开源软件缺陷报告自动摘要研究综述

刘翠兰[1,5], 张嘉元[3,5], 曹旭栋[4,5], 伍高飞[1,2,5], 朱笑岩[6], 任家东[7], 冯 涛[3]

[1]西安电子科技大学 广州研究院 广州 中国 510555
[2]桂林电子科技大学 广西密码学与信息安全重点实验室 桂林 中国 541004
[3]兰州理工大学 计算机与通信学院 兰州 中国 730050
[4]中国科学院大学 计算机科学与技术学院 北京 中国 101408
[5]国家计算机网络入侵防范中心(中国科学院大学) 北京 中国 101408
[6]西安电子科技大学 通信工程学院 西安 中国 710071
[7]燕山大学 信息科学与技术学院 秦皇岛 中国 066004

摘要 在开源软件开发的维护阶段, 开源软件缺陷报告为开发人员解决缺陷提供了大量帮助。然而, 开源软件缺陷报告通常是以用户对话的形式编写, 一个软件缺陷报告可能含有数十条评论和上千个句子, 导致开发人员难以阅读或理解软件缺陷报告。为了缓解这个问题, 人们提出了开源软件缺陷报告自动摘要, 缺陷报告自动摘要可以减少开发人员阅读冗长缺陷报告的时间。本文以综述的方式对开源软件缺陷报告自动摘要的研究做了系统的归纳总结。首先, 根据摘要的表现形式, 将开源软件缺陷报告摘要分类为固定缺陷报告摘要和可视化缺陷报告摘要, 再将固定缺陷报告摘要研究方法分类为基于监督学习方法和基于无监督学习方法, 之后总结了基于监督学习和无监督学习的开源软件缺陷报告摘要生成的工作框架, 并介绍了开源软件缺陷报告摘要领域常用数据集、预处理技术和摘要评估指标。其次, 本文以无监督学习为切入点, 分类阐述和归纳了无监督开源软件缺陷报告摘要方法, 将无监督开源软件缺陷报告摘要方法分类为: 基于特征评分方法、基于深度学习方法、基于图方法和基于启发式方法, 并对每类方法进行讨论与分析。再次, 从缺陷报告摘要的实用性出发, 对现有的缺陷报告可视化摘要研究成果进行总结, 并对固定缺陷报告摘要和可视化缺陷报告摘要的实用性做出分析。最后, 对现有研究成果及综述进行讨论和分析, 指出了开源软件缺陷报告摘要领域在缺陷报告数据集、抽取式摘要和黄金标准摘要三个方面面临的挑战和对未来研究的展望。

关键词 开源软件; 缺陷报告; 自动摘要; 文本摘要
中图法分类号 TP391.1 DOI 号 10.19363/J.cnki.cn10-1380/tn.2022.11.09

A survey of Automatic Summarization of Open Source Software Bug Reports

LIU Cuilan[1,5], ZHANG Jiayuan[3,5], CAO Xudong[4,5], WU Gaofei[1,2,5], ZHU Xiaoyan[6], REN Jiadong[7], FENG Tao[3]

[1]Guangzhou Institute of Technology, Xidian University, Guangzhou 510555, China
[2]Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin 541004, China
[3]School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China
[4]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 101408, China
[5]National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, Beijing 101408, China
[6]School of Telecommunication Engineering, Xidian University, Xi'an 710071, China
[7]School of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China

Abstract In the maintenance phase of open source software development, open source software bug reports provide a lot of help for developers to solve bugs. However, open source software bug reports are usually written in the form of user dialogue. A software bug report may contain dozens of comments and thousands of sentences, making it difficult for developers to read or understand the software bug report. In order to alleviate this problem, people have proposed automatic summarization of open source software bug reports, which can reduce the time for developers to read lengthy bug reports. In this paper, the research on automatic summarization of open source software bug reports is summarized systematically. First, according to the presentation form of the summary, the open source software bug report summary is classified into fixed bug report summary and visual bug report summary, and then the research methods of fixed bug report summary are

通讯作者: 伍高飞, 博士, 讲师, Email: wugf@nipc.org.cn
本课题得到国家自然科学基金项目(No. U1836210, No. 61941105, No. 61772406)、广西密码学与信息安全重点实验室研究课题(No.GCIS202123)、陕西省自然科学基础研究计划项目(No. 2021JQ-192)和河北软件工程重点实验室项目(No. 22567637H)资助。
收稿日期: 2022-07-04; 修改日期: 2022-10-11; 定稿日期: 2022-10-11

classified into supervised learning based method and unsupervised learning based method. After that, the work framework for generating open source software bug report summary based on supervised learning and unsupervised learning is summarized, it also introduces common data sets, preprocessing techniques and summary evaluation indicators in the field of open source software bug report summary. Secondly, this paper takes unsupervised learning as the starting point, elaborates and summarizes the unsupervised open source software bug report summary methods by category, and classifies the unsupervised open source software bug report summary methods into: feature based scoring methods, depth based learning methods, graph based methods, and heuristic methods, and discusses and analyzes each type of methods. Thirdly, starting from the practicability of bug report summary, the existing research results of visual bug report summary are summarized, and the practicability of fixed bug report summary and visual bug report summary is analyzed. Finally, the existing research results and reviews are discussed and analyzed, and the challenges faced by the open source software bug report summary field in three aspects of bug report dataset, abstract summary and gold standard summary are pointed out, as well as the prospects for future research.

Key words open source software; bug report; automatic summary; text summary

1 引言

开源软件在不断地迭代和使用过程中, 会出现很多软件缺陷[1]。软件缺陷通常以软件缺陷报告的形式存在于开源软件缺陷库中, 开源软件缺陷报告摘要研究常用开源软件缺陷库及其 URL 如表 1 所示。缺陷报告内蕴含了关于所描述软件缺陷的有价值信息, 对于软件开发人员来说, 参考缺陷报告来修复软件缺陷是至关重要的。

表 1 常用开源软件缺陷库

Table 1 Common open source software bug library

开源软件缺陷库	URL
ECLIPSE	https://bugs.eclipse.org/bugs/
MOZILLA	https://bugzilla.mozilla.org/home/
KDE	https://bugs.kde.org/
GNOME	https://gitlab.gnome.org/GNOME
APACHE PROJECT	https://issues.apache.org/jira/
LINUX KERNEL	https://bugzilla.kernel.org/

一个完整的缺陷报告通产包含 3 种类型的数据: 文本数据、元数据和附件。文本数据由标题、描述和评论组成, 标题是关于缺陷的简要介绍, 描述是缺陷的概述及重现步骤, 评论是报告者与用户以对话形式对缺陷的讨论。元数据是关于缺陷的各种信息, 比如缺陷编号、产品名称、软件版本、严重性、报告者等。附件是缺陷产生的日志信息, 目的是为了更清楚地说明缺陷[2]。图 1 显示了 Kernel 缺陷库中 ID 为 208253 的缺陷报告的一部分信息及对应的数据类型。多个用户以评论的形式交互讨论, 产生了大量的消息对话。为了理解缺陷报告, 开发人员必须仔细阅读所有的评论, 这是一个相当耗时且乏味的任务。缺陷报告自动摘要可以减少阅读冗长缺陷报告的时间。缺陷报告摘要的一种产生方式是指派一个项目成员(例如开发人员)在缺陷报告得到解决时编写报告摘要。然而, 考虑到大多数软件项目的时间和资源限制, 这不是一个实用的解决方案。因此, 自动缺陷报告摘要成为解决这一问题的重要方法[3]。

文本摘要在产生方法上可以分类为: 抽取式摘要和生成式摘要。抽取式摘要是指从文本中选取若干重要句子直接组合成摘要[4], 生成式摘要是通过对文本内容进行理解来概括出主要内容[5], 因此需要较大的数据集, 最终获得更贴切人类撰写的摘要。在缺陷报告摘要领域, 摘要方法主要是朝着抽取式摘要的方向发展[2]。缺陷报告摘要在表现形式上又可以分类为固定缺陷报告摘要和可视化缺陷报告摘要, 固定缺陷报告摘要的内容不可变, 而可视化缺陷报告摘要的内容可随关键词的变化而变化。固定缺陷报告摘要方法可以分为有监督学习方法和无监督学习方法。经过调研发现, 缺陷报告摘要的研究主要集中在 2010 年以后, 皆为抽取式摘要, 且基于无监督学习的方法较多。目前已存在 4 篇相关的英文综述, 文献综述[6-7]没有对缺陷报告摘要方法进行详细的分类, 文献综述[8]将缺陷报告摘要方法分类后只选择了五篇文献进行深入分析, 文献综述[9]总结的是 2010 年 1 月至 2016 年 4 月期间的缺陷报告摘要研究成果。由此可见, 目前尚缺乏对缺陷报告自动摘要任务进行系统研究的中文综述。

我们分别检索了万方、知网、IEEE、ACM、SCI、Springer、EI 等数据库在 2010 至 2022 年 6 月期间收录的研究工作, 然后进一步根据标题、摘要和研究内容筛选论文, 最后选出切题文献共 40 篇, 如图 2 所示。

本文的主要贡献有 3 个方面:

1) 广泛收集并调研了缺陷报告自动摘要领域的现有相关文献, 阐述了基于监督学习和无监督学习抽取式缺陷报告摘要生成框架。以无监督学习为切入点, 分类阐述对比现有无监督缺陷报告摘要方法。此外, 还介绍了常用缺陷报告数据集、预处理技术和摘要评估指标。

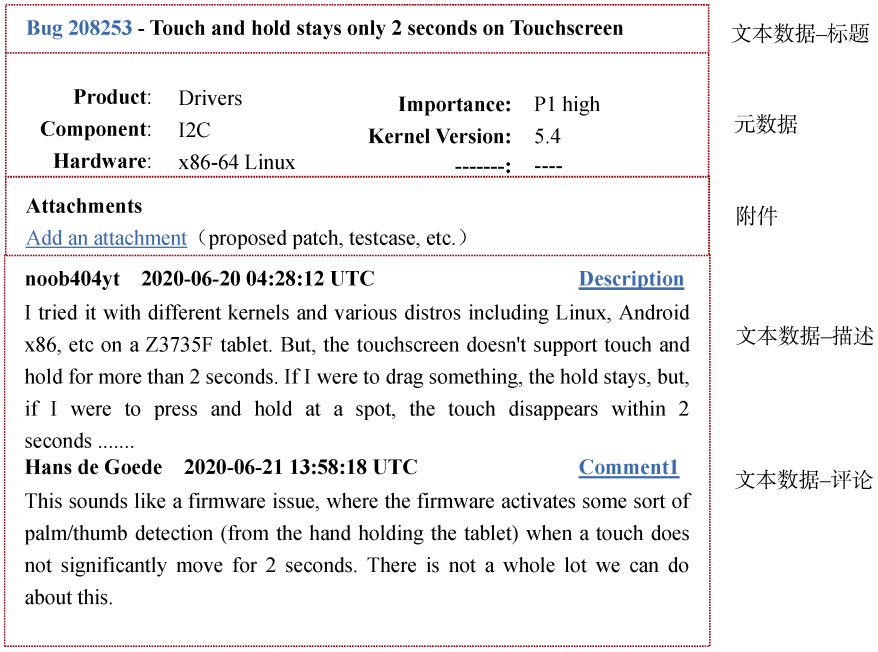

图 1　Kernel 缺陷库中 ID 为 208253 的缺陷报告的部分内容及对应数据类型

Figure 1　Some contents and corresponding data types of the bug report with ID 208253 in the Kernel bug library

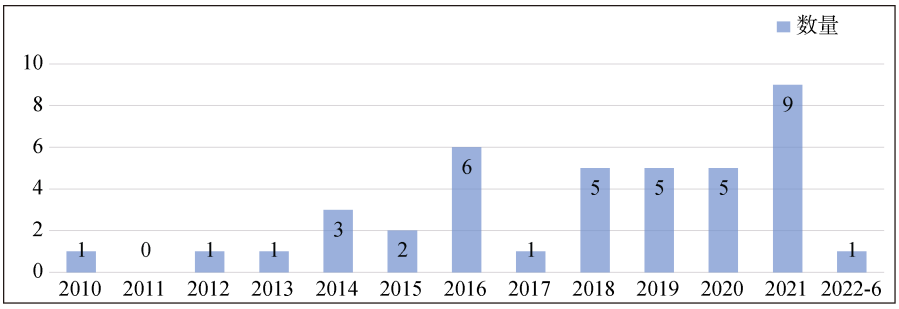

图 2　缺陷报告摘要领域研究文献数量

Figure 2　Number of research literatures in bug report summary field

2) 从缺陷报告摘要的实用性出发, 对现有的缺陷报告可视化摘要研究成果进行总结, 并对固定缺陷报告摘要和可视化缺陷报告摘要用于实际应用场景的可能性做出讨论与分析。

3) 对已有开源软件缺陷报告自动摘要研究工作进行总结, 分析当前开源软件缺陷报告自动摘要领域面临的三大挑战和机遇, 并对未来的研究趋势进行展望。

2　基于监督学习的缺陷报告摘要

基于监督学习的缺陷报告摘要方法使用经过训练的模型来计算句子权重, 再选择权重值较高的句子构成摘要。本节分析了缺陷报告摘要有监督学习方法框架, 并归纳了常用缺陷报告数据集、预处理技术、提取特征方法和摘要评估指标。

2.1　监督学习摘要方法框架

通过调研现有的研究工作, 我们给出了监督学习抽取式缺陷报告摘要的一般流程, 包括缺陷报告收集、模型训练和模型测试三个阶段, 如图 3 所示。

在缺陷报告收集阶段, 从开源软件缺陷库中收集缺陷报告或使用已公开的缺陷报告数据集作为训练集和测试集。

在模型训练阶段, 将训练集进行预处理, 例如分段、标记化、移除停止词和词干提取等[10], 来整理原始缺陷报告并进行格式化。从预处理后的数据中提取文本特征并将句子表示为向量形式, 将句子向量和句子标签输入模型, 评估文本特征以训练统计模型, 如逻辑回归、决策树等。

在模型测试阶段, 测试集预处理并提取文本特征后, 将句子向量输入统计模型来预测句子属于摘

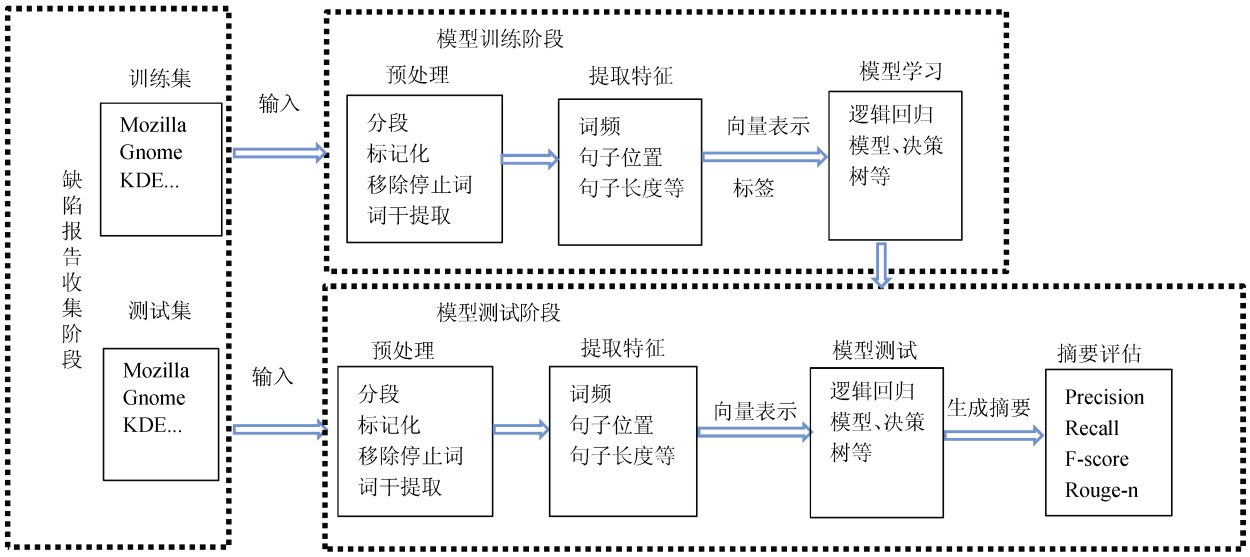

图 3　缺陷报告摘要有监督学习方法框架图

Figure 3　Bug report summary supervised learning methodology framework

要的概率, 并选择概率值较高的句子组成摘要。最后, 使用评估指标来评估摘要。

2.2　缺陷报告数据集

在缺陷报告摘要领域, 常用的公开数据集为 SDS(Summary Data Set)。SDS 数据集由文献[11-12]提出, 包括 36 个缺陷报告。这 36 个缺陷报告来自四个开源软件项目, 分别是 Eclipse、Mozilla、KDE 和 Gnome, 共包含 2361 个句子。SDS 数据集由来自英属哥伦比亚大学计算机系的十名研究生来注释, 每个注释员从 4 个软件项目中选择一个缺陷报告。对于每个缺陷报告, 注释员用他们自己的句子(最多 250 个单词)编写缺陷报告的生成式摘要。接着, 将生成式摘要中的每一个句子映射到缺陷报告中的一个或者多个句子, 再根据映射写出缺陷报告的抽取式摘要。因为总结是一个主观过程, 所以为每个缺陷报告分配了 3 个注释员, 并使用 kappa 测试[13]来衡量注释员们将生成式摘要句子映射到缺陷报告句子的一致程度。Kappa 测试结果为 0.41, 显示为中等程度的一致性。之后结合缺陷报告的 3 个生成式摘要, 根据将生成式摘要句子映射到缺陷报告中句子的注释员人数, 为缺陷报告中的每个句子打分。

对于缺陷报告中的句子, 如果没有任何注释员将其映射, 那么分数为 0, 如果有 3 个注释员在其生成式摘要中都与该句子有映射, 那么分数为 3。如果一个句子的得分为 2 分及以上, 则该句子被视为摘要的一部分。对于每个缺陷报告, 得分为 2 分或 2 分以上的句子集称为黄金标准摘要(Gold Standard Summary, GSS), GSS 即为人工编写的缺陷报告摘要。

2.3　预处理技术

缺陷报告预处理包括句子预处理和句子过滤。句子预处理步骤包括分段、标记化、删除停止词、词干提取[10]。句子过滤是指过滤掉对摘要提取无意义的句子, 常见的被过滤的句子类型有代码片段、堆栈跟踪。

分段: 使用分隔符(如“.”、“!”、“？”)将缺陷报告分成多个句子。

标记化: 把句子标记为基本的语言单位[14], 即单词、短语。

删除停止词: 停止词是指对文本数据没有意义的单词, 常见的停止词有“the”、“a”、“this”等等。

词干提取: 词干指的是单词的词根, 通过去除前缀和后缀, 将单词转变为词根形式[10]。例如, 单词“absorbing”被提取为词根形式“absorb”。

文献[15]增加了一个预处理步骤, 从缺陷报告中过滤掉噪音句子。他们将句子分类为疑问句、调查句、代码句子、其他句子, 并将代码句子和其他句子作为噪音过滤掉。文献[16-17]提出了句子可信度来衡量交互讨论中的一个句子被认可与否的程度, 来减少有争议的句子被选入摘要的可能性。

2.4　特征提取方法

分析并总结重要性句子的特征, 再通过是否拥有这些特征来判断缺陷报告中每个句子的重要性。文献[11-12]首次提出使用监督学习来提取缺陷报告摘要, 提出了 24 种特征, 并将这 24 种特征分为四大类: 结构特征、参与者特征、句子长度特征和词汇特征。文献[14]将所有句子分为 6 类: 问题、代码、调

查、人为因素、程序和其他。其他类句子会被过滤掉, 剩下的句子被认为是信息性句子, 再将文献[11-12]中的 24 种特征用于信息性句子来提取摘要。文献[18]提出基于众包属性逻辑回归(Logistic Regression with Crowdsourced Attributes, LRCA)方法, 它们通过筛选提取众包数据来构建特征。文献[19-20]使用术语频率-逆文档频率(Term Frequency–Inverse Document Frequency, TF-IDF)[21]、卡方检验、信息增益方法和句子复杂度来选取关键词和关键句。文献[22]将缺陷报告中的句子意图分为七类: 缺陷描述、解决方案、表达的意见、信息搜索、提供信息、代码、情感表达, 训练一个意图分类器来自动获取缺陷报告中句子的意图及其权重。文献[23]考察了文献[11-12]在不使用复杂特征的情况下是否还能创建合理的缺陷报告摘要, 发现缺少一些复杂的特征会导致摘要的精确度和召回率降低, 并确认句子长度是缺陷报告摘要的最重要特征。

缺陷报告句子提取特征之后, 将句子表示为向量形式。在现有的监督学习缺陷报告摘要研究中, 句子向量的维数与特征的总数相同, 一维代表一个特征, 若句子中包含该特征, 则句子向量中对应维的值是 1, 否则为 0。句子标签值为 1 或 0, 标签值为 1 表示一个摘要句子, 标签值为 0 表示一个普通句子。将句子向量和句子标签输入模型来评估特征, 计算每个特征的权重, 句子权重值为包含的特征权重之和, 将句子权重值按降序排序, 选择排名靠前的句子组成摘要。文献[11-12]选择前 25%句子组成摘要, 并且被认为是摘要的最合适长度, 广泛应用于现有的缺陷报告摘要研究中。

2.5 摘要评估指标

可以从准确性和可读性两个角度来评估摘要方法的性能[16-17]。准确性是指与黄金标准摘要对比, 生成摘要(Generated Summary, GS)句子的准确性, 衡量方法的准确性指标包括精确度(*Precision*)、召回率(*Recall*)和 F 分数(*F-score*)。可读性是指生成摘要的连贯性, 衡量方法的可读性指标主要是 *Rouge-n*。*Rouge-n* 量化生成摘要和黄金标准摘要之间重叠的单词数, *n* 通常取 1(对应于 1-gram)和 2(对应于 2-gram)[24]。对以上所有指标来说, 评估指标的值越大, 说明生成摘要的质量越高。

精确度: 选择生成摘要(*GS*)和黄金标准摘要(*GSS*)中出现的句子数除以生成摘要(*GS*)中的句子总数[10], 计算如公式 1 所示。

$$Precision = \frac{|GS \cap GSS|}{GS} \tag{1}$$

召回率: 选择生成摘要(*GS*)和黄金标准摘要(*GSS*)中出现的句子数除以黄金标准摘要(*GSS*)中的句子总数[10], 计算如公式 2 所示。

$$Recall = \frac{|GS \cap GSS|}{GSS} \tag{2}$$

F 分数: 表示计算出的精确度和召回率的调和平均值, 并描述了所提出方法的总体性能[10], 计算如公式 3 所示。

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3}$$

Rouge-n: 分母是黄金标准摘要(*GSS*)*n-gram*[25]的个数, 分子是黄金标准摘要(*GSS*)和生成摘要(*GS*)共有的 *n-gram* 的个数, 计算如公式 4 所示。

$$Rouge - n = \frac{\sum_{s \in GSS} \sum_{n-gram \in s} Count_{match}(gram_n)}{\sum_{s \in GSS} \sum_{n-gram \in s} Count(n-gram)} \tag{4}$$

3 基于无监督学习的缺陷报告摘要

基于无监督方法的缺陷报告不使用经过训练的模型来生成摘要, 通常是根据缺陷报告中句子的重要性为每个句子分配一个度量值, 并选择度量值排名靠前的句子形成摘要。本节分析了缺陷报告摘要无监督学习方法框架, 并对为句子赋予权重的方法进行分类总结。

3.1 无监督学习摘要方法框架

从开源软件缺陷库中收集缺陷报告后, 对缺陷报告进行预处理, 使用基于特征评分、基于深度学习和基于启发式等方法为句子赋予权重, 并使用权重值靠前的句子组成摘要。最后, 使用评估指标来评估摘要, 过程如图 4 所示。基于监督学习和基于无监督学习的缺陷报告摘要在数据集收集、预处理和摘要评估方面使用的方法相同。

3.2 无监督学习摘要方法分类

本文在开源软件缺陷报告摘要领域现有研究的基础上, 将句子赋予权重的方法分类为: 基于特征评分、基于深度学习、基于图和基于启发式方法, 方法对比见表 2。

3.2.1 基于特征评分的方法

基于特征评分的方法是指为句子中的特征赋予权重, 为包含这些特征的句子赋予度量值, 再将度量值较高的句子组成摘要。特征包括词频、与标题或主题的相似度, 以及句子长度、句子位置等[26]。

文献[10]提出了基于关键字和句子的缺陷报告摘要方法, 他们将特征分为关键字特征和句子特征,

关键字特征提取使用 TF-IDF 方法和快速自动关键词提取(Rapid Automatic Keyword Extraction, RAKE)[27]方法, 句子特征提取使用模糊 C 均值聚类算法[28]等。然后, 根据关键字和句子特征构建规则引擎来选择缺陷报告中的句子组成摘要。对生成的摘要再使用层次聚类算法来去除冗余, 最终生成缺陷报告摘要。

文献[15]中介绍了中心(Centroid)[29]和最大边缘相关(Maximal Marginal Relevance, MMR)[30]两种基于特征评分的无监督学习方法。Centroid 方法是基于句子的词频特征来生成摘要。缺陷报告中的句子表示为 TF-IDF 的加权向量, 通过加权向量来寻找中心句子。其中, 中心句子是一个伪句子, 其向量的权重等于缺陷报告中所有句子向量的平均值。中心句子中包含的词称为中心词, 而包含中心词的句子更能代表文档的主题。每个句子中包含的中心词权重之和称为句子中心值, 按句子中心值的降序排序, 并选择排名靠前的句子构建摘要。MMR 方法除了使用词频特征外, 还使用了句子相异度特征。句子相异度值为句子余弦相似度的负值。在摘要句子选择过程中, MMR 方法通过向摘要句子集中递增句子来生成摘要, 如果一个句子除了中心性之外, 与已经选择的摘要句子具有最小的相似性, 那么该句子应该包含在摘要中。

文献[31]提出了基于两级特征提取生成缺陷报告摘要的方法, 将特征分为评论选择特征和句子选择特征。先使用评论选择特征进行评论选择, 再从这些评论中使用句子选择特征选取句子创建评论摘要。其次使用 PageRank 算法[32]和基于词典的方法[33]来创建描述字段的摘要, 再使用基于图模型的方法来提取标题。最后再将标题、描述摘要和评论摘要结合在一起形成最终摘要。

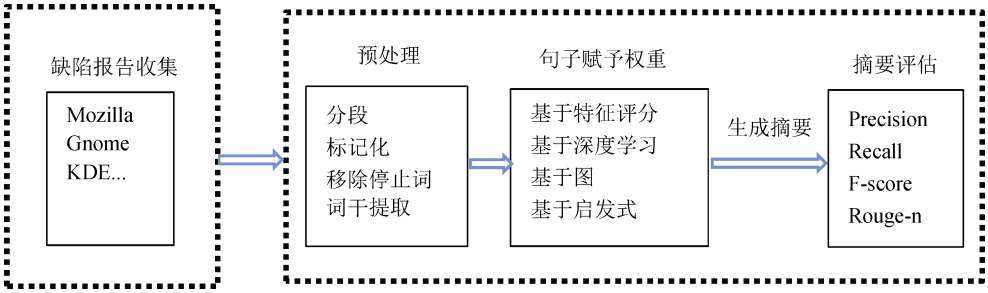

图 4 缺陷报告摘要无监督方法框架图

Figure 4 Framework of unsupervised method for bug report summary

表 2 基于无监督学习的缺陷报告摘要方法分类

Table 2 Classification of bug report summary method based on unsupervised learning

方法分类	描述	缺点
特征评分	据词频、与标题的相似度、句子长度、句子位置等特征选取句子组成摘要	内容不全面、语句冗余、不连贯
深度学习	利用自动编码网络、循环神经网络等神经网络模型对原文建模得到文本单元表示后进行文本单元的抽取形成摘要	对数据要求较高, 参数较多
基于图	句子构成图的顶点, 两个句子间的相似度作为边的权重, 利用图排序算法选择句子组成摘要	任意句子相似性计算导致运行速度相对比较慢, 句子冗余
基于启发式	使用粒子群算法、蚁群算法等启发式算法来选取最优句子形成摘要	参数设置和迭代停止条件等只能依赖经验调整, 运算复杂

讨论 1. 本节总结归纳了基于特征评分的缺陷报告生成摘要方法, 分别从特征类型、特征描述、数据集的选择和摘要评估指标等多个角度进行分析和对比, 如表 3 所示。从特征类型和特征描述而言, 特征可以根据表示范围分为评论特征、句子特征和关键字特征。每个句子拥有的特征越多, 其重要性就越大, 则被选入缺陷报告摘要的可能性就越大。从数据集类型上看, 基于特征评分的缺陷报告摘要方法主要使用 SDS 数据集。从评估指标上看, 不同的数据集对摘要评估的影响不同, 文献[10]生成的缺陷报告摘要质量较好。

观点 1. 基于特征评分的缺陷报告摘要方法能够直接判断关键字或句子包含的特征, 并为特征赋予合适的度量值。句子的度量值为包含特征度量值的

表 3 基于特征评分的缺陷报告摘要工作对比

Table 3 Comparison of bug report summary based on feature scoring

文献	特征类型	特征描述	数据集	评估指标		
				精确度	召回率	*F* 分数
[10]	关键字特征、句子特征	将关键字特征和句子特征结合起来选择摘要句子	SDS	0.656	0.625	0.6396
[15]	句子中心性	与中心句子的相似性越高, 是摘要句子的可能性越大	SDS	0.42	0.43	0.43
[15]	句子中心性、句子相异性	选择与中心句子相似性高, 且与已选的摘要句子相异度值大的句子组成摘要	SDS	0.47	0.49	0.48
[31]	评论特征、句子特征	使用评论特征在缺陷报告中选择评论, 再从这些评论中使用句子特征选取句子创建评论摘要	SDS	0.46	0.76	0.512

总和, 将句子度量值按降序排列, 选择排名靠前的句子组成摘要。通过分析和对比现有的研究工作, 本文发现: 1)根据词频和句子中心性来选择摘要句子, 会造成语句冗余、内容单薄; 2)句子相异性可以有效减缓由句子中心性来带的语句冗余问题。

3.2.2 基于深度学习的方法

深度学习方法利用自动编码网络、循环神经网络(Recurrent Neural Network, RNN)等神经网络模型对原文建模得到文本单元表示后进行文本单元的抽取形成摘要[26]。

文献[16-17]提出了 BugSum(Bug Summarization)方法, 他们考虑了缺陷报告对话中蕴含的表达赞成或反对态度的行为进行评估, 并提出了句子可信度特征。被其他句子支持的句子更可信, 其可信度较高; 而有争议的句子可能不正确, 其可信度较低。使用经过训练的自动编码网络[34]将句子转换为向量, 并将句子的可信度评分合并到向量中, 以减少有争议的句子被选入摘要的可能性。最后, 使用一种同时考虑到句子信息性和全面性的动态策略来优化缺陷报告摘要的自动生成过程。

文献[35]提出了基于深度学习的缺陷报告摘要(Deep Summary, DeepSum)方法, DeepSum 方法的核心是一个阶梯式自动编码网络。DeepSum 方法的基本思想是: 深层神经网络的隐藏层是缺陷报告中句子的特征向量的压缩表达式。通过使用句子选择算法测量每个句子对这个压缩表达式的权重, 来选取缺陷报告摘要中的句子。DeepSum 方法使用评估增强模块来过滤冗余句子, 然后, 它将不同句子类型的向量逐步输入到自动编码器网络中, 来自动测量不同句子类型中单词的权重。最后, 采用动态规划的方法提取缺陷报告摘要中的句子。

文献[36]提出的方法是: 缺陷报告预处理后, 使用句子嵌入[37]将句子转换为向量形式。再使用 K-均值聚类算法[38]将句子聚类, 在每一类中选择一个句子, 根据句子的信息性对句子进行排名, 选择排名靠前的句子组成缺陷报告摘要。句子嵌入使用 Skip-Thought[39]方法, Skip-Thought 方法带有 GRU (Gated Recurrent Unit)激活的 RNN 编码器和 GRU 激活的 RNN 解码器。

文献[40]提出一种基于深度学习和句子重要性因素的缺陷报告摘要方法。此方法有 4 个输入: 第 1 个输入是要提取摘要的缺陷报告, 第 2 个输入是句子重要性因素列表, 第 3 个输入是句子重要性因素的权重, 第 4 个输入是输出摘要的长度, 最终输出为缺陷报告摘要。在该方法中, 句子重要性因素包括句子可信度、句子间衔接程度和话题关联程度。其中, 句子可信度是指该句子在交互讨论中被认可的程度, 句子间衔接程度为句子与句子之间衔接自然连贯的程度, 话题关联程度即为这句话与标题之间的相似性。缺陷报告预处理后, 使用自动编码网络提取句子特征, 再根据句子重要性因素及其相应的权重计算句子分数, 最后根据输入的摘要长度选择句子组成摘要。

讨论 2. 表 4 总结了现有的基于深度学习的缺陷报告摘要研究, 并从方法描述、数据集和评估指标角度进行对比和分析。从方法描述而言, 自动编码网络在基于深度学习的缺陷报告摘要中较受欢迎。从数据集上看, 基于深度学习的缺陷报告摘要研究中的数据集以 SDS 为主。从评估指标而言, 文献[16-17]生成摘要的准确性和可读性较好。

观点 2. 在现有研究中, 基于深度学习的缺陷报告摘要方法使用自动编码网络来生成摘要, 首先将句子转为向量形式, 再将句子向量输入自动编码网络进行训练来得到单词的权重。本文发现: 1) 使用动态规划句子选择算法可以高效地生成不同长度的句子摘要; 2)在生成摘要的过程中, 自动编码网络需要

大量的训练数据, 所以运行时间较长, 使用深度学习生成缺陷报告的速度较慢; 3)句子可信度可以有效避免有争议的句子选入摘要, 提高摘要的准确性和可读性。

3.2.3 基于图的方法

缺陷报告摘要的文本单元是句子, 基于图的方法是句子构成图的顶点, 两个相似的点用边连接起来, 将文本构建成拓扑结构图[26]。常见的图排序算法有 PageRank 和 TextRank[41]等。

文献[15]中介绍了 Grasshopper (Graph random-walk with absorbing that hop's among peaks for ranking)[42]和 DivRank(Diverse Rank)[43]两种基于图文档排序的无监督学习方法。Grasshopper 方法强调多样性, 它将文档概括为一个图, 其中每个句子概括为一个节点, 这些节点之间的边被分配了一个等于句子相似性的权重。Grasshopper 算法在这个图上执行随机游动, 以获得图的平稳分布, 即获得随机游动访问节点的概率, 其中概率较高的节点对文档更重要, 所以选择访问概率较高的句子组成缺陷报告摘要。DivRank 方法在数据的图形表示中也使用了随机游动, 该方法中的随机游动是一种时间变异游动。从一个节点到另一个节点的转移概率通过之前访问该节点的次数得到加强, 从而导致转移概率随着时间的推移而变化。

文献[44]提出了 Hurried 方法来模拟匆忙阅读缺陷报告时如何总结摘要。Hurried 方法基于一个假设模型, 即当时间紧迫时, 一个人将把注意力集中在他认为最重要的句子上, 并对读者认为重要的句子类型提出三个假设: 讨论与主题相关的句子、被其他句子评论的句子, 以及与标题和描述相似的句子。使用该假设模型根据句子被阅读的概率对其进行排序, 并用概率最高的句子组成摘要。Hurried 方法使用马尔科夫链对这三个假设进行建模, 马尔科夫链是一种有向图, 其中节点表示状态, 节点之间的边模拟状态之间的转移概率。为了模拟匆忙阅读缺陷报告读取过程, 将缺陷报告中的每个句子表示为马尔科夫链中的一个节点, 其中每个边 $m_{i,j}$ 表示从句子 s_i 转换到句子 s_j 的概率。第一个假设是用户在阅读一个句子后, 接下来要阅读的句子是与上一个句子谈论类似主题的句子。这在马尔科夫链中建模表示为: 谈论类似主题的句子之间的转移概率应该高于不谈论类似主题的句子之间的转移概率。第二个假设是用户应该注意已经被其他句子评论过的句子, 这在马尔科夫链中意味着从一个句子到被评价句子的转换概率应该比到其他句子的转换概率更高。第三个假设是用户将注意力集中在讨论缺陷报告标题和描述上, 因此在描述中的每个句子中添加一个链接, 来提高与缺陷描述主题相似的句子的相关性。

表 4 基于深度学习的缺陷报告摘要工作对比

Table 4 Comparison of bug report summary based on deep learning

文献	方法描述	数据集	评估指标		
			精确度	召回率	Rouge-1
[16]	使用经过训练的自动编码网络将句子转换为向量, 并将句子的可信度评分合并到向量中	SDS	0.629	0.413	0.589
[35]	根据阶梯式自动编码网络的隐藏层推断出缺陷报告摘要	SDS	0.621	0.388	0.563
[36]	句子嵌入使用Skip-Thought方法将句子转换为向量形式, 再使用 *K*-均值聚类算法将句子聚类选择摘要	SDS	—	—	0.584
[40]	使用自动编码网络提取句子特征, 再根据句子重要性因素及其相应的权重计算句子分数	SDS	0.525	0.449	—

缺陷库中存在大量重复的缺陷报告, 重复的缺陷报告是指使用不同的文本描述相同缺陷的缺陷报告集。重复缺陷报告集中选择一个缺陷报告作为主缺陷报告, 剩余缺陷报告称为重复缺陷报告。文献[45]提出了基于 PageRank 的摘要技术(PageRank-based Summarization Technique, PRST), 创建了 MBRC(Modified Bug Report Corpus)数据集和 OSCAR 数据集。PRST 方法的基本思想是, 利用主缺陷报告和重复缺陷报告之间的文本信息, 对主缺陷报告进行有效总结, 重复缺陷报告中可以提取额外的有用信息, 来进一步完善摘要。主缺陷报告和重复缺陷报告的描述和评论通过内容拆分器拆分成句子, 这些句子被传递到排名模块。PageRank 算法使用向量空间模型[46]、Jaccard[47]和 WordNet[48]三种相似性度量在 MBRC 和 OSCAR 数据集上计算主缺陷报告和重复缺陷报告句子之间的相似性, 并对句子相似

性进行排名。PRST 方法将 PageRank 算法得到的句子分数和文献[11-12]BRC 方法得出的句子分数结合起来, 根据合并排名模块获得每个句子的最终分数, 对缺陷报告中的句子进一步排序。最后, 选择排名靠前的句子组成主要缺陷报告摘要。

文献[49]提出基于关系的缺陷报告摘要(Relation-based Bug Report Summarizer, RBRS)方法。RBRS 分析并收集缺陷库中有关输入缺陷报告的重复项、阻塞项和依赖项。重复项是指与该缺陷报告重复的缺陷报告, 阻塞项是指阻碍该缺陷报告修复的缺陷报告, 依赖项是指依赖于该缺陷报告修复的缺陷报告。缺陷报告与其阻塞项和依赖项的关系相反, 如果缺陷 A 阻塞缺陷 B 或缺陷 B 依赖于缺陷 A, 则必须先修复缺陷 A, 再修复缺陷 B。接下来, RBRS 使用预处理模块对缺陷报告进行预处理, 并将预处理的缺陷报告及其关系项形成一个缺陷报告图。然后, RBRS 在缺陷报告图上执行加权 PageRank 算法, 计算每个句子的 PageRank 分数, 另外, 还使用文献[11-12]提出的 BRC 方法为每个句子计算分数。RBRS 通过排名合并模块对这两个分数进行加权合并, 最终选择排名前 25%的句子组成摘要。

讨论 3. 本节归纳了基于图的缺陷报告摘要方法, 分别从方法名称、方法描述、数据集选择和摘要评估指标等多个角度进行分析和对比, 如表 5 所示。从方法描述而言, 随机游动和马尔科夫链这两种随机过程是基于图的缺陷报告摘要研究的热点。从数据集选择上看, 基于图的缺陷报告摘要方法主要使用 SDS 数据集。从评估指标而言, 基于图的缺陷报告摘要方法生成的摘要精确度较高。

观点 3. 基于图的缺陷报告摘要方法是在文本拓扑结构图上执行随机过程, 来获得每个句子节点的权重。通过比对分析现有的研究工作, 本文发现: 1)图排序算法严重依赖于句子相似性, 所以生成摘要的句子冗余度较高; 2)由于每个句子都要计算相似性, 计算量大, 所以基于图的缺陷报告摘要方法运行速度较慢。

3.2.4 基于启发式的方法

启发式算法是相对于最优化算法提出的基于直观或经验构造的算法, 通常在可接受的时间和空间花费下给出待解决组合优化问题每个实例的一个可行解[26], 该可行解与最优解的偏离程度一般不能被预计。现阶段, 启发式算法以仿自然体算法为主, 在缺陷报告摘要领域, 主要利用粒子群算法、差分进化算法等将文本摘要问题形式化表示为优化问题, 提取最优句子形成摘要[26]。

文献[24]使用多目标差分进化算法来生成摘要。差分进化算法是一种基于群体进化的算法, 具有记忆个体最优解和种群内信息共享的特点[50], 即通过种群内个体间的合作与竞争来实现对优化问题的求解, 它的整体结构类似于遗传算法, 与遗传算法的主要区别在变异操作上, 差分进化的变异操作是基于染色体的差异向量进行的, 其余操作则和遗传算法类似[51]。文献[24]将差分进化算法用于多目标优化问题上, 利用多目标二元差分进化的搜索能力, 同时优化缺陷报告摘要的不同方面, 包括句子之间的多样性、句子相关性、句子长度以及基于关键字的目标函数, 以提高缺陷报告摘要的质量。

文献[52]使用粒子群优化算法来选择最优摘要, 粒子群优化算法属于单目标优化算法, 其基本思想是, 通过设计一种无质量的粒子来模拟鸟群中的鸟, 粒子仅有两个属性: 速度和位置, 速度代表移动的快慢, 位置代表移动的方向。每个粒子在飞行过程中所经历过的最好位置, 就是粒子本身找到的最优解。整个群体所经历过的最好位置, 就是整个群体目前找到的最优解。前者叫做个体极值, 后者叫做全局极值。粒子群中所有粒子根据当前的个体极值和全局极值来调整自己的速度和位置[53]。将缺陷报告预处理后, 使用 TF-IDF 和 n-gram 两种特征评分方法来为每个句子重要性打分。据用户输入的百分比来生成摘要, 若用户输入 30%, 则随机选取缺陷报告 30%的句子作为摘要, 会得到一个大小为 n 的摘要集, 摘要集中的每个摘要子集都由缺陷报告中 30%的句子组成, 据句子分数来计算每个摘要子集分数。摘要子集作为粒子, 摘要子集分数为最初位置, 使用粒子群优化算法在摘要集中寻找最优摘要子集。

文献[54]提出了两种无监督的抽取式摘要方法, 第一种是使用粒子群优化子集选择(Particle Swarm optimization subset Selection, PSS)方法生成摘要, 第二种是将粒子群优化子集选择和混合蚁群优化子集选择方法的结合版本(Hybrid version of Particle Swarm optimization subset Selection and Ant Colony optimization subset Selection, Hybrid PSS_ACS)应用于摘要。混合 PSS_ACS 克服了粒子群优化算法搜索能力弱和蚁群优化算法开发能力弱的缺点, 在搜索过程中, 粒子群优化算法缩小了搜索空间, 蚁群优化算法对缩小的搜索空间进行探索。文献[54]的实验表明, 混合 PSS_ACS 在缺陷报告摘要方面具有较好的性能。

讨论 4. 本节总结归纳了基于启发式的缺陷报告摘要方法, 分别从方法名称、方法描述、数据集选择

和摘要评估指标多个角度进行分析和对比, 如表 6 所示。通过对现有研究调研, 发现使用启发式算法来获得缺陷报告摘要的方法较少。将粒子群优化算法和混合蚁群优化算法结合起来生成摘要的可读性较高。

观点 4. 基于启发式的缺陷报告摘要方法是是将缺陷报告摘要问题看作一个求最优解问题, 使用启发式算法得出最优解作为摘要。通过分析对比现有的研究工作, 发现基于启发式的缺陷报告摘要方法参数较多、运算复杂, 应用于实际开发中较为困难。

表 5 基于图的缺陷报告摘要工作对比

Table 5 Figure based bug report summary work comparison

文献	方法名称	方法描述	数据集	评估指标		
				精确度	召回率	*F* 分数
[15]	Grasshopper	Grasshopper 算法在图上执行随机游动, 随机游动访问概率较高的节点对文档更重要	SDS	0.49	0.51	0.50
[15]	DivRank	DivRank 在图中也使用了随机游动, 该方法中的随机游动是一种时间变异游动	SDS	0.45	0.46	0.46
[44]	Hurried	Hurried 方法对读者认为重要的句子类型提出三个假设, 并使用马尔科夫链来模拟这三个假设	SDS (ℓ_{all})	0.71	0.30	0.41
[45]	PRST	PageRank 算法使用三种相似性度量来计算主缺陷报告和重复缺陷报告句子之间的相似性, 并对句子进行排名	MBRC (WordNet)	0.5578	0.3280	0.3989
[49]	RBRS	将预处理的缺陷报告及其关系项形成一个缺陷报告图, 之后在缺陷报告图上执行加权 PageRank 算法	—	0.5169	0.5182	0.5176

表 6 基于启发式的缺陷报告摘要工作对比

Table 6 Heuristic based bug report summary work comparison

文献	方法描述	数据集	评估指标		
			精确度	召回率	ROUGE-1
[24]	使用多目标差分进化算法来选择最优句子组成摘要	SDS	0.742	0.390	0.601
[52]	使用粒子群优化算法在摘要集中选择最优摘要子集	SDS	—	—	—
[54]	将粒子群优化子集选择和混合蚁群优化子集选择方法的结合版本选择最优摘要子集	SDS	—	—	0.926

4 缺陷报告可视化摘要

缺陷报告的主题可能会随时间变化而改变, 固定的缺陷报告摘要不能满足需求。另外, 现有缺陷报告摘要皆为抽取式摘要, 但是抽取式摘要句子不连贯, 脱离上下文很难理解。所以有研究提出缺陷报告可视化摘要, 缺陷报告可视化摘要根据关键字将缺陷报告中与关键字相关的句子标注出来, 帮助开发人员看到标注句子的上下文。固定缺陷报告摘要及可视化缺陷报告摘要对比如表 7 所示。

文献[55]提出了缺陷报告可视化分析系统, 该系统对缺陷报告执行两个可视化任务, 第一个任务展示了从多个缺陷报告中衍生出来的主题随时间的演变, 第二个任务是将摘要句子在缺陷报告中以不同的颜色显示。一旦开发人员选择了一个数据集, 该系统就会自动对数据集应用主题建模, 提取每个主题的关键词, 再提取每个时间间隔的时间敏感关键词。借助 Java 图表库 JFreeChart[56]来演变可视化主题。系统提供了搜索选项, 开发人员可以使用与主题相关的关键词搜索缺陷报告, 系统会为搜索出的每个缺陷报告计算可视化摘要, 并将可视化摘要与缺陷报告一起呈现给开发人员。

缺陷报告摘要的质量是使用人工创建的黄金标准摘要来评估的, 但是创建摘要的注释员们在创建黄金标准摘要问题上存在分歧, 并且注释员数量低于稳定注释的既定值, 这造成缺陷报告自动摘要方法不可能获得较高的评估分数。另外, 固定的摘要长度并不适用于每个缺陷报告。所以文献[3, 57]认为创建一个单一的黄金标准摘要来评估自动缺陷报告摘要的方法既不可行也不现实, 他们提出了基于任务的可视化摘要, 将缺陷报告中的句子分类为: 描述、线索词、原始句子、引用句子、主题、离题句子、URL、代码、解析。在缺陷报告界面的顶部有复选框的标签, 这些复选框允许用户根据特定的句子类

别在评论中显示或隐藏句子。用户可以选择感兴趣的句子类别, 创建与当前信息需求相关的缺陷报告的自定义摘要。

讨论 5. 固定的缺陷报告摘要不能随时间和关键词的变化而变化, 而且抽取式摘要脱离上下文不易理解。可视化摘要可以随开发人员提供的关键词的变化而变化, 而且在缺陷报告中将与关键词相关的句子标注出来有助于开发人员对可视化摘要的理解。

观点 5. 通过调研现有研究工作, 本文发现固定的缺陷报告摘要的准确性和可读性指标并不高, 用于实际应用场景中的阻碍较大。面对冗长的缺陷报告, 可视化摘要可以根据开发人员提供的关键词从缺陷报告中将相关句子标注出来, 帮助开发人员了解相关句子的上下文, 增加了可视化摘要的易懂性, 可以尝试用于实际应用场景。

表 7 固定缺陷报告摘要和可视化缺陷报告摘要对比

Table 7 Comparison of fixed bug report summary and visual bug report summary

摘要名称	摘要描述	优点	缺点
固定缺陷报告摘要	使用缺陷报告摘要方法将缺陷报告内容压缩为独立的简短文本。固定缺陷报告摘要的内容和大小不可变	独立简短的固定缺陷报告摘要可以帮助开发人员减少阅读冗长缺陷报告的时间	目前固定缺陷报告摘要技术还不成熟, 生成摘要的准确性和可读性不高
可视化缺陷报告摘要	根据关键字将相关性句子在缺陷报告中标注出来。可视化缺陷报告摘要的内容和大小可变	可视化缺陷报告摘要可以根据关键字的变化而变化, 使开发人员更容易理解	需要在缺陷报告页面添加额外的网页标签, 增加了软件缺陷库网站的维护成本

5 未来研究展望

本综述梳理和归纳了 2010 年至 2022 年 6 月期间的有关缺陷报告摘要的研究成果。一方面, 本文从基于监督学习和基于无监督学习两种方法角度缺陷报告摘要进行了分类梳理, 并在 2~3 节分别阐述了两种学习方法的抽取式缺陷报告摘要生成框架。通过调研发现现有的缺陷报告摘要研究中基于无监督学习抽取式摘要方法较多, 所以在 3.2.1~3.2.4 节分类阐述并对比了现有的基于无监督学习缺陷报告摘要研究工作, 同时给出了详尽的讨论和分析**(讨论&观点 1-4)**。另一方面, 本文从缺陷报告摘要的实用性出发, 对现有的缺陷报告可视化摘要研究成果进行总结, 具体表述详见第 4 节, 并对固定缺陷报告摘要和可视化摘要用于实际应用场景的可能性做出讨论和分析**(讨论&观点 5)**。由于缺陷报告格式的复杂性和现有自动摘要技术不完备等原因, 实现缺陷报告自动摘要仍然还有很长的路要走。相信随着新技术的发展、模型性能的提升, 缺陷报告摘要的质量将会更好, 能更有效的帮助开发人员解决软件缺陷问题。

本节基于第 2 节和第 3 节对现有研究成果及文献综述[6-9]的一些讨论和分析**(讨论&观点 1-5)**, 尝试总结缺陷报告摘要领域现阶段面临的主要挑战, 并在已有研究基础上, 对未来的重要研究方向主要从三个方面做出了展望。缺陷报告摘要领域研究的三大挑战和机遇如表 8 所示。

5.1 缺陷报告数据集

缺陷报告自动摘要生成首先面临的挑战是数据集的获取(挑战①), 通过对现阶段该领域文献调研, 发现各项研究在性能评估阶段以公开数据集 SDS 为主。现有公开的缺陷报告数据集并不多, 而且数据集往往不大, 这影响了缺陷报告自动摘要结果的可靠性。因此, 为了辅助缺陷报告自动摘要研究的发展, 必须构建一个公开规范的大型缺陷报告数据集。

5.2 抽取式摘要

抽取式摘要句子不够连贯影响了摘要的可读性(挑战②), 生成式摘要与人类撰写的摘要更相似。因现有公开数据集太小限制了缺陷报告生成式摘要的发展。为了提高缺陷报告摘要的质量, 在构建大型标准缺陷报告数据集的基础上, 应该关注缺陷报告生成式摘要的研究。

5.3 黄金标准摘要

缺陷报告自动摘要与人工编写的黄金标准摘要对比来评估自动摘要的性能。黄金标准摘要不够标准, 会造成自动摘要的评估结果不准确(挑战③)。经过调研发现, 不同注释者对缺陷报告有不同的理解, 注释者无法就黄金标准摘要达成一致, 且注释者数量低于稳定注释的既定值。因此, 为了使自动摘要的评估结果更准确, 应该增加注释者数量至稳定注释的既定值, 并且提高注释者创建黄金标准摘要一致性的要求。

表 8 缺陷报告自动摘要领域面临的挑战和机遇

Table 8 Challenges and opportunities in the area of automatic summary of bug reports

挑战	机遇
①缺陷报告数据集	构建公开规范的大型缺陷报告数据集
②抽取式摘要	在大型缺陷报告数据集的基础上, 研究生成式摘要
③黄金标准摘要	增加注释者数量, 提高注释者创建黄金标准摘要一致性的要求

6 结束语

随着文本自动摘要技术的不断发展, 将文本自动摘要技术用于缺陷报告缓解了软件开发人员阅读冗长乏味缺陷报告的问题。本文通过梳理和总结2010年至2022年6月期间缺陷报告自动摘要研究成果, 归纳其整体工作流程和技术路线, 并从无监督缺陷报告摘要方法为切入点, 将现有研究方法分类为基于特征评分方法、基于深度学习方法、基于图方法和基于启发式方法。同时也总结了缺陷报告可视化摘要的研究进展。最后, 对该领域面临的挑战和机遇进行展望。本文认为在缺陷报告摘要未来的研究工作中, 构建大型数据集对生成式缺陷报告摘要等问题进一步分析, 克服现有研究方法的局限性, 将有助于提升缺陷报告摘要的质量。

参考文献

[1] Liu W J. *Software defect reports severity prediction*[D]. Dalian: Dalian University of Technology, 2020.
(刘文杰. 软件缺陷报告严重性预测研究[D]. 大连: 大连理工大学, 2020.)

[2] Gupta S, Kumar S. A Systematic Study of Duplicate Bug Report Detection[J]. *International Journal of Advanced Computer Science and Applications*, 2021, 12(1).

[3] Anvik J, Galappaththi A. Are Automatic Bug Report Summarizers Missing the Target? [C]. *The IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020: 149-152.

[4] Hou L W, Hu P, Cao W L. Automatic Chinese Abstractive Summarization with Topical Keywords Fusion[J]. *Acta Automatica Sinica*, 2019, 45(3): 530-539.
(侯丽微, 胡珀, 曹雯琳. 主题关键词信息融合的中文生成式自动摘要研究[J]. *自动化学报*, 2019, 45(3): 530-539.

[5] Sun Y, Li J. Short Text Summarization Based on IF-PGN Model[J]. *Journal of Jiamusi University* (*Natural Science Edition*), 2021, 39(1): 41-44.
(孙岩, 李晶. 基于 IF-PGN 模型的短文本摘要生成[J]. *佳木斯大学学报(自然科学版)*, 2021, 39(1): 41-44.)

[6] Tarar M I N, Ali M, Butt W H. Bug report summarization: A systematic literature review[C]. *The 2019 11th International Conference on Education Technology and Computers*, 2019: 257-261.

[7] Hima Bindu Sri S, Dutta S R. A Survey on Automatic Text Summarization Techniques[J]. *Journal of Physics*: *Conference Series*, 2021, 2040(1): 012044.

[8] Gupta S, K G S. Comparative Study of Bug Report Summarization Techniques[J]. *Indian Journal of Computer Science and Engineering*, 2019, 10(6): 158-167.

[9] Nazar N, Hu Y, Jiang H. Summarizing Software Artifacts: A Literature Review[J]. *Journal of Computer Science and Technology*, 2016, 31(5): 883-909.

[10] Jindal S G, Kaur A. Automatic Keyword and Sentence-Based Text Summarization for Software Bug Reports[J]. *IEEE Access*, 8: 65352-65370.

[11] Rastkar S, Murphy G C, Murray G. Automatic Summarization of Bug Reports[J]. *IEEE Transactions on Software Engineering*, 2014, 40(4): 366-380.

[12] Rastkar S, Murphy G C, Murray G. Summarizing Software Artifacts: A Case Study of Bug Reports[C]. *The 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, 2010: 505-514.

[13] Bujang M A, Baharum N. Guidelines of the minimum sample size requirements for Kappa agreement test[J]. *Epidemiology, Biostatistics and Public Health*, 2017, 14(2).

[14] Yang C Z, Ao C M, Chung Y H. Towards an Improvement of Bug Report Summarization Using Two-Layer Semantic Information[J]. *IEICE Transactions on Information and Systems*, 2018, E101.D(7): 1743-1750.

[15] Mani S, Catherine R, Sinha V S, et al. AUSUM: Approach for Unsupervised Bug Report Summarization[C]. *The ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, 2012: 1-11.

[16] Liu H R, Yu Y, Li S S, et al. BugSum: Deep Context Understanding for Bug Report Summarization[C]. *The 28th International Conference on Program Comprehension*, 2020: 94-105.

[17] Liu H R, Yu Y, Li S S, et al. How to Cherry Pick the Bug Report for Better Summarization? [J]. *Empirical Software Engineering*, 2021, 26(6): 119.

[18] Jiang H, Li X C, Ren Z L, et al. Toward Better Summarizing Bug Reports with Crowdsourcing Elicited Attributes[J]. *IEEE Transactions on Reliability*, 2019, 68(1): 2-22.

[19] Lin T, Gao J H, Fu X, et al. Extraction Approach for Software Bug Report[J]. *Computer Science*, 2016, 43(6): 179-183.
(林涛, 高建华, 伏雪, 等. 面向软件缺陷报告的提取方法[J]. *计算机科学*, 2016, 43(6): 179-183.)

[20] Lin T, Gao J H, Fu X, et al. A Novel Bug Report Extraction Approach[M]. Algorithms and Architectures for Parallel Processing. Cham: Springer International Publishing, 2015: 771-780.

[21] Ramos J. Using tf-idf to determine word relevance in document queries[C]. *The first instructional conference on machine learning*. 2003, 242(1): 29-48.

[22] Huai B, Li W, Wu Q, et al. Mining Intentions to Improve Bug Report Summarization[C]. *SEKE*. 2018, 2018: 320-363.

[23] Galappaththi A, Anvik J. Feature Evaluation for Automatic Bug

Report Summarization (S)[C]. *SEKE*. 2019: 205-274.

[24] Shastri A, Saini N, Saha S, et al. MEABRS: A multi-objective evolutionary framework for software bug report summarization[C]. *2021 IEEE International Conference on Systems*, *Man*, *and Cybernetics*, 2022: 2006-2011.

[25] Cheng W, Greaves C, Warren M. From N-Gram to Skipgram to Concgram[J]. *International Journal of Corpus Linguistics*, 2006, 11(4): 411-433.

[26] Li J P, Zhang C, Chen X J, et al. Survey on Automatic Text Summarization[J]. *Journal of Computer Research and Development*, 2021, 58(1): 1-21.
(李金鹏, 张闯, 陈小军, 等. 自动文本摘要研究综述[J]. *计算机研究与发展*, 2021, 58(1): 1-21.)

[27] Rose S, Engel D, Cramer N, et al. Automatic Keyword Extraction from Individual Documents[M]. Text Mining. Chichester, UK: John Wiley & Sons, Ltd, 2010: 1-20.

[28] Bezdek J C, Ehrlich R, Full W. FCM: The Fuzzy C-Means Clustering Algorithm[J]. *Computers & Geosciences*, 1984, 10(2/3): 191-203.

[29] Radev D R, Jing H Y, Styś M, et al. Centroid-Based Summarization of Multiple Documents[J]. *Information Processing & Management*, 2004, 40(6): 919-938.

[30] Carbonell J, Goldstein J. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries[C]. *The 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 1998: 335-336.

[31] Gupta S, Gupta S K. An Approach to Generate the Bug Report Summaries Using Two-Level Feature Extraction[J]. *Expert Systems With Applications*, 2021, 176: 114816.

[32] Page L, Brin S, Motwani R, et al. The PageRank citation ranking: Bringing order to the web[R]. Stanford InfoLab, 1999.

[33] Hull D A, Grefenstette G. Querying across Languages: A Dictionary-Based Approach to Multilingual Information Retrieval[C]. *The 19th annual international ACM SIGIR conference on Research and development in information retrieval*, 1996: 49-57.

[34] Zhang Y C, Zhang R C, Liu J, et al. Network Security Situation Evaluation Using Deep Auto-Encoders Network[J]. *Computer Engineering and Applications*, 2020, 56(6): 92-98.
(张玉臣, 张任川, 刘璟, 等. 应用深度自编码网络的网络安全态势评估[J]. *计算机工程与应用*, 2020, 56(6): 92-98.)

[35] Li X C, Jiang H, Liu D, et al. Unsupervised Deep Bug Report Summarization[C]. *The 26th Conference on Program Comprehension*, 2018: 144-155.

[36] Nawaz Tarar M I, Ahmed F, Butt W H, et al. Automated Summarization of Bug Reports to speed-up software development/maintenance process by using Natural Language Processing (NLP)[C]. *2020 15th International Conference on Computer Science & Education*, 2020: 483-488.

[37] Lin Z H, Feng M W, Santos C N D, et al. A Structured Self-Attentive Sentence Embedding[EB/OL]. 2017: arXiv: 1703.03130. https://arxiv.org/abs/1703.03130

[38] Ahmad A, Dey L. A K-Mean Clustering Algorithm for Mixed Numeric and Categorical Data[J]. *Data & Knowledge Engineering*, 2007, 63(2): 503-527.

[39] Kiros R, Zhu Y K, Salakhutdinov R, et al. Skip-Thought Vectors[C]. *The 28th International Conference on Neural Information Processing Systems - Volume 2*, 2015: 3294-3302.

[40] Koh Y, Kang S, Lee S. Deep Learning-Based Bug Report Summarization Using Sentence Significance Factors[J]. *Applied Sciences*, 2022, 12(12): 5854.

[41] Mihalcea R, Tarau P. Textrank: Bringing order into text[C]. *The 2004 conference on empirical methods in natural language processing*, 2004: 404-411.

[42] Zhu X, Goldberg A B, Van Gael J, et al. Improving diversity in ranking using absorbing random walks[C]. *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. 2007: 97-104.

[43] Mei Q Z, Guo J, Radev D. DivRank: The Interplay of Prestige and Diversity in Information Networks[C]. *The 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010: 1009-1018.

[44] Lotufo R, Malik Z, Czarnecki K. Modelling the 'Hurried' Bug Report Reading Process to Summarize Bug Reports[J]. *Empirical Software Engineering*, 2015, 20(2): 516-548.

[45] Jiang H, Nazar N, Zhang J X, et al. PRST: A PageRank-Based Summarization Technique for Summarizing Bug Reports with Duplicates[J]. *International Journal of Software Engineering and Knowledge Engineering*, 2017, 27(6): 869-896.

[46] Salton G, Wong A, Yang C S. A Vector Space Model for Automatic Indexing[J]. *Communications of the ACM*, 1975, 18(11): 613-620.

[47] Niwattanakul S, Singthongchai J, Naenudorn E, et al. Using of Jaccard coefficient for keywords similarity[C]. *The international multiconference of engineers and computer scientists*, 2013, 1(6): 380-384.

[48] Miller G A. WordNet[J]. *Communications of the ACM*, 1995, 38(11): 39-41.

[49] Kim B, Kang S, Lee S. A Weighted PageRank-Based Bug Report Summarization Method Using Bug Report Relationships[J]. *Applied Sciences*, 2019, 9(24): 5427.

[50] Yang H X, Liu Z W, Wang J, et al. Modified PSO Hybrid Algorithm[J]. *Journal of Computer Applications*, 2010(6):1516-1518.
(杨恢先, 刘子文, 汪俊, 等. 改进的 PSO 混合算法[J]. *计算机应用*, 2010(6):1516-1518.)

[51] Shi Q. *Research on the multi-objective optimization problem based on differential evolution algorithm*[D]. Shanghai: Donghua University, 2016.
(侍倩. 基于差分进化算法的多目标优化问题的研究[D]. 上海: 东华大学, 2016.)

[52] Kukkar A, Mohana R. An Optimization Technique for Unsupervised Automatic Extractive Bug Report Summarization[M]. International Conference on Innovative Computing and Communications. Singapore: Springer Singapore, 2018: 1-11.

[53] Zhang L B, Zhou C G, Ma M, et al. Solutions of Multi-Objective Optimization Problems Based on Particle Swarm Optimization[J]. *Journal of Computer Research and Development*, 2004, 41(7): 1286-1291.
(张利彪, 周春光, 马铭, 等. 基于粒子群算法求解多目标优化

问题[J]. *计算机研究与发展*, 2004, 41(7): 1286-1291.)

[54] Kukkar A, Mohana R. Bug Report Summarization by Using Swarm Intelligence Approaches[J]. *Recent Advances in Computer Science and Communications*, 2020, 13(1): 53-67.

[55] Yeasmin S, Roy C K, Schneider K A, et al. Interactive visualization of bug reports using topic evolution and extractive summaries[C]. *2014 IEEE International Conference on Software Maintenance and Evolution*, 2014: 421-425.

[56] Gilbert D. The JFreeChart Class Library [J]. *Developer Guide. Object Refinery*, 2002, 7.

[57] Galappaththi A. Automatic sentence annotation for more useful bug report summarization[M]. University of Lethbridge (Canada), 2020.

刘翠兰 于 2021 年在河北农业大学网络工程专业获得学士学位。现在西安电子科技大学电子信息专业攻读硕士学位。主要研究方向为网络与信息系统安全、AI 安全。Email: liucl@nipc.org.cn

张嘉元 于 2019 年在西安电子科技大学信息安全专业获得学士学位。现在兰州理工大学计算机系统结构专业攻读硕士学位。主要研究方向为网络与信息安全和模糊测试。Email: zhangjy@nipc.org.cn

曹旭栋 于 2022 年在中国科学院大学计算机技术专业获得硕士学位。现在中国科学院大学计算机应用技术专业攻读博士学位。主要研究方向为网络与系统安全。Email: caoxd@nipc.org.cn

伍高飞 于 2015 年在西安电子科技大学获得博士学位。现任西安电子科技大学讲师, 硕士生导师。主要研究方向为网络与信息系统安全、AI 安全、密码学。Email: wugf@nipc.org.cn

朱笑岩 1979 年生, 博士, 教授, 博士生导师, 现任西安电子科技大学教授。主要研究方向为网络与信息安全。Email: xyzhu@mail.xidian.edu.cn

任家东 1967 年生, 博士, 教授, 博士生导师, CCF 高级会员, IEEE 和 ACM 会员。现任燕山大学教授。主要研究方向为数据挖掘、时态数据建模和软件安全。Email: dren@ysu.edu.cn

冯涛 1970 年生, 博士, 教授, 博士生导师, 兰州理工大学计算机与通信学院院长。主要研究方向为网络与信息安全技术、现代密码学理论。Email: fengt@lut.edu.cn