软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.006021]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于词频-逆文件频率的错误定位方法*

张 卓[1], 雷 晏[2], 毛晓光[1], 常 曦[3], 薛建新[1,3], 熊庆宇[2]

[1](国防科技大学 计算机学院,湖南 长沙 410072)

[2](重庆大学 大数据与软件学院,重庆 400044)

[3](上海第二工业大学 计算机与信息工程学院,上海 200127)

通讯作者: 雷晏, E-mail: yanlei@cqu.edu.cn

摘 要: 错误定位方法大多通过分析语句覆盖信息,以标识出导致程序失效的可疑语句.其中,语句覆盖信息通常以语句执行或语句未执行的二进制状态信息来表示.然而,该二进制状态信息仅表明该语句是否被执行的信息,无法体现该语句在具体执行中的重要程度,可能会降低错误定位的有效性.为解决这个问题,本文提出基于词频-逆文件频率的错误定位方法.该方法采用词频-逆文件频率技术识别出单个测试用例中语句的影响程度高低,从而构建出具有语句重要程度识别度的信息模型,并基于该模型来计算语句的可疑值.实验结果表明,本文方法大幅提升错误定位的效能.

关键词: 错误定位;词频;逆文件频率;可疑值;

中图法分类号: TP311

中文引用格式: 张卓,雷晏,毛晓光,常曦,薛建新,熊庆宇.基于词频-逆文件频率的错误定位方法.软件学报. http://www.jos.org.cn/1000-9825/6021.htm

英文引用格式: Zhang Z, Lei Y, MAO X, Chang X, Xue J and Xiong Q. Fault Localization Using Term Frequency–Inverse Document Frequency. Ruan Jian Xue Bao/Journal of Software, 2019 (in Chinese). http://www.jos.org.cn/1000-9825/6021.htm

Fault Localization Using Term Frequency–Inverse Document Frequency

ZHANG Zhuo[1], LEI Yan[2], MAO Xiao-Guang[1], CHANG Xi[3], XUE Jian-Xin[1,3], XIONG Qing-Yu[2]

[1](School of Computer, National University of Defense Technology, Changsha 410072, China)

[2](School of Big Data & Software Engineering, Chongqing University, Chongqing 400044, China)

[3](College of Computer and Information Engineering, Polytechnic University, Shanghai 200127, China)

Abstract: Most existing fault localization approaches utilize statement coverage information to identify suspicious statements potentially responsible for failures. They generally use the binary status information to represent the statement coverage information, indicating a statement executed or not executed. However, the binary information just shows whether a statement is executed or not whereas it cannot evaluate the importance of a statement in a specific execution. Consequently, this may degrade fault localization performance. To address this issue, this paper proposes a fault localization approach using term frequency-inverse document frequency. Specifcally, our appraoch constructs an information model to suceesfully identify the influence of a statement in a test case, and uses the information model to evaluate the suspiciousness of a statement of being faulty. The experiments show that the proposed approach significantly improves fault localization effectiveness.

Key words: fault localization; term frequency; inverse document frequency; suspiciousness;

调试是软件开发过程中耗费成本较高的活动.为了降低调试成本,研究人员提出了许多错误定位方法(例

* 基金项目: 国家自然科学基金(61620106007, 61602504, 61502296, 61672529);中央高校基本科研业务费(2019CDXYRJ0011)

Foundation item: National Natural Science Foundation of China (61620106007, 61602504, 61502296, 61672529); Fundamental Research Funds for the Central Universities (2019CDXYRJ0011)

收稿时间: 2019-07-18; 修改时间: 2019-10-28, 2019-12-22; 采用时间: 2020-01-18; jos 在线出版时间: 2020-04-21

如:文献[1,2,3,4,5]).其中,基于频谱的错误定位方法(Spectrum-based Fault Localization, SFL)[2]是被广泛研究和使用的错误定位方法,能有效减少检查代码量,提升调试效率.SFL 方法的原理是首先通过分析测试用例集运行信息,构建出语句覆盖信息矩阵;然后基于该矩阵采用可疑值度量公式评估出语句为错误的可疑值;最后,依据可疑值对所有语句进行降序排列,并根据语句排序列表来定位错误语句,其中的语句覆盖信息矩阵是 SFL 方法的关键基础,影响着后续可疑值计算模型的精度.

然而,语句覆盖信息矩阵仅仅记录了语句是否被测试用例执行的二进制状态信息,即被执行为 1,未被执行为 0.二进制状态信息表达信息能力很有限,无法展示出语句在具体测试用例执行中的重要程度,限制着错误定位精度[6].目前,一些研究[7]尝试将语句在测试用例中的执行次数融入到覆盖矩阵信息.然而,这些研究并没有考虑到该语句在所有测试用例中的特征,由此产生一些偏差,对错误定位效果反而产生不利的影响[40].因此,有必要研究如何融合更为丰富的有效信息到信息覆盖矩阵,使之能更准确地反映出语句和测试用例之间的关系,解决信息表达能力弱对定位精度上限的限制问题.

基于上述分析,本文提出基于词频-逆文档频率(Term Frequency-Inverse Document Frequency,TF-IDF)[8]的错误定位方法,以增强信息表达能力来实现错误定位效力的提升.该方法试图将词频-逆文档频率技术融入到错误定位中,综合全局与局部两个维度获取语句与测试用例之间更为丰富的有效关系信息,以此构建出信息表达能力更强的信息模型.根据构建的信息模型,利用 SFL 的方法原理重新定义其中的参数来计算语句的可疑值.本文在 8 个开源程序的大量错误版本上展开了实验,与 8 种典型错误定位方法进行了对比.实验结果表明,相比较于 8 种典型错误方法,基于 TF-IDF 的错误定位方法能大幅提升缺陷定位效能.具体来说,本文方法在高效能区间(即检查可执行语句的比例从 5%到 10%区间)取得最明显的效能提升,并且在严格统计学意义下本文方法取得 7 个更优结果和 1 个相似结果.

文章组织结构:本文第 1 节介绍 TF-IDF 技术和基于频谱的错误定位技术.第 2 节具体描述本文方法.第 3 节为实验描述和结果分析.第 4 节为相关工作.第 5 节总结了本文.

1 相关背景

(1)词频-逆文档频率

在信息检索领域中,词频-逆文档频率(Term Frequency-Inverse Document Frequency,TF-IDF)是一种数值统计技术,旨在反映单词对文档集或一个语料库中的一份文件的重要程度.在自然语言处理领域,TF-IDF 是最流行的词语加权方案之一,通常应用于信息搜索、文本挖掘和用户建模[8].TF-IDF 的主要思想是:假设有一个文本集合由多个文本组成,如果某个词在一个文本中出现的次数较多,而在文本集合中的其他文本出现的次数较少,则这个词对于区分这个文本集具有较好的能力,比较适合用来做分类.TF 表示词频(Term Frequency),IDF 是逆文本频率指数(Inverse Document Frequency).词频指的是一个词在一篇文章中出现的次数,逆文本频率指数指的是一个词在所有的文档中出现的频度[8].如果一个词在一篇文档中出现的次数较少,那么这个词的词频值较低,反之如果这个词出现的次数较多则它的词频值较高,这个次数是对词数量的归一化,防止它偏向于较长的文本文件(同一个词无论重要与否,在短文件中出现的次数可能较少,而在长文件中较多).相反的,一个词如果在许多文档中都出现,则它的逆文本频率指数值较低,反之在较少的文档中出现则逆文本频率指数值则较高.IDF 是一个词语重要性的度量,具有较低 IDF 的词语表示区分能力不强,反之则是区分能力较强.下面举例说明 TF-IDF 的一般计算方法:

$$tf_i = \frac{n_i}{\sum_k n_k} \tag{1}$$

公式(1)是 *tf* 的计算公式,假设一个文档集合 *D* 中的一个文本 *L* 包含 *k* 个词,按照从 1 到 *k* 的顺序进行编号,第 *i* 个词的编号为 *i*,则第 *i* 个词的 *TF* 值为 tf_i.公式(1)的分子为词 *i* 在文本 *L* 中出现的次数,记为 n_i,分母为文本 *L* 中从 1 到 *k* 所有词的出现次数之和,它们的比值即为词 *i* 的 *TF* 值.

$$idf(t_i, D) = log\frac{N}{|\{d \in D: t_i \in d\}|} \tag{2}$$

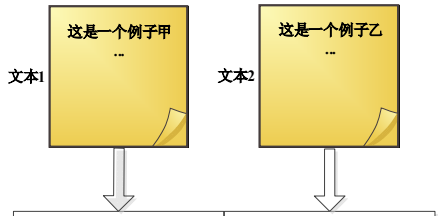
$$TFIDF_i = tf_i * idf(t_i, D) \tag{3}$$

公式(2)是 *idf* 的计算公式, *D* 为一个文档集合,文档的总数为 *N*,*idf*(*t*,*D*)表示一个单词 *t* 在文档集合 *D* 中的逆文本频率指数.对于文档集合 *D*,*d* 为 *D* 中的一个文档,如果这个 *t* 在 *d* 中出现,则 *t* 的数目加 1, $|\{d \in D: t \in d\}|$ 指的是在文档集合 *D* 中包含词语 *t* 的文档 *d* 的个数.为了防止分母为 0,某些情况下会在分母的后面加 1.

TF-IDF 值 $TFIDF_i$ 指的是 *tf* 与 *idf* 的乘积,如公式(3),如果一个词语在整个文件集合中具有较低的频率,但是在其中一个文件中具有较高的频率,那么这个词具有较高的 TF-IDF 值.因此,TF-IDF 技术可以对比较重要的词语进行保留,而丢弃掉相对不重要的词语.

例如,图 1 为 TF-IDF 示例.假设文本集 *D* 中有两个文本,分别为文本 1 和文本 2.文本 1 中有内容“这是一个例子甲”,文本 2 中有内容“这是另一个例子乙”.则根据公式(1)、公式(2)和公式(3):*tf*(甲,文本 1)=*tf*(甲,文本 2)=1/5;*idf*(甲,*D*)=*log*(2/1)=0.3010;*TFIDF*(甲,文本 1)=*tf*(甲,文本 1)* *idf*(甲,*D*)=0.0602;同理可以计算出 *TFIDF*(乙,文本 2)=0.0602,而“这”、“是”、“一个”、“例子”四个词的 TF-IDF 值为 0.从上述计算可以看出,由于“这”、“是”、“一个”、“例子”四个词在文本集合 D 中的文本 1 和文本 2 中都有出现,因此这四个词不属于文本的关键词,也就是说这四个词对区分文档的贡献度为 0,文本 1 中的“甲”和文本 2 中的“乙”TF-IDF 值最高,这表明“甲”和“乙”分别是文本 1 和文本 2 的关键词,这两个词对文本的贡献度最高.

在搜索领域当中,如果一个词在一个文件中出现的次数较少,那么这个词的权重也应该比较大.反之如果一个词在大量文件中都存在,那么根据这个词是不清楚要找的目标,这个词的权重应该较小.因此,TF-IDF 技术可以应用到错误定位领域中,如果一个语句在所有的测试用例中都被运行,那么这个语句对于区分不同的测试用例意义并不大,这样的语句应该被赋予较小的权值;如果一个语句只有在部分测试用例中执行,而且这些测试用例执行的语句数也比较少,那么这个语句对这个测试用例的意义就会比较大,也应该被赋予较高的权值.



文本1		文本2	
单词	出现次数	单词	出现次数
这	1	这	1
是	1	是	1
一个	1	一个	1
例子	1	例子	1
甲	1	乙	1

Fig.1 An Example Illustrating TF-IDF

图 1 TF-IDF 示例

(2)基于频谱的错误定位

程序谱(Program Spectrum)指的是一个程序在特定测试用例下的运行信息,比如语句覆盖信息、条件分支以及循环路径等[9].程序谱可以用来追踪程序的行为,当一个程序执行导致程序失效时,这个程序行为就可以用来寻找导致程序失效的可疑代码的位置.程序覆盖信息指的是在一个测试用例运行完之后,程序的哪些部分被这个测试用例覆盖到.获得程序覆盖信息后,就可以确定程序的哪些部分和错误输出有着关联关系,也就缩小了错误查找的搜索域[6].基于频谱的错误定位技术(Spectrum-based Fault Localization, SFL)的主要思想就是利用程

序在运行完某些测试用例之后的语句覆盖信息构建程序谱,之后利用可疑值计算公式以程序谱作为输入来计算出程序中每个语句的可疑值并进行排序[9].下面是 SFL 的一些定义:

P 一个程序,包含 N 个语句

T 程序 P 的测试用例集,共包含 M 个测试用例

s_i 程序 P 中第 i 个可执行语句

t_i 测试用例集 T 中第 i 个测试用例

$a_{np}(s_i)$ 未执行语句 s_i 的通过测试用例数

$a_{nf}(s_i)$ 未执行语句 s_i 的失败测试用例数

$a_{ep}(s_i)$ 执行语句 s_i 的通过测试用例数

$a_{ef}(s_i)$ 执行语句 s_i 的失败测试用例数

$$\begin{matrix} & N\ statements & & & errors \\ \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix} & & & & \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_M \end{bmatrix} \end{matrix}$$

Fig.2 The coverage of M executions

图 2 M 个测试用例执行后的覆盖信息

根据以上定义,图 2 为程序 P 执行完测试用例集 T 后的语句覆盖信息. $P=\{s_1,s_2,...,s_N\}$, $T=\{t_1,t_2,...,t_M\}$. t_1 到 t_M 中至少包含一个错误的测试用例.x_{ij} 表示程序 P 中的语句 s_j 在测试用例 t_i 执行下的覆盖情况:1 表示被覆盖,即测试用例 t_i 执行了语句 s_j;0 表示未被覆盖, 即测试用例 t_i 没有执行语句 s_j.左侧的 $M\times N$ 矩阵即为语句的覆盖情况,右侧为结果向量 e. 如果 t_i 是成功的测试用例,则 e_i 为 0,否则为 1.

Table 1 Maximal Formulas of SFL

表 1 最优 SFL 公式

Name		Formulas	Name		Formulas
ER1′	Naish1	$\begin{cases}-1 & \text{if } a_{ne}>0\\ a_{np} & \text{if } a_{ne}\le 0\end{cases}$	ER5	Wong1	a_{ef}
	Naish2	$a_{ef}-\frac{a_{ep}}{a_{ep}+a_{np}+1}$		Russel&Rao	$\frac{a_{ef}}{a_{ef}+a_{nf}+a_{ep}+a_{np}}$
	GP13	$a_{ef}(1+\frac{a_{ep}}{2a_{ep}+a_{ef}})$		Binary	$\begin{cases}0 \text{ if } a_{ne}>0\\ 1 \text{ if } a_{ne}\le 0\end{cases}$
GP02		$2\left(a_{ef}+\sqrt{a_{np}}\right)+\sqrt{a_{ep}}$	GP03		$\sqrt{\lvert a_{ef}^2-\sqrt{a_{ep}}\rvert}$
GP19		$a_{ef}\sqrt{\lvert a_{ep}-a_{ef}+a_{nf}-a_{np}\rvert}$	Dstar（D*）		$\frac{a_{ef}^*}{a_{nf}+a_{ep}}$
Ochiai		$\frac{a_{ef}}{\sqrt{(a_{ef}+a_{nf})(a_{ef}+a_{ep})}}$	Tarantula		$\frac{\frac{a_{ef}}{a_{ef}+a_{nf}}}{\frac{a_{ef}}{a_{ef}+a_{nf}}+\frac{a_{ep}}{a_{ep}+a_{np}}}$

根据 SFL 定义中的四个参数 $a_{np},a_{nf},a_{ep},a_{ef}$,可以由可疑值计算公式计算出程序 P 中 N 个语句的可疑值. SFL 在构建可疑性度量公式时,假设错误语句具有相对较高的 a_{ef} 值,较低的 a_{ep} 值.其理想情况是错误语句在所有的失败的测试用例中都有执行而在通过的测试用例中均没有被执行,这时错误语句的 a_{ef} 值为最大,而 a_{ep} 值为最小.可疑性度量公式在此情况下会赋予错误语句最大可疑值[1].表 1 列举了 SFL 最优的典型可疑值计算公式.一般来说,不同度量公式根据给这四个参数赋予不同的权值从而产生不同的排名.Xie 等[2,10]理论分析了大量 SFL

度量公式,总结出了 5 种最优公式,即 ER1'、ER5、GP02、GP03 和 GP19.除了这 5 种理论最优公式,D*是实证调查[4]中错误定位效力最优的定位方法.需要特别进行说明的是,D*公式中的*通常被赋值为 2[4],这也是本文实验所采取的赋值.

SFL 以其较为简单的工作原理和较好的定位效果得到了广泛研究和应用.然而,它使用的信息覆盖矩阵只是简单的二进制信息矩阵,0 和 1 仅代表语句是否被测试用例执行.其信息表达能力有限,无法显示语句执行对测试用例的影响程度,限制了其错误定位精度.尤其在程序规模大的情况下,不相关的语句将迅速增加,这对软件调试带来更大干扰.因此,融入丰富的有效信息对覆盖信息矩阵进行优化,获得更为精准的定位有效信息,有利于提高错误定位精度.

2 基于词频-逆文档频率的错误定位方法

(1)方法原理

基于词频-逆文档频率的错误定位方法的基本思想是利用 TF-IDF 技术构建覆盖信息矩阵.这个矩阵中的每一个元素的值不再是代表执行或未执行的二进制信息,而是表示语句对测试用例的影响程度.$TF(s,t)$对应的是语句 s 在测试用例 t 中的 TF 值.一个测试用例中执行的语句数越多,按照公式(1)中 TF 值的计算,测试用例 t 中每个语句出现一次,分子为 1,分母为语句数的总和,因此 $TF(s,t)$值越小.$IDF(s)$是语句 s 在所有测试用例的执行频度.如果 s 在许多测试用例中执行,则 $IDF(s)$值较小,如果 s 在少数测试用例中执行,则 $IDF(s)$值较大.

$$\begin{matrix} N\ statements & & & errors \end{matrix}$$
$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_M \end{bmatrix} \rightarrow \begin{bmatrix} TFIDF(x_{11}) & TFIDF(x_{12}) & \cdots & TFIDF(x_{1N}) \\ TFIDF(x_{21}) & TFIDF(x_{22}) & \cdots & TFIDF(x_{2N}) \\ \vdots & \vdots & \ddots & \vdots \\ TFIDF(x_{M1}) & TFIDF(x_{M2}) & \cdots & TFIDF(x_{MN}) \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_M \end{bmatrix}$$
$$\begin{matrix} & N\ dimensional & & errors \end{matrix}$$

Fig.3 The binary matrix and the TF-IDF matrix of M executions

图 3 M 个测试用例的二进制矩阵和 TF-IDF 矩阵

假设程序 P 包括 N 个语句,分别为$\{s_1,s_2,\ldots,s_N\}$,这些语句被 M 个测试用例 T 执行.测试用例 T 为$\{t_1,t_2,..,t_M\}$,T 中至少包含一个失败的测试用例.如图 3 所示,所有的测试用例运行完毕后, 左侧的 $M\times(N+1)$的二进制矩阵由语句覆盖情况与测试结果组成.其中 $M\times N$ 矩阵表示语句执行情况.如果语句 s_j 由测试用例 t_i 执行,则 x_{ij} 为 1,否则为 0.矩阵最右侧的结果向量 $errors$ 表示测试结果,它是由 M 个测试用例的结果$\{e_1,e_2,..,e_M\}$组成.如果 t_i 是失败测试用例,则 e_i 的值为为 1,否则值为 0.可以发现,二进制矩阵仅表示语句在测试用例中是否被执行,并不能得出语句对测试用例的影响程度.因此,本文方法采用 TF-IDF 对矩阵进行重新计算.在左侧 $M\times(N+1)$的二进制矩阵的基础上,对矩阵中的元素 x_{ij} 计算出 $TFIDF(x_{ij})$,具体计算方法为公式(4)、(5)和(6).

$$TF(x_{ij}) = x_{ij} * \frac{1}{1+\log(N(t_i))} \tag{4}$$

$$IDF(x_{ij}) = log_{10}^{M/DF(s_j)} \tag{5}$$

$$TFIDF(x_{ij}) = TF(x_{ij}) * IDF(x_{ij}) \tag{6}$$

公式(4)计算 x_{ij} 的 TF 值.在公式(4)中,x_{ij} 为图 3 左侧二进制矩阵中的一个元素,值为 1 或 0.$N(t_i)$为测试用例 t_i 中被执行的语句的个数,即图 3 左侧二进制矩阵 $M\times N$ 中第 i 行值为 1 的个数.公式(5)计算 x_{ij} 的 IDF 值.M 为测试用例的总数,$DF(s_j)$为所有测试用例中执行语句 s_j 的测试用例数(如果 $DF(s_j)$为 0 时,取 $IDF(x_{ij})$为 0).参照实验数据,公式(4)和公式(5)选取常用的 log 和 log10.最后,基于公式(4)和(5),公式(6)计算 x_{ij} 的 TF-IDF 值.

本文基于构建的 TF-IDF 矩阵重新定义语句 s_j 的四个参数 a_{np},a_{nf},a_{ep},a_{ef}.公式(7),(8),(9)和(10)分别为语句 s_j 的四个参数 a_{np},a_{nf},a_{ep},a_{ef}的计算公式.最后,本文方法根据四个重新定义的参数,采用表 2 的可疑值计算公式计算出每个语句的可疑值.

$$a_{np}(s_j) = \sum_{i=1}^{M}(1 - TFIDF(x_{ij})) * (1 - e_i) \tag{7}$$

$$a_{nf}(s_j) = \sum_{i=1}^{M}(1 - TFIDF(x_{ij})) * e_i \tag{8}$$

$$a_{ep}(s_j) = \sum_{i=1}^{M} TFIDF(x_{ij}) * (1 - e_i) \tag{9}$$

$$a_{ef}(s_j) = \sum_{i=1}^{M} TFIDF(x_{ij}) * e_i \tag{10}$$

(2) 例子演示

如图 4 所示,本小节通过包含错误语句 s_6 的程序 P 展示基于 TF-IDF 的错误定位方法工作原理.这个例子选择 Russel&Rao 来计算 8 个语句的可疑值.左侧 s_1 到 s_8 之下的单元格表示该语句是否被测试用例执行(1 表示执行,0 表示未执行);右侧 8 列的每个语句之下的单元格表示该语句在相对应测试用例中的 TF-IDF 值;而最右侧 R 之下的单元格表示测试用例的执行是否失败(0 表示通过,1 为失败).根据语句的二进制覆盖信息和测试用例结果,Russel&Rao 输出可疑值降序排列的语句表:{s_1,s_2,s_3,s_4,s_6,s_7,s_8,s_5}.由于 s_1,s_2,s_3,s_4 四个语句在 6 个测试用例均被执行,二进制信息矩阵无法体现出这些语句对测试用例结果的差异性影响.转换为 TF-IDF 信息矩阵之后,根据新的 a_{np},a_{nf},a_{ep},a_{ef}, Russel&Rao 计算出新语句排序表:{s_6,s_7,s_8,s_4,s_5,s_1,s_2,s_3}.由于 s_1,s_2,s_3,s_4 四个语句的 TF-IDF 值均为 0,错误语句 s_6 则由原来二进制矩阵下的第 5 名提升到了第 1 名.因此,基于 TF-IDF 的方法改进了错误定位效力.

Program P(maximal value of a,b,c)										Bug information									
S1:Read(a); S2:Read(b); S3:Read(c); S4:if(c > a) and (c > b){ S5: max = c; } S6:else if (a < b){								S7:max = a; S8:else {max = b;}		S6 is faulty. Correct form: else if (a > b){									
T	a,b,c	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	T	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	R
t1	1,2,3	1	1	1	**1**	1	0	0	0	t1	0	0	0	0	0.18	0	0	0	0
t2	-2,-7,5	1	1	1	1	1	0	0	0	t2	0	0	0	0	0.18	0	0	0	0
t3	5,-6,-8	1	1	1	1	0	1	0	1	t3	0	0	0	0	0	0.06	0	0.17	1
t4	5,4,3	1	1	1	**1**	0	1	0	1	t4	0	0	0	0	0	0.06	0	0.17	1
t5	**4,7,1**	1	1	1	1	0	1	1	0	t5	0	0	0	0	0	0.06	0.17	0	1
t6	**-1,2,1**	1	1	1	1	0	1	1	0	t6	0	0	0	0	0	0.06	0.17	0	1
SFL	value	0.67	0.67	0.67	**0.67**	0	0.67	0.33	0.33	TFIDF	0	**0**	0	0	0	0.11	0.08	0.07	
	rank	1	2	3	4	8	5	6	7		6	7	8	4	5	1	2	3	

Fig.4 An example illustrating the TF-IDF-based fault localization approach

图 4 基于 TF-IDF 的错误定位方法例子说明

3 实验

(1) 实验设计

实验在8组典型程序集上,对比基于TF-IDF的错误定位方法与表1中的8种典型SFL方法.表2描述了8组实验程序集,依次为程序名、简要描述、实验使用的错误版本数量、语句行数和测试用例数.这些程序代码行数最少为5,000行,最多为491,000行.其中,python、gzip和libtiff从ManyBugs[11]获取,space、nanoxml_v1、nanoxml_v2、nanoxml_v3和nanoxml_v5从SIR[12]获取.

为了评估本文提出的错误定位方法的效力,实验采用错误定位精度(称为*Exam*)[13]作为评价标准.*Exam*表示在找到真正的错误语句之前要检查的可执行语句的百分比,即缺陷语句在排序表中的排名,除以整个程序可执行语句总数.越低值的*Exam*表示查找到真正的错误语句时检查的语句的百分比越低,代表越好的缺陷定位性能[15].

Table 2 The subject Programs

表 2 实验程序集

Program	Description	Versions	KLOC	Test
python	General-purpose language	8	407	355
gzip	Data compression	5	491	12
libtiff	Image processing	12	77	78
space	ADL interpreter	35	6.1	13585
nanoxml_v1	XML parser	7	5.4	206
nanoxml_v2	XML parser	7	5.7	206
nanoxml_v3	XML parser	10	8.4	206
nanoxml_v5	XML parser	7	8.8	206

实验环境为:CPU为I5-2640、16G物理内存、GPU为12G的NVIDIA TITAN X Pascal、操作系统版本为Ubuntu16.04.3、matlab版本为MATLAB R2016b.

(2) 结果分析

图5为基于TF-IDF的方法和8种SFL方法的*Exam*对比图,对于每个子图,横坐标表示在所有程序中检查的可执行语句的百分比.纵坐标表示定位方法找出错误的比例.图5中的每一个点表示在检查可执行语句的百分比时,定位到的错误的百分比.从图中可以看出,除了Ochiai之外,其余方法在检查可执行语句到5%或10%时则出现了明显的效果提升.因此,基于TF-IDF的方法提升了缺陷定位的性能.

表3显示了基于TF-IDF的方法和五种SFL方法的*Exam*对比情况.对于每种SFL方法,实验提供了三种类型的*Exam*对比,分别为best、average和variance.best指的是在程序的所有错误版本中最小的*Exam*值,即表示最好的定位效果;average为所有错误版本的*Exam*平均值,即表示平均定位效果;variance表示所有错误版本的*Exam*方差,即表示定位效果的稳定性.表3中标粗的数据代表最优值.以ER1'为例,基于TF-IDF的ER1'和原始的ER1'对比,它获得8个最优best、8个最优average和5个最优variance.这表明基于TF-IDF的ER1'比原始的ER1'相比,其最好定位效果和平均定位效果高于ER1',并且定位效果具有更好的稳定性.表3的实验结果表明,基于TF-IDF的错误定位方法能提高错误定位效果,且具有更好的稳定性.

同时,表3显示出应用了TF-IDF技术后,在一些实验程序取得的average值,并没有取得更好的效果.本文深入分析实验数据后,发现实验程序中少数错误版本定位效能的高偏向性,会影响整个实验对象的结果.以一个例子来描述这种高偏向性带来的这种结果.假定在实验程序的少数错误版本上,错误定位方法FL1的效能比FL1(TF-IDF)显著高出很多;在其它大多数错误版本上,FL1的效能低于FL1(TF-IDF),而效能低出量不大.在这种情况下,FL1在少数错误版本上取得的效能高偏向性,会抵消和扭转其在大多数版本上取得效能劣势,最终呈现出FL1(TF-IDF)的average值比FL1会低很多,即FL1的效能要好于FL1(TF-IDF).因此,这种情况存在少数错误版本的效能高偏向性问题,并不能得出FL1优于FL1(TF-IDF)的结论.为了消除这种偏向性并更深入地评价本文方法,实验采用Wilcoxon有符号秩检验(称为*Wilcoxon-Signed-Rank Test*)[14].*Wilcoxon-Signed-Rank Test*把观测值和零假设的中心位置之差的绝对值的秩,分别按照不同的符号相加作为其检验统计量[14].它适用于T检验中的成对比较,不要求成对数据之差服从正态分布,只要求对称分布即可[14]. *Wilcoxon-Signed-Rank Test*结果为Better表明定位效果较好,Similar表明定位效果相似,Worse表明定位效果较差.

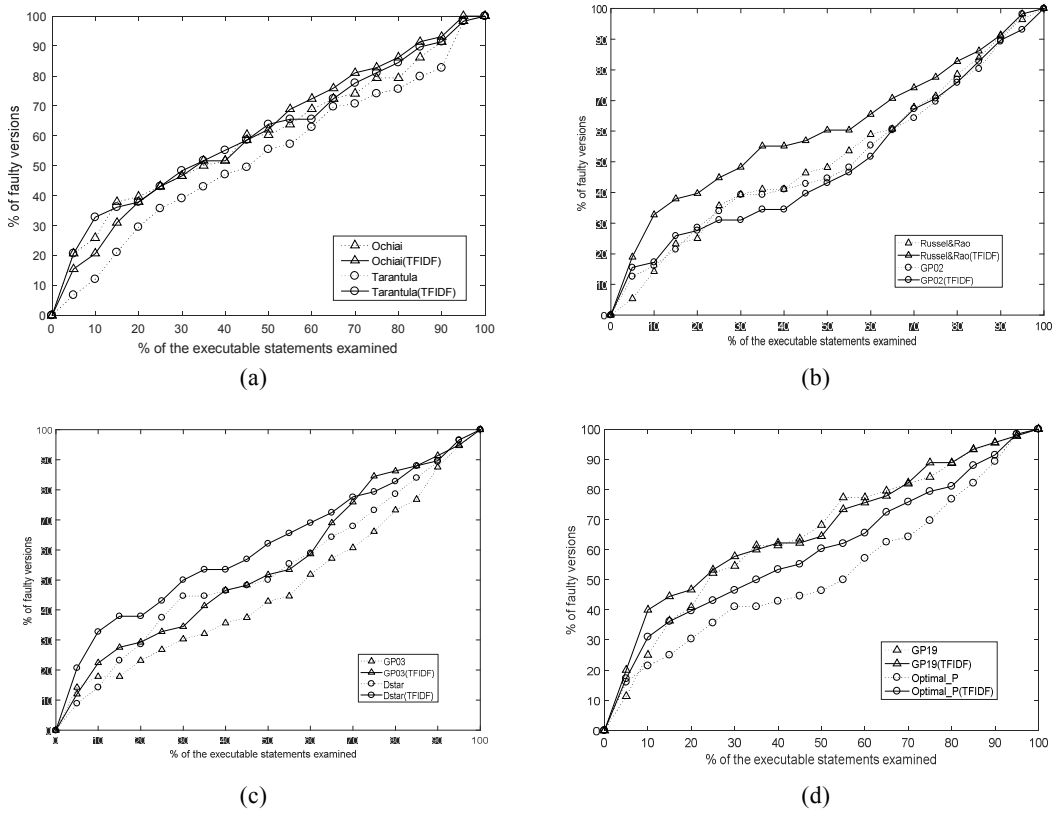


Fig.5 *Exam* score of TF-IDF over SFL

图 5 基于 TF-IDF 的方法和 SFL 的 *Exam* 值对比图

具体来说,给定一个程序,错误定位方法FL1在该程序所有错误版本的*Exam*值为G(y);FL1(TF-IDF)在该程序所有错误版本的*Exam*值为F(x).在给定的显著性水平 ϕ 下(本文设定 ϕ 值为0.05),可以使用2-tailed的p值和1-tailed的p值来得出结论.在2-tailed检验中,如果$p \geq \phi$,那么原假设H0: F(x)和G(y)不存在显著差异会被接受,即FL1(TF-IDF)在所有错误版本的*Exam*值与FL1不存在显著差异,从而FL1（TF-IDF）和FL1具有相似的定位效能(用Similar表示);否则,备择假设H1: F(x)和G(y)存在显著差异会被接受.对于1-tailed的p值,有两种情况:1-tailed(right)和1-tailed(left).在1-tailed(right)检验中,如果$p \geq \phi$,那么原假设H0: F(x)不趋于显著地大于G(y)会被接受;否则,备择假设H1: F(x)趋于显著地大于G(y)会被接受,即FL1(TF-IDF)在所有错误版本的*Exam*值趋于显著地大于FL1,从而FL1（TF-IDF）的定位效能低于FL1(用Worse表示).在1-tailed(left)检验中,如果$p \geq \phi$,那么假设H0: F(x)不趋于显著地小于G(y)会被接受;否则,备择假设H1: F(x)趋于显著地小于G(y)就被接受,FL1(TF-IDF)在所有错误版本的*Exam*值趋于显著地小于FL1,即FL1（TF-IDF）的定位效能高于FL1(用Better表示).

Table 3 The best, average and variance of *Exam* score of TF-IDF over SFL

表3 基于TF-IDF的方法和SFL的最好*Exam*,平均*Exam*和*Exam*方差对比

		python	**gzip**	**libtiff**	**space**	**nanoxml _v1**	**nanoxml _v2**	**nanoxml _v3**	**nanoxml _v4**
ER1' (TF-IDF)	best	**0.04895**	**0.00338**	**0.04239**	**0.01619**	**0.03049**	**0.02260**	**0.01493**	**0.01852**
	average	**0.40150**	**0.08778**	**0.31594**	**0.60321**	**0.06707**	**0.04331**	**0.17936**	**0.35217**
	variance	0.40480	**0.09589**	0.35192	**0.26374**	**0.03168**	**0.03112**	**0.20934**	0.34688

ER1’	best	0.05008	0.00791	0.18812	0.03457	0.03068	0.28409	0.02985	0.06897
	average	0.40967	0.10061	0.44217	0.60890	0.32720	0.27083	0.38126	0.55126
	variance	**0.39562**	0.11939	**0.29987**	0.28954	0.49774	0.41498	0.23180	**0.31936**
GP02 (TF-IDF)	best	**0.10461**	**0.00338**	**0.04239**	0.08807	0.03049	0.02260	**0.01493**	**0.03241**
	average	0.57820	0.38448	**0.37312**	0.61519	0.36382	**0.21846**	0.39151	**0.54923**
	variance	**0.31786**	0.46971	0.41699	**0.28261**	**0.29181**	**0.33435**	0.33491	0.32546
GP02	best	0.11161	0.02606	0.19093	**0.03457**	**0.00614**	**0.00568**	0.02985	0.06897
	average	**0.44159**	**0.23090**	0.44643	**0.61102**	**0.32720**	0.26326	**0.38126**	0.55126
	variance	0.36465	**0.26134**	**0.30791**	0.28743	0.51935	0.42177	**0.23180**	**0.31936**
GP19 (TF-IDF)	best	**0.05008**	**0.00791**	**0.04239**	0.01948	**0.03049**	**0.02260**	**0.01493**	**0.01852**
	average	**0.40443**	**0.17705**	**0.33361**	0.52907	**0.07114**	**0.24670**	0.22010	**0.19031**
	variance	**0.38950**	**0.29108**	0.34426	**0.36126**	**0.03520**	0.34041	0.24701	0.18565
GP19	best	**0.05008**	**0.00791**	0.22665	**0.01564**	**0.03049**	0.07910	**0.01493**	0.09677
	average	0.42315	0.23541	0.45331	**0.52393**	**0.07114**	0.28437	**0.20585**	0.23673
	variance	0.43119	0.35418	**0.28787**	0.36566	**0.03520**	**0.30791**	**0.17894**	**0.14901**
ER5 (TF-IDF)	best	**0.05008**	**0.00791**	**0.04239**	**0.01975**	**0.03049**	**0.02260**	**0.01493**	**0.01852**
	average	**0.40587**	0.12532	**0.37312**	0.62266	**0.06707**	**0.04331**	**0.12602**	**0.26643**
	variance	0.40934	0.14485	0.41699	**0.27089**	**0.03168**	**0.03112**	**0.12162**	**0.21831**
ER5	best	**0.05008**	0.02702	0.23267	0.04033	0.12883	0.12068	0.07463	0.05172
	average	0.41545	**0.10010**	0.49282	**0.60484**	0.40491	0.34659	0.37967	0.53555
	variance	**0.39509**	**0.08305**	**0.35444**	0.28544	0.40113	0.29871	0.20702	0.31985
GP03 (TF-IDF)	best	**0.05355**	0.05375	**0.04239**	**0.01619**	0.13415	**0.02260**	**0.01493**	**0.01852**
	average	0.57223	0.42830	**0.55258**	**0.49606**	0.39024	**0.26365**	**0.34801**	**0.43795**
	variance	0.34659	0.39770	0.48527	0.29507	**0.28365**	**0.33335**	**0.28587**	**0.27952**
GP03	best	0.08459	**0.00791**	0.37637	0.03457	**0.03068**	0.02841	0.02985	0.06897
	average	**0.51524**	**0.24378**	0.71171	0.63429	**0.33538**	0.27083	0.37493	0.55126
	variance	**0.32120**	**0.26674**	**0.30939**	**0.27341**	0.51191	0.41498	0.22820	0.31936
Dstar (TF-IDF)	best	**0.05008**	**0.00791**	**0.04239**	**0.01646**	**0.03049**	**0.02260**	**0.01493**	**0.01852**
	average	**0.40587**	0.12532	**0.37312**	0.60953	**0.06707**	**0.04331**	**0.12602**	**0.26829**
	variance	0.40935	**0.14485**	0.41699	**0.26482**	**0.03168**	**0.03112**	**0.12162**	**0.21614**
Dstar	best	**0.05008**	**0.00791**	0.23267	0.02881	0.20859	0.10227	0.02985	0.05172
	average	0.41375	**0.10534**	0.49282	**0.59605**	0.46830	0.18371	0.49203	0.35786
	variance	**0.40049**	0.11496	**0.35444**	0.28473	0.44453	0.07961	0.36532	0.29766
Ochiai (TF-IDF)	best	**0.05008**	**0.02703**	**0.04239**	**0.33333**	0.09756	**0.02259**	**0.01492**	**0.01851**
	average	**0.33541**	**0.17939**	**0.37312**	**0.63127**	0.14837	0.14500	0.18360	0.34619
	variance	**0.31012**	**0.20895**	**0.41699**	0.18938	0.14837	0.10901	0.21176	**0.21500**
Ochiai	best	**0.05007**	**0.02702**	**0.04239**	0.42770	**0.02439**	**0.02259**	**0.01492**	**0.01851**
	average	0.34629	0.21024	**0.37312**	0.63360	**0.04674**	**0.06403**	**0.16687**	**0.31036**
	variance	0.31401	0.21468	**0.41699**	**0.15405**	**0.04674**	**0.06692**	**0.18173**	0.24267
Tarantula (TF-IDF)	best	**0.05007**	**0.02702**	**0.04239**	**0.02908**	**0.02439**	**0.02824**	**0.01932**	**0.09677**
	average	**0.40586**	**0.12532**	**0.37312**	0.52501	**0.06707**	**0.05084**	**0.12268**	**0.42345**
	variance	0.40934	0.14484	**0.41699**	0.32069	**0.06707**	**0.02589**	0.11506	**0.19641**

Tarantula	best	**0.05007**	**0.02702**	**0.04239**	0.32290	0.14024	0.14124	0.18357	**0.09677**
	average	0.41545	0.13774	**0.37312**	**0.44051**	0.22967	0.21280	0.28463	0.44210
	variance	**0.39509**	**0.13946**	**0.41699**	**0.22718**	0.22967	0.13385	**0.10917**	0.30340

表 4 展示了在真实错误(real faults)、植入错误(seeded faults)和所有错误(total)这三种情况下的 *Wilcoxon-Signed-Rank Test* 统计结果,给出具体的 p 值和对比结果.以 ER1'(TF-IDF)和 ER1'在真实错误(real faults)上的对比为例,其 2-tailed、1-tailed(right)和 1-tailed(left)的 p 值分别为 6.55e-03、8.14e-01 和 4.32e-02.这表明在真实错误上,运用 TF-IDF 方法后,ER1'的 *Exam* 值减小了.因此,在真实错误上,基于 TF-IDF 的定位方法改进了 ER1'的定位效果,结果为 Better.如表 4 所示,与八种定位方法(即 ER1'、GP02、GP03、GP19、ER5、Dstar、Ochiai 和 Tarantula)的对比结果可以看出:在真实错误、植入错误和所有错误三种情况上,基于 TF-IDF 的方法均取得 7 个 Better 和 1 个 Similar 的结果.同时,可以看出基于 TF-IDF 方法在不同的 SFL 公式中有不同的效能.究其原因是不同的 SFL 公式对四个参数 a_{np},a_{nf},a_{ep},a_{ef} 计算时的侧重不同,导致基于 TF-IDF 方法在对这四个参数重新计算后的侧重也存在不同,从而在不同的 SFL 公式上得到了不同的效能提升结果.

Table 4 *Wilcoxon-Signed-Rank Test* of the effectiveness relationship

表4 基于*Wilcoxon-Signed-Rank Test*的有效性对比

Comparison		2-tailed	1-tailed(right)	1-tailed(left)	Conclusion
ER1'(TF-IDF) vs ER1'	real faults	6.55e-03	8.14e-01	4.32e-02	Better
	seeded faults	1.09e-02	9.69e-01	9.07e-04	Better
	total	1.66e-02	9.18e-01	8.37e-03	Better
GP02(TF-IDF) vs GP02	real faults	9.44e-01	4.86e-01	5.42e-01	Similar
	seeded faults	5.00e-03	7.91e-01	2.95e-02	Better
	total	9.90e-03	5.07e-01	4.97e-02	Better
GP03(TF-IDF) vs GP03	real faults	3.17e-02	9.77e-01	4.03e-02	Better
	seeded faults	2.47e-02	9.85e-01	4.16e-03	Better
	total	2.88e-02	8.57e-01	1.45e-02	Better
GP19(TF-IDF) vs GP19	real faults	5.11e-03	7.91e-01	2.34e-02	Better
	seeded faults	7.47e-03	9.70e-01	4.67e-03	Better
	total	1.40e-02	9.32e-01	7.14e-03	Better
ER5(TF-IDF) vs ER5	real faults	2.47e-02	9.85e-01	4.16e-03	Better
	seeded faults	1.80e-02	9.63e-01	1.86e-02	Better
	total	1.95e-02	9.90e-01	9.87e-03	Better
Dstar(TF-IDF) vs Dstar	real faults	3.45e-02	8.60e-01	2.09e-02	Better
	seeded faults	1.09e-02	9.69e-01	9.07e-02	Better
	total	1.88e-02	9.91e-01	9.54e-03	Better
Ochiai(TF-IDF) vs Ochiai	real faults	6.55e-03	8.14e-01	5.00e-03	Better
	seeded faults	9.44e-01	4.86e-01	5.42e-01	Similar
	total	9.75e-01	5.00e-01	5.25e-01	Similar
Tarantula(TF-IDF) vs Tarantula	real faults	6.55e-03	8.14e-01	5.00e-03	Better
	seeded faults	2.47e-02	9.85e-01	4.16e-03	Better
	total	4.68e-03	9.98e-01	2.45e-03	Better

由以上实验结果可以看出,基于 TF-IDF 的定位方法提升了程序缺陷的定位效能.基于二进制状态信息仅能表示执行和未执行两种状态,无法提供更多的语义信息.与二进制状态信息相比,基于 TF-IDF 的定位方法利用

词频（TF）获取语句在单个测试用例执行情况和利用逆文本频率指数（IDF）获取语句在整个测试用例集执行情况,最后综合局部和全局两个维度的信息,形成更丰富的语句和测试用例的关系信息,能识别更多二进制状态信息无法发现的差异性影响,从而提升定位效能.以 gzip 程序中错误版本 v3 为例.其中,第 405 行为非缺陷语句.该语句被所有的测试用例覆盖,基于二进制状态信息的矩阵无法体现出第 405 行语句对测试用例结果的差异性影响.而基于 TF-IDF 的方法能将第 405 行对测试用例结果的影响降为 0,有效防止了第 405 行对缺陷定位结果的干扰.因此,基于 TF-IDF 的方法融合了更丰富的有效信息到信息覆盖矩阵中,更准确地反映出语句和测试用例之间的关系,从而解决了信息表达能力弱对定位精度上限的限制问题,提升了缺陷定位效能.

(3)有效性威胁

本文实验有如下有效性威胁:

本实验的前提是失败的测试用例运行的时候含有缺陷的语句是被执行的,但是当程序存在多个缺陷并且这些缺陷是在一个传播链上进行传播的时候这个假设并不成立,比如多个缺陷中有的缺陷是在其他缺陷激活时才会发生的情况.但是本实验进行的单错误的研究是多错误研究的基础,因此这也是当前的研究大都基于单错误的原因.

其次,本实验选用的实验对象只是广泛应用于缺陷定位研究的典型程序,现实世界中的程序多种多样,这些实验程序并不能覆盖现实中的各种类型的错误,也不能预测到现实程序调试中的许多未知因素.但是本实验选用的程序是错误定位领域具有代表性的实验程序,具有说服力.同时在将来的工作中,我们将关注更多的现实工程中的程序并在其基础上进行实验.

4 相关工作

基于覆盖信息的错误定位技术研究是当前研究的热点,本节将对这些工作进行简要的介绍,更多的关于各类错误定位的工作可以参考综述文章[1].

基于程序覆盖信息的缺陷定位技术的基本思想为首先对大批量的测试用例在目标程序上的运行信息进行收集,之后根据这些信息试图寻找与程序错误输出相关联的位置信息,即程序实体为缺陷的可疑值,或者根据这些信息提取出某种统计模型,以此模型作为参照,通过对比实际模型与参照模型的差异来定位缺陷.基于频谱的错误定位技术（Spectrum-based Fault Localization,SFL）[16],是最典型的基于统计的错误定位技术.SFL 具有简单高效的特点,也是当前应用最广泛和影响力最大的错误定位技术.使用 SFL 技术的时候不需要了解程序内部的逻辑结构和运行机制,只需要根据成功和失败的测试用例的运行情况就能成功定位程序的错误所在. SFL 技术中有许多代表性的工作,比如, Jaccard 技术和 Tarantula 技术,他们分别是由 Chen 等人[17]和 Jones 等人[18]提出的,其中 Tarantula 是在后续研究中被广泛使用和对比的技术. 之后 Abreu 等人[19]提出了 Ochiai 技术,在与 Tarantula 进行的实验对比中 *EXAM* 值平均提高了 5%.后来他们又对 Tarantula 等技术进行了有效性的评估并应用 Ochiai 评估了测试用例[20].Wong 等[21] [22]在研究了程序的数据和控制流程的基础上,提出了 Wong1-3、Wong3'和一些缩减搜索域的方法.Wong 等[23,24]还提出当前最优的 SFL 技术 Dstar(D*),它是一种基于交叉表的方法,本文实验和 Dstar 进行了对比.Lee[7]通过赋予执行次数 FC（Frequency Execution Count,FC）高的语句以更高的可疑值权重来进行错误定位.谢等人[10]理论上研究了 GP 进化公式.谭[25]提出通过增大边际权重提高基于频谱的错误定位方法.很多新技术也基于 SFL 来应对各种错误定位领域的问题.为了在多错误条件下实施有效错误定位,Dean 等[26]在 SFL 的基础上引入了线性方法,提升了定位多缺陷错误的能力.Jones 等[27]运用聚类技术将由同一个缺陷引发的程序错误的失败测试用例归为一类,之后对每一个类别中的单缺陷再进行缺陷定位,但是此种方法对互相影响和传播的错误没有办法实现很好的定位.Abreu 等[28][29]通过将 SFL 技术与基于模型的诊断技术进行集成,以此提出基于频谱的多错误定位方法 Zoltar-M,实现了多错误的有效定位.Abreu 等[30]引入不变式将原有测试结果失败与否的判断,转变为程序状态错误与否的判断,从而缓解“预言家难题”.Xie X 等[31][32]采用

蜕变测试重新定义了测试结果,解决SFL需要测试预言的问题.本文方法在上述SFL方法的基础上,采用TF-IDF优化覆盖信息矩阵,提高了错误定位效能.

偶然正确性（Coincidental Correctness）问题是指出现执行了错误语句的成功测试用例的情况.偶然正确性问题会对错误定位性能产生不利的影响,如何识别出这种执行力错误语句的成功测试用例是解决偶然正确性问题的关键所在.Wang 等人[33]提出使用匹配上下文模型（Context Pattern）的方法来识别偶然正确性问题.Gopinath 等[34]在 SFL 方法中融入了规约分析并提升了错误定位的性能.Masri 等[35] [36]定义了描述具有偶然正确性问题的成功测试用例的方法,并度量出了成功测试用例和失败测试用例之间的相似度,这个相似度是基于Euclidean 标准的.Miao 等[37]采用聚类的思想对不同类型的测试用例进行分类,从而有助于发现.Bandyopadhyay[38]对具有偶然正确性问题的成功测试用例进行预测并根据预测结果提升错误定位的有效性.Lei 等人[39]系统研究了测试用例集的效能并提出了偶然正确性问题是影响错误定位效能最大的因素.本文方法关注程序覆盖信息矩阵的重塑和优化.本文方法不关注优化具有偶然正确性问题的成功测试用例,且采用的TDIDF技术不能根除偶然正确性,然而,本文方法对信息矩阵元素的重新赋值,抑制了具有偶然正确性问题的成功测试用例的负效应,从而间接地缓解了偶然正确性问题.

此外,TF-IDF 技术也被应用到缺陷查找等工作.Saha 等[41]提出 BLUiR,利用错误报告的错误相似数据进行缺陷查找.Wang 等[43]提出用于查找相关错误文件的方法 AmaLgam+.Wang 等[43]还提出了 $VSM_{composite}$ 方法,通过分析 AspectJ、Eclipse 和 SWT 的错误报告来进行相关的关键报告的检索.实验中 $VSM_{composite}$ 方法和基于TF-IDF 技术的 $VSM_{natural}$ 方法进行了比较.这些方法主要使用 TF-IDF 技术来处理错误报告.本文方法与这些方法不同:本文方法将 TF-IDF 技术用于信息覆盖矩阵的重构,从而构建出信息表达能力更强的信息模型,以此提升错误定位效能.

5 结论

本文提出一种基于词频-逆文档频率的错误定位方法.该方法以 TF-IDF 技术识别出语句执行的影响程度,重新构建具有影响识别度的信息模型,并结合 SFL 的可疑性度量方法,构建语句的可疑值排序.实验结果表明,与五种典型 SFL 定位方法对比,基于 TF-IDF 的方法能显著缩减了代码检查的范围,大幅提升了错误定位性能.未来工作包括方法优化和多错误定位的应用,并进一步提高其准确性.

References:

[1] Wong WE, Gao R, Li Y, Rui A, "A survey on software fault localization," in IEEE Transactions on Software Engineering, vol.42, pp.1-1, 2016.

[2] X.Xie, F.-C.Kuo, T.Y.Chen, S.Yoo, and M.Harman, "Provably Optimal and Human-Competitive Results in SBSE for Spectrum Based Fault Localisation," in Proceedings of the 5th Symposium on Search-Based Software Engineering, St.Petersburg, Russia, 2013, pp.224-238.

[3] M.Acharya and B.Robinson."Practical Change Impact Analysis Based on Static Program Slicing for Industrial Software Systems." in International Conference on Software Engineering, vol.111, pp.746–765, 2011.

[4] Pearson S, Campos J, Just R, et al. "Evaluating and Improving Fault Localization," in International Conference on Software Engineering. Buenos Aires, Argentina, 2017, pp.609-620.

[5] Yoo, S.(2012),"Evolving human competitive spectra-based fault localisation techniques," in Proceedings of the 4th International Symposium on Search-Based Software Engineering, Italy, 2012, pp.244-258.

[6] 张卓,谭庆平,毛晓光,雷晏,常曦,薛建新.增强上下文的错误定位技术.软件学报,2019,30(2):266-281

[7] Lee H J, Naish L, Ramamohanarao K, "Effective software bug localization using spectral frequency weighting function," in Proceedings of the 34th Annual Computer Software and Applications Conference (COMPSAC), Seoul, Korea, 2010, pp.218–227.

[8] A. Rajaraman and J.D. Ullman, "Mining of massive datasets, " CambridgUniversity Press, 2011.

[9] Lee naish, Hua jie Lee, and Kotagiriramamohanarao, University of Melbourne, "A Model for Spectra-Based Software Diagnosis," in ACM Transactions on Software Engineering and Methodology, vol.20, pp.11, 2011.

[10] X.Xie, T.Y.Chen, F.-C.Kuo, B.XU, "A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization," in ACM Transactions on Software Engineering and Methodology , vol.22, pp.31, 2013.

[11] ManyBugs, http://repairbenchmarks.cs.umass.edu/manybugs/.

[12] SIR, http://sir.unl.edu/portal/index.php.

[13] X. Mao, Y. Lei, Z. Dai, Y. Qi, and C. Wang, "Slice-based statisticalfault localization," in Journal of Systems and Software, vol. 89, no. 1, pp.51 – 62, 2014.

[14] G. W. Corder and D. I. Foreman, "Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach," in International Statistical Review,2010, vol. 78.

[15] Y.Lei, X.Mao, Z.Dai, C.Wang, "Effective statistical fault localization using program slices," in Proceedings of the 36th Annual International Computer Software and Applications Conference, Izmir, Turkey, 2012, pp.1-10.

[16] Abreu R, Zoeteweij P, van Gemund A, "Spectrum-based multiple fault localization," in Proceedings of the 24th International Conference on Automated Software Engineering (ASE), Auckland, New Zealand, 2009, pp. 88-99.

[17] M.Chen, E.Kiciman, E.Fratkin, A.Fox, E.Brewer, "Pinpoint: problem determination in large, dynamic internet services," in Proceedings of International Conference on Dependable Systems and Networks, Bethesda, MD, USA, 2002, pp.595 – 604.

[18] J.A.Jones, "Fault localization using visualization of test information," in Proceedings of the 26th International Conference on Software Engineering, Edinburgh, Scotland, UK, 2004, pp.54–56.

[19] R.Abreu, P.Zoeteweij, A.J.C.van Gemund, "An evaluation of similarity coeffcients for software fault localization," in Proceedings of the 12th Pacific Rim International Symposium on Dependable Computing, Riverside, USA, 2006, pp.39–46.

[20] R.Abreu, P.Zoeteweij, R.Golsteijn, A.J.C.van Gemund, "A practical evaluation of spectrum-based fault localization," in Journal of Systems and Software, vol.82, pp.1780–1792, 2009.

[21] W.E.Wong, Y.Qi, L.Zhao, K.Y.Cai, "Effective fault localization using code coverage," in the 31st Annual International Computer Software and Applications Conference, Beijing, China, 2007, pp.449-456.

[22] W.E.Wong, V.Debroy, B.Choi, "A family of code coverage-based heuristics for effective fault localization," in Journal of Systems and Software, vol.83, pp.188–208, 2010.

[23] W.E.Wong, T.Wei, Y.Qi, L.Zhao, "A crosstab-based statistical method for effective fault localization," in Proceedings of the 1st International Conference on Software Testing, Verification and Validation, Lillehammer, Norway, 2008, pp.42–51.

[24] W.E.Wong, V.Debroy, Y.H.Li, R.Z.Gao, "Software fault localization using dstar (d*)," in Proceedings of the 6th IEEE International Conference on Software Security and Reliability, Gaithersburg, Maryland, USA, 2012, pp.21–30.

[25] 谭德贵, 陈林, 王子元, 丁晖, 周毓明, 徐宝文, "通过增大边际权重提高基于频谱的错误定位效率," 计算机学报, vol.33, pp. 2335-2342, 2010.

[26] Dean B C, Pressly W B, Malloy B A, Whitley A A, "A Linear Programming Approach for Automated Localization of Multiple Faults," in Proceedings of the 24th International Conference on Automated Software Engineering (ASE), Auckland, New Zealand, 2009, pp.640-644.

[27] Jones J A, Bowring J F, Harrold M J, "Debugging in Parallel," in Proceedings of the International Symposium on Software Testing and Analysis (ISSTA), London, United Kingdom, 2007, pp.16-26.

[28] Abreu, R, Zoeteweij, P, van Gemund, A, "Localizing software faults simultaneously," in Proceedings of the 9th International Conference on Quality of Software (QSIC), Jeju, Korea, 2009, pp.367-376.

[29] Abreu R, Zoeteweij P, van Gemund A, "Simultaneous debugging of software faults," in Journal of Systems and Software, vol.84, pp. 573-586, 2010.

[30] Abreu R, Gonzalez A, Zoeteweij P, van Gemund A, "Automatic Software Fault Localization Using Generic Program Invariants ," in Proceedings of the ACM Symposium on Applied Computing (SAC), Fortaleza, Ceara, Brazil, 2008, pp.712-717.

[31] Xie X, Wong W E, Chen T Y, Xu B, "Metamorphic Slice: An Application in Spectrum-Based Fault Localization ," in Information and Software Technology, vol.55, pp. 66–879, 2013.

[32] Xie X, Wong W E, Chen T Y, Xu B, "Spectrum-Based Fault Localization: Testing Oracles are No Longer Mandatory," in Proceedings of the 11th International Conference On Quality Software (QSIC), Madrid, Spain, 2011, pp.1-10.

[33] Wang X, Cheung S C, Chan W K, "Taming coincidental correctness: Coverage refinement with context patterns to improve fault localization," in International Conference on Software Engineering, Vancouver, Canada, 2009, pp.45-55.

[34] Gopinath D, Zaeem R N, Khurshid S, "Improving the effectiveness of spectra-based fault localization using specifications," in Proceedings of the 27th International Conference on Automated Software Engineering (ASE), Essen, Germany, 2012, pp.40–49.

[35] Masri, W, Assi, R.A, "Cleansing test suites from coincidental correctness to enhance fault-localization," in Software Testing, Verification and Validation, 2010, pp. 165–174.

[36] Masri, W, Assi, R.A, "Prevalence of coincidental correctness and mitigation of its impact on fault localization," in ACM Transactions on Software Engineering and Methodology, vol.23, pp. 8, 2014.

[37] Yi Miao, Zhenyu Chen, Sihan Li, Zhihong Zhao, Yuming Zhou, "Identifying Coincidental Correctness for Fault Localization by Clustering Test Cases," in Proceedings of the 24th International Conference on Software Engineering and Knowledge Engineering (SEKE 2012), Redwood City, San Francisco, 2012, pp.267-272.

[38] Aritra Bandyopadhyay, "Mitigating the Effect of Coincidental Correctness in Spectrum Based Fault Localization," in Proceedings of the 5th International Conference on Software Testing, Verification and Validation, Kolkata, India, 2012, pp. 479-482.

[39] Yan Lei, Chengnian Sun, Xiaoguang Mao, Zhendong Su, "How test suites impact fault localisation starting from the size," in IET Software, vol.12, pp.190-205, 2018.

[40] Y. Lei, X. Mao, Z. Min, J. Ren, and Y. Jiang, “Toward understanding information models of fault localization: Elaborate is not always better,” in Proceedings of the 41st Annual Computer Software and Applications Conference (COMPSAC), pp.57–66, 2017.

[41] Saha, R.K., Lease, M., Khurshid, S. and Perry, D.E, “Improving bug localization using structured information retrieval,” in 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 345-355, 2013.

[42] Wang, S. and Lo, D, “Amalgam+: Composing rich information sources for accurate bug localization,” in Journal of Software: Evolution and Process, vol.28(10), pp.921-942, 2016.

[43] Wang, S., Lo, D. and Lawall, J, “Compositional vector space models for improved bug localization,” in 2014 IEEE International Conference on Software Maintenance and Evolution, pp. 171-180, 2014.