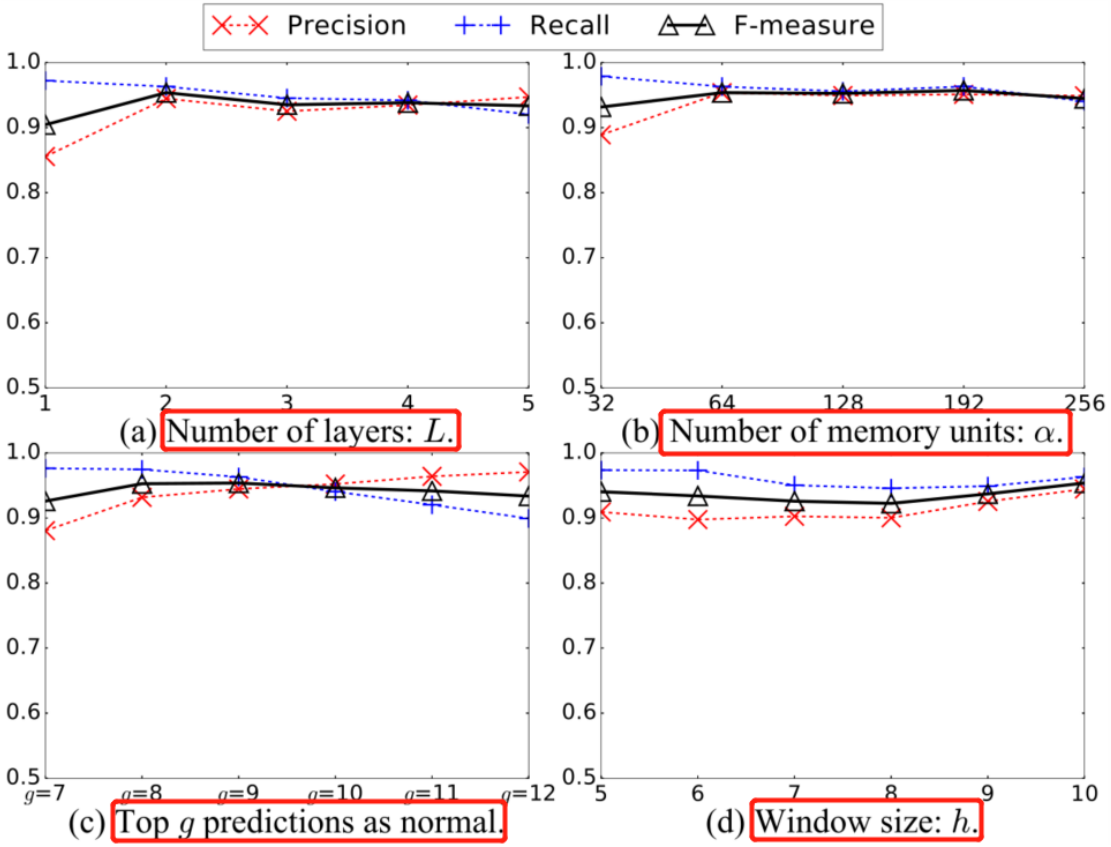
# 算法原理 实验结果

- 在OpenStack数据集上不同方法的指标数据
  - OpenStack I日志是随机生成的,某些log key可能出现多少次是不确定的
  - 生成了OpenStack II具有确定性模式的数据集，如(Create Delete)+
  - DeepLog在两个OpenStack日志上都展示了出色的性能，F-measure分别为98%和97%



# 算法原理 实验结果

- 研究了DeepLog中各种参数的性能影响，包括g、h、L和α
  - DeepLog的性能相对于不同的值是相当稳定的，也就是说，它对这些参数值的任何一个或组合的调整都不是非常敏感。这使得其在实践中更易于部署和使用



# 算法原理 优劣分析

- 优势：
  - 实现了利用非结构化的日志数据进行数据挖掘分析和异常检测
  - 有效地解决了从海量日志数据中进行在线异常检测的问题
  - 不需要依赖先验知识，对未标注数据集具有兼容性
  - 仅使用“少部分”正常日志条目组成的数据集进行训练

- 局限性：
  - DeepLog输入数据采用one-hot编码，无法学习出两个模板之间的语义相似度

# 算法原理

# 算法原理 LogAnomaly

| | |
|---|---|
| T | 检测日志异常 |
| I | 系统/应用运行产生的日志数据 |
| P | 1.解析日志并抽取log keys (i.e., template)<br>2. 将log key序列转换为向量序列，计算对应计数向量<br>3.根据滑窗长度、步长等参数获取训练样本<br>4.训练LSTM模型<br>5.对待检测日志数据进行预处理、预测和判定 |
| O | 待检测日志数据是/否异常 |

| | |
|---|---|
| P | log key索引丢失日志的精确语义信息、导致任务中断 |
| C | 可正常解析的未受攻击日志数据（篡改等） |
| D | 如何将log key转换为嵌入向量 |
| L | IJCAI (CCF-A会议) |

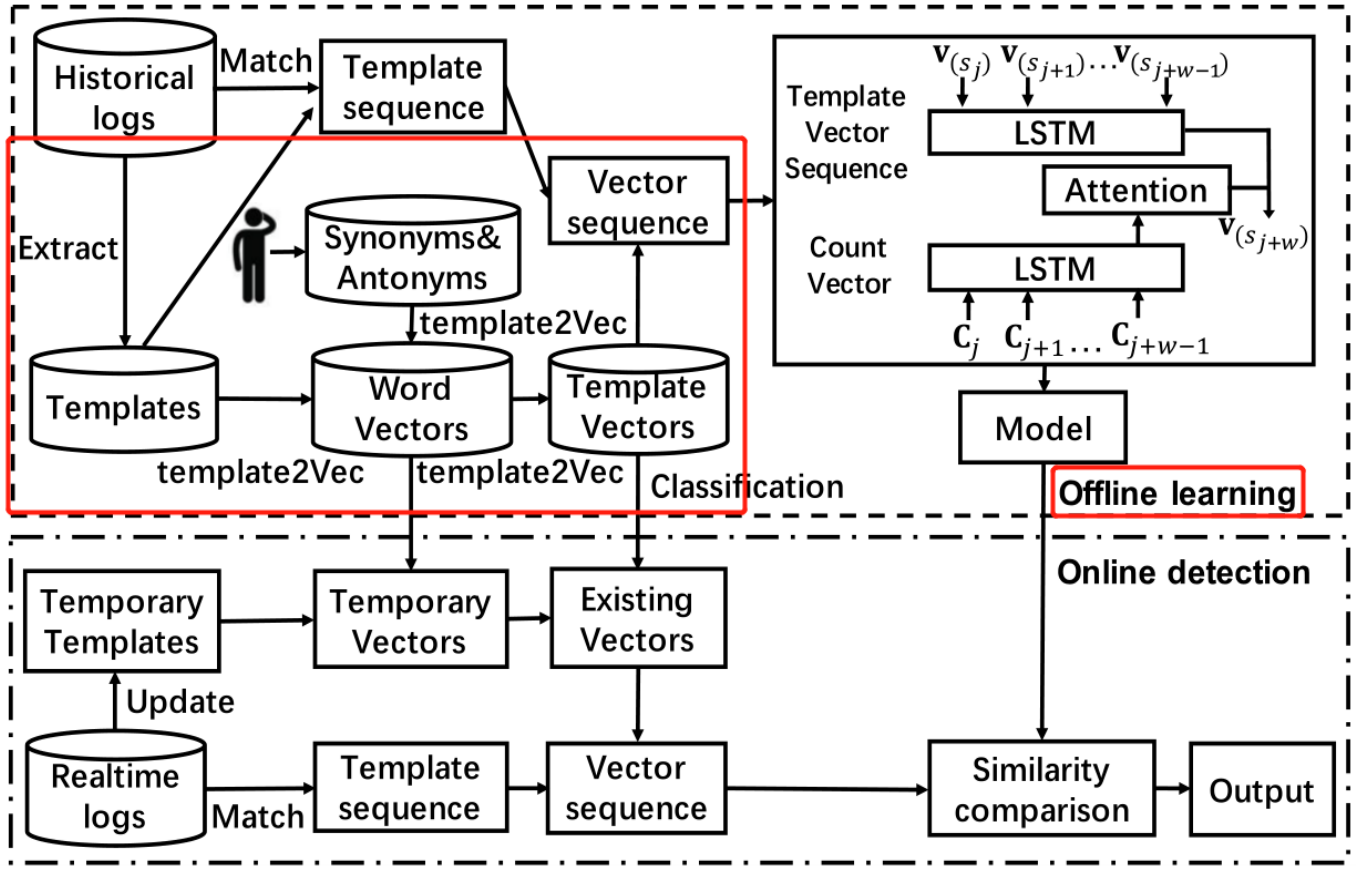
# 算法原理 LogAnomaly

- 总体框架
  - 两个主要部分：

    离线训练部分、在线检测部分

  - 离线训练部分：

    Template2Vec

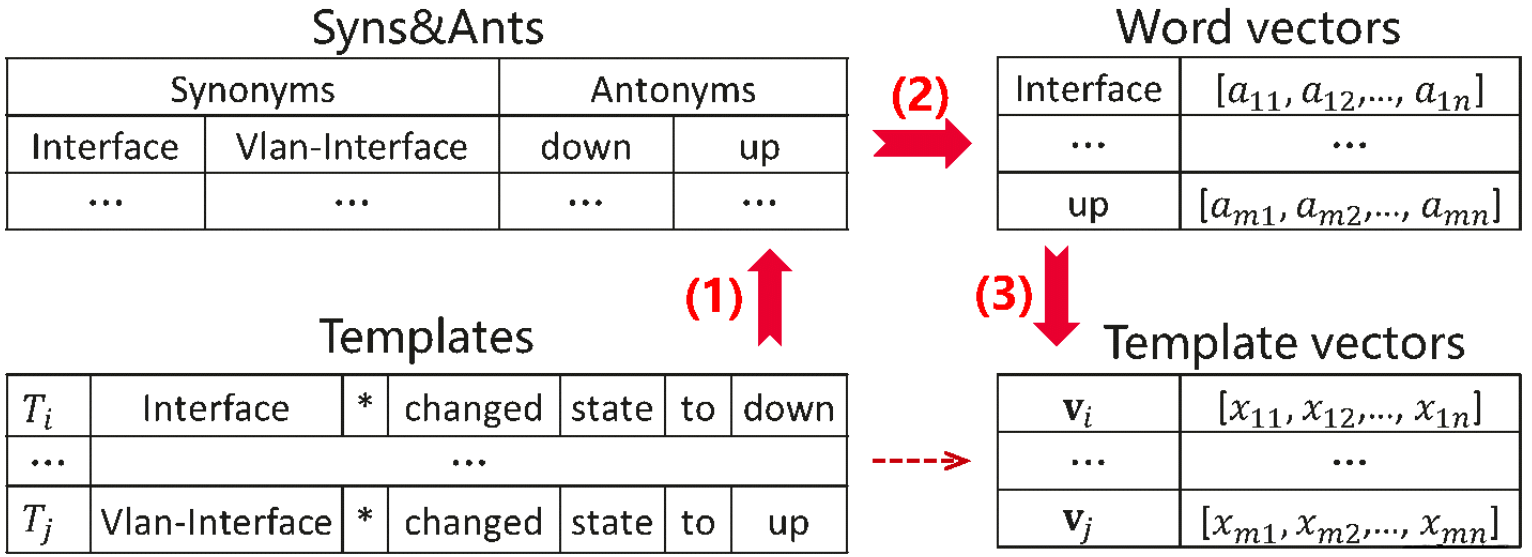


# 算法原理 LogAnomaLy

- Template2Vec
  - 为了表征和学习log key的语义信息和具体业务含义
  - 受Word2Vec的启发，设计Template2Vec将log key编码为语义向量，替代DeepLog中的one-hot编码
  - 突出语义与业务含义不同日志之间差异性，捕获语义与含义相似日志之间相似性
  - 而对于新出现的log key，没有在原有的log key空间中出现，则将其即时转换为语义最接近的已有log key向量，保证检测任务的连续性，避免模型重训练

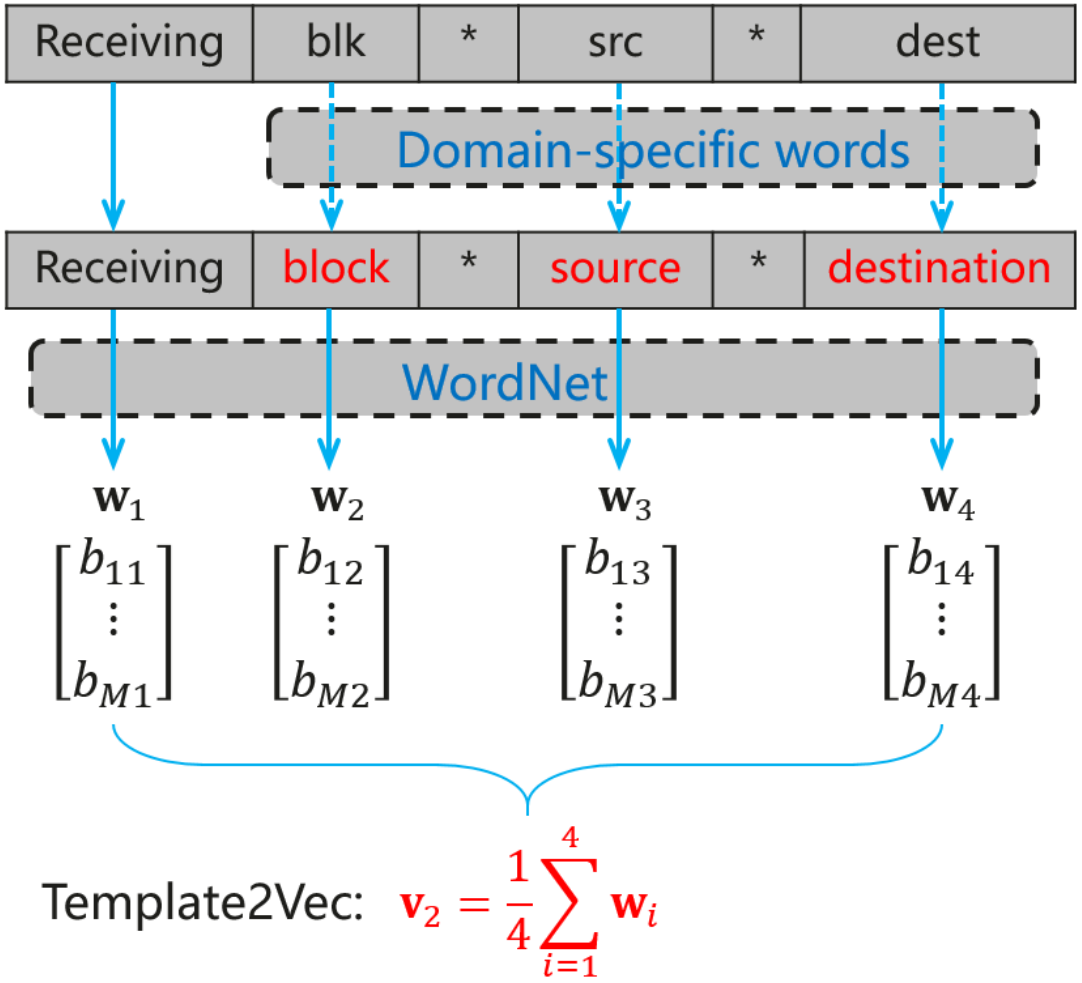
# 算法原理 LogAnomaly

- Template2Vec
  - 在WordNet中对模板内容中的自然语言单词进行同义词和反义词搜索(如图中的down和up)，之后，对具有业务知识的词汇识别同义词和反义词(如图中的Interface和Vlan-Interface)，并将其转化为正常的自然语言词汇
  - 应用dLCE生成模板中单词的词向量，如图中的Word vectors
  - 模板向量是模板中单词的词向量的加权平均值。如图中的Templates vectors



# 算法原理 LogAnomaly

- Template2Vec
  - Template2Vec结合了运维人员的专业领域知识和自然语言处理中的dLCE模型，以便准确生成模板向量。例如对模板Receiving blk src dest的Template2Vec求解过程如右图。
  - 借助Template2Vec将模板序列转换为语义向量序列，之后的LSTM模型训练与日志异常检测过程与DeepLog类似。

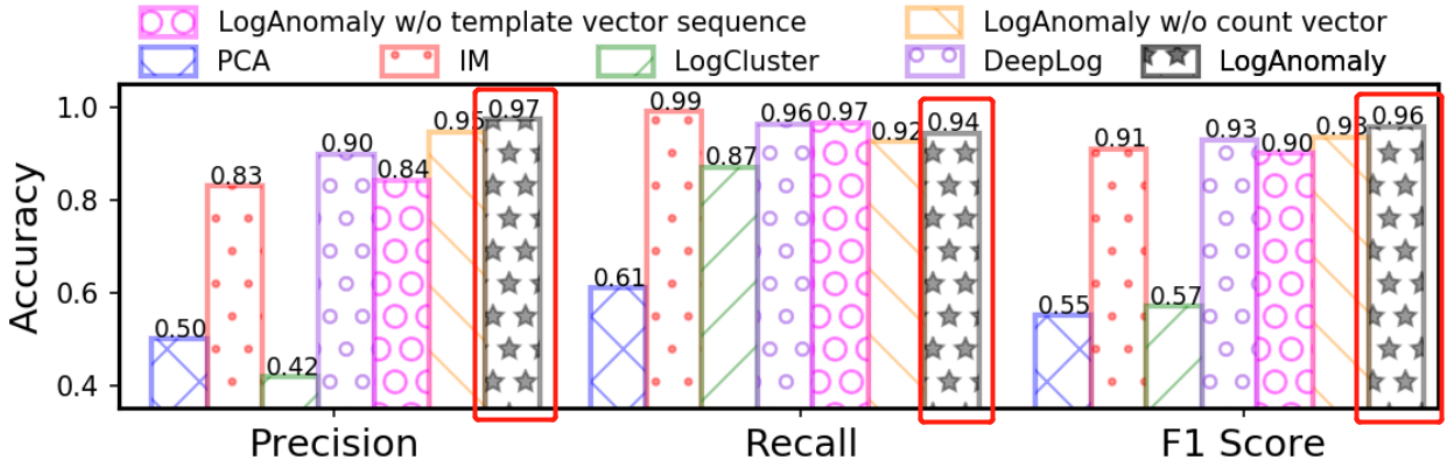


- 数据集
  - BGL数据集
    - 包含4747963条日志。每个BGL日志被手动标记为异常或正常，348460个日志被标记为异常。BGL数据集由Blue Gene/L超级计算机生成，部署在劳伦斯·利弗莫尔国家实验室（LLNL）
  - HDFS数据集

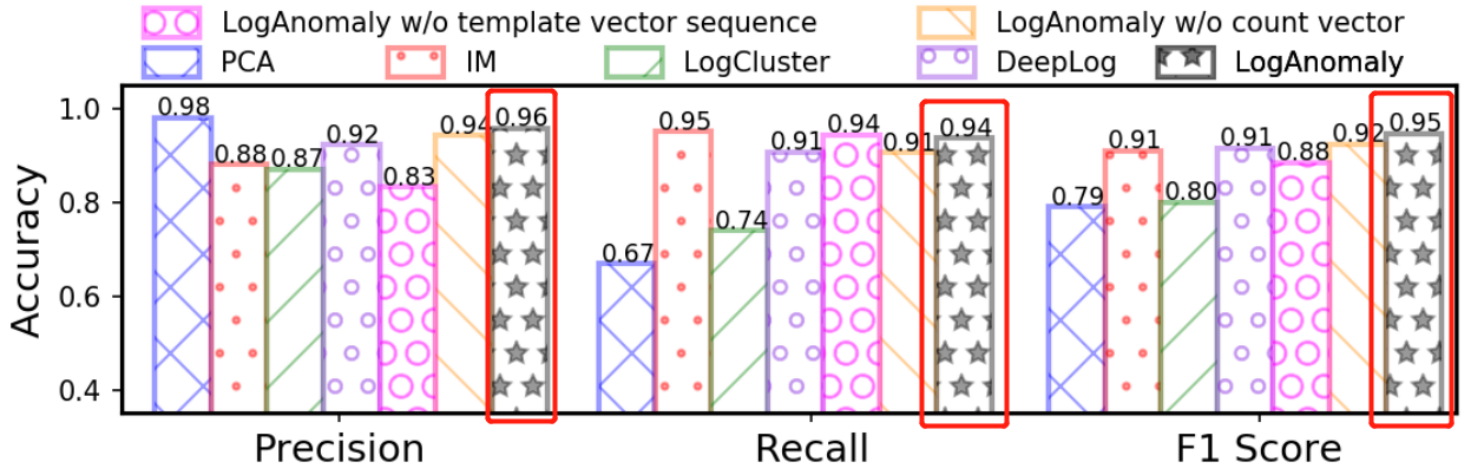
| Datasets | Duration | # of logs | # of anomalies |
|---|---|---|---|
| HDFS | 38.7 hours | 11,175,629 | 16,838(blocks) |
| BGL | 7 months | 4,747,963 | 348,460(logs) |

# 算法原理 实验结果

- BGL数据集实验结果




- HDFS数据集实验结果



- 优势：
  - 简单而有效的提取日志模板中隐藏的语义信息，解决潜在的误报问题
  - 可以避免在检测阶段出现新的日志模板时而引发的任务中断等

- 局限性：
  - 不能在训练阶段或检测阶段处理日志中词汇表（word vectors）外的新词汇

# 应用总结

# 应用总结

- 应用
  - 根据日志数据判定当前系统运行情况是否存在异常
  - 给出异常点对应的关键日志，精准定位问题
  - 确定异常对应的日志上下文，辅助运维/安全人员分析异常、研判根因、排查故障

- 拓展方向
  - 解决现有框架的问题，如OOV(表外词汇)的处理与嵌入等
  - 利用因果等逻辑关系的多模态数据源(log key、参数、统计数据等)融合表示
  - 系统运行故障和异常的预测，进行提前预警
  - 从现成的审计日志构建端到端攻击故事情节，进行安全事件的攻击策略的抽象建模和分析取证，如2021年USINIX公布的ATLAS(基于序列的攻击调查学习方法)

# 参考文献

[1] Du, Min, et al. “DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning.” Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1285–1298.

[2] Meng, Weibin, et al. “LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs.” Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, 2019, pp. 4739–4745.

[3] https://segmentfault.com/a/1190000039060586

[4]https://blog.csdn.net/qq_41773806/article/details/116052589?utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7Eessearch%7Evector-12.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7Eessearch%7Evector-12.no_search_link

大成若缺，其用不弊。
大盈若冲，其用不穷。
大直若屈。大巧若拙。
大辩若讷。静胜躁，寒
胜热。清静为天下正。

谢谢！