文章编号:1007-5321(2020)01-0068-06 **DOI**:10.13190/j.jbupt.2019-033

基于归一化特征判别的日志模板挖掘算法

双 锴[1,2], 李怡雯[1], 吕志恒[1], 韩 静[3], 刘建伟[3]

(1. 北京邮电大学 网络与交换技术国家重点实验室, 北京 100876;
2. 通信网信息传输与分发技术重点实验室, 石家庄 050081; 3. 中兴通讯股份有限公司, 深圳 518057)

摘要: 针对传统日志模板挖掘时需要日志聚类数目作为先验信息的问题,提出了一种基于归一化特征判别的日志模板挖掘算法. 首先,对日志数据进行压缩,以提高后续处理效率;其次,进行日志聚类过程,使用归一化的日志统计特征判断聚类是否满足要求,若满足,则聚类成功;若不满足,则采用二分搜索的方式调整日志聚类的数目,重新进行聚类;最后,从聚类结果中提取日志模板,设计了一种衡量模板挖掘效果的评价指标. 在真实数据集上的实验结果表明,算法的模板挖掘匹配度优于基准方法,并且具有良好的泛化性能.

关 键 词: 模板挖掘; 日志分析; 文本聚类; 归一化特征

中图分类号: TP391 **文献标志码**: A

Log Template Extraction Algorithm Based on Normalized Feature Discrimination

SHUANG Kai[1,2], LI Yi-wen[1], LÜ Zhi-heng[1], HAN Jing[3], LIU Jian-wei[3]

(1. State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China;
2. Science and Technology of Information Transmission and Dissemination in Communication Networks Laboratory, Shijiazhuang 050081, China;
3. ZTE Corporation, Shenzhen 518057, China)

Abstract: A log template extraction algorithm based on normalized feature discrimination is proposed, aiming at the problem that the number of clusters needs to be provided as a priori information in traditional log template extraction. First, log data is initially compressed to reduce data redundancy. Then, a log clustering process is implemented, and the normalized feature is used to discriminate whether the clustering result meets requirement: if so, the clustering process is successfully completed; if not, the number of log clusters is adjusted by using binary search and redo clustering. Finally, the log template is extracted via clustering results. In addition, an evaluation metric that measures the effectiveness of template extraction is designed. Experiments on real data indicated that the algorithm can achieve more stable and accurate template extraction performance than the benchmark method, and it had good generalization performance.

Key words: template extraction; log analysis; text clustering; normalized feature

日志记录了系统内部操作的重要信息,如系统状态、程序崩溃等,运维人员可以通过日志数据对系统进行分析与优化,并进行异常检测. 随着大数据技术的发展,各种网络服务和分布式系统变得愈加

收稿日期: 2019-03-22
基金项目: 国家重点研发计划项目(2016QY01W0200); 上海市青年科技英才扬帆计划项目(18YF1423300); 通信网信息传输与分发技术重点实验室开放基金课题(SXX18641X024)
作者简介: 双 锴(1977—), 男, 副教授, 硕士生导师,E-mail:shuangk@bupt.edu.cn.

复杂,系统生成日志的规模越来越大. 如何从海量日志中挖掘出有效信息,快速准确地提取日志模板,成为日志处理领域亟待解决的问题.

提出一种基于归一化特征的日志模板挖掘算法,在对日志进行初步的聚类过程后,使用归一化聚类后日志的统计特征作为判别聚类成功的标准,解决了传统日志聚类方式需要日志聚类数目作为先验信息的问题,提高了算法的泛化能力. 通过对不同日志数据进行实验的结果表明,算法有效地提高了模板挖掘的准确率.

1　相关工作

由于日志集合中不同类型的日志长度不一,词汇量大,传统的基于 01 词袋模型聚类的模板挖掘方法仅仅使用日志单词本身的基本特征进行关键词判别,忽略了日志集合的整体特征,因此在实际应用时效果不佳[1];基于聚类后提取模板的方式是当前主流的日志模板挖掘方法:Vaarandi 等[2]提出的简单日志聚类法主要采用基于密度的聚类算法,通过人工设定阈值的方式对聚类大小进行控制,阈值越小,每个聚类中的文本相似程度越高;Fu 等[3]使用以单词为最小单位的编辑距离对日志进行聚类,并提取每个聚类中的频繁词序列或公有日志文本片段作为日志模板;迭代分区日志挖掘法(IPLoM, iterative partitioning log mining)[4-5]根据日志的长度、频繁项、词位关系等逐步将日志集合划分成多个聚类,并根据聚类中的频繁项从每个聚类中构造日志模板,尽管该方法相比于传统方法能够取得更好的效果,但由于划分聚类的标准较为单一且固定,无法在较为复杂的场景下取得满意的效果;Li 等[6]在预先给出日志聚类数目的情况下,通过建立基于词对关系的聚类转移增益计算构造最终聚类,聚类效果较好;Nandi 等[7]在聚类构建过程中引入日志时间序列关系辅助判断,提高了模板挖掘的准确率;日志分析系统 LogCluster 中的日志聚类算法使用聚合层次聚类,在得到聚类结果后,选择每个聚类中与其他序列距离最小的序列作为日志模板[8].

以上方法通过聚类的方式结合了日志集合的整体信息和词项的单位信息,然而大多需要人工干预的过程,与自动化进行模板挖掘的目标不符. 针对上述问题,提出一种基于归一化特征的日志模板挖掘算法,通过对日志进行聚类完成模板挖掘过程,并使用归一化特征判断聚类结果,不需人工干预即可自动完成聚类,可以在先验信息较少时取得更好的模板挖掘效果.

2　日志模板挖掘算法

2.1　问题描述

针对日志模板挖掘问题,建立基本日志模板挖掘问题模型 M. 定义:

$$M=(D,T,V),D=\{d_i\mid 1\leqslant i\leqslant N_d\},T=\{t_j\mid 1\leqslant j\leqslant N_t\},$$
$$V=\{v_{ij}\mid 1\leqslant i\leqslant N_d,1\leqslant j\leqslant N_t\},V_{ij}=f(d_i,t_j)$$

满足以下条件约束:

$$\forall v_{ij}\in V,v_{ij}=0\vee v_{ij}=1$$
$$\forall d_i\in D,\exists j(v_{ij}=1\wedge t_j\in T)$$
$$\forall d_i\in D,\neg\ \exists j\exists k(v_{ij}=1\wedge v_{ik}=1)$$

其中:D 为原始日志集合;N_d 为原始日志数量;T 为日志模板集合;N_t 为模板数量;V 为日志和日志模板的对应关系,用一个二元组 $V_{ij}=f(d_i,t_j)$ 表示;v_{ij} 表示日志 d_i 是否可以与模板 t_j 匹配,$v_{ij}=1$ 表示是,$v_{ij}=0$ 表示否. 模型的 3 个约束表示对于每条日志和每条模板只存在匹配和不匹配 2 种情况,匹配且仅匹配一条模板.

对于原始日志集 D,模型 M 将 D 中任一元素匹配至模板集合 T 中的仅一条元素,该过程表示原始日志和日志模板的生成过程和多对一的匹配关系.

2.2　解决思路

当前的主流方法是基于聚类的日志模板挖掘方法. LogSig 算法在预先给出日志聚类数目的情况下,通过建立基于词对关系的聚类转移增益计算构造最终聚类. 其中,预先给出日志聚类数目需要人类的先验知识,而在实际情况下,对于未知的日志数据数目难以取得,严重限制了算法的应用场景. 如果能够设置某种条件对聚类数目做出预先估计,将大大增加算法的应用价值.

对于不同的日志类型,挖掘出的模板类别数,即算法需要的聚类数目是相差较大的[9],若对其设置相同的聚类数目则不能很好地适应,但对于算法而言,需要有一套统一的判别标准. 使用的基于归一化特征进行判别的方式则能够较好地解决聚类数目差别较大的问题.

特征归一化处理是数据挖掘的基础工作之一. 不同的评价指标往往有不同的量纲和量纲单位,这种情况会很大程度上影响数据分析的效果. 为消除评价指标因量纲不同产生的影响,需要对数据进行归一化处理,以解决数据之间的可比性. 原始数据

在经过了归一化处理后,各指标处于同一数量级,适合进行相互间的对比评价.

在模板挖掘问题中,算法所需的聚类数目并不适合进行归一化,因为不同日志系统需要的聚类数目在归一化操作后依然差距较大,因此,为使不同日志系统产生的结果具有可比性,需要采用更合适的归一化特征进行判别. 在实际的日志系统中可以发现,不同的日志系统,其单条日志中参数部分所占比例是在一定范围内的,通过基于频繁项的参数判别方式构造参数比例相关的归一化特征,可以作为判别依据,从而搜索出合适的聚类数目[10].

所提出的日志模板挖掘流程如图1所示. 在当前日志的总类别数未知时,可通过在搜索过程中设置统计归一化特征进行判别的方式搜索出合适的聚类数目,并进行模板挖掘.

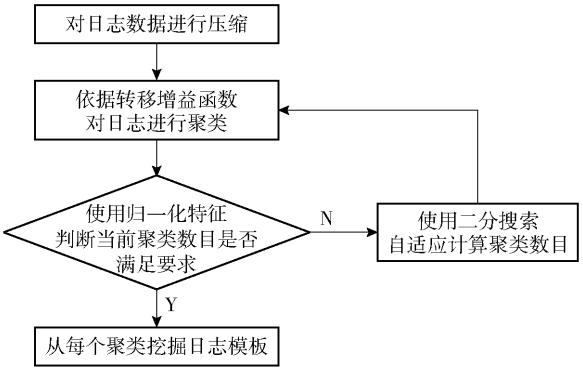

图1 基于归一化特征的日志模板挖掘算法流程

2.3 算法描述

基于归一化特征的日志模板挖掘算法实现过程如下.

算法1 基于归一化特征的日志模板挖掘算法

输入:日志数据集 Data, 最大类别数 k, 关键词阈值 λ_1, 关键词比例阈值 λ_2, 日志比例阈值 λ_3

输出:日志模板集 T

1 压缩日志集 Data 生成新日志集 D
2 do 二分搜索:
3 当前搜索值 mid
4 随机初始化日志聚类 C, C 中包含 mid 个类别
5 设置 C'为空集
6 while $C! = C'$ do
7 $C = C'$
8 遍历搜索 D 中每条日志的 $\Delta_{i=j}^{x}\Phi(D)$ 最高值,构成新聚类 C
9 end while
10 根据 λ_1, 统计生成 C 中各类别关键词
11 根据 λ_2, 统计生成 D 中聚类成功日志所占比例 r
12 根据 λ_3 和 r, 更新二分搜索区间值
13 while 二分搜索区间 >1
14 根据日志聚类 C, 提取日志模板集 T
15 return T

算法实现过程的详细步骤如下.

1) 对待处理的日志进行数据压缩

日志输出系统中存在着大量形式固定、易于检出的参数,如 uuid、IP 地址等. 首先使用正则表达式去除日志中已知形式的参数;之后采用统计词频的方式构筑关键词词典. 在对单词在全体日志中出现的频数进行统计后,统一设定一个阈值,频数大于该阈值的单词构成初步关键词表. 为了实现对关键词更精确的提取,对初步关键词中的单词判断其是否属于以下3个类别:

① 符合英文基本语义的单词,如:apple, book;

② 计算机领域的专业词汇,如:dns, dhcp;

③ 对于特定的日志集,常出现特定表示,例如:BUPT.

对于符合英文基本语义的单词选择,使用 PyEnchant 作为筛选器辅助筛选.

若单词不属于以上3种类别中的任何一个,则剔除出关键词表.

利用关键词词典,使用01词袋模型对单条日志构筑一个 n 维的01向量,向量相同的日志只保留一条,实现对原始数据进行压缩. 由此方式可以使得进入聚类算法的日志数量大大降低,减少由数据冗余带来的额外系统资源消耗,有效提高算法效率.

2) 对日志进行聚类

算法首先依据二分搜索初始值给出一个聚类数目的初始值 k. 先将日志随机划分成数量均匀的 k 类,在二分搜索每一次的判别过程中对每条日志进行转移增益函数计算,将其分到增益函数最大的聚类.

给定一个聚类 C,$T(C)$表示聚类 C 的日志中所有有序单词对的集合. 例如,以下是来自某应用程序的一条日志:

2017-08-23 14:46:28.579ERROR: Not exist file "config.py"

通过正则表达式,去除时间参数,得到

ERROR: Not exist file "config.py"

两两提取日志中的单词,并保留其顺序,形成有序单词对:

{ERROR:, Not}, {ERROR:, exist},
{ERROR:, file}, {ERROR:, "config. py"},
{Not, exist}, {Not, file}, {Not,"config. py"},
{exist, file}, {exist,"config. py"},
{file,"config. py"}

经过转换,有序单词对集合不仅可以保留日志中单词的顺序,并且与直接对日志进行操作相比,可以降低后续计算的时间复杂度,进一步提高算法效率.

对于任意有序单词对 $t \in T(C)$, $N(t,C)$ 表示 C 中包含 t 的日志的数量,$s(t,C) = N(t,C)/|C|$ 表示 C 中包含 t 的日志所占比例.

定义 $\phi(C)$ 为聚类 C 的聚类纯净值,

$$\phi(C) = \sum_{t \in T(C)} N(t,C)[s(t,C)]^2 \tag{1}$$

聚类纯净值用来衡量聚类中日志的"纯净"程度. 当聚类包含的日志属于同一类的比例越高,所包含日志条目数越大,则该纯净值越大.

由此,定义日志集合 D 的整体纯净值为

$$\Phi(D) = \sum_{i=1}^{k} \phi(C_i) \tag{2}$$

其中:D 被划分为 k 个聚类,$\phi(C_i)$ 为每个聚类的纯净值,$i = 1,2,\cdots,k$.

转移增益函数的计算方式为

$$\Delta_{i \xrightarrow{x} j}\Phi(D) = [\phi(C_j + \{X\}) - \phi(C_j)] + [\phi(C_i - \{X\}) - \phi(C_i)] \tag{3}$$

其中:$\Delta_{i \xrightarrow{x} j}\Phi(D)$ 表示将 $X \in D$ 从聚类 C_i 转移到 C_j $(i,j = 1,\cdots,k, i \neq j)$ 时 $\Phi(D)$ 的变化量,$\phi(C_j + \{X\}) - \phi(C_j)$ 表示在聚类 C_j 中添加日志 X 产生的聚类纯净值变化,$\phi(C_i - \{X\}) - \phi(C_i)$ 表示从聚类 C_i 中移除日志 X 产生的聚类纯净值变化,二者相加表示当前日志发生转移之后对日志集合整体纯净值产生的变化.

反复进行 $\Delta_{i \xrightarrow{x} j}\Phi(D)$ 的计算,直到 $\Phi(D)$ 达到极大值,认为当所有日志当前所属的类都是转移增益函数最大的类时,聚类达到稳定.

3) 使用归一化特征判断当前聚类数目是否满足要求

对当前聚类结果进行判断. 若聚类数目可以满足实际需求,则聚类结束,进入下一步骤;若聚类数目过多或过少,则需进一步调整 k 值,重新进行聚类过程.

由于不同日志集合的 k 值不同,甚至差别非常大的,在设定了参数提取成功的要求后,满足要求的日志即认为模板挖掘成功的日志数量是一定的,因此提出基于一个归一化的特征进行判断,通过设定满足要求的日志在总体中所占得比例 r,在这个比例和 k 值有正相关关系时,使用多次二分搜索的方式得出 k 值.

假设日志集合中任意一条日志为 d,每条日志中的一个单词为 w,设定日志模板挖掘成功的条件如下.

① 对于模板挖掘成功的日志,其参数数量应当小于 x,参数所占比例 $p < \lambda_1$,该比例能够根据不同的日志系统进行调整. 其中对于参数的判断条件为:某单词在聚类中的日志样本里出现的比例 $q < \lambda_2$,认为该单词是参数.

$$p_i = \frac{\sum_{j}^{|d_i|}(q_j < \lambda_2)}{|d_i|} \tag{4}$$

$$q_j = \frac{\sum_{i}^{N_d} E(w_j,d_i)}{N_d} \tag{5}$$

其中:$E(w_j,d_i)$ 表示单词 w_j 是否出现在日志 d_i 中,若是,值为 1;否则值为 0.

② 模板挖掘成功的日志在全体日志集合中所占比例 $r > \lambda_3$.

$$r = \frac{\sum_{i}^{N_d}(p_i < \lambda_1)}{N_d} \tag{6}$$

设定以上条件后,k 值和 r 值就具有了一定的正相关性,在 OpenStack 日志数据集的实验结果如图 2 所示. 因此,利用二分搜索的方式可以自动地定位到 k 值. 为获得较为稳定的 k 值,可以进行多次二分搜索,并对得到的结果取平均值作为最终的

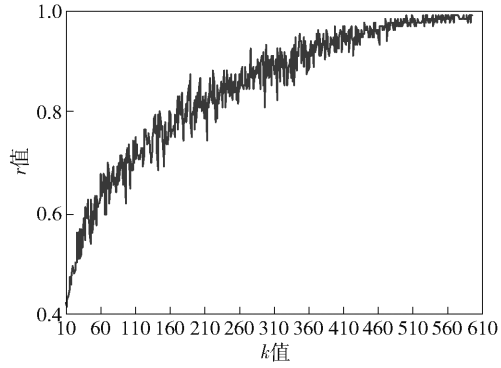

图 2　OpenStack 日志数据集上 k 值与 r 的相关性

聚类数目. 由于 λ_1、λ_2、λ_3 是归一化到[0,1]区间内且有物理意义的特征,所以受不同类型日志文本特征的影响较小. 这样,对于不同类型的日志集合,都能够得到相应的聚类数目 k,并根据该 k 值自动进行日志聚类以及后续的模板挖掘过程.

2.4 算法分析

现有的模板挖掘提取算法大多需要人工干预的过程,所提出的无监督、可以自更新的模板挖掘算法解决了传统的基于聚类的模板挖掘算法中聚类数目 k 需要人工预先设定的问题,对不同类型的日志数据集可搜索出合适的聚类数目;此外,考虑到日志数量可能非常庞大,算法对日志数据先进行压缩再处理,使得进入算法的日志数量大大降低,在处理大规模日志数据时仍能保持较高的效率.

3 评价标准

现有日志集合 Data,经过数据压缩后得到包含压缩去重后的 n 条日志的集合 D,设有 G_i 条日志对应去重后的日志 D_i.

标准模板(此处将人工提取的日志模板称为标准模板)拥有一个模板集合 T,包含 m 个日志模板,即 T 将 D 划分成了 m 类,类别 n 中包含了 K_i 条日志,且有 $\sum_i^m K_i = n$. 其中对于类别 n,其模板对应为 T_i.

经过模板挖掘算法后得到一个模板集合 T',其中包含 m'个日志模板,D 中任意一条日志对应且仅对应 T'中的一个模板.

现提出一个衡量日志模板匹配度的评价标准,将模板挖掘算法产生的 T'在 D 上与参考标准 T 的差异度定义为

$$\mathrm{NotMatch}(T,T') = \frac{\sum_i^n \{[\max(|F(D_i)|,|F'(D_i)|) - |\mathrm{WLCS}(F(D_i),F'(D_i))|]\}\max(\log_{10}G_i,1)}{\sum_i^n |F(D_i)|\max(\log_{10}G_i,1)} \tag{7}$$

匹配度即为

$$\mathrm{Match}(T,T') = 1 - \mathrm{NotMatch}(T,T') \tag{8}$$

其中:$F(D_i)$ 表示日志 D_i 在 T 中对应的模板,$F'(D_i)$表示日志 D_i 在 T'中对应的模板;$|F(D_i)|$表示模板 $F(D_i)$ 中包含的单词个数;$\mathrm{WLCS}(d,d')$ 表示以单词为最小单位进行比较的两条日志的最长公共子(单词)序列.

对于单条压缩后的日志,使用基于最长公共子序列(WLCS, word longest common subsequence)的方法判断日志与真实模板的相似度,经取补变换后获得差异度. 单条压缩后日志产生的差异度与其在源数据集中数量的对数相乘,并加权求和,可以解决源数据分布不均的问题;最终结果除以总差异度从而归一化到[0,1]区间,并百分化,使得评判标准能够很好地衡量日志模板的匹配程度.

4 实验

4.1 实验数据

实验在2个数据集上进行. 其中 OpenStack 数据集采集自 OpenStack 分布式系统日志,大小约439 MB,共约700万条日志,经正则表达式去参和01词袋模型压缩后共有767条互不相同的日志,人工标注类别257类;Ubuntu 数据集采集自 Ubuntu 操作系统日志,大小约1.37 MB,共有19 228条日志,经正则表达式去参和01词袋模型压缩后包含2 706条互不相同的日志,人工标注类别8类.

4.2 实验结果与分析

分别执行所提出的基于归一化特征的模板挖掘算法与传统的基于01词袋模型的模板挖掘算法(简称基于01词袋模型)、基于编辑距离的模板挖掘算法(简称基于编辑距离)和 IPLoM 算法,比较模板挖掘的匹配度,得到的实验结果如表1所示.

表1 4种日志模板挖掘算法的匹配度对比 %

算法	OpenStack 数据集	Ubuntu 数据集
基于01词袋模型	96.50	68.20
基于编辑距离	96.88	69.96
IPLoM 算法	98.10	88.46
本文算法	99.24	95.50

从表1可以看出,基于01词袋模型和基于编辑距离的模板挖掘算法和 IPLoM 算法在操作系统日志数据集上参数识别效果差;在 OpenStack 数据集上,4种算法的匹配效果差异不大. 而本算法在2个数据集上都取得了较好的效果,在 Ubuntu 操作系统日志数据集上的效果尤为明显.

由于 OpenStack 日志数据集的数据量、参数量较多,形式较为复杂,符合正则表达式规则的部分也比较大,仅使用正则表达式去参并利用词频进行压缩后,已经能够取得一定的效果. 若进一步进行模

板挖掘，虽能够识别一定量的参数，但也会对其中部分模板元素产生误识别，从而降低总体的识别效果. Ubuntu 操作系统日志集形式较为简单，参数量少，但参数在日志中的占比相对较大，容易对部分算法产生误导.

对于复杂的 OpenStack 日志数据集来说，尽管通过初步的参数识别过程已经能够取得较好的效果，但各算法的错误依然集中在将部分关键词也视为参数，导致各算法将很多相似类别的日志识别为同一类；而这部分关键词多数为符合英文语义的单词，在算法的第 1 步中已经进行过滤. 为验证这一步骤的必要性，设计实验验证了筛选关键词对模板挖掘匹配度的影响. 实验结果如表 2 所示.

表 2　4 种日志模板挖掘算法 + 关键词筛选的匹配度对比

是否筛选关键词	OpenStack 数据集/%		Ubuntu 数据集/%	
	否	是	否	是
基于 01 词袋模型	96.50	68.20	62.70	68.20
基于编辑距离	96.88	69.96	62.70	69.96
IPLoM 算法	98.10	88.46	74.93	88.46
本算法	99.24	95.50	86.20	95.50

从表 2 可以看出，在加入关键词筛选后，各算法的表现均有提升. 对于 OpenStack 日志，由于其本身形式较复杂，基于 01 词袋模型和基于编辑距离的算法在该数据集上的效果提升不明显；IPLoM 和本算法在提供了英文单词的先验信息后效果提升明显.

通过设计和日志聚类内容相关的纯净值特征表示，本算法不仅能够保证聚类中样本之间是相似的，也可保证整个聚类能够提取出一个公共部分，有效避免了模板元素被识别为参数的情况. 基于归一化特征判别的模板挖掘算法通过设定和聚类数目无关的归一化特征，能够避免不同日志集的类别数量差异，从而具备较好的泛化能力.

5　结束语

日志模板挖掘是日志分析领域的重要部分，是进行系统异常检测的基础工作. 针对大规模日志的模板挖掘，现有方法在缺乏先验信息的情况下效果较差. 笔者提出基于归一化特征的日志模板挖掘算法，在依赖的先验信息更少时能够取得更好的效果. 实验证明算法的模板挖掘准确率高，并且相对于其他算法可以更好的适应各种类型的日志，具有较好的泛化能力.

参考文献：

[1] 董蜻灵，李芳，何婷婷. 基于 LDA 模型的文本聚类研究[C]//孙茂松，陈群秀. 中国计算语言学研究前沿进展(2009r 2011). 北京：清华大学出版社，2011：455-461.

[2] 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). Kansas City：IEEE，2003：119-126.

[3] 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. Miami：IEEE，2009：149-158.

[4] Makanju A，Zincirheywood A N N，Milios E E. Clustering event logs using iterative partitioning[C]//Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York：ACM，2009：1255-1264.

[5] Makanju A，Zincirheywood A N N，Milios E E. Investigating event log analysis with minimum apriori information[C]//Ifip/IEEE International Symposium on Integrated Network Management. Ghent：IEEE，2013：962-968.

[6] 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. New York：ACM，2011：785-794.

[7] Nandi A，Mandal A，Atreja S，et al. Anomaly detection using program control flow graph mining from execution logs[C]//ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York：ACM，2016：215-224.

[8] 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). Barcelona：IEEE，2015：1-7.

[9] Aharon M，Barash G，Cohen I，et al. One graph is worth a thousand logs：uncovering hidden structures in massive system event logs[C]//Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Berlin：Springer，2009：227-243.

[10] Vaarandi R. A breadth-first algorithm for mining frequent patterns from event logs[C]//International Conference on Intelligence in Communication Systems. Berlin：Springer，2004：293-308.