计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(12):26−34 [doi: 10.15888/j.cnki.csa.007687]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

面向用户的软件缺陷报告特征重要度分析①

李文华[1], 李浩杰[2]

[1](青岛科技大学 信息科学技术学院, 青岛 266061)
[2](青岛科技大学 高密校区机电系, 潍坊 261500)
通讯作者: 李浩杰, E-mail: 1010803363@qq.com

摘　要: 缺陷的及时反馈与修复是开源软件长久健康发展的基本保障, 面对每天提交的大量、包含多方面因素的开源软件缺陷报告, 很多缺陷报告因为描述遗漏或不准确, 而影响缺陷有效判定和缺陷修复进程; 而面对需要填报的复杂缺陷报告信息, 报告提交者也无法确定哪些属性是关键的、哪些需要重点描述致使提交缺陷报告质量不高. 综合文献对缺陷报告描述的 5 个维度 28 种特征进行分析, 按照维度间和维度内多特征两个角度对缺陷报告特征进行重要度排序对比, 结果显示按照维度分析: 文本特征和报告者经验特征两个维度的 *F*1 值和准确率指标最高, 而每个维度内特征也显示了不同重要度, 从而可以指导缺陷提交者规范缺陷报告提交, 也可以指导修复者进行缺陷判定和缺陷修复.

关键词: 缺陷报告; 特征提取; 特征排序; 特征推荐; 缺陷修复

引用格式: 李文华,李浩杰.面向用户的软件缺陷报告特征重要度分析.计算机系统应用,2020,29(12):26–34. http://www.c-s-a.org.cn/1003-3254/7687.html

Importance Analysis of User Oriented Software Defect Reporting Features

LI Wen-Hua[1], LI Hao-Jie[2]

[1](College of Information Science and Technology, Qingdao University of Science and Technology, Qingdao 266061, China)
[2](Department of Mechatronics, Qingdao University of Science and Technology, Weifang 261500, China)

Abstract: The timely feedback and repair of bug is the basic guarantee for the long-term and healthy development of open source software. Facing a large number of open source software bug reports submitted every day, including many factors, many bug reports affect the effective judgment and repair process of defects due to missing or inaccurate descriptions. For the complex bug report information that needs to be filled in, the report submitter is also impossible to determine which attributes are critical and which need to be highlighted, which results in poor quality of submitted bug report. Based on the analysis of 28 features of 5 dimensions described in bug report in the comprehensive literature, the importance ranking and comparison of bug report features are carried out from two perspectives of inter dimension and multi feature within dimension. The results show that the *F*1 value and accuracy index of two dimensions, text feature and bug reporter experience feature, are the highest, and the features within each dimension also show different importance, therefore, it can guide the bug reporter to standardize the submission of bug report, or guide the repairer to determine and repair defects.

Key words: bug report; feature extraction; feature ranking; feature recommendation; bug repair

① 基金项目: 国家自然科学基金 (61973180); 山东省重点研发计划 (2018GGX101052)
Foundation item: National Natural Science Foundation of China (61973180); Key Research and Development Program of Shandong Province (2018GGX101052)
收稿时间: 2020-04-07; 修改时间: 2020-05-13; 采用时间: 2020-05-21; csa 在线出版时间: 2020-11-30

1 前言

软件发展的过程中, 在软件工程领域软件的开源和软件系统的不断兴起和发展使得软件的开发与软件管理变得更加方便快捷. 开源软件的开发管理团队通常依靠开放性的缺陷跟踪管理系统, 对软件使用过程中产生的缺陷进行管理与跟踪, 如: Bugzilla、Jira 等均为众人比较熟知的开放性缺陷报告提交平台. 据统计, 自 2007 年起, Eclipse 缺陷报告库中新的缺陷报告以每天 30 个的速度递增[1]; 2001 年至 2010 年, Mozilla 的缺陷报告库中的缺陷报告以平均每天 307 个的速度增加[2]. 分析显示, 大量的缺陷报告因为描述遗漏或不准确, 而影响缺陷有效判定和缺陷修复进程, 致使缺陷修复时间一般都较长[3], 也存在因描述不准确而长时间得不到修复的缺陷报告. 而面对需要填报的复杂缺陷报告信息, 缺陷报告发布者也无法确定哪些属性是关键的、哪些需要重点描述. 因此, 能够区分缺陷报告的属性的重要度, 不仅能够指导报告者规范缺陷报告的提交信息从而提升缺陷报告质量, 也能够辅助开发者提高缺陷有效性判定和修复的效率.

Zhang 等[4] 将修复过程分解为 5 个阶段, 发现缺陷提交后的确认阶段、缺陷分派后到开始修复的分析阶段耗时较长. Saha 等[5] 对 4 个开源项目进行实例研究, 发现影响缺陷修复时间的因素是复杂的、多方面的, 缺陷报告质量不高是主要因素, 同时, 调查发现由于缺陷报告描述不准确, 40% 的长时间未修复的缺陷可能只修改几行代码或修改 1 个文件. Karim 等[6] 研究高影响力缺陷 (HIB) 修复问题, 研究发现缺陷报告质量不高是影响缺陷时间修复的主要因素, 同时也发现测试用例, 代码示例, 重现步骤和堆栈跟踪是代码修复的重要因素. Tantithamthavorn 等[7] 研究了缺陷建模的 pitfalls and challenges 指出缺陷预测模型实际应用的最大问题是缺乏高质量的数据, 不能提取预测模型所需要的高质量缺陷特征. Fan 等[8] 全面提取报告提交者、修复者、社交网络、缺陷报告描述文本、缺陷报告描述可读性等特征, 采用 5 种机器学习方法进行有效性预测, 并通过特征的是否参与发现文本类特征是较重要的特征, 但其并没有给出不同特征的重要度排序, 也没有对每一维度下的特征进行重要分析.

基于上述的问题, 本文在文献 [9–12] 等提出的缺陷报告特征描述的基础上, 综合得到 5 个维度 28 个特征, 给出面向维度和维度内特征的重要度排序方法. 首先, 基于不同维度, 采用 5 种不同的机器学习分类器进行有效性判定学习和训练, 根据模型预测的准确性对每个维度的特征重要性进行排序; 其次采用 XGBoost[13] 和 GBDT[14] 两个特征选择常见的方法, 对每个维度内的特征进行排序. 将排序结果推荐给缺陷报告提交者, 使其能够在提交缺陷报告时, 更加注重重要度较高的属性描述, 提升缺陷报告的质量; 将排序结果推荐给开发者, 可以辅助开发者更准确的识别缺陷, 实现缩短确认和修复时间的目的.

本文第 2 部分综合历史文献给出常见的缺陷描述特征; 第 3 部分给出特征重要度选择方法, 并给出实验的设计方案; 第 4 部分给出实验结果, 最后给出本文的总结.

2 缺陷报告特征提取

本部分综述目前文献在进行缺陷有效性判定时提取的特征, 共分为 5 个维度, 并分析每一维度的特征含义及包含的属性信息.

2.1 发布者经验

在 Just 等[15] 研究表明在软件修复缺陷过程中缺陷报告有效性和缺陷发布者的经验是相关的. 在后续的研究中他们发现有经验的开发者或缺陷报告发布者发布的软件缺陷报告可能被修复的概率更高.

发布者经验的维度上, Guo 等的工作提出[9], 发布者的经验是定义了发布者的能力的一个关键特征. 在某一个领域的经验越高, 代表着他对这个领域了解的更多. 相应的, 也代表在对应领域的经验也相对的丰富. 本文定义发布者经验包括 3 个属性: 报告者的报告数量 (bug-num)、缺陷报告的有效率 (validrate) 和最近缺陷报告数量 (recent-bug-num) 3 种特征. 缺陷报告的数量有效率决定了报告者的基础经验, 理论上缺陷报告越多, 也就代表着发布者的潜在基础经验越高. 但是发布者有可能在后续没有产生相应的缺陷报告, 而且缺陷报告的数量也不能代表着缺陷报告的有效性越好. 所以提出了最近的报告数量和有效率. 最近的报告数量决定了报告者的活跃度, 代表了开发者在时间段内保持着一定的活跃度和粘性. 有效率是指有效缺陷报告和缺陷报告总数的比率, 其中报告者的活跃度使用了 120 天内的缺陷报告发布数量. 发布者经验维度特征如表 1 所示.

表 1 发布者经验维度特征

特征名称	编号
报告数量	RE1
最近报告数量	RE2
报告有效率	RE3

2.2 完整性

缺陷报告定义了软件缺陷的所有信息, 如图 1. 一个完整的缺陷报告应当要求包含对应的信息. 缺陷报告包含对整个缺陷报告的总结、缺陷的基本内容包括平台和基本信息、缺陷发生时的附件或者截图、缺陷的详细描述以及后续补充评论的内容等, 这些内容构成了一个完整的缺陷报告. 开发者会经常利用重现步骤、栈轨迹、代码样例、测试样例和截屏等信息处理和修复缺陷. 缺陷报告中缺少这些信息是一个严重的问题, 这些问题在缺陷修复过程中也经常遇到. 在实验中, 我们用在缺陷报告的描述中有无这些专有信息作为报告有效性的一种推断. 这些专有信息包括栈轨迹 (stack trace)、重现步骤 (steps to reproduce)、代码样例 (code samples)、补丁 (patches)、测试样例 (test case) 和截屏 (screenshots).

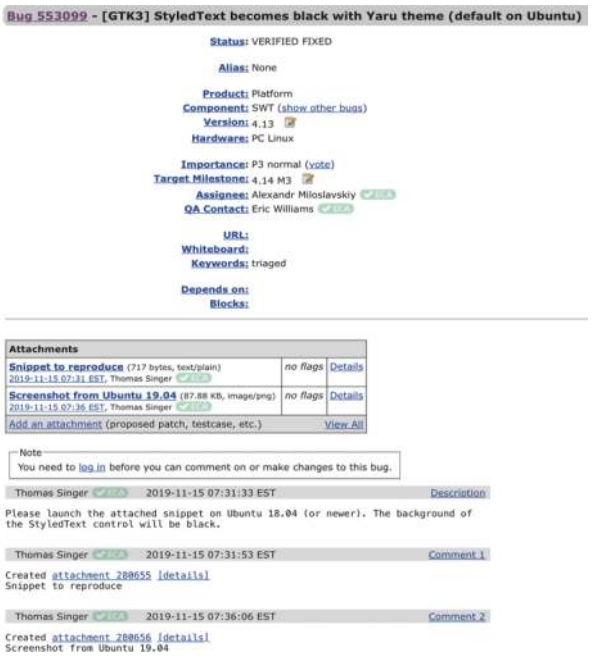

图 1 缺陷报告样例 553 099

文献 [10,16] 的工作介绍了总结了缺陷报告的结构和内容而且研究提出了缺陷报告最重要的阶段. 缺陷报告的完整度越高, 代表着缺陷报告修复的可能性也越高. 所以缺陷报告的完整度也是缺陷报告是否能够尽快修复的一个关键特征.

Bettenburg 等[10] 在缺陷报告的研究中使用了多个关键的正则表达从缺陷报告中提取出的关键信息, 包括缺陷重现 (steps to reproduce)、栈轨迹 (stack trace)、测试样例 (test case)、代码 (code samples)、附件 (patches) 和截图 (screenshots), 有无此类信息关系到缺陷报告的整个完整性. 在排除掉信息有效的情况下, 缺陷报告包含的内容越充分, 代表着缺陷报告的完整度越高. 所以在完整性维度上, 本文使用了上述 6 种特征代表了缺陷报告的完整度特征.

定义了缺陷报告的所有关键特征中包括是否包含步骤, 截图, 代码, 补丁, 测试用例和截图等. 每个特征取 1 或 0, 分别代表缺陷报告中有无对应相应的信息. 完整性维度特征如表 2 所示.

表 2 完整性维度特征

特征名称	编号	特征名称	编号
堆栈	C1	补丁	C4
步骤	C2	测试用例	C5
代码	C3	截图	C6

2.3 可读性

缺陷报告的描述和内容会对整个缺陷的阅读理解产生影响, 缺陷报告者在描述缺陷上需要清晰明了, 这其中包括缺陷描述的理解程度和的文本阅读的难度[17].

Zimmermann 等的研究认为可读性是影响缺陷报告的重要因素, 缺陷报告的可读性也与缺陷报告的有效性成正比, 缺陷报告的可读性影响体现在文本阅读难度和理解难度[11].

要计算可读性的特征需要对文本中特殊单词进行定义, 对于文本中的特殊单词可以分为复杂单词 (complex words)、长单词 (long words) 和期 (period) 多音节词 (polysyllables) 这几个概念进行定义.

大部分较为复杂的单词其中通常包含 3 个或更多音节, 所以定义由 3 个或更多音节组成的单词称为复杂单词, 其中不包括专有名词、行业用语和地方词. 如果一个单词中存在超过 6 个字符那么可以定义为长单词. 定义 period 可以从单词是否包含像句号 (.)、冒号 (:) 或大写的首字母. 多音节词是从分析提取的文本中组合获得, 由文本的开头、中间和结尾 3 部分分别取 10 个单词组成 3 组句子, 则这三组句子中有 3 个或更多音节的单词称为多音节词.

文本中抽取的所有概念包括字符数 (*Characters*)、单词 (*Words*)、音节 (*Syllables*)、句子 (*Sentence*)、复杂单词 (*ComplexWords*)、长单词 (*LongWords*)、期 (*Period*) 和多音节词 (*Polysyllables*).

$$fog = 0.4\frac{Words}{Sentences} + 40\frac{ComplexWords}{Words} \quad (1)$$

$$lix = \frac{Words}{Periods} + 100\frac{LongWords}{Words} \quad (2)$$

$$flesch = 206.835 - 1.015\frac{Words}{Sentences} - 84.6\frac{Syllables}{words} \quad (3)$$

$$kincaid = 0.39\frac{Words}{Sentences} + 11.8\frac{Syllables}{Words} - 15.59 \quad (4)$$

$$ari = 4.71\frac{Characters}{Words} + 0.5\frac{Words}{Sentences} - 21.43 \quad (5)$$

$$coleman - liau = 5.88\frac{Characters}{Words} + 29.6\frac{Sentences}{Words} \quad (6)$$

$$smog = 3 + \sqrt{Polysyllables} \quad (7)$$

可读性计算最初应用计算文本的理解程度, 以便推荐确定是否与人的理解能力相匹配. 以上的 7 个不同的可读性计算公式可以涵盖所有文本的可读性特征包括: *flesch* 是通过计算元音和音节数来定义可读性的方法[18]; *lix*[19] 最初改进 *flesch* 用在文本的困难度上的测试; *fog*[20] 常用来衡量音节对可读性的影响; *smog*[21] 则常被用来衡量多音节词对可读性的影响. *kincaid*[22]、*ari*[23] 以及 *coleman-liau*[24] 都是可读性的计算, 但是与大多数其他不同, 不需要分析单词的字符内容, 只需要分析单词的字符长度, 只依赖于字符而不是每个单词的音节.

通过 NLTK 中的 readability 包, 最后可以得到 7 个不同的可读性特征. 通过这 7 类特征, 可以统计出每个缺陷报告的可读性特征分布. 通过不同的缺陷报告的可读性特征不同, 分析不同缺陷报告可读性特征的重要度. 可读性维度特征如表 3 所示.

表 3 可读性维度特征

特征名称	编号	特征名称	编号
flesch	R1	*ari*	R5
fog	R2	*coleman-liau*	R6
lix	R3	*smog*	R7
kincaid	R4		

2.4 社交网络

在缺陷报告中, 报告发布者与开发者存在密切的问答关系, 而缺陷报告与开发者存在着分派关系. 缺陷报告的问答关系可以提取出人员的社交网络关系, 问答关系涉及不同的参与人员, 直接体现出不同参与人员的知识水平和社交活跃度. 缺陷报告中的每一条评论都包含有参与人员的信息和回复目标, 本文通过正则表达式抽取出用户和回复关系, 通过参与人员的回复关系建立节点和映射, 进而转换为所有参与人员的社交网络关系.

Zanetti 等[12] 都对缺陷报告的社交网络分析中使用了 9 种方法, 得出社交网络的特征关系表现为 9 个相关特征. 在缺陷报告中提取 9 个相关的特征代表了报告发布者在社交网络中的嵌入度, 而且具有很好的效果. 本文使用了他们的工作在社交网络的特征方面选取了同样的特征处理. 社交网络维度特征如表 4.

表 4 社交网络维度特征

特征名称	编号	特征名称	编号
lcc-membership	CN1	k-coreness	CN6
in-degree	CN2	closeness-centrality	CN7
out-degree	CN3	betweenness-centrality	CN8
total-degree	CN4	eigenvector-centrality	CN9
clustering-coefficient	CN5		

这 9 种特征分别是最大连同分量 (lcc-membership)、入度 (in-degree)、出度 (out-degree)、总度 (total-degree)、聚类系数 (clustering-coefficient)、k-核系数 (k-coreness)、接近中心性 (closeness-centrality)、中介中心性 (betweenness-centrality) 和特征向量中心性 (eigenvector-centrality).

最大连同分量 (lcc-membership) 表示缺陷报告发布者所在节点在社交网络中是否属于最大连通分量. 入度 (in-degree)、出度 (out-degree) 和总度 (total-degree) 这 3 个特征衡量了发布者到其他发布者和开发者的连接数量. 入度表示通过发布者节点的输入边与之连接的其他节点数量, 出度表示通过发布者节点的输出边与之连接的其他节点数量, 总度为入度与出度的和, 即与发布者节点相连接的节点数量. 入度、出度和总度的特征值越大意味着发布者在社交网络中越活跃.

社交网络分解的出 k 核系数 (k-coreness); 局部聚类系数能够反映单个节点的在网络中的性质; 全局聚类系数能够反映出整个网络的特性; 聚类系数 (clustering-coefficient) 用来表示网络中节点聚集程度的系数. 例如发布者对应节点的聚类系数越高, 说明发布者和他的临近节点在协作活动中的联系越密切; 临

近中心性 (closeness-centrality) 体现一个节点到所有其它节点的难易程度: 节点的介数中心性 (betweenness-centrality) 是指网络中包含此节点的最短路径的数量占所有最短路径数量的百分比. 某个节点的介数中心性越高, 说明此节点对社交网络的控制力越强, 因为很多信息会从此节点经过; 特征向量中心性 (eigenvector-centrality) 也是社交网络的重要特征. 社交网络中存在这样一种概念, 连接到中心性高的节点会增加节点本身的中心性.

在缺陷报告中有明显的缺陷分配的关系和问答关系. 分配关系仅体现了报告发布者与开发者之间的联系, 其中缺陷报告中只有一条最终的分配关系. 而缺陷报告中的问答关系却涉及到多名开发者和参与者, 存在的交互关系更为复杂. 因此, 问答关系网络中会包含更多的信息, 能更有效和实际地体现每个节点在网络中的权重关系. 在实验中, 本文建立了基于问答关系的社交网络图.

2.5 文本

缺陷报告中内容的文本挖掘可以帮助缺陷报告的分类, 以往对缺陷报告描述内容的研究大多使用了文本挖掘进行分类[25].

文本表示的特征的工作由早期的词映射模型表示到神经网络模型经历了多种变迁, 其中经典的模型有 Word2Vec[26] 和 Glove[27]. 其方法是把文本中的单词转化为向量, 然后在相近的单词余弦相似度较小的原理来做分类, 早期词向量的模型生成的语义与上下文无关, 无法更好、更完整的表达文本语义的信息.

Fan 等[8] 使用了文本内容的数据集, 分别放入分类器中进行分类和训练. 所有的训练集训练出一个分类器, 将此分类器用于对测试集的分类, 用分类器预测测试集中每篇缺陷报告文本特征向量有效的概率, 将此概率作为测试集的一个文本度量. 这种方法使用的基于数据集的学习分类, 使得文本的有效性可能会高于原本文本的度量值. 基于数据集的分类器很大程度上会依赖于数据集本身的结果, 使文本的度量值可能偏高.

在 2018 年提出后成为了最具代表性的预训练语言模型 BERT (Bidirectional Encoder Representations from Transformers) 模型[28]. BERT 使用了 attention 注意力机制对文本中的句子进行建模和双向深度的 transformer 的训练模型, 采用两阶段模型进行预训练和微调, 使通过 BERT 模型表达的文本向量包含更多的文本语义信息.

本维度缺陷描述信息主要包含综述, 描述和评论 3 类信息. 标题作为一个缺陷报告的总结内容、包含的目的性和错误信息更加的准确和精简; 描述信息是缺陷报告出错信息的详细描述, 包含作者出错的栈轨迹信息、缺陷的现象描述及产生缺陷时所执行的测试用例等信息, 缺陷详细描述信息可以帮助开发者更好的重现缺陷; 评论信息则是在缺陷修复过程中, 报告者、修复者和其他参与用户关于缺陷的各种讨论信息, 是逐步发现缺陷根因和缺陷需要采用的方法汇总.

采用 BERT 模型分别对标题、描述和评论 3 类信息分别进行训练, 分别产生 3 类文本信息的作为文本维度的 3 个特征: 标题特征和描述特征和评论特征, 共同代表了文本特征的总体. 文本维度特征如表 5.

表 5 文本维度特征

特征名称	编号
标题	T1
描述	T2
评论	T3

3 特征重要度分析

3.1 数据准备

在缺陷报告库中, 其中缺陷报告状态为 FIXED(被修复)、WONTFIX(真实的缺陷, 但由于资源或权限等问题无法修复) 的缺陷报告是有效的缺陷报告, 而缺陷报告状态为 DUPLICATE (重复)、INVALID (无效)、WORKSFORME (无法重现) 和 INCOMPLETE (不完整) 的缺陷报告为无效报告.

使用 Python 从 Eclipse 的缺陷报告库中爬取截止到 2019 年 8 月份的所有缺陷报告信息. 并且提取出的所有文本信息按照特征值作为处理后并归一化得到缺陷报告中所有的特征. 按照缺陷报告的状态对所有的缺陷报告做分类统计, 统计的报告数量和缺陷报告的问答数量如表 6 所示.

表 6 报告数据统计

缺陷报告状态	缺陷报告数量	缺陷报告中问答数量
DUPLICATE	9092	68271
FIXED	10961	196859
INCOMPLETE	6159	72281
WONTFIX	8321	96747
INVALID	10914	75595
WORKSFORME	9827	96214

表 6 中实验选取缺陷报告数据为缺陷报告的确认阶段和修复阶段; 在缺陷报告的确认阶段针对缺陷报告的特征重要度时选取无效的缺陷报告: 包含 resolution 为 INVALID (无效)、WORKSFORME (无法重现) 和 INCOMPLETE (不完整) 的缺陷报告. 其中重复 DUPLICATE 的缺陷报告认为仍然拥有相对有效的信息, 会对实验的准确度产生影响. 选取所有有效缺陷报告是排除无效的缺陷报告的所有缺陷报告, 包含状态为 FIXED(被修复)、DUPLICATE(重复) 的缺陷报告.

3.2 特征重要度排序方法设计

在缺陷重要度排序中, 本文遵循在缺陷的各类预测模型中, 不同特征在相同方法预测越准确特征重要度越高的判定标准. 本文分别考虑不同维度下特征集合的重要性和同一维度下不同特征的重要性. 在不同维度的排序中, 每一维度都包含多个特征, 通过多种机器学习分类方法的有效性判定, 根据准确率、召回率和 $F1$ 值综合确定每个维度的重要性, 从而形成每个维度的排序; 在同一维度的不同特征排序中, 采用基于决策树的嵌入方法进行维度内的每个特征排序.

采用有效性预测模型进行重要度排序. 有效性预测不仅是缺陷确认阶段的关键活动, 也是影响缺陷分配质量的关键活动. 为了保证结果的客观性和数据的有效性, 考虑到不同的分类方法对分类效果可能效果不同, 因此我们使用了多层感知机, 朴素贝叶斯, 决策树, 支持向量机, 逻辑回归 5 种常用的机器学习方法对缺陷报告每个维度的特征集合进行有效性预测, 根据预测精度进行特征重要度排序.

为了分析内部特征的重要性影响, 在对 3 个维度的特征使用了特征重要性排序, 是为了获得每个特征的重要度, 分别使用了在本文数据集更适合的 XGBoost 和 GBDT (Gradient Boosting Decision Tree) 对维度的单独特征进行排序, 得到特征排名和特征重要度.

GBDT 在 Facebook 中用于自动发现有效的特征和特征组合, 也在淘宝的搜索和预测中发挥了重要的作用. XGBoost 和 GBDT 两种方法实际上都是属于树形模型, 都具有高解释性, 而且可以处理多种混合类型的特征, 具有伸缩不变形, 对于缺失值也可以自然处理, 可以更好的处理缺陷报告多种维度的特征. GBDT 使用梯度下降, 使用多个弱分类器的加权求和合成强分类器, 每次迭代的过程中会产生一个弱分类器, 而当前的弱分类器在之前的分类器的残差基础上进行训练, GBDT 使用回归树模型而 XGBoost 使用了 L1, L2 正则化项的 CART 树模型.

树模型可以定义为所有树组成的函数空间, 回归树可以看作为一个把特征向量映射为某个值的函数.

$$J_j^2 = \frac{1}{M}\sum_{m=1}^{M} J_j^2(T_m) \tag{8}$$

特征 j 的全局重要度通过特征在单棵树中的重要度的平均值来衡量, M 是树的数量. 则特征 j 在单棵树中的重要度的如下:

$$\widehat{J_j^2}(T) = \sum_{t=1}^{L-1} i_t^2 1(v_t = j) \tag{9}$$

其中, L为树中的叶子节点数量和, $L-1$即为树的非叶子节点数量 (构建的树都是具有左右孩子的二叉树), v_t是和节点t相关联的特征, 节点t分裂之后平方损失的减少值为i_t^2.

为了探讨单独特征对于缺陷报告的影响, 基于特征选择器的原理对于每一个特征分别使用 XGBoost 和 GBDT 这两种方法. 实验中单独对每个维度单独特征进行特征重要度排序时, 单独使用每一个维度的特征而排除其他维度特征的影响, 以每个特征所在缺陷报告是否最终修复, 使用两种特征选择方法计算每个特征对其结果的影响, 最后两种方法得到的影响率转化为百分比, 取两种方法结果的均值作为最后的排名的依据.

3.3 评价指标

为了衡量特征选取的性能, 首要选取使用分类的准确率 (Precision, P) 和召回率 (Recall, R), 但是准确率和召回率也不能代表整个特征选取的性能指标. 若希望将有效的缺陷报告尽可能多地预测出来, 则可通过增加要预测的报告的数量来解决, 将所有报告预测为有效, 真正有效的报告必然被预测为有效, 但这样会降低查准率; 若希望预测出的有效报告中真正有效的比例尽可能高, 则只将有效的可能性大的报告预测为有效, 但这样会漏掉不少真正有效的报告, 造成查全率较低. 因此, 准确率与召回率是一对矛盾的度量. 如果部分样本准确率高, 而小部分样本没有被召回则我们更要关注样本 $F1$ 值的情况, 所以在实验性能的度量方面, 分别使用了准确率、召回率和 $F1$ 值作为效果对比. 表 7 为分类结果混淆矩阵.

$$P = TP/(TP+FP) \quad (10)$$

$$R = TP/(TP+FN) \quad (11)$$

$$F1 = 2PR/(P+R) = 2TP/(\text{样例总数}+TP-TN) \quad (12)$$

表 7 分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	*TP* (真正例)	*FN* (假反例)
反例	*FP* (假正例)	*TN* (真反例)

4 实验与分析

为了对比维度上的重要度, 本文首先将缺陷报告数据集首先按照对应属性维度分为 5 组. 输入每组的数据包含各自维度的特征, 缺陷报告的是否有效作为数据判定标准, 分别对数据集采用分层随机采样策略进行划分, 其中使用了 30% 的数据集作为测试集和 70% 的训练集作为样本进行了 30 次迭代实验, 计算最后结果度量的平均值, 然后进行比较. 同时为了验证每个分类方法对于综合维度分类的准确性, 我们计算了所有维度的分类效果的平均值.

实验中使用了 5 种不同的分类方法, 首先对于每个维度的评估值如表 8 所示, 经试验表示在文本维度和发布者经验维度方面两者对于不同的方法各有优劣, 也从侧面反应了对于缺陷报告来讲发布者经验和文本是重要的维度. 另外我们也发现在综合所有维度逻辑回归和支持向量机对于本文数据的分类效果最好.

表 8 特征维度重要度

特征	多层感知机		朴素贝叶斯		决策树		支持向量机		逻辑回归		平均	
	准确率	*F*1	准确率	*F*1	准确率	*F*1	准确率	*F*1	准确率	*F*1	准确率	*F*1
可读性	0.6823	0.4901	0.535	0.2833	0.6147	0.4336	0.6745	0.1756	0.6687	0.1588	0.635	0.3083
完整性	0.7195	0.4369	0.7112	0.4800	0.7166	0.4251	0.7237	0.4438	0.7195	0.4381	0.7181	0.4448
社交网络	0.7465	0.6373	0.7214	0.4738	0.7422	0.6108	0.7795	0.6414	0.7268	0.4973	0.7433	0.5721
发布者经验	0.8738	0.8125	0.8555	0.7617	0.8443	0.8618	0.8697	0.8084	0.8739	0.8063	0.8634	0.7901
文本	0.9442	0.9149	0.8891	0.8465	0.9138	0.8718	0.9485	0.9213	0.9114	0.8728	0.9214	0.8855
综合维度平均	0.6350	0.3083	0.7181	0.4448	0.7433	0.5721	0.8634	0.7901	0.9214	0.8855	0.7763	0.6002

可读性以及社交网络特征不能很好的表征每个单独特征对缺陷报告的影响, 所以实验仅对完整性, 发布者经验以及文本做特征重要度排序, 以此来验证缺陷报告中涉及的特征对于缺陷报告有效性的影响.

在报告者经验中, 特征选择得到的特征重要度为图 2 所示, 在报告者经验中, 通过两种方法的特征选择结果发现: 缺陷报告数量以及缺陷报告有效率特征占比均高于最近缺陷报告数量; 其次缺陷报告有效率对缺陷报告有效性的影响达到了 48.9%. 基于此分析报告者的历史发布数量和报告的有效率对于缺陷报告的最终是否修复影响更大. 虽然报告者发布的缺陷数量越多以及缺陷报告发布者的历史缺陷报告有效率越高则对其以后发布缺陷报告修复的影响更大, 但也并不是绝对的, 需要综合最近缺陷报告的数量等综合影响.

而在完整性的维度中如图 3 所示, 完整性的特征较多, 分布可能不明显. 通过两种特征选择的方法可以发现: 在缺陷报告的完整性方面, 堆栈以及截图特征的影响率均优于其他特征, 最高的影响率达到了 35.7%, 表明如果缺陷报告中包含栈轨迹 (stack traces) 的缺陷报告更容易最终被修复. 栈轨迹和截图是完整性维度中包含最多的特征之一, 其中包含了重要的信息, 而且容易上传和进行提交, 因此对于缺陷修复的影响也可能更大.

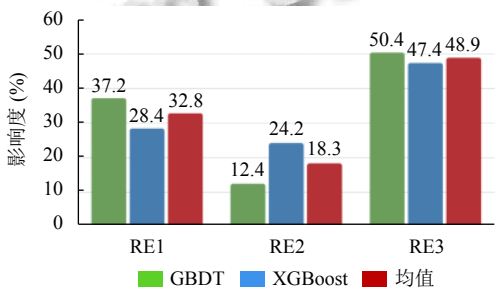

图 2 报告者经验特征重要度

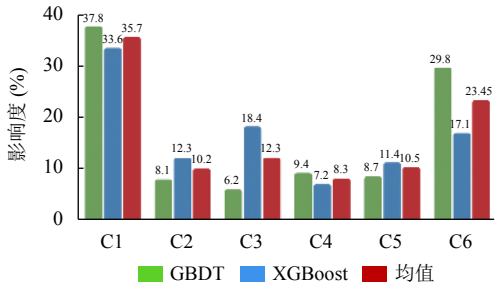

图 3 完整性特征重要度

在文本维度中定义了总结, 描述和评论中的特征, 使用了 BERT 生成向量特征能够更好的表示文本特征的差异, 在试验结果中, 评论文本内容的特征重要度要优于其他两项达到了 58.2% 的占比, 其次是描述文本的重要度如图 4 所示. 由此发现: 由于评论包含的文本信息包含更多与开发者交互的问答内容, 能够在缺陷报告提交后及时与开发者交流和补充相关信息, 其次缺陷报告描述的信息是开发者能够与提交者交互的重要支撑. 而标题文本则所占的重要度更低, 可能由于标题文本的字符数较少而且包含的有效信息更少的原因.

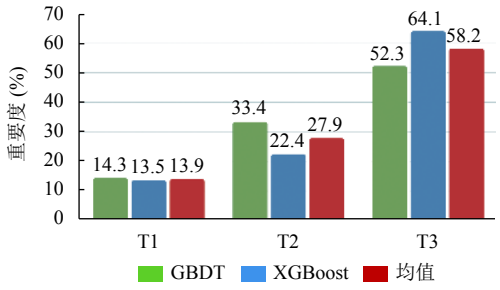

图 4 文本特征重要度

由此可见, 在本文分析的 3 种特征维度的单独特征分析结果表明: 对于报告者首先应当着重完善缺陷报告本身的质量和提交更多的缺陷报告; 对于缺陷报告完整度本身, 应当尽量提交栈轨迹和截图, 也需要完善补充所有的内容; 对于文本维度而言报告者应当尽量完善描述的同时, 需要在发布缺陷报告后与开发者更多的交互;

5 结论与展望

在缺陷报告的特征当中, 每一维度对应的特征的重要度不同, 报告者对于缺陷报告的关注度也不同. 带来的缺陷报告的有效性的判定也是不同的. 所以报告者对于缺陷报告特征重要度的关注度其实间接着影响了缺陷报告的有效性以及后续缺陷报告被修复的可能性. 本文首先综合所有维度提取出缺陷报告的全部特征信息, 然后对于每一维度的特征重要度进行了分析, 通过这些实验和统计做出的分析推荐给缺陷报告的报告者以及开发人员, 希望他们关注于缺陷报告中特征的重要度, 来提高缺陷报告的有效性和修复成功率. 但本文没有考虑时间对于缺陷报告重要度的影响, 修复的效率也是影响特征重要度的关键. 其中缺陷报告修复及确认时间的长短, 同样对缺陷报告的特征本身有很重要的影响. 未来基于缺陷报告每个特征的重要度和特征值, 研究作为预测缺陷报告是否可以有效地面向用户和开发者推荐缺陷报告.

参考文献

1 Anvik J, Hiew L, Murphy GC. Who should fix this bug? Proceedings of the 28th International Conference on Software engineering. Shanghai, China. 2006. 361–370.
2 Liu C, Yang JQ, Tan L, *et al*. R2fix: Automatically generating bug fixes from bug reports. Proceedings of the 2013 IEEE 6th International Conference on Software Testing, Verification and Validation. Luxembourg. 2013. 282–291.
3 Kim S, Whitehead EJ. How long did it take to fix bugs? Proceedings of the 2006 International Workshop on Mining Software Repositories. Shanghai, China. 2006. 173–174.
4 Zhang F, Khomh F, Zou Y, *et al*. An empirical study on factors impacting bug fixing time. Proceedings of the 2012 19th Working Conference on Reverse Engineering. London, UK. 2012. 225–234.
5 Saha RK, Khurshid S, Perry DE. An empirical study of long lived bugs. 2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering. Antwerp, Belgium. 2014. 144–153.
6 Karim MR, Ihara A, Yang X, *et al*. 2017. Understanding key features of high-impact bug reports. 2017 8th International Workshop on Empirical Software Engineering in Practice (IWESEP). Tokyo, Japan. 2017. 53–58.
7 Tantithamthavorn C, Hassan AE. An experience report on defect modelling in practice: Pitfalls and challenges. 2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track. Gothenburg, Sweden. 2017. 286–295.
8 Fan YR, Xia X, Lo D, *et al*. Chaff from the wheat: Characterizing and Determining Valid Bug Reports. IEEE Transactions on Software Engineering, 2020, 46(5): 495–525.
9 Guo PJ, Zimmermann T, Nagappan N, *et al*. Characterizing and predicting which bugs get fixed: An empirical study of Microsoft Windows. Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering. Cape Town, South Africa. 2010. 495–504.
10 Bettenburg N, Premraj R, Zimmermann T, *et al*. Extracting structural information from bug reports. Proceedings of the 2008 International Working Conference on Mining Software

Repositories. Leipzig, Germany. 2008. 27–30.
11 Zimmermann T, Premraj R, Bettenburg N, *et al*. What makes a good bug report? IEEE Transactions on Software Engineering, 2010, 36(5): 618–643.
12 Zanetti MS, Scholtes I, Tessone CJ, *et al*. Categorizing bugs with social networks: A case study on four open source software communities. 2013 35th International Conference on Software Engineering. San Francisco, CA, USA. 2013. 1032–1041.
13 Chen TQ, Guestrin C. XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining. San Francisco, CA, USA. 2016. 785–794.
14 Friedman JH. Greedy function approximation: A gradient boosting machine. Annals of Statistics, 2001, 29(5): 1189–1232.
15 Just S, Premraj R, Zimmermann T. Towards the next generation of bug tracking systems. Proceedings of the 2008 IEEE Symposium on Visual Languages and Human-Centric Computing. Herrsching, Germany. 2008. 82–85.
16 Zhang T, Jiang H, Luo XP, *et al*. A literature review of research in bug resolution: Tasks, challenges and future directions. The Computer Journal, 2016, 59(5): 741–773.
17 Hooimeijer P, Weimer W. Modeling bug report quality. Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering. Atlanta, GA, USA. 2007. 34–43.
18 Flesch R. A new readability yardstick. Journal of Applied Psychology, 1948, 32(3): 221–233.
19 Anderson J. Lix and Rix: Variations on a little-known readability index. Journal of Reading, 1983, 26(6): 490–496.
20 Guerrero LA, Mejías B, Collazos CA, *et al*. Collaborative learning and creative writing. Proceedings of the First Conference on Latin American Web Congress. Santiago, Chile. 2003. 180–186. [doi: 10.1109/LAWEB.2003.1250295]
21 Mc Laughlin GH. Smog grading —A new readability formula. Journal of Reading, 1969, 12(8): 639–646.
22 Kincaid P, Fishburne RP, Rogers PL, *et al*. Derivation of new readability formulas for navy enlisted personnel. Millington: Chief of Naval Training, 1975.
23 Smith EA, Senter RJ. Automated readability index [Technical Report]. University of Cincinnati OH, 1967.
24 Coleman M, Liau TL. A computer readability formula designed for machine scoring. Journal of Applied Psychology, 1975, 60(2): 283–284.
25 Zhou Y, Tong YX, Gu RH, *et al*. Combining text mining and data mining for bug report classification. Journal of Software: Evolution and Process, 2016, 28(3): 150–176.
26 Mikolov T, Chen K, Corrado G, *et al*. Efficient estimation of word representations in vector space. arXiv preprint arXiv: 1301.3781, 2013.
27 Pennington J, Socher R, Manning CD. GloVe: Global vectors for word representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. Doha, Qatar. 2014. 1532–1543.
28 Devlin J, Chang MW, Lee K, *et al*. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv: 1810.04805, 2013.