基于抖音崩溃数据的 Android 应用崩溃聚类系统实现

王　凯,刘歆宁
(大连东软信息学院 计算机与软件学院,辽宁 大连 116023)

摘　要:随着应用的用户体量的扩张,应用的崩溃量也随之出现大规模的增长。为了实现对崩溃的高效分析,需要具备一套能对崩溃进行准确聚类的系统。针对现有的 Android 崩溃聚类系统聚合不够准确的问题,提出了一种基于崩溃特征的二级聚类方法。首先,从收集的崩溃上报堆栈中,分析出异常类型、异常描述、业务代码行和系统代码行。然后,对异常描述中的干扰信息进行标准化处理得到标准化异常描述,对业务代码分析出根因行,从业务代码行和系统代码行中获取完整的崩溃代码路径。最后,结合异常类型、标准化异常描述、业务代码根因计算出的哈希值作为崩溃的一级特征聚合值,形成崩溃的一级聚合列表。使用异常类型、标准化异常描述和异常路径计算出的哈希值作为崩溃的二级特征聚合值,形成二级聚合列表。该系统已经在某大型互联网公司内部系统部署,经过抖音、今日头条等应用的大规模验证,满足了对崩溃进行准确聚类的需求。

关键词:Android 崩溃;崩溃聚类;特征分析;异常描述;异常代码路径
中图分类号:TP311　　**文献标识码**:A　　**文章编号**:1673-629X(2024)06-0053-06
doi:10.20165/j.cnki.ISSN1673-629X.2024.0076

Implementation of Android Application Crash Clustering System Applied on Douyin Crash Data

WANG Kai,LIU Xin-ning
(School of Computer and Software,Dalian Neusoft University of Information,Dalian 116023,China)

Abstract:With the expansion of active users of applications,the amount of crashes has also experienced large-scale growth. In order to achieve efficient analysis of crashes,a system that can accurately cluster crashes is required. We present a two-level crash clustering method aiming at Android App Crash based on crash features. Firstly,from the collected crash reporting stacks,the types of Java crashes,exception descriptions,business code lines,and system code lines are analyzed. Then,the interference information in the exception description is standardized to obtain the standardized exception description,business code lines are decomposed to obtain root case code line and the full exception code path is resolved from business code lines and system code lines. Finally,the crash type,standardized exception description and business code root cause are used to calculate the first hash value,which is used to cluster the crashes into the first level list. The crash type,standardized exception description and full code path are used to calculate the second hash value,which is used to cluster the crashes into the second level list. The system has been deployed in the internal APM system of one big internet company,and has been validated through large-scale testing in applications such as Douyin and Toutiao,meeting the demand for accurate clustering of crashes.

Key words:Android crash;crash clustering;feature analysis;exception description;exception code path

0　引　言

软件运行过程中出现的崩溃可能给软件的使用者带来巨大损失,同时给软件的开发者带来恶劣的口碑影响以及面对软件使用者的流失。2006 年,美国宇航局发射的火星探测器[1]软件系统发生崩溃,失去了与地面人员之间的联系,经济损失达到了 2.4 亿美元,也是人类太空探索事业上的一次惨痛教训。2021 年 11 月,抖音应用发生异常[2],用户无法查看个人作品,而

收稿日期:2023-09-07　　**修回日期**:2024-01-10
基金项目:辽宁省高等学校基本科研项目(LJKMZ20222007);大连市青年科技之星项目(2021RQ068);大连东软信息学院科技创新基金项目(TIFP202307)
作者简介:王　凯(1987-),男,硕士研究生,研究方向为人工智能。

且收藏的视频也全都不见了,对关键词进行搜索无任何结果出现,消息也无法发送出去,首页重复推送刷过的视频。同时,抖音直播的后台异常,买家无法付款或付款后无相关信息。在社交平台上被用户广泛批评。

因此,及时修复崩溃,是软件开发人员必须要完成的一项任务。要完成崩溃修复任务,开发人员需要拥有一套崩溃的收集和聚合系统,在软件运行过程中发生崩溃后,能够通过该系统看到用户端发生的崩溃,然后进行崩溃的分析和解决。这被称之为崩溃报告系统(Crash Reporter)或者应用性能管理系统(APM)。常见的用于 Android 应用的商用崩溃报告系统包括 Google 公司的 Firebase[3]、腾讯公司的 Bugly[4]、听云公司的基调听云[5]等。这些系统能将用户侧发生的崩溃信息提交,其崩溃处理流程基本一致,即采集、聚类和指派给修复人员,大致流程如图 1 所示。

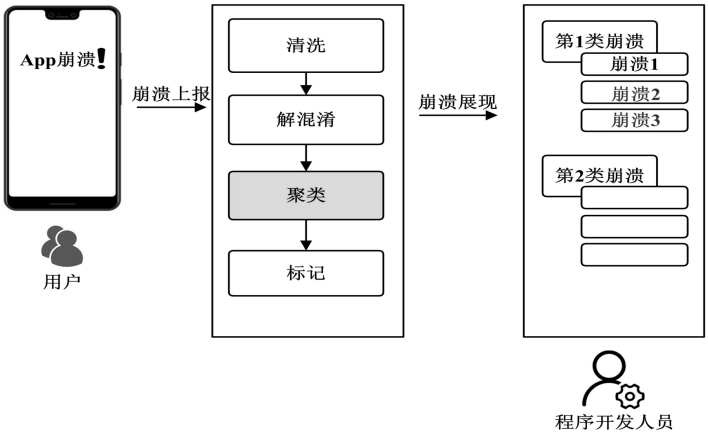

图 1 崩溃报告系统的数据处理流程

该文要解决的是如何对收集到的崩溃信息进行有效聚合。随着用户数量的急剧增长,要处理的崩溃类型和数量也大幅增长,在使用其他商业崩溃报告系统以及原始自研系统时,发现存在聚合不准确的问题,这对程序开发人员快速解决崩溃的能力形成了制约。基于实践,该文提出了一种基于崩溃特征和崩溃路径的二级异常聚合方法,并应用在自研的崩溃处理系统中。

崩溃信息的聚合错误分成两种:第一种,在某类异常中发现了不属于该类异常的样本。第二种,发现某两种或者更多种的异常应该归属于同一类,但在系统的聚合结果中,分成了不同的种类。这时,把聚类样本数最多的聚类记为正确,其他聚类下的样本均记为错误。当第一类聚类异常发生时,会导致程序开发人员遗漏要解决的异常,造成异常继续在用户中扩散,具有很高的危害性。当第二类聚类异常发生时,会浪费程序开发人员的排查时间,导致异常解决效率的降低,同样具备较高的危害性。

该文的主要贡献如下:

(1)减少了第一种错误的发生。原自研聚合系统仅根据崩溃的堆栈的前 7 行进行聚类,这忽视了同样行数的部分代码行可能会引起不同类型的异常。有部分商业系统在聚合时,针对异常信息,仅提取异常类型,不考虑异常的描述信息,这同样忽视了不同描述信息下崩溃的差别。

(2)减少了第二种错误的发生。为了减少第一种错误,该方法在将异常的描述信息加入到特征值的计算后,发现带来了第二种错误的大比例发生,该文提出了新的解决办法,即消除描述信息里的干扰元素,从而实现了同步减少第二种错误。

(3)提出了二级聚类,即结合异常类型、异常错误行进行一级聚类,然后使用异常堆栈提取崩溃发生的代码路径进行二级聚类。方便开发人员深入分析。

(4)计算过程资源消耗极少,单日处理百万级崩溃日志,最大时延小于 5 分钟。

1 相关工作

为了高效解决崩溃,对崩溃进行正确聚类成为软件开发公司必须要解决的问题。研究者提出了不同的聚类方法进行改进。研究的数据来源包括业务产品在运行过程中上报的崩溃信息,也包括自动化测试时产生的崩溃信息。前者包括了业界常见的大型软件,如 Mozilla 公司的 Firefox 和微软公司的系列软件产品等,后者形成了相应的测试数据集。

Dhaliwal 等人[6]针对 Mozilla Firefox 收集到的崩溃报告案例进行了分析。仅根据崩溃栈顶的函数签名来聚类时,会导致不同类型的崩溃被分配到同一类崩溃中的错误。他们提出了根据堆栈路径的相似性来进行聚类的两层聚类方法,首先根据栈顶签名进行第一层聚类,然后对同一聚类下的崩溃计算莱文斯坦距离(Levenshtein distance)进行第二层聚类。Kim 等人[7]针对微软公司的崩溃报告系统 WER 进行了研究。WER 系统存在一个被称为第二桶(second – bucket

problem)的问题,即把同一类型的崩溃分配到了另一种聚类。为此,他们提出了崩溃图(crash graph)的方式。崩溃图是根据崩溃堆栈构建出来的树状结构图,其中每个节点代表堆栈中不同的函数,每条边代表堆栈中的函数调用关系,函数出现次数和函数调用次数被作为图中节点和边的权重。崩溃图方法首先根据崩溃报告构建出每组崩溃的崩溃图,接着通过比较不同崩溃组的崩溃图的相似度来判断两组崩溃是否是一类崩溃。Dang 等人[8]同样针对 WER 系统崩溃报告聚类过程中存在的问题进行了分析,包括两类问题:(a)第二桶问题(the second bucket problem);(b)长尾问题(the long tail problem),即很多聚类中仅包含一个或几个崩溃。为了改善聚类的精确性,他们提出了基于堆栈相似度的 ReBucket 方法,设计了位置相关模型(position dependent model)来对崩溃进行度量,实现了更准确的聚类。Tonder 等人[9]提出的崩溃方法和前述方法均为不同,他们提出的方法是语义崩溃桶(semantic crash bucketing),该方法利用补丁模版和语义反馈来自动化生成近似的修复程序补丁,然后利用修复程序进行崩溃报告的聚类。该方法在三个测试数据集 AFL-Fuzz[10]、CERT BFF[11]和 Honggfuzz [12]上取得了优秀的聚类效果。Golagha 等人[13]提出了一种用在自动化测试领域的对失败的测试进行聚类的方法。他们使用的是非代码的特征,包含组件和文件等一般性特征、测试历史、损坏或修复以及 Jira 历史等,来构成特征向量,然后利用此对于失败的测试输入进行聚类。顾咏丰[14]在研究软件崩溃分析方法时,对如何定位崩溃根因提出了基于堆栈迹挖掘的崩溃定位方法。王文祥等人[15]对包含动态链接库信息的崩溃聚类进行了分析,在基于软件崩溃堆栈进行聚类的算法基础上,结合动态链接库信息,通过区分系统动态链接库和用户动态链接库,结合用户代码位置信息,得到用户关注的函数集合,以此为基准进行崩溃的界定聚类。不限于对崩溃进行聚类,金安[16]描述了某互联网公司内部 APM 系统的实现,包括了崩溃数据、性能数据和网络数据等,重点给出了大数据存储的聚合策略。

抖音是中国市场占有率排名第一的短视频类应用,月活跃人数达到 7.15 亿[17]。当抖音发生崩溃时,应用内部会把捕获的崩溃信息,包括崩溃堆栈、运行系统的版本、应用版本、内存占用等,上报到崩溃处理系统的服务端,进行崩溃数据的分析处理。最终,呈现给开发人员的是处理后的数据,能够看到崩溃的聚类情况,包括各类崩溃的总次数、用户数以及相应的比例,开发人员根据每类崩溃出现的次数等判断崩溃的修复优先级。

因为用户分布的广泛性,系统每日接收的 Java(包括 Kotlin)崩溃数量达到百万级别,因此对崩溃进行高效准确的聚类,成为系统必须要解决的问题。原始自研系统采用的取崩溃堆栈顶部 7 行计算得到的哈希值进行聚类,存在显著缺陷。上述研究成果当应用在本系统中时,主要存在以下不足:

(1)没有对 Android 平台下的 Java 崩溃做针对性研究。

仅使用栈顶签名和路径针对 Java 崩溃聚类时没有结合 Java 崩溃异常类型等信息,在同一聚类下存在多种不同类型的崩溃的问题。需要结合动态链接库信息对绝大部分的 Java 崩溃聚合不适用,因为最常见的 Java 崩溃没有动态链接库相关的信息。

(2)计算方式较为复杂。

在对大数据量崩溃进行聚合时,需要以极低的时延和 CPU 消耗来完成聚类任务。以 Tonder 等人[9]的语义崩溃聚类方法为例,需要首先生成代码补丁,然后根据修复程序来聚类,这对资源的消耗不满足系统的要求。

基于此,该文提出了一种新的高效实用的崩溃聚类方法。

2　基于崩溃特征的二级聚类方法

对上报的崩溃数据的处理分为 5 个步骤,如图 2 所示。

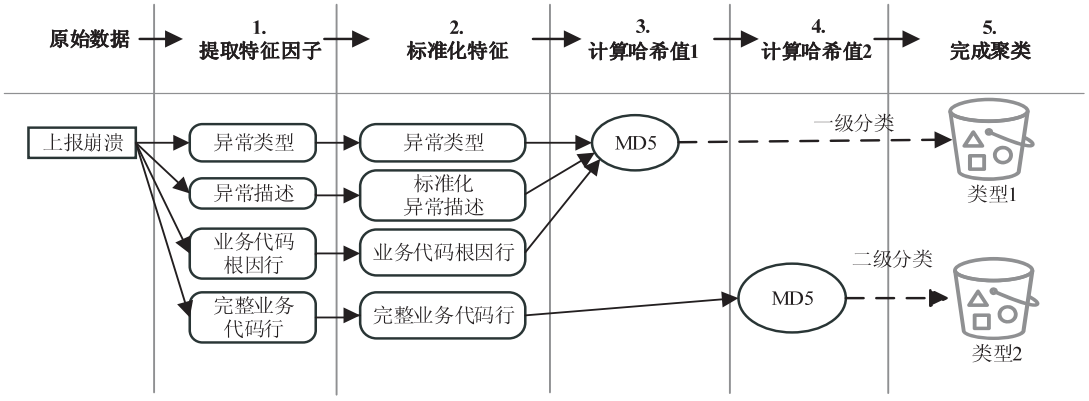

图 2　对上报崩溃的聚合处理流程

详细处理过程如下：

第 1 步：提取特征因子。

一个典型的 Java 崩溃上报的信息（见图 3）可以分解为 3 部分，分别是：①异常类型。比如空指针异常。②异常描述。比如空指针异常发生时，异常描述会说明是在什么对象上调用方法或引用属性时发生异常；内存 OOM 时，会描述当前是申请多少字节的内存造成内存不足从而导致异常的。③异常调用栈。堆栈表达了从系统层到软件业务层代码的逻辑调用关系，大部分调用栈的最底部是系统代码，最上层是业务代码。栈中的每一行可以提取出包名、类名、方法名和文件名。

```
java.lang.ArrayIndexOutOfBoundsException: length=101; index=101
        at com.[redacted] DeviceListAdapter.internalGetItemViewType(DeviceListAdapter.kt:39)
        at com.[redacted] DeviceListAdapter.internalBindViewHolder(DeviceListAdapter.kt:44)
        at com.[redacted] DeviceListAdapter.onBindViewHolder(DeviceListAdapter.kt:65)
        at com.[redacted] DeviceListAdapter.onBindViewHolder(DeviceListAdapter.kt:12)
        at androidx.recyclerview.widget.RecyclerView$Adapter.onBindViewHolder(RecyclerView.java:7065)
        at androidx.recyclerview.widget.RecyclerView$Adapter.bindViewHolder(RecyclerView.java:7107)
        at androidx.recyclerview.widget.RecyclerView$Recycler.tryBindViewHolderByDeadline(RecyclerView.java:6012)
        at androidx.recyclerview.widget.RecyclerView$Recycler.tryGetViewHolderForPositionByDeadline(RecyclerView.java:6279)
        at androidx.recyclerview.widget.GapWorker.prefetchPositionWithDeadline(GapWorker.java:288)
        at androidx.recyclerview.widget.GapWorker.flushTaskWithDeadline(GapWorker.java:345)
        at androidx.recyclerview.widget.GapWorker.flushTasksWithDeadline(GapWorker.java:361)
        at androidx.recyclerview.widget.GapWorker.prefetch(GapWorker.java:368)
        at androidx.recyclerview.widget.GapWorker.run(GapWorker.java:399)
        at android.os.Handler.handleCallback(Handler.java:938)
        at android.os.Handler.dispatchMessage(Handler.java:99)
        at android.os.Looper.loopOnce(Looper.java:210)
        at android.os.Looper.loop(Looper.java:299)
        at android.app.ActivityThread.main(ActivityThread.java:8105)
        at java.lang.reflect.Method.invoke(Native Method)
```

图 3 一例 Android 崩溃的堆栈

首先把堆栈信息的 3 部分拆解，得到异常类型（crash-type）、异常描述（raw-description）和异常调用栈，然后需要提取根因代码行的类名和方法名。为了实现这一目标，异常调用栈需要进一步分解为业务代码和系统代码，这一步是通过提取 Android 系统和 JDK 代码的包名前缀进行区分。同时，也将 Retrofit 等经过长期验证过的三方开源类库加入到系统聚类中。最终梳理出的包名前缀实现对堆栈的拆分。完成业务代码的提取之后，把业务代码的栈顶行的类名、方法名和文件名提取为崩溃根因（biz-root-cause），随后把包含栈顶行在内的完整业务代码堆栈作为崩溃路径（biz-code-path）进行记录。

如图 3 所示的崩溃信息，提取到的特征 1 是 java.lang. ArrayIndexOfBoundsExcetpion，特征 2 是 length = 101；index = 101，特征 3 是提取到的文件名 DeviceListAdapter、类名 DeviceListAdapter 和方法名 internalGetItemViewType，排除系统代码后，得到崩溃时业务代码的执行路径作为特征 4，每一行均提取文件名、类名、方法名进行字符串拼接，即第 1 行 DeviceListAdaper、DeviceListAdaper、internalGetItemViewType，第 2 行 DeviceListAdapter、DeviceListAdapter、internalBindViewHolder，第 3 行 DeviceListAdapter、DeviceListAdapter、onBindViewHolder 和第 4 行 DeviceListAdapter、DeviceListAdapter、onBindViewHolder，第 5 行及以下属于系统代码，不作为特征进行提取。

第 2 步：标准化特征。

部分异常描述中会包含个体设备特定运行环境的信息，以图 3 所示的崩溃为例，异常描述中包含了数组的长度数值，具体数值根据每次运行时不同。在常见的 OOM 崩溃（见图 4）中，异常描述中会包含申请内存大小的具体数值，这类数值信息在不同设备上或者同一设备上不同次使用时均不相同，但其实际表达的是一种异常。这就要求把此类元素消除。从实践中发现，此类异常描述存在很多的可能。如果将异常描述中存在的数值、地址类型的信息直接消除会产生新的副作用，比如会把 Class1 和 Class2 这样的包含数值的不同类名混同，而在实际的业务实现中，这是比较常见的一种行为。所以，有效的作法是从实践中提取负面案例构建案例库进行修正。修正后的异常描述称之为标准化异常描述。针对图 4 的 OOM 崩溃，根据异常类型是 OOM 崩溃时，采取的方案是消除异常描述信息，即可以取空字符串作为标准化后的异常描述。

```
java.lang.OutOfMemoryError: Failed to allocate a 10485776 byte allocation with 1040144 free bytes and 1015KB until OOM, target footprint 268435456, growth
        at com.[redacted].HomeActivity.initChartData(HomeActivity.kt:164)
        at com.[redacted].HomeActivity.initClickEvents$lambda$0(HomeActivity.kt:173)
        at com.[redacted].HomeActivity.$r8$lambda$ZeguPYejFgzBWdyNOOw737PbddY(Unknown Source:0)
        at com.[redacted].HomeActivity$$ExternalSyntheticLambda0.onClick(Unknown Source:2)
        at android.view.View.performClick(View.java:7753)
        at android.view.View.performClickInternal(View.java:7730)
        at android.view.View.access$3700(View.java:861)
        at android.view.View$PerformClick.run(View.java:29136)
        at android.os.Handler.handleCallback(Handler.java:938)
        at android.os.Handler.dispatchMessage(Handler.java:99)
        at android.os.Looper.loopOnce(Looper.java:210)
        at android.os.Looper.loop(Looper.java:299)
```

图 4 包含干扰信息的异常描述案例

用 raw-description 表示提取的原始异常描述，获取标准化异常描述 standard-description 的方法即对原始异常描述进行正则替换，将其中的数字等个体性信息消除，如式 1 所示：

standard - description = regex(raw - description)　　(1)

针对图 3 所示的崩溃信息,提取到的原始异常描述是“length = 101; index = 101”,用于消除干扰信息的正则匹配字符串为“length = 数字; index = 数字”,相应得到的消除正则表达式为“^length = \d+; index = \d+ $”,从而得到标准化异常描述为“length = 0; index = 0”。

第 3 步:计算基于根因的哈希值。

选取异常类型、标准化异常描述和根因业务代码行这三个要素做为崩溃的根因特征值。计算哈希值如式 2 所示:

$$F = \mathrm{MD5}(\text{crash - type}, \text{standard - description}, \text{biz - root - cause}) \quad (2)$$

因此,针对图 3 所示的崩溃,得到的崩溃哈希值为:

$$F = \mathrm{MD5}(\text{"java.lang.ArrayIndexOfBounds Exception"}, \text{"length = 0; index = 0"}, \text{"DeviceListAdapter"}, \text{"DeviceListAdapter"}, \text{"internalGetItemViewType"}) \quad (3)$$

第 4 步:计算基于路径的哈希值。

选取崩溃堆栈中所有的业务代码行计算崩溃的路径哈希值,如式 4 所示:

$$F = \mathrm{MD5}\left(\sum_{i=0}^{n} \text{ith biz - code}\right) \quad (4)$$

其中, biz - code 表示从业务代码行(biz-code-path)里的每一行提取到的文件名、类名和方法名。

针对图 3 所示的崩溃信息,可以得到二级聚类的崩溃哈希值是:

$$F = \mathrm{MD5}(\text{"DeviceListAdapter"}, \text{"DeviceListAdapter"}, \text{"internalGetItemViewType"}, \text{"DeviceListAdapter"}, \text{"DeviceListAdapter"}, \text{"internalBindViewHolder"}, \text{"DeviceListAdapter"}, \text{"DeviceListAdapter"}, \text{"onBindViewHolder"}, \text{"DeviceListAdapter"}, \text{"DeviceListAdapter"}, \text{"onBindViewHolder"}) \quad (5)$$

第 5 步:呈现二级聚类效果。

在最终呈现给开发者时,一级聚类列表由根因特征值计算的哈希值决定,从实践中看,绝大部分的崩溃在该层已经得到了很好的区分度,两种错误聚类均得到了很好的控制。当开发人员点击进入该类崩溃时,可进一步查看二级聚类的列表。

3　实验分析

针对两种错误聚类,分别定义错误指标。记录总样本数为 n。第一种,在某类异常中发现了不属于该类异常的样本,记录样本数 x。第二种,发现某两种或者更多种的异常应该归属于同一类,但在系统的聚类结果中,分成了不同的种类。这时,把归类样本数最多的聚类记为正确,其他归类下的样本均记为错误,记录为样本数 y。

第一种错误的比例 $e_1 = x/n$,第二种错误的比例 $e_2 = y/n$。对系统的指标(仅针对一级聚类)约束为 $e_1 \leqslant 0.05\%$, $e_2 \leqslant 0.10\%$。

本项目在实验时,选用的数据集为抖音应用实际上报的崩溃日志。实验方法为选择某一天上报的 10 000条日志,经过系统聚类处理后,再通过人工逐条检查日志聚类的合理性,根据错误聚类修正聚类实现,第二天使用新上报的 10 000 条日志做效果验证。错误聚类的案例,主要包括崩溃描述里干扰元素排除的遗漏、系统代码错误归类到业务代码错误等。对此,提取干扰元素的处理正则,补充到干扰消除正则库里,对于业务代码提取错误的,则更新到系统代码包名前缀库里,如此反复调整,直到人工检查确认聚类错误指标均在约束范围内。表 1 是原自研系统和采用文中方法的新自研系统对崩溃样本的聚类错误率。

表 1　聚类结果　%

项目名称	原系统错误率	本系统错误率
第一种错误	0.05	0.03
第二种错误	2.13	0.09

在系统后续的上线运行过程中,接收程序开发人员使用过程中的反馈对错误样本进行修正。在长期运行中,系统对崩溃日志聚类的准确性得到了验证。

4　结束语

为了解决 Android 崩溃信息聚类,提出了一种基于特征根因和路径的二级聚类方法,方法准确度高,满足了抖音平台的日志聚类需求。通过抖音和今日头条等应用大规模的数据验证,该方法的可靠性得到了尽可能的保证。存在的不足是,没有使用更多的数据集进行测试,在应用到特定类型的软件中时,可能存在增大的偏差。目前,使用同样方法进行崩溃聚类的系统也已经对抖音外部的公司开放,后续会在更大的数据集下进行优化。

参考文献：

[1] KIM J. Fatal bugs：disasters and revelations from software defects[M]. 叶蕾蕾，译. 北京：人民邮电出版社，2016：11-27.

[2] 中国经济网. 抖音回应 APP 发生崩溃：已经修复完毕[EB/OL]. (2021-11-02). http://finance1. ce. cn/stock/gsgdbd/202111/02/t20211102_37052828. shtml.

[3] Firebase. Firebase Crashlytics[EB/OL]. (2023-11-17). https://firebase. google. cn/docs/crashlytics? hl=zh-cn.

[4] 腾讯. 异常上报功能简介[EB/OL]. [2023-11-23]. https://bugly. qq. com/docs/introduction/bugly - introduction/? v=1.0.0.

[5] 基调听云. 基调听云 App[EB/OL]. [2023-11-23]. https://www. tingyun. com/tingyun-app.

[6] DHALIWAL T, KHOMH F, ZOU Y. Classifying field crash reports for fixing bugs：a case study of mozilla firefox[C]// 2011 27th IEEE international conference on software maintenance (ICSM). Williamsburg：IEEE, 2011：333-342.

[7] KIM S, ZIMMERMANN T, NAGAPPAN N. Crash graphs：an aggregated view of multiple crashes to improve crash triage[C]//2011 proceedings of international conference on dependable systems and networks, Hong Kong：IEEE, 2011：486-493.

[8] DANG Y N, WU R X, ZHANG H Y, et al. ReBucket：a method for clustering duplicate crash reports based on call stack similarity[C]//2012 proceedings of the 34th international conference on software engineering. Zurich：IEEE, 2012：1084-1093.

[9] TONDER R V, KOTHEIMER J, GOUES C L. Semantic crash bucketing[C]//2018 proceedings of the 33rd ACM/IEEE international conference on automated software engineering. Montpellier：IEEE, 2018：612-622.

[10] American fuzzy lop. AFL-Fuzz[EB/OL]. (2018-04-26). http://lcamtuf. coredump. cx/afl/.

[11] CMU Software Engineering Institute. CERT BFF[EB/OL]. (2018-04-26). https://www. cert. org/vulnerability-analysis/tools/bff-download. cfm.

[12] Google. Honggfuzz[EB/OL]. (2018-04-26). https://github. com/google/honggfuzz.

[13] GOLAGHA M, LEHNHOFF C, PRETSCHNER A, et al. Failure clustering without coverage[C]//2019 28th ACM SIGSOFT international symposium on software testing and analysis. Beijing：Association for Computing Machinery, 2019：134-145.

[14] 顾咏丰. 面向根源的软件崩溃分析方法[D]. 武汉：武汉大学，2021.

[15] 王文祥，高 庆，许 可，等. 一种结合动态链接库信息的崩溃输入聚类方法[J]. 软件学报，2023，34(4)：1594-1612.

[16] 金 安. 一种移动端 APM 系统框架设计与实现[J]. 软件导刊，2022，21(1)：205-209.

[17] Questmobile 网. QuestMobile2022 中国移动互联网年度大报告[EB/OL]. (2023-02-21). https://www. questmobile. com. cn/research/report/1627881652360417282.