计算机科学
COMPUTER SCIENCE

基于主题一致性保持和伪相关反馈库扩展的缺陷报告重构方法

刘文杰, 邹卫琴, 蔡碧瑜, 陈冰婷

引用本文

刘文杰, 邹卫琴, 蔡碧瑜, 陈冰婷. 基于主题一致性保持和伪相关反馈库扩展的缺陷报告重构方法[J]. 计算机科学, 2024, 51(7): 1-9.

LIU Wenjie, ZOU Weiqin, CAI Biyu, CHEN Bingting. Bug Report Reformulation Method Based on Topic Consistency Maintenance and Pseudo-correlation Feedback Library Extension [J]. Computer Science, 2024, 51(7): 1-9.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230400069

基于主题一致性保持和伪相关反馈库扩展的缺陷报告重构方法

刘文杰　邹卫琴　蔡碧瑜　陈冰婷
南京航空航天大学计算机科学与技术学院　南京 211106
(wenwenmu@nuaa.edu.cn)

摘　要　为了加快开发人员定位软件缺陷,研究人员提出了一系列基于文本检索的缺陷定位技术,自动为用户所提交的缺陷报告推荐可疑的代码文件。由于用户的专业知识不同,编写的缺陷报告质量不一致,因此某些低质量的缺陷报告无法被成功定位。对低质量的缺陷报告进行重构从而改进其定位效果,是常见的解决方案。现有基于查询扩展和查询缩减的主流重构方法,容易出现重构前后查询主题不一致或所依赖伪相关库质量差导致重构质量低的问题。对此,提出了一种基于主题一致性保持和伪相关反馈库扩展的缺陷报告重构方法,由主题一致性保持的查询缩减阶段和伪相关反馈库扩展的查询扩展阶段两部分组成。查询缩减阶段将缺陷报告的概要问题描述和从问题描述文本中提取的关键词合并来解决主题不一致性问题;查询扩展阶段综合使用多种定位工具(即 Lucene,BugLocator 和 Blizzard)来获得伪相关反馈库,并从中提取查询扩展关键词,以解决现有伪相关反馈库质量差导致的重构质量低的问题;最后将查询缩减和扩展阶段的输出合并得到重构后的查询。通过在 6 个 Java 项目上进行实验发现,对于使用现有缺陷定位方法无法在 TOP 10 可疑推荐文件中定位的低质量缺陷报告,使用所提重构方法后,能定位其中 21%～39%的缺陷即 Accuracy@10,MRR@10 为 10%～16%。对比现有重构技术,所提重构方法在 Accuracy@10 和 MRR@10 两个指标上分别可以提升 7%～32%和 2%～13%。

关键词:缺陷定位;查询重构;查询缩减;查询扩展;伪相关反馈库;缺陷报告质量
中图分类号　TP311

Bug Report Reformulation Method Based on Topic Consistency Maintenance and Pseudo-correlation Feedback Library Extension

LIU Wenjie,ZOU Weiqin,CAI Biyu and CHEN Bingting
College of Computer Science and Technology,Nanjing University of Aeronautics and Astronautics,Nanjing 211106,China

Abstract　To enhance the speed of locating software bugs for developers,a set of bug location techniques based on text retrieval has been proposed. These techniques aim to automatically recommend potentially suspicious code files associated with bug reports submitted by users. However,due to varying levels of professional expertise among users,the quality of bug reports tends to be inconsistent. As a result,some low-quality bug reports cannot be successfully located. To improve the quality of those bug reports,it is common to refactor the bug reports. Existing mainstream methods for reformulation,which involve query extension and query reduction,often face issues such as inconsistent query topics before and after reformulation or the utilization of poor-quality pseudo-correlation libraries. To address this problem,this paper proposes a bug report reformulation method that focuses on maintaining topic consistency and extending pseudo-correlation feedback libraries. This method consists of two parts:the query reduction stage,which aims to maintain topic consistency through combining a concise problem description with keywords extracted from the text,and the query expansion stage,which involves using various locating tools(Lucene,BugLocator,and Blizzard) to comprehensively obtain a pseudo-correlation feedback library. From this library,additional keywords for query expansion are extracted to address the issue of low reformulation quality caused by the inadequacy of the existing pseudo-correlation feedback library. Ultimately,the outputs of the query reduction and expansion stages are combined to form the reformulated query. Through experiments conducted on six Java projects,it is discovered that for low-quality bug reports that could not be identified among the top 10 recommended files using the existing bug location method,21%～39% of them can be located using the proposed reformulation method,i.e.,Accuracy@10 and MRR@10 is 10%～16%. Compared with existing reformulation techniques,the Accuracy@10

到稿日期:2023-04-11　返修日期:2023-07-31
基金项目:国家自然科学基金(62002161,62272225);南京航空航天大学前瞻布局科研专项资金
This work was supported by the National Natural Science Foundation of China(62002161,62272225) and Fund of Prospective Layout of Scientific Research for Nanjing University of Aeronautics and Astronautics.
通信作者:邹卫琴(weiqin@nuaa.edu.cn)

and MRR@10 of the proposed reformulation method can improve by 7%～32% and 2%～13%, respectively.

Keywords Bug localization, Query reformulation, Query reduction, Query expansion, Pseudo-correlation feedback libraries, Quality of bug report

1 引言

软件缺陷修复是整个软件产品生命周期中的关键环节之一，由于软件规模和复杂性的增加，开发人员需要耗费许多精力来对缺陷进行定位。为了帮助开发人员快速且准确地定位软件缺陷，研究人员提出了许多自动为缺陷报告推荐可疑文件的缺陷定位方法。目前，基于文本检索的缺陷定位方法是现有缺陷定位技术的一类主流方法[1-3]，该类方法主要是将缺陷报告作为查询，将源代码库作为语料库，通过计算缺陷报告与源代码库中文件之间的文本相似性来获取其对应的可疑文件推荐列表，其定位效果较好且速度较快。然而，受到缺陷报告质量的影响，某些缺陷报告在该类缺陷定位方法中无法成功定位，导致开发人员在定位这类缺陷时耗费大量时间和精力。因此，提高缺陷报告的质量是近年来的研究热点。

查询重构被广泛认为是提高缺陷报告质量和改善基于文本检索缺陷定位方法性能的有效方法。查询重构方法主要分为查询缩减和查询扩展两类。查询缩减方法中，一类是通过单词的长度、位置以及词性等来判断单词是否有意义，并删除带有噪声的词，从而缩小查询范围，提高检索质量[4]；另一类是根据单词的频率以及单词之间的共现关系等提取关键词，使用关键词替换原始查询[5]。查询扩展方法是对查询进行扩充，以获得更多与语料库相关的文本信息，提高检索准确性。其中，一类方法是根据开发人员提供的查询词扩展查询；另一类方法是从伪相关反馈库中根据单词的统计信息及单词的共现关系计算单词权重，将权重高的单词作为扩展词[6-8]。

现有的查询重构方法能够有效提升缺陷报告质量，但仍旧存在一些问题。对于使用关键词替换原始查询的查询缩减方法，现有方法依据单词频率等统计信息和单词依赖关系从缺陷报告中提取出关键词，并使用关键词构建重构后的查询。然而，由于缺陷报告中包含了许多主题无关词，使用关键词替换的查询减少方法无法区分该关键词是否为主题相关词，因此可能导致使用关键词替换的重构后的查询偏离主题。对于现有使用伪相关反馈的查询扩展技术，伪相关反馈库指通过缺陷报告运行缺陷定位工具获取前 K 个文件，再从伪相关反馈库中获取重要单词来扩展查询。由于现有方法的伪相关反馈库无法保证前 K 个文件中包含与缺陷报告相关的缺陷文件，因此从伪相关反馈库中获得的关键词可能与该缺陷报告没有很大关联。

针对上述存在的查询重构前后主题不一致和伪相关反馈库质量差的问题，本文提出了一种基于主题一致性保持和伪相关反馈库扩展的缺陷报告重构方法，由主题一致性保持的查询缩减阶段和伪相关反馈库扩展的查询扩展阶段组成。在主题一致性保持的查询缩减阶段，本文首先根据缺陷报告中单词的共现关系来构建文本图，接着使用 TextRank 算法从文本图中提取前 K 个关键词，最后将缺陷报告中的问题概要描述与前 K 个关键词合并作为查询缩减策略的输出。在伪相关反馈扩展的查询扩展阶段，本文首先使用 BugLocaor，Blizzard 以及 Lucene 这 3 种工具获得的可疑文件推荐列表构成其伪相关反馈库；然后根据伪相关反馈库中源代码文件中单词的共现性构成文本图，使用 CodeRank 算法从文本图中获得扩充词，并将其作为查询扩展的输出；最后将查询缩减的输出与查询扩展的扩充词合并作为一个重构后的查询。为了验证方法的有效性，在 6 个不同的 Java 项目数据集上进行实验并评估，实验结果表明，所提方法对于实验数据集的低质量缺陷报告的 Accuracy@10 指标和 MRR@10 指标分别为 21%～39%，10%～16%。对比现有重构技术，本文方法在 Accuracy@10 和 MRR@10 两个指标上分别可以提升 7%～32%和 2%～13%。

本文的主要贡献如下：

1）提出了基于主题一致性保持和伪相关反馈库扩展的缺陷报告重构方法，并在 6 个项目数据集上进行了实验。实验结果表明，对于这些低质量的缺陷报告，该方法的 Accuracy@10 指标能达到 21%～39%，MRR@10 指标能达到 10%～16%。

2）对比现有的查询重构方法，该方法的 Accuracy@10 指标的提升范围是 7%～32%，MRR@10 指标的提升范围是 2%～13%。

2 研究背景和相关工作

2.1 查询重构

目前许多研究使用查询重构来提升缺陷报告质量，这些研究分为查询缩减方法和查询扩展方法。查询缩减方法往往从初始查询中删除无用词或使用同义词库方式以及从缺陷报告中提取关键词的方法来进行查询重构。查询扩展方法往往应用来自开发人员的相关反馈、运行检索工具获得伪相关反馈库，并从中提取关键词及使用查询同义词库等重新制定查询。

对于使用查询缩减来重新制定查询，往往通过删减噪声词来提升查询质量。Kevic 等[4]考虑了一系列启发式方法，如频率、位置、词性和任务描述中术语的符号，并使用一个预测模型从变更请求中识别搜索词。Chaparro 等[9]发现，使用 OB 和报告标题的组合内容替代整份缺陷报告来定位能够达到更好的效果。Rahman 等[5]根据单词共现和词性依赖关系提出使用 TextRank 和 POSRank，从变更请求中确定合适的搜索词。至于使用同义词库来做查询缩减策略，Howard 等[10]根据注释和源代码文件的方法名来构建特定软件工程上的单词相似性库。由于只能挖掘出数量较少的单词相似性对，Tian 等[11]利用 StackFlow 帖子的文本内容，根据单词之间的相似性构建软件工程领域的单词相似性库。Cao 等[12]提出了一种基于深度学习的、特定于软件的自动查询重构方法，利用 Stack Overflow 提供的查询日志，构建了一个大规模

的查询重构语料库。本文提出的缺陷报告重构方法的查询缩减阶段，相比现有的查询缩减方法，既使用了 TextRank 提取关键词来减少噪声对缺陷报告质量的影响，也使用缺陷报告的问题概要描述来保持重构后的查询与初始查询的主题一致性。

对于应用查询扩展来扩展查询，Lemos 等[13]使用 WordNet 和一个包含软件相关词关系的同义词库来自动查询扩展。Hill 等[14]将方法或字段签名中查询术语的存在作为其相关性的指标，并建议使用它们的自然语言短语作为重新表述的查询。Sisman 等[15]选择代码文件中固定窗口大小内与查询术语共存的单词，并将这些单词作为重构词。Rocchio[6]和 Carpinteo 等[7]使用基于 TF-IDF 的指标来确定一个术语的重要性。Rahman 等[8]将一个初始查询作为输入，使用一个新的术语权重从源代码中识别出适当的搜索词，然后使用文档结构、查询质量分析和机器学习对初始查询提出最佳的重新表述。Rahman 等[16]提出了 Blizzard 方法，首先根据缺陷报告中包含的内容进行分类，接着将分类后的缺陷报告根据不同的关键词提取算法提取出关键词进行查询重构。本文提出的缺陷报告重构方法中的伪相关反馈库扩展阶段，相比现有的查询扩展方法不但使用了 Lucene 构建伪相关反馈库，而且使用了 BugLocator 和 Blizzard 扩展构建伪相关反馈库，以提高从伪相关反馈库中提取的关键词与缺陷报告之间的关联性。

2.2 基于文本检索的缺陷定位方法

基于文本检索的缺陷定位（TRBL）在现有的故障定位方法中起着核心作用。现有的 TRBL 研究主要采用文本检索方法，通过基于缺陷报告信 息检索相关文档来定位故障。与动态缺陷定位方法不同，基于文本检索的缺陷定位不需要执行程序来获取相关的执行信息。

目前，研究者已经提出一系列静态缺陷定位方法，主要分为下列 4 类：1）变更检索模型，找到适用于缺陷定位的模型；2）使用更多特征来提升模型性能；3）通过查询重构来提升缺陷定位方法的性能（如 2.1 节所述）；4）应用深度学习方法进行缺陷定位。

对于哪种模型在缺陷定位上更合适这一问题，许多研究者将不同的文本检索模型应用于缺陷定位上，如向量空间模型（VSM）[17]、隐含狄利克雷分布主题模型（LDA）[18]、潜在语义分析模型（LSI）[19]等。Thomas 等[20]探究了文本检索模型中不同的主题数目对缺陷定位性能的影响。

在使用更多特征来提升模型性能的研究上，Zhou 等[1]利用向量空间模型（VSM）和缺陷报告之间的相似性特征提出了 BugLocator 方法。Wong 等[2]建议使用代码执行过程中产生的堆栈信息特征，利用堆栈信息中的代码文件是更可疑的特征来提升缺陷定位性能。Wang 等[3]指出最近导致错误的文件在不久的将来很可能会导致其他错误，根据项目的版本历史特征、相似性特征和结构特征来定位缺陷。

随着深度学习技术的发展，Lam 等[21]将深度学习方法与向量空间模型结合来定位缺陷，使用深度学习方法来提取缺陷报告与源代码的语义信息。Xiao 等[22]通过将缺陷报告和源代码文件字符集编码，提出了一个基于字符级别的深度学习技术。Qiu 等[23]使用神经语言模型来捕获变更集中代码行的语义，进而学习代码行的自然性，获得每个代码行的可疑分数。

3 基于主题一致性保持和伪相关反馈库扩展的缺陷报告重构方法

3.1 整体框架

图 1 给出了一种基于主题一致性保持和伪相关反馈库扩展的缺陷报告重构方法的整体架构。该框架由两部分组成，即主题一致性保持的查询缩减阶段和伪相关反馈库扩展的查询扩展阶段。对于任意的一个低质量缺陷报告，首先将缺陷报告的问题概要描述和问题详细描述字段作为查询，将查询输入到查询缩减阶段过程中获得查询缩减的输出（见 3.2 节），然后将查询输入到查询扩展阶段获得其扩展词（见 3.3 节），最后将查询缩减的输出与扩展词合并构成重构后的查询（见 3.4 节）。使用重构后的查询运行 Lucene 文本检索工具，获得其可疑文件推荐列表。

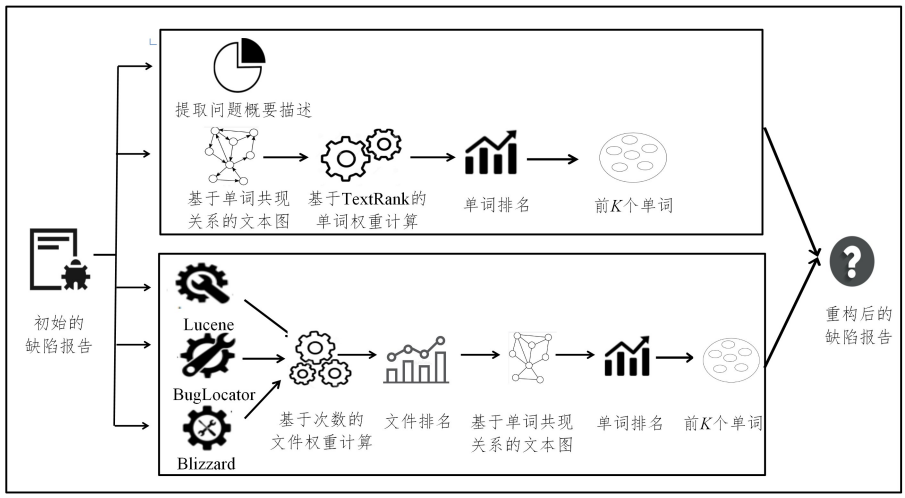


图 1 基于主题一致性保持和伪相关反馈库扩展的缺陷报告重构方法的整体架构

Fig. 1 Overall architecture of bug report reformulation method based on topic consistency maintenance and pseudo-correlation feedback library extension

3.2 主题一致性保持的查询缩减方法

3.2.1 提取问题概要描述

为了减小噪声词对查询质量的影响，一些研究人员提出从原始查询中抽取关键词作为新的查询。然而，现有查询缩减方法无法区分关键词是否为主题相关词，导致重构后的查询偏离主题进而影响查询质量。由于缺陷报告的问题概要描述通常简要概括了缺陷报告内容，因此在主题一致性保持的查询缩减阶段，我们不仅使用 TextRank 方法提取关键词来减小噪声词的影响，而且使用缺陷报告的问题概要描述来帮助保持重构后的查询与初始查询主题的一致性。

图 2 给出了从 Zookeeper 项目的缺陷跟踪系统中收集的缺陷报告，缺陷报告通常是由 BugID(如图 2 中的数字 3，用来唯一标识缺陷报告)、问题概要描述(summary)、问题详细描述(description)等字段组成，基于文本检索的缺陷定位方法往往使用缺陷报告中的 summary 和 description 字段作为查询来代替整个缺陷报告。然而，summary 和 description 中通常包含了许多无用词，由于 summary 是缺陷报告中的问题概要描述，简要地描述了缺陷的性质等信息，是缺陷报告中最重要的部分之一。因此，本文在查询缩减时采用 summary 来代替整个缺陷报告，使用 summary 来帮助保持重构后的查询与初始查询之间主题的一致性。

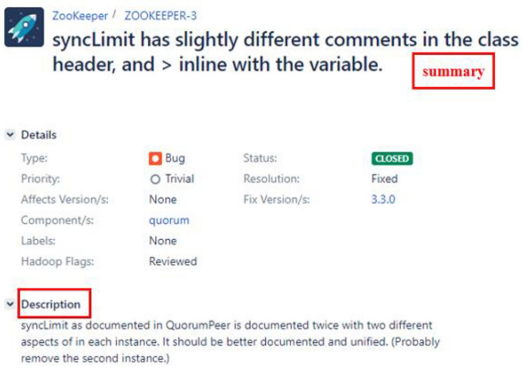


图 2 Zookeeper 缺陷报告片段示例

Fig. 2 Example of bug report in Zookeeper

3.2.2 查询文本图构建

对于关键词的提取，本文首先从缺陷报告中提取 summary 和 description，接着以句子划分 summary 和 description 中的内容并收集每个缺陷报告中的所有句子，然后使用标准的预处理操作(即分词、删除停止词、去符号、去数字、大写转小写)。分词是按照空格划分单词并分割以点组成的单词以及按照驼峰命名的词。Dit 等[24]提出的以点组成的单词(org. hibernate. mysql. Dialect)通常包含不同的技术术语，因此本文应该划分它们来进行单独的分析(org，hibernate，mysql，DiaLect)。由于缺陷报告中也经常包含驼峰命名的单词，源代码文件中对变量和方法名的命名通常会遵循驼峰命名的方式。为了减小缺陷报告与源代码文件的语义差异，将每个以驼峰命名的单词(createSystemView)划分成单独的单词(create，System，View)，接着根据从 summary 和 description 中划分后的单词之间的共现性构成文本图。文本图是一种文本表示方法，其中文本中的每个唯一术语(如单词)被表示为一个节点，而文本中的术语共现性则用节点之间的边表示。文本单词之间的共现关系的基本思想是，在一个固定窗口内同时出现的所有术语之间都存在语义关联或依赖关系[25]。例如，对于预处理后的单词短语“Illegal argument”，可以发现该短语在语义上是相互依赖的，使用单词共现在统计学上可以获取这种依赖。在具体实验中，我们设置滑动窗口大小为 2 作为单词的语义单元[25]，来捕获共现单词之间的依赖关系，最后将单词之间的共现关系编码到文本图的边上。图 3 给出了这个 BugID 为 3 的缺陷报告的 summary 和 description 构建的文本图。

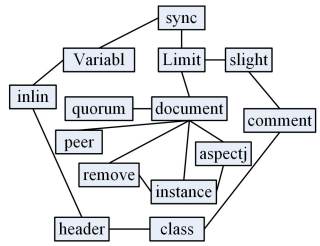


图 3 根据单词共现关系构建的文本图

Fig. 3 Text graph based on word co-occurrence

3.2.3 TextRank 计算单词权重

在将预处理之后缺陷报告的 summary 和 description 构成文本图之后，我们使用 TextRank 算法来计算每个单词顶点的权重。TextRank 算法是基于图的排序算法，是由网页重要性排序算法 PageRank 改进而来。PageRank 被提出[26]用于 web 链接分析，这是一种迭代求解算法，根据网页重要性进行排序。TextRank 具体的计算式如下：

$$W(v_i)=(1-\varphi)+\varphi\sum_{j\in V_{(v_i)}}\frac{W(v_j)}{|V_{(v_j)}|} \tag{1}$$

其中，$V_{(v_j)}$ 表示所有与顶点 j 相连接的顶点；φ 表示阻尼系数，取值范围为 0～1，代表文本图中顶点到其他任意顶点之间的概率。对于参数的设置，本文中将阻尼系数设置为0.85，并使用默认值 0.25 初始化图中的每个顶点得分，然后运行算法的迭代版本，直到所有项的分数收敛于某个阈值或达到最大迭代极限(即 Blanco 等[27]建议的 100)，并使用了一个阈值为 0.0001 的启发式阈值来进行收敛性检查[25]。

对文本图使用 TextRank，为图中每个单词顶点计算分数后，将该分数作为单词顶点的重要性，接着对每个单词顶点按照重要性排名，获取 TOP 10 个单词作为查询中的关键词，最后将 summary 与 TOP 10 个关键词合并构成查询缩减的输出。

3.3 伪相关反馈库扩展的查询扩展方法

3.3.1 伪相关反馈库

为了扩充查询，通常需要添加一些具有意义且与主题相关的单词。为此，许多基于伪相关的反馈方法被提出，该类方法的主要思想是对初始查询运行检索工具，获得推荐列表，将推荐列表的前几个文件作为伪相关反馈库，从中选出关键词来扩展查询。

然而，现有查询重构技术只使用单独的定位工具本身(如 Lucene 检索工具)来构建伪相关反馈库，对于质量差的缺陷报告，其伪相关反馈库不存在对应的缺陷文件。因此，从这样

的伪相关反馈库中提取的重要单词可能与缺陷报告无关。Zou 等[28]指出基于文本检索缺陷定位工具之间具有互补性,在 Lucene 中定位不到的缺陷报告可能在其他基于文本检索缺陷定位方法中成功定位。而 BugLocator 和 Blizzard 的运行速度较快,且对 Lucene 具有一定的互补性,因此本文使用 BugLocator 和 Blizzard 对伪相关反馈库进行互补,构建高质量的伪相关反馈库来解决这个问题。具体来说,对于任意给定的一个缺陷报告,获取其 summary 和 description 字段,运行 Lucene,BugLocator 和 Blizzard 这 3 种文本检索缺陷定位工具,获取其对应的 3 个可疑文件推荐列表。我们选择 3 个可疑文件推荐列表的前 10 个文件,按照它们在不同工具中出现的次数对文件进行排名,选择排名前五的文件作为伪相关反馈库。

3.3.2 CodeRank 计算单词权重

在获得缺陷报告的伪相关反馈库之后,将提取伪相关反馈库中源文件的方法签名和字段签名[13]。由于开发人员通常将他们在代码和域词汇表背后的意图编码到精心设计的标识符名称中,其中连接了多个术语,因此,需要将标识符划分成每一个单词,例如方法名 addSourceLineTask 可以转换成(add,Source,Line),每个单词之间相互共现传递着一个信息,它们是语义相关联的。本文对这些短语进行了进一步的分析,并利用这些术语之间的共现关系来构建文本图。

在获取到每个候选词后,对每个候选词进行验证,验证候选词是否是有效的(删除单词长度小于 2 和停止词的候选词)。接着取出每个唯一的候选词作为图的顶点,设置滑动窗口大小为 2 作为单词的语义单元,根据单词之间的共现性构建两个单词顶点之间的边。

对于上述根据源代码文件标识符构成的文本图,我们应用 CodeRank 来计算每个顶点的权重,将上述计算单词权重的 TextRank 应用到源代码文件中,称之为 CodeRank,接着可以计算出每个单词顶点的分数,将每个单词顶点获得的分数作为它们的权重,再将单词按照权重进行排名,取前 5 个单词作为查询扩展词。

3.4 组合和推荐

在查询缩减和查询扩展阶段之后,将 summary 与查询缩减使用 TextRank 算法获取的前 10 个单词以及查询扩展提取的前 5 个单词合并构成重构后的缺陷报告,再用重构后的缺陷报告运行 Lucene 获取可疑文件列表,查看可疑文件列表中的前 10 个文件中是否包含与缺陷报告相关的缺陷文件。

4 实验评估

4.1 数据集构建

本文在 6 个开源项目上进行实验,即 Zookeeper,Tomcat,Openjpa,Hibernate,Lucene 和 Aspectj。这些项目大多被应用于评估基于文本检索的缺陷定位技术的实验效果[29-30],它们来自不同的领域且具有不同的项目规模,能较好地评估所提技术的泛化性。我们从这些项目的缺陷跟踪系统(JIRA 和 Blizzard)中爬取截至 2018 年 12 月的 7 609 个状态为修复的缺陷报告。接着,为每个缺陷报告链接到其对应的缺陷文件,而任意缺陷报告对应的修复 commit 中修复的源代码文件,被认为是该缺陷报告所对应的缺陷文件。通过人工分析缺陷报告与 commit 之间的关系,我们发现开发者往往在 commit log 中添加项目名-缺陷报告的 ID,来将该 commit 对应的缺陷报告告知他人。因此本文首先使用启发式规则 projectName-bugid 来链接缺陷报告与其对应的缺陷文件。然后对于使用启发式规则无法链接到缺陷文件的缺陷报告,我们进一步在 commit log 中通过搜索 BugID 的方式获取到 BugID 的可疑文件。经过两步搜索的策略,获取到每个缺陷报告的可疑缺陷文件,接着人为验证每个缺陷报告的可疑文件是否为其缺陷文件。对于这些链接到多个 commit 的缺陷报告,或与其他缺陷报告共享同一个 commit 的缺陷报告,是不能确定其缺陷文件的,因此将其丢弃。

对于上述剩下的缺陷报告,我们提取其 summary 和 description 字段作为查询,接着对每个查询及其所对应的源代码文件进行预处理,然后运行 Lucene 文本检索工具(Lucene 是基于倒排索引的搜索引擎,广泛应用于信息检索等领域),获得每个缺陷报告所对应的可疑文件推荐列表。如果在前 10 个可疑文件中能找到缺陷文件,则认为其质量好。删除这些质量好的缺陷报告,因为这些缺陷报告不需要被重构。对于不能在前 10 个可疑文件中找到缺陷文件的缺陷报告,我们取缺陷报告的 summary 和 description 作为查询来构建实验数据集(Baseline)。最终我们收集了 1 956 个缺陷报告,如表 1 所列。

表 1 实验数据集

Table 1 Experimental datasets

Project	Queries
Asepctj	328
Lucene	402
Hibernate	594
Openjpa	233
Tomcat	296
Zookeeper	103

4.2 实验问题

本文实验希望能解决以下几个问题:

问题 1:本文提出的查询重构方法性能如何?

问题 2:主题一致性保持的查询缩减性能如何?

问题 3:基于伪相关反馈库扩展的查询扩展性能如何?

4.3 实验评估标准

为了评估本文方法的性能,我们使用 Accuracy@K,Mean Reciprocal Rank@K 和 Query Effectiveness 这 3 个指标来衡量本文方法的性能,这些指标经常被用于评估查询重构策略的有效性[5,8]。3 个评估指标的具体描述如下:

1)Accuracy@K,表示在可疑文件列表中排名的前 k 个文件中至少包含一个缺陷文件的概率,TOP k 的值越大,表示该方法的检索性能越好。

2)Mean Reciprocal Rank@K(MRR@K),通常用来衡量检索方法性能的一个指标,用来评估系统能否将最相关的可疑文件放在最前面,该指标先对每个查询的缺陷文件的推荐

位置取倒数，再对所有查询取平均值。

$$MRR=\frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \tag{2}$$

MRR 的值越大，表示该方法的性能越好。

3)Query Effectiveness(QE)，定义为查询返回的第一个缺陷文件的排名，如果一个排名的 effectiveness 分数越低，这个查询越好。

5 实验结果

5.1 本文提出的查询重构方法的性能

为了回答问题 1，首先展示本文提出的缺陷报告重构方法在实验数据集上的效果。其次，展示本文方法与当前效果最好的查询缩减方法 Strict 以及与本文提出的查询扩展阶段最相关的查询扩展方法 ACERcomb 的实验结果对比。最后通过具体的案例分析展示本文方法的有效性。

如表 2 所列，本文提出的缺陷报告重构方法，对于 6 个项目的实验数据集，有 57%～67%的重构后缺陷报告的缺陷文件排名有所提升，有 32%～42%的重构后缺陷报告的缺陷文件排名下降。这些结果表明，本文提出的缺陷报告重构方法能够提升实验数据集缺陷报告对应的缺陷文件排名。

表 2 缺陷报告重构方法在 Query Effectiveness 指标上的结果

Table 2 Results of bug report reformulation method on query effectiveness metric

Project	Queries	(QE) Improvement	(QE) Worsening	(QE) Preserving
Asepctj	328	176(57.0%)	129(41.7%)	4(10.3%)
Lucene	402	244(67.4%)	117(32.3%)	1(0.27%)
Hibernate	594	362(63.3%)	207(36.2%)	3(0.5%)
Openjpa	233	139(62.6%)	81(36.5%)	2(0.9%)
Tomcat	296	187(65.8%)	90(31.7%)	7(2.5%)
Zookeeper	103	62(64.6%)	34(35.4%)	0(0%)
	Total=1 956	Avg=63.5%	Avg=35.6%	Avg=2.4%

如表 3 所列，我们提出的缺陷报告重构方法，在 MRR@10 和 Accuracy@10 指标上相比 Baseline 和 Strict 以及 ACERcomb 方法有所提升。在实验数据集的 6 个项目中，我们提出的缺陷报告重构方法的 Accuracy@10 和 MRR@10 指标分别为 21%～39%和 0.1～0.16。这些结果表明了本文方法是有效的，可以提升实验数据集的缺陷报告质量。其次，对比先进的 Strict 方法，本文提出的缺陷报告重构方法在 Accuracy@10 和 MRR@10 两个指标上分别可以提升 7%～12%和 2%～5%。最后，对比 ACERcomb 方法，本文所提的缺陷报告重构方法在 Accuracy@10 和 MRR@10 两个指标上分别可以提升 17%～32%和 8%～13%。

表 3 缺陷报告重构方法在 MRR@10 和 Accuracy@10 指标上的结果

Table 3 Results of bug report reformulation method on MRR@10 and Accuracy@10 metric

Project	Baseline		ACERcomb		Strict		Our Method	
	MRR@10	*Accuracy*@10%	*MRR*@10	*Accuracy*@10/%	*MRR*@10	*Accuracy*@10/%	*MRR*@10	*Accuracy*@10/%
Asepctj	0.03	0	0.02	3.2	0.06	11.0	0.10	20.7
Lucene	0.03	0	0.03	3.6	0.11	23.2	0.16	35.4
Hibernate	0.03	0	0.03	2.8	0.08	14.9	0.12	26.6
Openjpa	0.03	0	0.03	2.7	0.09	19.8	0.14	26.6
Tomcat	0.03	0	0.04	3.2	0.09	24.3	0.15	32.0
Zookeeper	0.04	0	0.05	7.3	0.13	26.0	0.15	38.5

从表 4 中的示例 1 可以发现，对于 BugID 为 6272 的缺陷报告，其缺陷报告中不包含问题的详细描述。如果使用 Strict 方法对其进行重构，重构后的查询信息量不充足，若使用 ACERcomb 方法进行重构，在增加查询信息量的同时也会引入新的噪声。从表 4 中的示例 2 可以发现，对于 BugID 为 9202 的缺陷报告，其缺陷报告的详细问题描述中不仅包含描述信息，还包括堆栈信息和日志文件。从 Strict 重构后的查询可以发现，重构后查询的主题偏离了初始查询主题。而从 ACERcomb 的重构后查询中可以发现，当初始查询包含太多噪声时，使用该方法不仅不能减少噪声，反而增加了噪声。相比之下，对于表中的两个示例，本文提出的缺陷报告重构方法不仅有助于保持重构后查询与初始查询主题的一致性，而且该缺陷报告重构方法返回了部分相同的关键词，可以提高部分关键词的重要性，更有助于缺陷定位。

表 4 两种不同重构方法对缺陷报告重构的例子

Table 4 Examples of bug report reformulation by two different methods

示例 1	示例 2
BugID:6272 项目:Hibernate-orm 概要问题描述:More logging fix ups 详细问题描述: 修复文件:hibernate-core/src/main/java/ org/hibernate/id/factory/DefaultIdentifierGeneratorFactory.java Strict:fix log ACERcomb:fix log bind collection parameter type position Our Method:fix log fix log log name bind cache fix	BugID:9202 项目:Hibernate-orm 概要问题描述:NPE in OneToOneLinkTest on oracle12c 详细问题描述: Stacktrace:java.lang.NullPointerException at org.hibernate.test.onetoone.link.OneToOneLinkTest.testOneToOneViaAssociationTable(OneToOneLinkTest.java:76) at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) Output:17:20:44 587INFO Configuration:709-HHH000221:Reading mappings from resource:org/hibernate/test/onetoone/link/Person.hbm.xml17:20:44 907 DEBUG SQL:104-drop sequence hibernate_sequence 修复文件:core/src/test/java/ org/hibernate/test/onetoone/ link/ OneToOneLinkTest.java Strict:person acquire alb node employe debug info walker sun reflet ACERcomb:npe one one link test oralce stacktrace null point exception hibernate test one one link one one link test test one one via association table one one……name test queue remove entry Our Method:npe one one link test oracle node employe factory join jdbc sql debug drop hibernate table link factory connect driver jdbc

综上，本文提出的缺陷报告重构方法可以有效地提升缺陷报告的质量，其次本文方法对比 Baseline，Strict 和 ACER-comb 方法都有显著提升。

5.2 本文提出的主题一致性保持的查询缩减的性能

本节首先展示了主题一致性保持的查询缩减策略在实验数据集上的效果。主题一致性保持的查询缩减策略是由缺陷报告的问题概要描述即 summary 和 TextRank 提取关键词两个部分组成。为了了解这两个部分对主题一致性保持的查询缩减阶段性能的影响，将主题一致性保持的查询缩减策略与单独使用 summary 作为重构后的查询，以及使用从查询中用 TextRank 算法提取 10 个关键词的查询缩减方法进行对比实验。

如表 5 所列，使用本文提出的主题一致性保持的查询缩减方法，在 6 个项目的实验数据集上，有 60%～70%的重构后缺陷报告的缺陷文件排名有所提升，有 27%～39%的重构后的缺陷报告的缺陷文件排名下降。这些结果显示，本文提出的主题一致性保持的查询缩减方法能够提升实验数据集缺陷报告对应的缺陷文件排名。

表 5 主题一致性保持的查询缩减策略在 Query Effectiveness 指标上的结果

Table 5 Results of query reduction strategies with topic consistency maintenance on Query Effectiveness

Project	Queries	(QE) Improvement	(QE) Worsening	(QE) Preserving
Asepctj	328	185(59.9%)	119(38.5%)	5(1.6%)
Lucene	402	249(68.8%)	111(30.7%)	2(0.6%)
Hibernate	594	397(69.4%)	166(29.0%)	9(1.6%)
Openjpa	233	146(65.8%)	73(32.9%)	3(1.4%)
Tomcat	296	195(68.7%)	81(28.5%)	8(2.8%)
Zookeeper	103	62(64.6%)	26(27.1%)	8(8.3%)
	Total=1956	Avg=66.2%	Avg=31.1%	Avg=2.7%

如表 6 所列，本文提出的主题一致性报告的查询缩减策略，在 MRR@10 和 Accuracy@10 指标上对比单独使用 summary 作为重构后的查询以及从缺陷报告中使用 TextRank 提取 10 个关键词的查询缩减方法有所提升。对比 summary 方法，本文提出的基于主题一致性的查询缩减方法在 Accuracy@10 和 MRR@10 两个指标上分别可以提升 3%～9%和 2%～5%。其次对比使用 TextRank 提取 10 个关键词的方法，本文提出的基于主题一致性的查询缩减方法在 Accuracy@10 和 MRR@10 两个指标上分别可以提升 4%～9%和 1%～6%。

表 6 主题一致性保持的查询缩减策略在 MRR@10 和 Accuracy@10 指标上的结果

Table 6 Results of query reduction strategy with topic consistency maintenance on MRR@10 and Accuracy@10

Project	Summary		Keyword10(TextRank)		Summary+Keyword10(TextRank)	
	MRR@10	*Accuracy*@10/%	*MRR*@10	*Accuracy*@10/%	*MRR*@10	*Accuracy*@10/%
Asepctj	0.09	15.9	0.07	14.6	0.08	18.8
Lucene	0.17	30.4	0.12	27.6	0.18	35.1
Hibernate	0.13	22.9	0.08	17.8	0.13	26.2
Openjpa	0.09	20.3	0.11	21.2	0.13	25.7
Tomcat	0.12	26.8	0.12	26.4	0.15	31.0
Zookeeper	0.19	31.3	0.12	32.3	0.17	40.6

综上，我们提出的主题一致性的查询缩减方法对比单独使用 summary 和单独使用 TextRank 提取关键词的方法的效果有所提升。

5.3 本文提出的伪相关反馈扩展的查询扩展的性能

为了回答这个问题 3，我们先是对比了使用 Lucene 和 BugLocator 以及 Blizzard 构成的伪相关反馈库与只使用 Lucene 构成的伪相关反馈库，包含缺陷文件的情况以及缺陷文件的排名位置。进行上述分析，旨在说明 BugLocator 和 Blizzard 对 Lucene 存在定位结果的互补性，这为本文提出将其定位结果用于扩展伪相关反馈库提供了数据支撑。为了探究 BugLocator 和 Blizzard 在多大程度上能改善查询重构的效果，我们对比了基于 Lucene 的伪相关反馈库和同时基于 Lucene，BugLocator，Blizzard 构建的伪相关反馈库的查询重构性能。

如表 7 所列，我们提出使用 BugLocator 和 Blizzard 与 Lucene 共同构建伪相关反馈库是可能包含缺陷文件的。在 TOP 5 上，BugLocator，Blizzard 以及 BugLocatr 和 Blizzard 对 Lucene 的互补范围分别为 25%～45.8%，27.6%～56.7%，40.1～67.2%。在 TOP 10 上，BugLocator，Blizzard 以及 BugLocatr 和 Blizzard 对 Lucene 的互补范围分别为 33%～50.7%，35.6%～62.7%，51.1%～72.4%。

表 7 BugLocator 和 Blizzard 对 Lucene 方法的互补性结果

Table 7 Complementary results of Lucene method by BugLocator and Blizzard

(%)

Project	BugLocator		Blizzard		BugLocator+Blizzard	
	TOP-5	TOP-10	TOP-5	TOP-10	TOP-5	TOP-10
Asepctj	25.0	44.7	27.6	44.3	40.1	57.5
Lucene	45.8	50.7	56.7	62.7	67.2	72.4
Hibernate	26.8	36.2	31.3	37.5	45.9	52.9
Openjpa	30.1	33.0	29.1	35.6	42.9	51.1
Tomcat	33.8	40.2	42.4	50.0	54.4	61.1
Zookeeper	40.8	46.6	46.4	50.5	57.7	63.1

如表 8 所列，使用本文提出的由 Lucene，BugLocator 和 Blizzard 构成的伪相关反馈库做查询扩展比仅使用 Lucene 构建的伪相关反馈库做查询扩展的效果更好。本文提出的伪相关反馈库扩展的查询扩展方法比仅使用 Lucene 构建的伪相关反馈库做查询扩展在 Accuracy@10 和 MRR@10 两个指标上分别可以提升 0～8%和 0.1%～1%。

综上所述，使用 Lucene，BugLocator 和 Blizzard 构建的伪相关反馈库更有可能包含缺陷文件。此外本文提出的伪相关反馈库扩展的查询扩展方法是有效的。

表 8 伪相关反馈库扩展的查询扩展策略在 MRR@10 和 Accuracy@10 指标上的结果

Table 8 Results of pseudo-correlation feedback-based query expansion strategies on MRR@10 and Accuracy@10

Project	Keyword5 (Lucene+CodeRank)		Keyword5 (Lucene+BugLocator+Blizzard+CodeRank)	
	MRR@10	*Accuracy*@10/%	*MRR*@10	*Accuracy*@10/%
Asepctj	0.020	3.2	0.029	3.6
Lucene	0.031	3.6	0.032	3.6
Hibernate	0.028	2.8	0.033	4.5
Openjpa	0.028	2.7	0.031	3.2
Tomcat	0.036	3.2	0.041	5.6
Zookeeper	0.050	7.3	0.056	15.6

6 局限和不足

本实验方法的局限性和不足如下：

1)在数据集构建过程中，我们使用了人工总结的启发式规则来链接缺陷报告及其缺陷文件，该规则不是完全正确的。

2)本文所有实验均基于 6 个 java 开源项目截至 2018 年 12 月的缺陷报告数据，我们不能保证本文所得结论一定能泛化到其他开闭源项目乃至本文 6 个项目的较新的缺陷报告。

结束语 基于文本检索的缺陷定位方法被用来辅助开发人员更快地定位缺陷，由于缺陷定位方法的性能受到缺陷报告质量影响，查询重构方法经常被用来提升缺陷报告质量。因此，为提升缺陷报告质量和解决已有查询重构方法的不足，我们提出了基于主题一致性保持和伪相关反馈扩展的查询重构方法。实验结果表明，对于这些使用先进缺陷定位方法在 TOP 10 可疑推荐文件中定位不到的低质量缺陷报告，该方法在 TOP 10 Accuracy 指标和 MRR@10 指标上分别能达到 39% 和 16%，对比现有重构技术，所提重构方法在 Accuracy@10 和 MRR@10 两个指标上分别可以提升 7%～32% 和 2%～13%，证明了所提方法的有效性。未来在查询和伪相关反馈库提取关键词时，不仅仅使用单词的共现关系来构建网络图，还可根据查询词的词性依赖或单词之间的其他关系，来实现查询扩展对查询缩减的互补。

参 考 文 献

[1] ZHOU J, ZHANG H, LO D. Where should the bugs be fixed? more accurate information retrieval-based bug localization based on bug reports[C]// International Conference on Software Engineering. 2012: 14-24.

[2] WONG C P, XIONG Y, ZHANG H, et al. Boosting bug-report-oriented fault localization with segmentation and stack-trace analysis[C]// International Conference on Software Maintenance and Evolution. 2014: 181-190.

[3] WANG S, LO D. Version history, similar report, and structure: Putting them together for improved bug localization[C]// International Conference on Program Comprehension. 2014: 53-63.

[4] KEVIC K, FRITZ T. Automatic Search Term Identification for Change Tasks[C]// International Conference on Software Engineering. 2014: 468-471.

[5] RAHMAN M M, ROY C K. STRICT: Information retrieval based search term identification for concept location[C]// International Conference on Software Analysis. Evolution & Reengineering, 2017: 79-90.

[6] ROCCHIO J J. The SMART Retrieval System—Experiments in Automatic Document Processing[C]// IEEE Transactions on Professional Communication. 1972: 17-17.

[7] CARPINETO C, ROMANO G. A Survey of Automatic Query Expansion in Information Retrieval[J]. ACM Computing Surveys, 2012, 14(1): 1: 50.

[8] RAHMAN M M, ROY C K. Improved query reformulation for concept location using coderank and document structures[C]// International Conference on Automated Software Engineering. 2017: 428-439.

[9] CHAPARRO O, FLOREZ J M, MARCUS A. Using bug descriptions to reformulate queries duringtext-retrieval-based bug localization[J]. Empirical Software Engineering, 2019, 25(4): 2947-3007.

[10] HOWARD M J, GUPTA S, POLLOCK L, et al. Automatically mining software-based, semantically-similar words from comment-code mappings[C]// Working Conference on Mining Software Repositories. 2013: 377-386.

[11] TIAN Y, LO D, LAWALL J. Automated construction of a software-specific word similarity database[C]// Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering. 2014: 44-53.

[12] CAO K, CHEN C, BALTES S, et al. Automated query reformulation for efficientsearch based on query logs from stack overflow[C]// International Conference on Software Engineering. 2021: 1273-1285.

[13] LEMOS O A L, PAULA A C, ZANICHELLI F C, et al. Thesaurus-based Automatic Query Expansion for Interface-driven Code Search[C]// Working Conference on Mining Software Repositories. 2014: 212-221.

[14] HILL E, POLLOCK L, VIJAY-SHANKER K. Automatically Capturing Source Code Context of NL-queries for Software Maintenance and Reuse[C]// International Conference on Software Engineering. 2009: 232-242.

[15] SISMAN B, KAK A C. Assisting Code Search with Automatic Query Reformulation for Bug Localization[C]// Working Conference on Mining Software Repositories. 2013: 309-318.

[16] RAHMAN M M, ROY C K. Improving IR-based bug localization with context-aware query reformulation[C]// Joint Meeting on European Software Engineering Conference & Symposium on the Foundations of Software Engineering. 2018: 621-632.

[17] WONG C P, XIONG Y, ZHANG H, et al. Boosting bug-report-oriented fault localization with segmentation and stack-trace analysis[C]// International Conference on Software Maintenance and Evolution. 2014: 181-190.

[18] DEERWESTER S, DUMAIS S T, FURNAS G W, et al. Indexing by latent semantic analysis[J]. Journal of the American Society for Information Science, 1990, 41(6): 391-407.

[19] NGUYEN A T, NGUYEN T T, AL-KOFAHI J, et al. A topic-

based approach for narrowing the search space of buggy files from a bug report[C]// International Conference on Automated Software Engineering. 2011:263-272.

[20] THOMAS S W,NAGAPPAN M,BLOSTEIN D,et al. The impact of classifier configuration and classifier combination on bug-localization[J]. IEEE Transactions on Software Engineering, 2013,39(10):1427-1443.

[21] LAM A N,NGUYEN A T,NGUYEN H A,et al. Bug localization with combination of deep learning and information retrieval [C] // International Conference on Program Comprehension. IEEE,2017:218-229.

[22] XIAO Y,KEUNG J. Improving bug localization with character-level convolutional neural network and recurrent neural network [C]// Software Engineering Conference. IEEE,2018:703-704.

[23] QIU F,GAO Z,XIA X,et al. Deep just-in-time defect localization[J]. IEEE Transactions on Software Engineering, 2021, 48(12):5068-5086.

[24] DIT B,GUERROUJ L,POSHYVANYK D,et al. Can better identifier splitting techniques help feature location? [C]// International Conference on Program Comprehension. 2011:11-20.

[25] MIHALCEA R,TARAU P. Textrank:Bringing order into text [C]// Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing. 2004:404-411.

[26] BRIN S,PAGE L. The anatomy of a large-scale hypertextual web search engine [J]. Computer Networks and ISDN Systems, 1998,30(1/2/3/4/5/6/7):107-117.

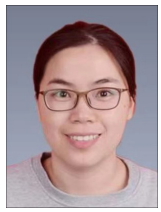
[27] BLANCO R,LIOMA C. Graph-based term weighting for information retrieval [J]. Information Retrieval,2012,15:54-92.

[28] ZOU D,LIANG J,XIONG Y,et al. An empirical study of fault localization families and their combinations[J]. IEEE Transactions on Software Engineering,2019,47(2):332-347.

[29] MORENO L,TREADWAY J J,Marcus A,et al. On the use of stack traces to improve text retrieval-based bug localization [C]// International Conference on Software Maintenance and Evolution. 2014:151-160.

[30] YE X,BUNESCU R,LIU C. Learning to rank relevant files for bug reports using domain knowledge[C]// Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. 2014:689-699.

LIU Wenjie, born in 1999, postgraduate. His main research interest is bug localization.



ZOU Weiqin, born in 1988, Ph.D, associate professor, is a member of CCF (No. D3300M). Her main research interests include bug localization and software repository mining.

(责任编辑:喻藜)