计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(2):31−39 [doi: 10.15888/j.cnki.csa.008299]

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

语义扩展连续查询的重复错误报告预测①

张骞月, 赵瑞莲, 王微微

(北京化工大学 信息科学与技术学院, 北京 100029)
通信作者: 王微微, E-mail: wangww@mail.buct.edu.cn

摘　要: 随着软件项目规模的增大与复杂性的增加, 测试过程产生了大量的错误报告, 其中重复的错误报告广泛存在. 重复错误报告的存在, 降低了开发人员修复错误的效率. 重复错误报告预测可有效地避免重复错误报告的产生, 是近年来的热门研究方向之一, 但其效率及准确率有待提高. 为此, 提出一种基于语义扩展连续查询的重复错误报告预测方法, 通过构建基于主题模型的错误报告索引词库, 对查询词序列进行语义扩展, 采用基于连续查询的错误报告检索算法, 在缩小索引空间的同时, 提升了预测准确率与效率. 实验表明, 相较于传统重复错误报告预测方法, 该方法减小了 50% 以上的错误报告索引空间, 最高提升了 33.6% 的预测效果, 且缩短了 41%–73% 的检索时间.

关键词: 重复错误报告; 语义扩展; 连续查询; 主题索引词库; 检索算法; 预测方法

引用格式: 张骞月,赵瑞莲,王微微.语义扩展连续查询的重复错误报告预测.计算机系统应用,2022,31(2):31–39. http://www.c-s-a.org.cn/1003-3254/8299.html

Prediction of Duplicate Bug Reports Based on Semantically Extended Continuous Queries

ZHANG Qian-Yue, ZHAO Rui-Lian, WANG Wei-Wei

(College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China)

Abstract: With the increase in scale and complexity of software projects, a large number of bug reports are generated during the testing process, among which duplicate bug reports are widely present, reducing the efficiency of developers in fixing bugs. The prediction of duplicate bug report has become one of the popular research fields in recent years, and its efficiency and accuracy need to be improved. Therefore, this study puts forward a prediction method of duplicate bug reports based on semantic extension and continuous queries. Through the construction of a bug report index thesaurus based on the theme model, the semantic extension of query sequences is conducted. Then, the bug report retrieval algorithm based on the continuous query is adopted to narrow the index space and improve the prediction accuracy and efficiency. Experimental results show that compared with the traditional prediction method of duplicate bug reports, the proposed method reduces the index space of bug reports by more than 50%, improves the prediction effect by up to 33.6%, and shortens the retrieval time by 41%–73%.

Key words: duplicate bug report; semantic extension; continuous query; subject index thesaurus; retrieval algorithm; prediction method

1　引言

为保证软件系统的质量与性能, 软件系统在发布或上线之前, 需要进行大量的测试[1]. 测试人员一旦检测出软件系统存在问题, 则将提交错误报告 (bug report), 以供开发人员对错误进行修复. 错误报告一般含有对错误的概要说明 (title)、详细描述 (description)、提交时间、提交人、错误表现截图、代码片段等相关信息.

① 基金项目: 国家自然科学基金 (62077003, 61872026)
收稿时间: 2021-04-09; 修改时间: 2021-05-11; 采用时间: 2021-05-19; csa 在线出版时间: 2022-01-17

由于软件项目规模的增大和复杂性的增加, 导致参与测试工作的人员较多, 不同测试人员可能会检测到同一个错误, 这样就会造成针对同一错误, 重复提交错误报告 (duplicate bug report) 的可能[2]. 而重复错误报告的存在, 将降低开发人员修复错误的效率. 因此, 如何避免重复错误报告的提交已成为近年来软件工程领域新的研究热点. 目前, 与重复错误报告相关的研究主要集中在重复错误报告检索与重复错误报告预测两方面.

重复错误报告检索是指: 针对错误报告库, 采用自然语言处理和信息检索技术, 检索出重复的错误报告. 如 Wang 等[3] 将错误报告中的自然语言描述与执行信息作为查询依据, 通过信息检索技术, 检测错误报告库中是否存在重复的错误报告. 该方法虽能通过对比错误报告中自然语言描述以及执行信息的相似度, 检索得出重复错误报告, 但该方法将错误报告的自然语言描述作为一个整体要素进行检索, 未对错误报告的自然语言描述进行深入的语义挖掘, 使得检索的准确率不高. 范道远等[4] 借鉴集成测试的思想, 提出了一种将错误报告文本信息与分类信息结合的重复错误报告检索方法, 该方法虽将分类信息与文本信息结合作为判定重复错误报告的依据, 但也未对错误报告文本信息进行语义挖掘, 其对重复错误报告检索准确率的提升仍然有限. Chaparro 等[5,6] 基于文本检索的错误定位方法, 通过对错误报告中的文本信息进行语义分析, 以错误报告中的可观测行为 OB (observed behavior)、错误复现步骤 S2R (steps to reproduce) 等作为查询要素, 通过各种组合方式, 进行重复错误报告检索, 提高了检索准确度. 但该方法中对于 OB、S2R 等语句的识别主要依靠人为判定与标注, 在引入人为偏差的同时, 也降低了重复错误报告检索的自动化程度. 为了更进一步地提高重复错误报告的检索效率, Sun 等[7] 提出一种基于判别模型的重复错误报告检索方法, 通过对错误报告的信息进行特征提取与对比, 在错误报告库中检索相似的错误报告. 该方法虽然较大程度提高了重复错误报告的检索准确率, 但动态建模开销较大.

由于重复错误报告检索是基于完整的错误报告, 无法避免重复错误报告的产生. 因此, Hindle 等[8] 借鉴搜索引擎的思想, 给出了一种重复错误报告预测方法. 重复错误报告预测是指: 在错误报告提交前对错误报告进行预测, 以避免重复错误报告的产生. Hindle 等提出的重复错误报告预测方法, 可在错误报告编写过程中, 对当前已输入的文本内容进行实时检索, 依据检索结果判定当前撰写的报告是否已在错误报告库中, 避免重复错误报告的提交. 但由于其检索过程采用倒排表索引技术[9], 仅通过查询词序列与错误报告的词法相关性进行检索, 并未对错误报告进行深入的语义挖掘. 因此, 该方法在重复错误报告预测的准确度仍有很大的改进空间. 此外, 随着错误报告库的不断增大, 索引文件所占的内存也在不断增大[10], 导致索引空间过大, 检索效率较低.

因此, 本文提出一种基于语义扩展连续查询的重复错误报告预测方法, 通过对错误报告进行主题语义挖掘, 确定错误报告的主题词, 构建基于主题模型的错误报告索引词库, 以减小错误报告索引空间, 建立主题词-错误报告之间的语义关系; 在此基础上, 对查询词序列进行同义词补充及基于语义的后续词组扩展, 以提升查询准确度; 同时, 在检索过程中, 利用语义扩展后的查询词序列对主题索引进行检索, 以提升检索效果及效率. 通过实验验证, 相较于传统方法, 本文方法减小了 50% 以上的索引空间, 在检索速度上提升了 41%–73%, 预测效果最高提升 33.6%.

2 基于主题模型的错误报告索引词库构建

传统的倒排表索引技术依据词与错误报告的包含关系建立词与错误报告的索引联系[11,12], 并且根据错误报告中包含的词全集建立错误报告的索引词库. 一方面, 仅靠包含关系建立起来的词与错误报告之间的语义关联可能导致检索的准确率低下; 另一方面, 以词全集作为错误报告索引词库也造成索引空间的不断增大, 降低了检索效率.

基于此, 本文提出一种基于主题模型的错误报告索引词库构建方法, 通过对错误报告进行主题语义挖掘, 深化词与错误报告之间的语义关联, 确定错误报告主题词, 建立错误报告主题词与错误报告之间的语义关系, 构建错误报告主题索引词库; 同时, 为减小错误报告索引空间, 在构建错误报告主题索引词库时, 依据挖掘得到的错误报告主题, 进行错误报告主题词确定, 以缩小错误报告索引词库, 从而减小索引空间.

2.1 错误报告主题挖掘

针对错误报告主题挖掘, 本文借鉴自然语言处理中文档主题分类思想, 利用常用的 LDA (latent Dirichlet allocation) 主题模型[13,14], 对库中的错误报告进行主题

挖掘, 获取错误报告的主题分布与针对每个主题的词项分布, 由此可计算每份错误报告的中心主题, 以及与中心主题高度相关的若干关键词, 这些关键词即为错误报告的主题词.

在 LDA 主题模型中, 假设错误报告主题与主题中词项的先验分布均为 Dirichlet 分布, 则对于任一错误报告, 其主题分布为:

$$\theta_d = Dirichlet(\alpha) \tag{1}$$

其中, α 为分布的超参数, 是一个 k 维向量; k 代表主题个数.

对于任一主题, 其词项分布为:

$$\beta_k = Dirichlet(\eta) \tag{2}$$

其中, η 为分布的超参数, 是一个 v 维向量. v 代表词库中词的个数.

在利用 LDA 主题模型对错误报告进行主题挖掘时, 设定的主题数目越多, 对错误报告的语义划分越细, 后续建立的词-错误报告关系就越多; 设定的主题数目越少, 对错误报告的划分越粗, 后续建立的词-错误报告关系就越少. 因此, 为了平衡检索的准确率与减小索引空间的需求, 需要合理设定主题数目.

2.2 错误报告主题词确定

由于 LDA 主题模型得到的词概率分布中所含的词全集作为错误报告的主题词, 则错误报告主题索引空间太大. 为了减小错误报告主题索引空间, 本文依据挖掘得到的错误报告主题, 确定错误报告的主题词, 剔除与主题关联度不大的词, 将与主题较为相关的词选入主题索引词库. 但由于设置的主题个数的局限性, 可能会遗漏一些在主题的词项分布中概率数值较低但对错误报告区分度很高的词, 为避免遗漏此类主题词, 本文在确定错误报告主题词的过程中, 利用 TF-IDF 进行词项权重判定, 以保留词频较低但较为重要的词项, 也可有效避免常用词对主题词的影响.

TF-IDF 公式计算如下:

$$TF\text{-}IDF = tf(t, d) \times \ln\left(\frac{N}{df(t)+1}\right) \tag{3}$$

其中, $tf(t, d)$ 表示词 t 在错误报告 d 中出现的频率; N 代表语料库中错误报告的总数; $df(t)$ 代表包含词 t 的错误报告数. $\ln(N/(df(t)+1))$ 又可写作 idf, 即逆文档频率.

TF-IDF 代表的含义为: 一个词的重要程度跟它在错误报告中出现的次数成正比[15], 跟它在语料库出现的次数成反比, 通过词频 (tf) 与逆文档频率 (idf) 的乘积计算, 其值越大, 则表明该词与错误报告的关联越大. 因此, 结合 TF-IDF 权衡词项权重, 可有效避免常用词对主题词的影响, 也能避免漏选低频主题词的情况.

结合 LDA 的错误报告主题语义挖掘与 TF-IDF 词项权重判定, 可得出与主题相关性较高及与错误报告关联度较大的词, 这些词可作为错误报告的主题词. 错误报告主题词确定流程如图 1 所示.

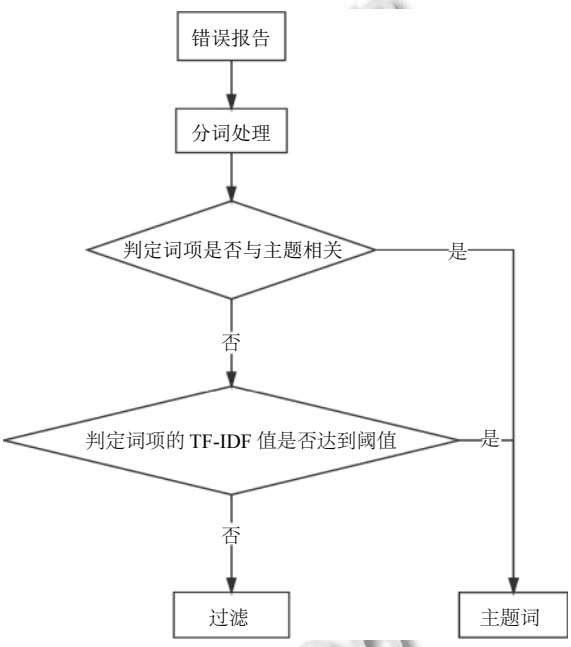

图 1 错误报告主题词确定流程

2.3 主题词-错误报告关系构建

为构建错误报告主题索引词库, 在确定了错误报告主题词的基础上, 还需要对错误报告主题词与错误报告本身建立关系.

在传统的错误报告索引词库构建中, 词与错误报告之间的关系为包含关系[16], 即若某词包含于若干错误报告中, 则该词与这些错误报告形成关联关系. 但这种包含关系不能体现词与错误报告更深层次的语义关系, 所以检索效果不佳.

因此, 本文通过建立错误报告主题词与错误报告的语义关系, 深化错误报告主题词与错误报告之间的语义关联, 以提升检索效果.

图 2 为主题词-错误报告关系示例图. 在主题词与错误报告的语义关系中, 主题是主题词与错误报告的关系媒介, 即主题词与错误报告的关联是通过同属于某一主题形成的.

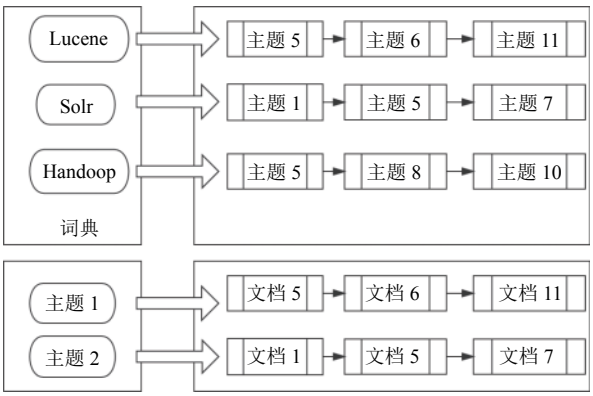

图 2 主题词-错误报告关系示例图

3 基于语义扩展连续查询的重复错误报告预测

为提高重复错误报告预测的准确率与检索效率, 本文提出一种基于语义扩展连续查询的重复错误报告预测方法, 框架如图 3 所示, 主要由错误报告主题索引词库构建、查询词序列扩展、基于连续查询的错误报告检索, 以及预测结果排序 4 个部分组成.

对初始查询词序列进行基于语义上下文的扩展, 可以得到初始查询词序列的同义词组及语义后续词组, 形成当前查询词扩展序列. 同时, 由于库中错误报告数量庞大, 为提升检索效率, 本文通过基于连续查询的错误报告检索算法, 对当前查询词扩展序列进行检索. 为保证预测结果与查询词序列在语义上更具相关性, 对检索得到的相关错误报告, 依据文本相似度进行排序, 以此为最终的预测结果.

3.1 查询词序列扩展

在重复错误报告连续查询时, 为提高重复错误报告的查全率, 本文通过当前词的同义词补充与后续词组预测, 对初始查询词序列进行基于语义上下文的扩展, 形成当前词的扩展序列.

(1) 当前词同义词补充

对当前词进行同义词补充的目的有两个, 一是若当前词在错误报告主题索引词库中不存在时, 对当前词进行同义词补充, 可以使补充的同义词在错误报告主题索引词库中; 二是通过当前词同义词补充, 扩大当前词关联的错误报告范围, 可提升查全率.

(2) 基于语义的后续词组预测

当查询词序列较短时, 由于输入的词较少, 查询信息难以完整, 仅依靠同义词补充, 并不能很好地将查询信息补充完整, 查询检索的准确率无法提升. 因此, 可通过对当前词的同义词组, 进行基于语义上下文的后续词组预测, 得到<当前词, 同义词, 后续词>的若干三元组, 作为当前词的扩展序列. 通过基于语义的后续词组预测, 得到当前词及其同义词组的语义下文词组, 从语义上对查询信息进行了完善, 有助于提升重复错误报告的查全率.

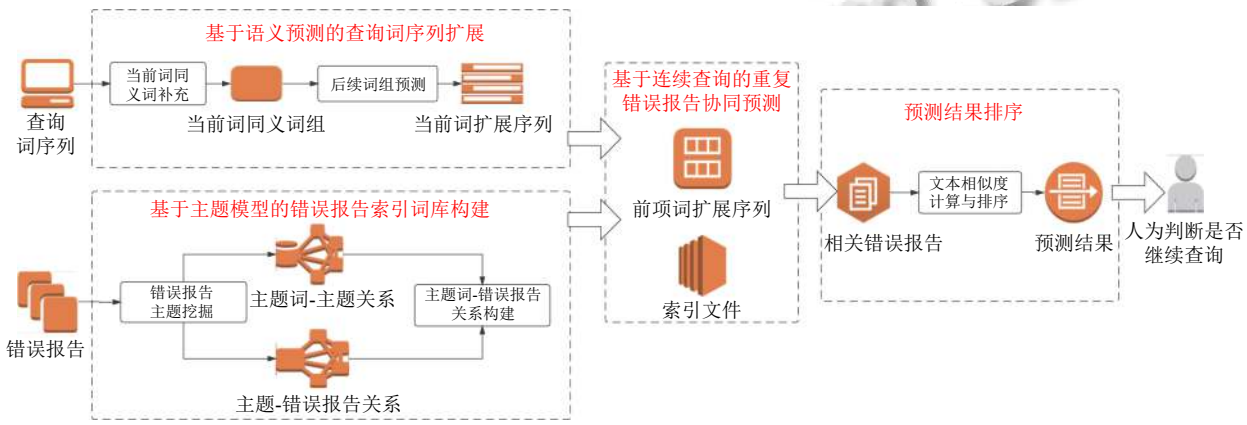

图 3 基于语义扩展连续查询重复错误报告预测方法

3.2 基于连续查询的错误报告检索算法

在传统的错误报告检索中, 检索算法大多基于倒排表设计. 倒排表索引词库中包含的词为语料库中所有的词; 同时, 倒排表索引结构的构建是基于词与错误报告的包含关系, 这就使得无论对检索算法如何改进, 由于索引空间是不变的, 检索效率的提升也是有限的. 因此, 为进一步提升检索效率, 本文提出一种基于错误报告主题索引词库的连续查询错误报告检索算法, 利用基于语义预测的前项查询词扩展序列, 在错误报告主题索引词库中进行检索时, 可以缩小检索空间, 大幅

提升检索效率.

在查询词序列扩展时, 对当前词的同义词组进行基于语义的后续词组预测, 得到当前词的扩展序列<当前词, 同义词, 后续词>, 后续词中有可能含有下一个要输入的词, 由此可知, 当前词也极有可能包含在其前项词扩展序列的后续词中. 扩展序列后续词的数量显然小于错误报告主题索引词库中词的数量. 所以, 若在错误报告主题索引词库中对当前词的扩展序列进行检索时, 先在其前项词的扩展序列中检索当前词, 若当前词包含在其前项词的扩展序列中, 借助前项词扩展序列的检索结果, 可以快速得到当前词扩展序列的检索结果, 较大程度地提升检索效率. 因此, 基于连续查询的错误报告检索算法如算法 1.

算法 1. 基于连续查询的错误报告检索算法

```
1) Buildsequence(input_word,synonym_word,conse-quent_word) //建立当前词扩展序列;
2) Length(initial_sequence,input_word) //判断当前词在查询词序列中的位置;
  If(Length(initial_sequence,input_word)==0)
  {
      Search index
  }//若当前词为第一个输入词, 即不存在前项词, 则在错误报告主题索引中进行检索;
3) Else
  Search preceding_sequence//否则检索其前项词的扩展序列;
  If succeed
  {
  return index_report//若在其前项词扩展序列中检索到当前词, 则返回相关错误报告;
  }
  Else
  Search index//否则继续在错误报告主题索引中检索.
```

3.3 重复错误报告预测结果排序

在错误报告主题索引中, 主题词-错误报告关系是由主题词与错误报告的主题相关性构建起来的, 因此, 检索得到的错误报告是与查询词序列具有主题相关性的. 但是由于主题数目的有限性, 使得仅通过主题相关性, 无法准确表征词在错误报告中的重要性. 所以, 仅以主题相关性作为预测结果是不全面的.

由于 TF-IDF、BM25 等文本相似性算法比较简单, 能够快速且较为准确地计算出查询语句与文档的相似性, 在实时搜索中具有较强的实用性. 因此, 为了保证检索实时性与查询结果的准确性, 本文采用文本相似性与主题相关性结合的方式对重复错误报告预测结果进行排序, 即在基于连续查询的错误报告检索结果的基础上, 对检索得到的错误报告与查询词序列进行文本相似性计算, 将计算结果与错误报告主题相似度求和, 以此作为重复错误报告预测结果的排序依据.

4 实验设计与结果分析

4.1 数据集

为验证方法的有效性, 针对 GoogleCode、LaunchPad、Bugzilla 三个错误报告管理系统, 选取了 12 个开源项目, 并收集了错误报告 254 883 份, 进行实验分析. 数据集信息如表 1 所示.

表 1 实验数据集

项目	平台	错误报告数	重复错误报告数
Android	GoogleCode	37 626	13 633
AppInventor	GoogleCode	2 098	265
Bazaar	LaunchPad	7 020	523
Cyan	GoogleCode	5 185	677
Eclipse	Bugzilla	45 234	4 341
K9mail	GoogleCode	4 309	909
Mozilla	Bugzilla	93 651	10 479
OpenOffice	Bugzilla	31 136	4 460
OpenStack	LaunchPad	17 077	211
Osmand	GoogleCode	1 026	73
Tempest	LaunchPad	2 085	98
Firefox	Bugzilla	8 436	691

4.2 实验设计

为验证本文提出方法的有效性, 本文设计了多组对比实验, 从索引空间、检索效率、预测效果 3 个方面对本文方法进行验证. 此外, 在检索效率的对比实验中, 单独设置消融实验, 以验证本文方法各模块对整体方法检索效率提升的贡献.

Lucene 是目前常用的全文检索引擎架构, 使用倒排表索引与全文检索算法. 本文将其作为对照组方法进行对比实验.

(1) 基于主题模型的错误报告索引空间分析

随着错误报告库的不断增大, 错误报告的索引空间也在不断增大. 过大的错误报告索引空间会降低错误报告检索效率. 本文通过构建基于主题模型的错误报告索引词库, 缩小错误报告索引空间. 将基于主题模型的错误报告索引词库与倒排表索引词库进行词数对比, 验证基于主题模型的错误报告索引词库在索引空间上的优越性.

本文使用 LDA 主题模型对数据集建模, 设置主题数 120, 分别得到该参数下基于主题的词与错误报告的概率分布, 再利用 TF-IDF 算法确定错误报告主题词,

构建基于主题模型的错误报告索引词库. 倒排表索引词库的构建方法为以 Lucene 全文检索框架自带的 Indexwriter 对数据集中的错误报告进行倒排表索引构建, 得到倒排表索引词库. 通过计算倒排表索引词库与基于主题模型的错误报告索引词库中的词数, 进行索引空间对比.

(2) 基于连续查询的错误报告检索效率分析

检索效率分析旨在验证本文提出的基于连续查询的错误报告检索算法的优越性与查询词序列扩展模块的必要性. 在评价检索效率时, 通用指标为查全率、查准率与检索速度, 检索速度通常采用检索时间的长短表征. 为了综合衡量查全率与查准率, 本文采用 *F*1-score 作为评价指标. 由于本文方法由多个模块组成, 需要设置消融实验, 进一步验证基于主题模型的错误报告索引词库、查询词序列扩展、基于连续查询的错误报告检索算法 3 个模块对本文方法在检索效率方面的提升.

基于连续查询的错误报告检索算法中, 需要使用查询词序列扩展模块中的变量, 因此, 将基于连续查询的错误报告检索算法与查询词序列扩展看作一个整体模块进行消融实验, 设计如下:

使用 Lucene 作为实验对照组.

将基于主题模型的错误报告索引词库替换为倒排表索引词库, 并与本文基于连续查询的错误报告检索算法相结合, 形成方法 1; 将基于主题模型的错误报告索引词库与 Lucene 的全文检索算法结合, 形成方法 2. 再将方法 1、方法 2 与 Lucene 方法进行对比实验, 计算检索时间与 *F*1-score, 以检验各模块在提升检索效率方面的作用.

(3) 基于语义扩展连续查询的重复错误报告预测效果分析

由于目前在重复错误报告预测研究领域, 仅有 Hindle 等 [8] 的实证研究得出了研究成果, 因此, 本文基于文献 [8] 实验的数据集, 构建基于语义扩展连续查询的重复错误报告预测实验, 并将实验结果与文献 [8] 中的结果进行对比.

同时, 为了实验的充分性, 在上述实验的基础上, 选取 Firefox 与 Mozilla 两个项目作为补充的数据集. 在补充数据集上复现文献 [8] 的实验, 将其实验结果与本文方法的实验结果进行对比.

4.3 实验结果与分析

(1) 基于主题模型的错误报告索引空间实验结果分析

如图 4 所示为基于主题模型的错误报告索引词库与倒排表索引词库的词数对比, 其中, 主题索引词库表示基于主题模型的错误报告索引词库. 可以看出, 基于主题模型的错误报告索引词库的词数比倒排表索引词库的词数少 50% 以上. 这是由于基于主题模型的错误报告索引词库, 对错误报告中的词项进行主题相关性与词项重要性的筛选, 得到的主题词远少于倒排表索引词库中的词, 较大程度地减小了索引空间.

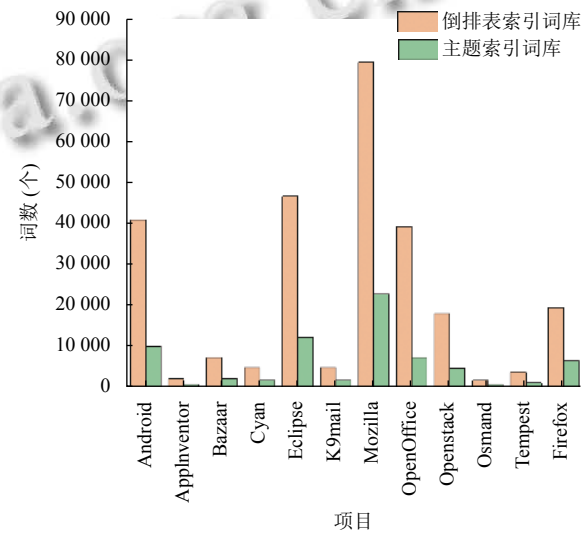

图 4 索引空间对比图

(2) 基于连续查询的错误报告检索效率结果分析

如图 5 所示为 Lucene 检索方法与本文方法、方法 1、方法 2 在检索时间上的对比. 其中, Lucene 检索代表 Lucene 检索方法; 连续查询检索代表本文方法.

检索时间计时起点为输入查询词序列, 终点为检索出相关错误报告. 索引的构建不纳入检索时间的考量.

从图 5 可以看出, 由于减小了索引空间, 并且在检索过程中采用了基于连续查询的错误报告检索算法, 在检索当前词时, 先检索前项词扩展序列, 而不是检索整个错误报告主题索引词库, 在检索的过程中减小了检索空间, 因此, 本文方法的检索时间较 Lucene 检索算法缩短了 41%–73%.

同时, 通过方法 1 与方法 2 的比较, 可以看出, 方法 1 与方法 2 的检索时间相较于 Lucene 检索方法都有减小, 且方法 2 的减小幅度更大, 这是由于基于主题模型的错误报告索引词库的建立, 较大幅度地缩小了错误报告的索引空间, 在同等情况下, 减少了检索时间.

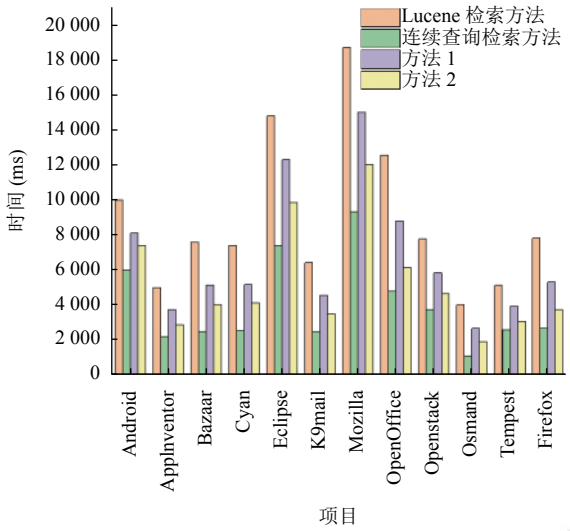

图 5 检索时间对比图

表 2 为 Lucene 检索方法与本文方法、方法 1、方法 2 在 *F*1-score 上的对比. *F*1-score 数值越大说明其检索效率越高. 由表 2 可得, 本文方法 (连续查询检索方法) 相较于 Lucene 检索方法, 在 *F*1-score 分数上提高了约 20%, 检索效率更高, 这是由于查询词序列扩展补全了查询信息, 使得检索的查全率与查准率得到提升. 方法 2 的 *F*1-score 分数最低, 因为方法 2 去除了查询词序列扩展模块, 且使用基于主题模型的错误报告索引词库, 索引词库中仅有数量较少的错误报告主题词, 因此在进行检索时, 其查准率较低, 由此导致 *F*1-score 分数低.

表 2 *F*1-score 对比

项目	Lucene方法	连续查询检索方法	方法1	方法2
Android	0.732	0.832	0.809	0.532
AppInventor	0.704	0.814	0.823	0.541
Bazaar	0.693	0.903	0.887	0.604
Cyan	0.867	0.887	0.904	0.487
Eclipse	0.712	0.804	0.813	0.544
K9mail	0.788	0.874	0.869	0.431
Mozilla	0.617	0.891	0.884	0.486
OpenOffice	0.784	0.940	0.927	0.521
OpenStack	0.801	0.914	0.920	0.577
Osmand	0.637	0.884	0.880	0.437
Tempest	0.706	0.834	0.851	0.419
Ave	0.731	0.871	0.870	0.507

(3) 基于语义扩展连续查询的重复错误报告预测效果实验结果分析

文献 [8] 的实验通过 MAP、MRR、TOP-k、AveP-TOPk、MRR-TOPk、MRR-TOPk-1 综合衡量重复错误报告的预测效果. MAP 代表平均精度, 是一种有效衡量信息检索精度的分数, 其值越高, 代表预测效果越好; MRR 代表重复错误报告排名的倒数平均值, 当重复错误报告排名越靠前, 其 MRR 分数越高; TOP-k 代表重复错误报告出现在给定的排名域中的个数, 数值越高表示预测效果越好.

但由于 MAP、MRR、TOP-k 具有局限性, 不能表征查询词序列长短对预测结果的影响, 因此, Hindle 等[8] 又提出了 AveP-TOPk、MRR-TOPk、MRR-TOPk-1 指标作为补充, 其数值越高表示预测效果越好. 其中, AveP-TOPk 是 MAP 与 TOP-k 的结合, AveP-TOPk 数值越高, 代表在同样的预测结果情况下, 使用的查询词数越少; MRR-TOPk 代表重复错误报告第一次出现在 TOPk 排名中时, 其 MRR 分数, 数值越大, 表明预测效果越好; MRR-TOPk-1 则表示重复错误报告第一次出现在 TOPk 排名中时, 平均所需要的查询词数, 数值越小表明预测效果越好.

基于文献 [8] 实验数据集的预测效果对比如表 3. AC 代表文献 [8] 方法, SECQ (semantically extended continuous query) 代表本文方法. 由表 3 可以看出, 本文方法在 6 个评价指标上均优于文献 [8] 方法, 其中, AveP-TOP5 提升了 20% 以上. 表 4 所示为基于补充数据集, 复现文献 [8] 方法实验, 得到的实验结果与本文实验结果的对比.

通过表 4 的对比可以看出, 在改变数据集的情况下, 本文方法仍在 6 个评价指标上均优于文献 [8] 方法, 最高提升 33.6%.

由此可见, 与传统方法相比, 本文方法减小了索引空间, 同时, 在检索效率与预测效果上均有提升.

5 结束语

本文提出了一种基于语义扩展连续查询的重复错误报告预测方法. 通过挖掘错误报告主题, 以错误报告主题为媒介, 确定错误报告的主题词, 构建错误报告主题索引词库, 建立错误报告主题词与错误报告的关系, 减小了错误报告索引空间; 在此基础上, 对查询词序列进行语义扩展, 提出基于连续查询的错误报告检索算法, 缩小检索空间, 在提高了预测准确率的同时, 也提升了检索的效率. 实验表明, 本文方法在索引空间与检索时间上较传统方法有较大减小, 同时与文献 [8] 实验结果的对比表明, 在预测效果上, 本文方法更好, 提升了 33.6%.

表 3 基于文献 [8] 实验数据集的预测效果对比

项目	方法	MAP	MRR	TOP-5	AveP-TOP5	MRR-TOP5	MRR-TOP5-1
Android	AC	0.223	0.231	0.300	0.274	0.141	7.099
	SECQ	0.235	0.242	0.313	0.291	0.153	6.526
AppInventor	AC	0.143	0.197	0.426	0.408	0.462	2.165
	SECQ	0.138	0.199	0.439	0.493	0.485	2.060
Bazaar	AC	0.185	0.185	0.278	0.275	0.148	6.749
	SECQ	0.192	0.191	0.301	0.311	0.162	6.174
Cyan	AC	0.239	0.247	0.264	0.249	0.158	6.310
	SECQ	0.243	0.262	0.261	0.308	0.162	6.168
Eclipse	AC	0.187	0.190	0.255	0.231	0.117	8.517
	SECQ	0.201	0.210	0.278	0.301	0.123	8.132
K9mail	AC	0.226	0.230	0.348	0.337	0.207	4.835
	SECQ	0.236	0.241	0.368	0.402	0.234	4.266
Mozilla	AC	0.216	0.220	0.288	0.258	0.124	8.086
	SECQ	0.230	0.234	0.291	0.303	0.137	7.289
OpenOffice	AC	0.170	0.175	0.220	0.196	0.098	10.211
	SECQ	0.193	0.184	0.231	0.224	0.117	8.531
OpenStack	AC	0.302	0.302	0.436	0.396	0.214	4.682
	SECQ	0.324	0.324	0.481	0.425	0.223	4.484
Osmand	AC	0.078	0.078	0.093	0.114	0.101	9.866
	SECQ	0.086	0.086	0.107	0.256	0.132	7.573
Tempest	AC	0.184	0.184	0.206	0.198	0.156	6.411
	SECQ	0.201	0.201	0.213	0.224	0.167	5.972
Ave	AC	0.196	0.204	0.283	0.267	0.175	6.812
	SECQ	0.207	0.216	0.298	0.322	0.190	6.107

表 4 基于补充数据集的预测效果对比

项目	方法	MAP	MRR	TOP-5	AveP-TOP5	MRR-TOP5	MRR-TOP5-1
Firefox	AC	0.223	0.299	0.211	0.328	0.162	6.913
	SECQ	0.346	0.294	0.232	0.335	0.173	5.773
Mozilla	AC	0.387	0.325	0.333	0.343	0.246	4.817
	SECQ	0.378	0.340	0.583	0.408	0.261	3.825
Ave	AC	0.213	0.220	0.281	0.277	0.180	6.667
	SECQ	0.231	0.294	0.315	0.329	0.195	4.799

参考文献

1 Sabor KK, Hamou-Lhadj A, Larsson A. DURFEX: A feature extraction technique for efficient detection of duplicate bug reports. 2017 IEEE International Conference on Software Quality, Reliability and Security. Prague: IEEE, 2017. 240–250.

2 Rakha MS, Shang WY, Hassan AE. Studying the needed effort for identifying duplicate issues. Empirical Software Engineering, 2016, 21(5): 1960–1989. [doi: 10.1007/s10664-015-9404-6]

3 Wang XY, Zhang L, Xie T, *et al*. An approach to detecting duplicate bug reports using natural language and execution information. Proceedings of the 30th International Conference on Software Engineering. New York: ACM, 2008. 461–470.

4 范道远, 孙吉红, 王炜, 等. 融合文本与分类信息的重复缺陷报告检测方法. 计算机科学, 2019, 46(12): 192–200. [doi: 10.11896/jsjkx.181102232]

5 Chaparro O, Florez JM, Marcus A. Using observed behavior to reformulate queries during text retrieval-based bug localization. 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME). Shanghai: IEEE, 2017. 376–387.

6 Chaparro O, Florez JM, Singh U, *et al*. Reformulating queries for duplicate bug report detection. 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER). Hangzhou: IEEE, 2019. 218–229.

7 Sun CN, Lo D, Wang XY. A discriminative model approach for accurate duplicate bug report retrieval. 2010 ACM/IEEE

32nd International Conference on Software Engineering. Cape Town: IEEE, 2010. 45–54.

8 Hindle A, Onuczko C. Preventing duplicate bug reports by continuously querying bug reports. Empirical Software Engineering, 2019, 24(2): 902–936. [doi: 10.1007/s10664-018-9643-4]

9 Lukins SK, Kraft NA, Etzkorn LH. Source code retrieval for bug localization using latent dirichlet allocation. 2008 15th Working Conference on Reverse Engineering. Antwerp: IEEE, 2008. 155–164.

10 Alipour A, Hindle A, Stroulia E. A contextual approach towards more accurate duplicate bug report detection. 2013 10th Working Conference on Mining Software Repositories (MSR). San Francisco: IEEE, 2013. 184–192.

11 Youm KC, Ahn J, Lee E. Improved bug localization based on code change histories and bug reports. Information and Software Technology, 2017, 82: 177–192. [doi: 10.1016/j.infsof.2016.11.002]

12 肖晗, 毛雪松, 朱泽德. 基于 HybridDL 模型的文本相似度检测方法. 电子技术应用, 2020, 46(6): 28–31, 35.

13 Tang ZQ, Zhang XA, Niu JM. LDA model and network embedding-based collaborative filtering recommendation. 2019 6th International Conference on Dependable Systems and Their Applications (DSA). Harbin: IEEE, 2019. 283–289.

14 王世杰, 周丽华, 孔兵, 等. 基于 LDA-DeepHawkes 模型的信息级联预测. 计算机科学与探索, 2020, 14(3): 410–425. [doi: 10.3778/j.issn.1673-9418.1903065]

15 上官明霞, 朱珊珊, 陈晓亮, 等. 基于融合自然语言处理的语义分析方法研究. 计算机与网络, 2018, 44(20): 65–67. [doi: 10.3969/j.issn.1008-1739.2018.20.054]

16 Soltani M, Hermans F, Bäck T. The significance of bug report elements. Empirical Software Engineering, 2020, 25(6): 5255–5294. [doi: 10.1007/s10664-020-09882-z]