软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2022,33(5):1699−1710 [doi: 10.13328/j.cnki.jos.006556]

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

软件问答社区的问题删除预测方法*

蒋 竞, 苗 萌, 赵丽娴, 张 莉

(北京航空航天大学 计算机学院, 北京 100191)

通信作者: 张莉, lily@buaa.edu.cn

摘 要: Stack Overflow 是最受欢迎的软件问答社区之一, 用户可以在该网站发布问题并得到其他用户的回答. 为了保证问题质量, 网站需要尽快发现并删除低质量或者不符合社区主题的问题. 当前, Stack Overflow 主要采用人工检查的方式发现需要被删除的问题. 然而这种方式往往不能保证问题被及时发现、删除, 而且加重了社区管理员的负担. 为了快速发现需要删除的问题, 提出了自动化预测问题删除的方法 MulPredictor. 该方法提取问题的语义内容特征、语义统计特征和元特征, 使用随机森林分类器计算问题会被删除的概率. 实验结果表明: 与现有方法 DelPredictor 和 NLPPredictor 相比, MulPredictor 的准确率在平衡测试集上分别提升了 16.34% 和 12.78%, 在随机测试集上分别提升了 12.38% 和 14.14%. 此外, 分析了影响问题删除的重要特征, 发现代码段、问题的标题和正文第 1 段的特征对问题删除有重要的影响.

关键词: 问题删除预测; 问题质量; 问题分类; 软件问答社区; Stack Overflow

中图法分类号: TP311

中文引用格式: 蒋竞, 苗萌, 赵丽娴, 张莉. 软件问答社区的问题删除预测方法. 软件学报, 2022, 33(5): 1699–1710. http://www.jos.org.cn/1000-9825/6556.htm

英文引用格式: Jiang J, Miao M, Zhao LX, Zhang L. Prediction Method for Question Deletion in Software Question and Answer Community. Ruan Jian Xue Bao/Journal of Software, 2022, 33(5): 1699–1710 (in Chinese). http://www.jos.org.cn/1000-9825/6556.htm

Prediction Method for Question Deletion in Software Question and Answer Community

JIANG Jing, MIAO Meng, ZHAO Li-Xian, ZHANG Li

(School of Computer Science and Engineering, Beihang University, Beijing 100191, China)

Abstract: Stack Overflow is one of the most popular software question and answer communities, where users can post questions and receive answers from others. In order to ensure the quality of questions, the website needs to promptly discover and delete questions with low quality or not conforming to the community's theme. Currently, Stack Overflow mainly relies on manual inspection to find questions that need to be deleted. However, this way usually hardly guarantees to discover and delete questions in time, and increases the burden of community administrators. In order to quickly find questions that need to be deleted, this study proposes a method to automatically predict question deletion, which is named MulPredictor. This method extracts the semantic content features, the semantic statistical features and the meta features of a question, and uses the random forest classifier to calculate the probability that it will be deleted. Experimental results showed that, compared with existing methods DelPredictor and NLPPredictor, MulPredictor increases the accuracy by 16.34% and 12.78% on balanced test set, and increases the accuracy by 12.38% and 14.14% on random test set. In addition, this study also analyzes important features in question deletion, and finds that the code segment, the question's title, and the first paragraph of the question's body have the most significant impacts on question deletion.

Key words: prediction of question deletion; question quality; question classification; software question and answer community; Stack Overflow

* 基金项目: 科技创新 2030-"新一代人工智能"重大项目 (2018AAA0102304); 国家自然科学基金 (62177003); 中央高校基本科研业务费 (YWF-20-BJ-J-1018)

本文由"领域软件工程"专题特约编辑汤恩义副教授、江贺教授、陈俊洁副教授、李必信教授以及唐滨副教授推荐.

收稿时间: 2021-08-10; 修改时间: 2021-10-09; 采用时间: 2022-01-10; jos 在线出版时间: 2022-01-28

在软件问答社区 Stack Overflow, 用户可以发布或回答问题、对问题或回答进行投票、发表评论以及编辑问题[1−3]. Stack Overflow 已经成为开发人员寻找软件开发相关问题解决方法的重要平台. 为了帮助用户提出符合要求的问题, Stack Overflow 提供明确、详细的提问指南. 尽管如此, Stack Overflow 仍然存在低质量或者脱离主题的问题, 干扰用户的注意力, 甚至导致用户流失[4].

在 Stack Overflow, 社区管理员或者高声誉用户人工查看并删除问题. 但是这种人工方式往往不能保证问题被及时发现、删除. Correa 等人[5]研究发现, 超过 80% 的问题在存活一个月后才会被删除. 在未被删除的时间里, 这些问题会持续“污染”问题库, 影响网站的整体问题质量. 此外, Correa 等人[5]的研究还表明, 从 2011 年开始, 高声誉用户对问题删除工作的参与度逐渐下降, 这导致社区管理员不得不承担更多的工作. 由于目前 Stack Overflow 的用户基数呈指数级增长, 问题数量也呈爆发趋势, 社区管理员的工作负载一直在稳步增加[6]. 因此, 需要一种自动预测问题删除的方法, 既可以辅助社区管理员快速发现并删除不符合要求的问题, 从而提高审核效率, 减轻社区管理员的负担; 同时还能缩短删除问题的生存周期, 有助于维持网站中的内容质量.

一些文献研究自动化预测被删除问题的方法. Correa 等人[5]提出基于元特征的问题删除预测方法. Xia 等人[7]结合元特征[5]和数字文本特征, 使用分类算法预测问题删除. Stack Overflow 制定了一个详细的提问指南 (How do I ask a good question? http://stackoverflow.com/help/how-to-ask), 指导用户提出符合要求的问题. 提问指南的要求可以指导设计问题删除预测方法. 比如, 提问指南强调标题的重要性, 因此问题删除预测方法可以单独分析问题的标题. 参考提问指南, 本文提出了一种基于多种特征的问题删除预测方法 MulPredictor (multiple feature based predictor), 提取问题的语义内容特征、语义统计特征和元特征[5], 使用随机森林分类器计算问题会被删除的概率. 为了评估其预测效果, 本文从 2018 年 11 月 16 日–2018 年 12 月 31 日期间发布的问题中, 随机抽取各 10 000 个被删除问题和未被删除问题组成平衡测试集, 并且随机抽取 20 000 个问题组成随机测试集, 然后分别在两个测试集上比较 MulPredictor 与对比方法 DelPredictor[7]和 NLPPredictor[8]的预测效果. 实验结果表明, MulPredictor 与两种对比方法相比, 预测的准确率在平衡测试集上分别提升了 16.34% 和 12.78%, 在随机测试集上分别提升了 12.38% 和 14.14%. 此外, 本文还分析特征重要性并进行解释, 指导用户提问.

本文的贡献主要在以下两方面.

1) 提出了一种软件问答社区的问题删除预测方法 MulPredictor. 该方法不仅考虑元特征[5], 还考虑了语义内容特征和语义统计特征. 为了分析问题语义是否符合主题, 语义内容特征统计问题和已被删除问题、已保留问题的主题是否相似. 语义统计特征考虑问题的可阅读性和可理解性, 统计问题标题、正文等不同文本段的音节数、flesch 易读性指数、文化水平等级等. MulPredictor 相比目前最好的同类方法[7,8], 具有更准确的预测效果.

2) 发现了影响问题删除的重要特征, 指导用户提问. 例如, 本文发现代码段长度是重要特征, 因此建议用户在提问时详细地给出代码信息; 此外还发现标题中单词的平均长度也是一个重要因素. 建议用户在撰写问题标题时尽量选择较为简短的单词, 以便于他人理解.

1 研究背景

Stack Overflow 是目前最热门的软件问答社区之一. 从 2008 年投入使用以来至 2020 年 10 月 12 日, Stack Overflow 已经积累了超过 2 020 万个问题、3 030 万个答案以及 1 320 万个注册用户, 平均每月有 2.621 亿访问量. 目前, Stack Overflow 越来越受到软件工程、人机交互、社会计算、数据挖掘等不同研究领域的关注[9−14], 研究人员通过研究 Stack Overflow 库了解开发问题和编程主题趋势等[15−17].

Stack Overflow 使用基于社区的声誉奖励机制. 如果用户发布了错误的答案或者类似垃圾邮件或广告的低质量问题, 就会导致声誉值降低; 相反, 如果用户提出高质量的问题或给出高质量的回答, 就会为其赢得声誉. 虽然声誉机制引导用户发布高质量内容, 但仍然存在大量低质量或者脱离主题的问题. 因此, Stack Overflow 采用人工方式删除问题. 社区管理员可以直接删除问题. 声誉值超过 10 000 的用户可以投票删除问题. 这种人工删除低质量问题的方法可以有效地维护网站内容质量, 然而这种方式既会导致不合格问题存活时间过长, 还因为需要社区管理员的参与而加重了社区管理员的负担.

在 Stack Overflow 网站, 用户可以发布软件开发相关的问题帖. 以 ID 为 64525237 的问题帖为例 (How to calculate the size of blocks of values in a list?), 它的基本结构如图 1 所示, 主要包括以下几个部分.

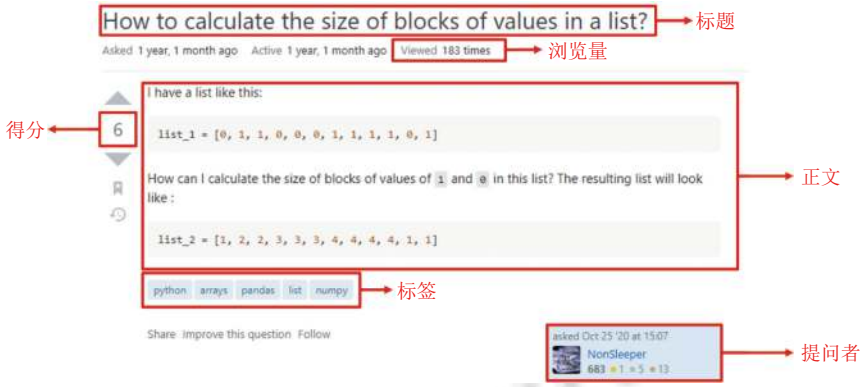

图 1 Stack Overflow 网站问题帖的结构

1) 标题, 即问题的题目. 图中问题的标题为“ How to calculate the size of blocks of values in a list? ”.

2) 浏览量, 即该问题被浏览的次数. 图中问题的浏览量为 61.

3) 得分, 标志着问题的质量. 浏览者可以对问题的质量进行评分. 图中问题的得分为 4.

4) 正文, 问题的主体部分, 包括自然语言描述和代码段.

5) 标签, 概括了问题所属的范畴, 由问题发布者设置. 图中问题的标签包括“Python”“arrays”“pandas”“list”和“numpy”.

6) 提问者, 显示提问者的一些基本信息, 例如图中提问者的用户名是“NonSleeper”.

2 相关研究

近年来, 一些研究人员针对 Stack Overflow 问题的关闭或删除预测进行了相关研究. Correa 等人[5]为了对问题删除进行预测, 从用户档案、社区、内容和句法等 4 个方面提取问题特征, 然后使用 AdaBoost 算法来学习预测模型. Xia 等人[7]在此基础上, 提出了一种两阶段混合方法——DelPredictor. 该方法在第 1 阶段通过文本处理和分类技术将问题标题、正文和标签字段中的文本转换为数字文本特征, 在第 2 阶段根据 Correa 等人的方法[5]提取相同的特征, 接下来分别使用两阶段特征训练独立的的分类模型, 最后使用联合分类器组合两阶段的分类结果. 由于 DelPredictor 预测删除问题的准确度高于 Correa 等人提出的方法[5], 因此本文选择对比方法时使用 DelPredictor. Tóth 等人[8]设计了一个预测问题是否会被关闭的方法, 为方便记述, 本文将该方法称为 NLPPredictor. 该方法将问题的标题、正文和标签文本经过一系列步骤进行预处理, 再编码为词向量, 然后使用 RNN 网络作为分类器, 输入词向量并输出问题关闭预测结果. 由于被关闭一定时间并且未重新开放的问题会被自动删除, 因此问题关闭预测和问题删除预测有一定的相关性, 故 NLPPredictor 也被选为本文的对比方法.

除了对问题关闭预测和问题删除预测的研究之外, 研究者们还对 Stack Overflow 问题开展其他研究[10−12,18−20]. Zhou 等人[10]发现, 有悬赏的问题更易于得到解答. Zhang 等人[12]对 Stack Overflow 中 API 滥用的普遍性和严重性进行了实证研究, 发现约 1/3 的问题包含潜在的 API 违规使用行为. Zhang 等人[18]研究了答案中的知识是如何过时的, 并确定了这些过时答案的特征. Ren 等人[19]设计了一种发现和总结问题中争议的方法. Singh 等人[20]提出一种基于标签关联和代码分析的问题标签推荐方法.

3 MulPredictor 方法设计

针对 Stack Overflow 面临的问题删除问题, 本文提出了自动化预测问题删除的方法 MulPredictor. 本节将对 MulPredictor 的设计思路和特征进行介绍.

3.1 设计思路

一个新问题被发布后可能被删除或者保留, 因此问题删除预测问题可以被转化为机器学习的二分类问题, 即问题的类型标记分别为“删除”和“不删除”. 本文提出自动化预测问题删除的方法 MulPredictor, 整体架构如图 2 所示.

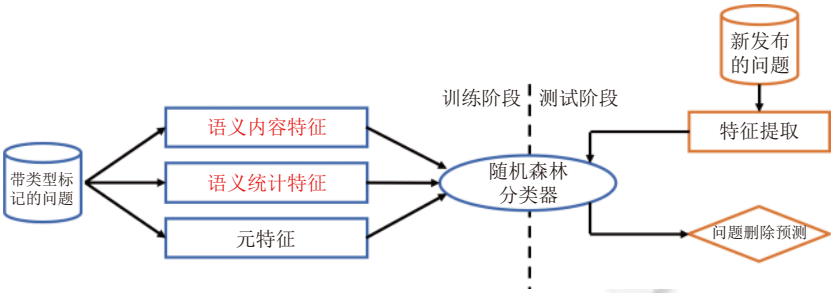

图 2 MulPredictor 的框架

MulPredictor 分为两个阶段: 训练阶段和预测阶段. 在训练阶段, 本文参考 Stack Overflow 给出的提问指南, 提出了影响问题删除预测的特征, 即语义内容特征、语义统计特征和元特征. 本文将在第 3.2 节详细介绍特征含义, 解释采用这些特征的理由. 图 2 中用红色字体标出了语义内容特征和语义统计特征, 表示这是本文新提出的特征; 用黑色字体标出元特征, 表示来源于参考文献 [5] 的特征. 接下来, MulPredictor 将特征输入随机森林分类器, 获得问题删除预测模型. 随机森林是利用多棵树对样本进行训练并预测的一种分类器. 它利用重采样方法, 从原始样本中抽取多个样本, 对每一个样本进行决策树建模, 然后组合所有树的分类投票结果, 输出得票最多的类别. 本文将在第 4.4.2 节详细介绍选择随机森林的原因. 在预测阶段, MulPredictor 获取测试集问题的语义内容特征、语义统计特征和元特征, 然后将特征输入到训练阶段获得的预测模型, 预测新发布问题是否会被删除.

3.2 特 征

MulPredictor 从 3 个维度提取问题的特征, 包括问题的语义内容特征、语义统计特征和元特征, 共计 67 个特征. 其中, 问题的语义内容特征、语义统计特征由本文提出, 问题的元特征由 Correa 等人[5]提出.

3.2.1 语义内容特征

脱离社区主题是问题删除的重要原因. 例如, Stack Overflow 上 ID 为 52584464 的问题是: “如果使用 PAL 拍摄电影, 那么该电影可以在 NTSC 中播放吗?”该问题的提问内容与编程和软件开发无关, 不符合 Stack Overflow 的社区主题, 因此被网站删除. 可以推理得知, 如果新问题和被删除问题的主题更相似, 那么新问题更可能被删除; 如果新问题和被保留问题的主题更相似, 那么新问题更可能被保留. 为了分析问题语义是否符合主题, 本文分别针对被删除和被保留两种情况, 计算新问题的语义内容删除概率和语义内容保留概率.

被删除问题和被保留问题是两类问题. 文本卷积神经网络模型 (text convolutional neural networks) 是由 Kim[21] 提出的使用卷积神经网络来处理自然语言问题的模型, 适用于捕捉语义信息进行文本分类任务. 本文采用 Text CNN 模型分析问题的标题和正文, 计算新问题的语义删除概率和语义保留概率. 具体来讲, 本文对问题的正文进行预处理, 删除正文中的代码段, 留下文本内容. 然后基于 Word2Vec 技术把标题和正文中的文本表示为词向量. 接下来, 输入已被删除问题和已被保留问题的词向量, 建立文本卷积神经网络模型. 最后, 提取新问题的词向量并输入文本卷积神经网络模型, 产生新问题的语义内容删除概率和语义内容保留概率.

3.2.2 语义统计特征

问题描述主要包括标题和正文两部分. 标题概述问题内容, 正文详细介绍问题. 为了指导用户提问, Stack Overflow 网站为用户制定了一个详细的提问指南. 提问指南首先强调标题的重要性, 即“The title is the first thing potential answerers will see, and if your title isn’t interesting, they won’t read the rest.” 此外, 提问指南也强调正文第 1 段的重要性, 即“The first paragraph in your question is the second thing most readers will see, so make it as engaging and informative as possible.” 因此, 本文从标题、正文和正文第 1 段 3 个方面提取问题的语义统计特征.

(1) 标题特征

问题标题是用户对该问题的第一印象, 如果用户对标题不感兴趣, 可能就不会仔细阅读问题的具体内容. 问题标题的可读性、可理解性、语法正确性都会影响用户对问题的印象. 基于以上分析, 本文提取了下列标题特征.

① 标题音节数. 音节数可以一定程度上反映英文单词的难度, 通常情况下单词越简单音节数越少, 超过 3 个音节的单词可能会带来阅读困难.

② 标题 flesch 易读性指数. flesch 易读性指数 (flesch reading ease) 一般被用于评估文档或表格的可读性, 其值越高表示越容易阅读, 它的计算方式如公式 (1) 所示[7].

$$flesch\ reading\ ease=206.835-1.015\times\frac{总单词数}{总句子数}-84.6\times\frac{总音节数}{总单词数} \tag{1}$$

③ 标题文化水平等级. 标题文化水平等级反映读懂一段特定文字需要的文化水平等级 (automated readability index, https://en.wikipedia.org/wiki/Automated_readability_index), 该值越大表示读懂该段文字需要的文化水平等级越高. 例如, 一段文本的总可读性为 8, 表示 8 年级及以上人群可以读懂该段文本.

④ 标题涵盖标签的比例. 标签是对问题主题的极简概括. 若标题含有标签中文本, 则有助于用户快速抓住问题主题. 标题涵盖标签的比例表示标题中含有的标签个数与该问题给出的标签个数的比例.

⑤ 标题语法错误数. 标题语法错误数反映标题中存在的语法错误数量. 该值越大表示标题语法错误越多.

(2) 正文特征

除标题外, 正文的可读性、可理解性等也会影响问题质量. 参考标题特征的选取, 本文也考虑: ① 正文的音节数、② flesch 易读性指数、③ 文化水平等级、④ 语法错误数.

除此之外, 本文还考虑⑤正文的情感倾向. 它分为消极和积极两种, 可能对阅读者的心理产生影响, 从而影响阅读者对问题的评价, 因此本文也加以考虑. 正文中不仅包含自然语言段, 也有可能包含代码段. 本文还考虑⑥正文代码段与自然语言段比例, 一般情况下, 比例大说明提供问题包含详细的代码信息, 有助于用户理解问题.

(3) 正文第 1 段特征

正文第 1 段通常是除标题外用户最先看到的部分. Stack Overflow 的提问指南建议: “In the body of your question, start by expanding on the summary you put in the title. Explain how you encountered the problem you're trying to solve, and any difficulties that have prevented you from solving it yourself.” 正文第 1 段通常概述问题内容, 其写作尤为重要. 因此, 本文单独考虑正文第 1 段特征. 参考标题, 本文提取: ① 正文第 1 段音节数、② 正文第 1 段 flesch 易读性指数、③ 正文第 1 段文化水平等级、④ 正文第 1 段语法错误数.

正文第 1 段一般用于对问题内容进行总体介绍. 如果正文第 1 段直接呈现代码, 而不是描述文字, 可能对用户理解该问题产生负面影响. 因此, 本文考虑: ⑤ 正文第 1 段是否为自然语言. 另外, 本文还考虑: ⑥ 正文第 1 段单词数、⑦ 正文第 1 段单词平均长度, 用以描述正文第 1 段的复杂程度. 如果正文第 1 段过于详细, 单词数量太多, 读者可能难以快速了解问题内容.

3.2.3 问题元特征

除了本文新提出语义内容特征和语义统计特征之外, 本文还采用了 Correa 等人[5]提出的 47 个元特征. 这些特征分为 4 类: 档案特征、社区特征、内容特征和句法特征. 元特征虽然不是本文新提出的, 但是已经被证明对问题删除预测有重要的参考价值[5,7], 因此 MulPredictor 采用了这些特征, 作为语义内容特征和语义统计特征的补充. 接下来, 本文概述元特征, 详细定义见现有工作[5].

(1) 档案特征

档案特征包含了提问者以往的提问情况、回答情况、获得的徽章数等特征. 档案信息反映提问者的历史提问水平. 提问者的档案特征包括以下内容: 提问者的注册时长, 提问者以往发布的得分为负、为零、为正的问题和回答数量, 提问者以往的提问总数量、回答总数量, 提问者获得的徽章数, 以及提问者的提问频率 (即提问者以往的提问总数量/提问者的注册时长) 和回答频率 (即提问者以往的回答总数量/提问者的注册时长).

(2) 社区特征

提问者的社区特征度量其历史提问或者回答情况, 主要包括: 问题的平均得分, 回答的平均得分, 平均问题浏

览量, 平均问题评论数, 平均问题喜欢票数, 以及平均接受回答数.

(3) 内容特征

问题的内容特征是根据问题的文本内容定义的. 在提取部分内容特征时, 本文用到了 LIWC (linguistic inquiry and word count) 分析工具, 分析问题中自然语言文本的心理计量特性[22,23]. 内容特征包括: 问题中的 URL 数量和标签数量, 问题的代码段长度, 以及人称代词、代词、空间词、相对单词、包含词、认知过程词、社交词和第一人称奇异代词的 LIWC 得分.

(4) 句法特征

句法特征主要度量问题文本的写作风格. 句法特征包括: 功能词、连词和介词的 LIWC 得分, 正文中的字符数、字母字符数、大写字母数、小写字母数、数字字符数、空格字符数、特殊字符数、标点符号数、单词数、短单词数、唯一单词数、单词的平均长度, 以及标题中的字符数、单词数和单词的平均长度.

4 实验评估

4.1 数据集

为了获取问题的特征数据, 本文使用了 Stack Overflow 网站的转储文件, 包括 Posts 文件、Users 文件和 Badges 文件. 本文从这 3 个文件中提取相应字段的信息并计算特征. 例如, 使用 Posts 文件中的标题和正文等字段构建语义内容特征; 使用 Posts 文件中的标题、正文和标签等字段构建语义统计特征; 使用 Posts 文件中的问题发布时间、问题得分、回答发布时间、回答得分、标题、正文、标签、(问题或回答的) 拥有者 ID、浏览量、评论数、喜欢人数、问题接受的答案 ID, 以及 Users 文件中的用户注册日期、用户 ID, 以及 Badges 文件中的用户 ID、用户获得勋章的时间等字段构建元特征.

Stack Overflow 的转储文件每隔一段时间就会更新, 例如 2018 年 1 月 1 日的 Posts 文件记录了从 2008 年网站投入使用以来至 2018 年 1 月 1 日期间发布的问题中未被删除的问题. 为了尽可能保证实验中使用的问题的类别正确性, 本文使用一定时间内保持稳定状态的问题构建数据集. 具体来讲, 使用 2019 年 1 月的转储文件以及 2020 年 1 月的转储文件构建数据集. 如果一个问题存在于 2019 年 1 月的 Posts 文件中, 却不在 2020 年 1 月的 Posts 文件中, 就认为这个问题被删除了; 如果该问题仍存在于 2020 年 1 月的 Posts 文件中, 也就是说截至 2020 年 1 月仍未被删除, 那么其后续被删除的概率也低, 本文认为它是保留问题. 通过上述方法, 本文从 2018 年 10 月 1 日–2018 年 12 月 31 日期间发布的问题中获取了共 76 745 个被删除问题和 449 769 个未被删除问题. 为了防止出现使用晚发布的问题预测早发布的问题是否会被删除的情况, 本文以 2018 年 10 月 1 日–2018 年 11 月 15 日作为训练集获取区间, 使用这期间发布的问题构建训练集; 以 2018 年 11 月 16 日–2018 年 12 月 31 日作为测试集获取区间, 使用这期间发布的问题构建测试集.

构建训练集时, 考虑到未被删除问题的数量远大于被删除问题, 为了防止过拟合现象的发生, 根据平衡训练集的思想, 本文从训练集获取区间中随机抽取了 10 000 个被删除问题和 10 000 个未被删除问题组成训练集. 在构建测试集时, 考虑到平衡测试集适用于反映方法的有效性, 而随机测试集适用于反映方法的真实使用情况, 本文分别构建了这两种测试集, 从测试集获取区间发布的问题中随机抽取 10 000 个被删除问题和 10 000 个未被删除问题组成平衡测试集, 然后随机抽取 20 000 个问题组成随机测试集.

4.2 评估指标

对于预测问题, 常用的评估指标是准确率 (*accuracy*)、精确率 (*precision*)、召回率 (*recall*) 和 F_1 值. 本文使用符号“*TP*”表示被删除问题预测为被删除问题的数量, “*TN*”表示被保留问题预测为被保留问题的数量, “*FP*”表示被保留问题预测为被删除问题的数量, “*FN*”表示被删除问题预测为被保留问题的数量.

准确率指标由公式 (2) 定义.

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{2}$$

精确率指标由公式 (3) 和公式 (4) 定义.

$$precision_d = \frac{TP}{TP+FP} \tag{3}$$

$$precision_{nd} = \frac{TN}{TN+FN} \tag{4}$$

上式中不同下标的精确率含义不同, $precision_d$ 表示将“被删除”视为正例的精确率 (下标“*d*”表示“deleted”), $precision_{nd}$ 表示将“被保留”视为正例的精确率 (下标“*nd*”表示“not deleted”), 后文中其他评估指标的下标含义同理.

召回率指标由公式 (5) 和公式 (6) 定义.

$$recall_d = \frac{TP}{TP+FN} \tag{5}$$

$$recall_{nd} = \frac{TN}{TN+FP} \tag{6}$$

F_1 值是精确率和召回率的调和平均值, 由公式 (7) 和公式 (8) 定义.

$$F_{1_d} = \frac{2\times precision_d \times recall_d}{precision_d + recall_d} \tag{7}$$

$$F_{1_{nd}} = \frac{2\times precision_{nd} \times recall_{nd}}{precision_{nd} + recall_{nd}} \tag{8}$$

此外, 为了比较不同方法的性能差异, 本文定义了增益值 (gain) 来计算方法 *i* 和方法 *j* 的效果差异, 包括准确率增益、精确率增益、召回率增益和 F_1 值增益, 这些增益值由公式 (9)–公式 (12) 定义.

$$gain(accuracy) = \frac{accuracy(i) - accuracy(j)}{accuracy(j)} \tag{9}$$

$$gain(precision_{d|nd}) = \frac{precision_{d|nd}(i) - precision_{d|nd}(j)}{precision_{d|nd}(j)} \tag{10}$$

$$gain(recall_{d|nd}) = \frac{recall_{d|nd}(i) - recall_{d|nd}(j)}{recall_{d|nd}(j)} \tag{11}$$

$$gain(F_{1_{d|nd}}) = \frac{F_{1_{d|nd}}(i) - F_{1_{d|nd}}(j)}{F_{1_{d|nd}}(j)} \tag{12}$$

4.3 研究问题

本文研究以下 3 个问题.

RQ1: MulPredictor 预测问题删除的效果如何?

本文将 MulPredictor 方法与 Xia 等人提出的 DelPredictor 方法[7]、Tóth 等人提出的 NLPPredictor 方法[8]相比较, 在平衡测试集和随机测试集验证 MulPredictor 方法对问题删除的预测效果.

RQ2: MulPredictor 应该采用哪种分类算法?

MulPredictor 使用机器学习分类算法来预测问题是否被删除. 本文尝试 4 种常见的分类算法——支持向量机、随机森林、逻辑回归和 XGBoost, 通过实验比较不同算法的预测效果, 将效果最好的分类算法应用于 MulPredictor.

RQ3: MulPredictor 使用的哪些特征对问题删除预测发挥重要作用?

MulPredictor 使用 scikit-learn 库实现预测模型. scikit-learn 库除了提供多种分类算法之外, 还支持分析各输入特征的重要性, 从而可以找出对问题删除预测影响较大的特征. 通过对这些特征进行分析, 本文还为用户如何在 Stack Overflow 社区提问给出指导意见.

4.4 评估结果

4.4.1 RQ1: 与同类方法的对比

Xia 等人[7]提出了一种两阶段混合的问题删除预测方法——DelPredictor. 该方法在第 1 阶段通过文本处理和分类技术将问题标题、正文和标签字段中的文本转换为数字文本特征, 在第 2 阶段根据 Correa 等人的方法[5]提取相同的特征, 接下来分别使用两阶段特征训练独立的分类模型, 最后使用联合分类器组合两阶段的分类结果.

Tóth 等人[8]设计了一个预测问题在发布后是否会被关闭的方法, 本文将其称为 NLPPredictor. 该方法将问题的标题、正文和标签文本经过一系列步骤进行预处理, 再编码为词向量, 然后使用 RNN 网络作为分类器, 输入词向量并输出问题关闭预测结果. 由于被关闭一定时间并且未重新开放的问题会被自动删除, 因此问题关闭预测和问题删除预测有一定的相似性. 本文使用 DelPredictor 和 NLPPredictor 作为 MulPredictor 的对比方法.

实验结果如表 1 和表 2 所示, 其中评估指标的下标为“*d*”或“*nd*”表示该指标针对删除问题或者未删除问题. 可以看到, MulPredictor 在平衡测试集和随机测试集的准确率分别达到 62.65% 和 64.10%.

表 1 MulPredictor 与现有方法的性能比较 (基于平衡测试集)(%)

方法	*accuracy*	$precision_d$	$recall_d$	F_{1d}	$precision_{nd}$	$recall_{nd}$	F_{1nd}
DelPredictor	53.85	53.98	52.20	53.72	53.72	55.50	54.59
NLPPredictor	55.55	55.27	58.20	55.86	55.86	52.90	54.34
MulPredictor	**62.65**	**63.22**	**60.50**	**61.83**	**62.13**	**64.80**	**63.44**

表 2 MulPredictor 与现有方法的性能比较 (基于随机测试集)(%)

方法	*accuracy*	$precision_d$	$recall_d$	F_{1d}	$precision_{nd}$	$recall_{nd}$	F_{1nd}
DelPredictor	57.04	20.77	40.80	27.53	80.50	61.10	69.47
NLPPredictor	56.16	24.22	56.00	33.81	**83.63**	56.20	67.22
MulPredictor	**64.10**	**39.75**	**57.50**	**47.00**	80.37	**66.63**	**72.86**

本文还对各个评估指标进行了增益计算, 用各评估指标的 *gain* 值表示, 计算结果如表 3 和表 4 所示. 从表 3 可以看出, 在平衡测试集上, MulPredictor 方法在准确率、F_1 值 (删除) 和 F_1 值 (未删除) 这些指标上分别优于 DelPredictor 方法 16.34%, 15.10%, 16.21%, 分别优于 NLPPredictor 方法 12.78%, 10.69%, 16.75%. 从表 4 可以看出, 在随机测试集上, MulPredictor 方法在准确率、F_1 值 (删除) 和 F_1 值 (未删除) 这些指标上分别优于 DelPredictor 方法 12.38%, 70.72%, 4.88%, 分别优于 NLPPredictor 方法 14.14%, 39.01%, 8.39%. 可以看到对于重要指标 F_1 值 (删除), 无论在平衡测试集还是随机测试集上, MulPredictor 相较于两个对比方法均有较大提升.

表 3 gain 值计算结果 (基于平衡测试集)(%)

方法	*accuracy*	$precision_d$	$recall_d$	F_{1d}	$precision_{nd}$	$recall_{nd}$	F_{1nd}
DelPredictor	16.34	17.12	15.90	15.10	15.66	16.76	16.21
NLPPredictor	12.78	14.38	3.95	10.69	11.22	22.50	16.75

表 4 gain 值计算结果 (基于随机测试集)(%)

方法	*accuracy*	$precision_d$	$recall_d$	F_{1d}	$precision_{nd}$	$recall_{nd}$	F_{1nd}
DelPredictor	12.38	91.38	40.93	70.72	−0.16	9.05	4.88
NLPPredictor	14.14	64.12	2.68	39.01	−3.90	18.56	8.39

综上所述, 无论在平衡测试集上还是随机测试集上, MulPredictor 方法的预测效果都优于对比方法 DelPredictor 和 NLPPredictor.

尽管 MulPredictor 在问题删除预测效果上相比两个对比方法更出色, 但是在一些情况下, MulPredictor 也会预测失效. 我们人工查看预测失败的问题, 发现一些典型例子, 指导我们未来改进方法.

(1) 问题的真实类别为“删除”, 预测结果为“不删除”. 例如 ID 8861969 的问题正文提到“I know this question has been asked severally. I need someone to check why the application displays a success message but no mail was received in my mail. Here is my code”. 提问者知道这个问题已经被询问过多次但是自己找不到答案, 所以又问了一次. 这种问题很可能会被管理员删除, 以防止社区充斥大量内容重复的问题.

(2) 问题的真实类别为“不删除”, 预测结果为“删除”. 例如 ID 3220999 的问题标题是“xlim in ggplot with POSIXct dates”. Stack Overflow 不支持带格式的标题文本, 因此不能在编写标题时把“POSIXct” “.AspNetCore.

Identity.Application”等词块标注为代码块, 故自动分析工具判断该问题的标题有 4 处语法错误, 进而预测该问题会被删除. 实际上, 该问题的标题描述准确, 所以没有被管理员删除.

RQ1: 与现有方法 DelPredictor 和 NLPPredictor 相比, MulPredictor 在平衡测试集和随机测试集上都具有更好的效果.

4.4.2 RQ2: 分类算法选择

MulPredictor 使用机器学习分类算法来预测问题删除. 本文尝试了 4 种机器学习分类算法, 包括支持向量机、随机森林、逻辑回归和 XGBoost. 支持向量机的原理是寻找一个最优分类超平面, 既能保证分类准确度, 又能最大化超平面两侧的空白区域. 随机森林利用重采样方法, 从原始样本中抽取多个样本, 并对每一个样本进行决策树建模, 然后组合所有树的分类投票结果, 输出得票最多的类别. 逻辑回归根据现有数据对分类边界线建立回归公式, 从而解决二分类问题. XGBoost 是一种基于梯度提升树的集成学习算法.

为了选择合适的分类算法, 本文评估 4 种分类算法在平衡测试集和随机测试集的问题预测效果. 实验结果如表 5 和表 6 所示. 其中评估指标的下标为“*d*”或“*nd*”表示该指标针对删除问题或者未删除问题. 结果显示, 机器学习分类算法对问题删除预测的影响较小, 不同机器学习分类算法的结果接近. 相对而言, 基于随机森林算法的问题删除预测方法在平衡测试集和随机测试集都取得最高的准确率. 因此, 本文使用随机森林算法实现问题删除预测. 另外, 从表 6 中还可以看出, 随机森林算法和 XGBoost 算法在随机测试集上的准确率相同, 因此在实际应用中, 可以根据具体项目的数据选择合适的分类算法.

表 5 不同分类算法性能比较 (基于平衡测试集)(%)

分类算法	*accuracy*	$precision_d$	$recall_d$	F_{1d}	$precision_{nd}$	$recall_{nd}$	F_{1nd}
支持向量机	58.60	58.74	57.80	58.27	58.46	59.40	58.93
随机森林	**62.65**	**63.22**	60.50	61.83	62.13	**64.80**	**63.44**
逻辑回归	60.55	61.53	56.30	58.80	59.72	**64.80**	62.16
XGBoost	61.35	60.62	**64.80**	**62.64**	**62.19**	57.90	59.97

表 6 不同分类算法性能比较 (基于随机测试集)(%)

分类算法	*accuracy*	$precision_d$	$recall_d$	F_{1d}	$precision_{nd}$	$recall_{nd}$	F_{1nd}
支持向量机	61.42	37.17	56.98	44.99	79.31	63.12	70.29
随机森林	**64.10**	39.75	57.50	47.00	80.37	66.63	**72.86**
逻辑回归	63.02	38.05	53.46	44.46	78.91	**66.68**	72.28
XGBoost	**64.10**	**39.81**	**58.01**	**47.22**	**80.51**	66.42	72.79

RQ2: 基于随机森林算法的问题删除预测方法在平衡测试集和随机测试集都取得最高的准确率.

4.4.3 RQ3: 特征重要性分析

本文提取了问题的 2 个语义内容特征、18 个语义统计特征和 47 个元特征, 共计 67 个特征, 采用随机森林分类器作为分类算法实现 MulPredictor 方法. 由于 scikit-learn 库提供的随机森林算法提供了“feature_importances_”接口, 可以获得每个特征对问题删除预测的影响大小. 第 i 个特征的影响因子值占最大影响因子值的比例 $ratio_i$ 的计算方法如公式 (13) 所示. 其中 $importance_i$ 表示第 i 个特征对问题删除预测的影响大小, $importance_{\max}$ 表示影响最大的特征的影响因子值.

$$ratio_i = \frac{importance_i}{importance_{\max}} \tag{13}$$

表 7 列举了所有特征中对随机森林分类算法影响比例最大的前 10 个特征. 这其中包括本文新提出的 5 个特征, 分别是语义内容删除概率 (第 1)、语义内容保留概率 (第 2)、代码段与自然语言段比例 (第 4)、正文第 1 段单词数 (第 6) 以及正文第 1 段语法错误数 (第 10). 这一结果表明本文新提出的语义内容特征和语义统计特征是有效的, 在元特征的基础上提高了预测问题删除效果.

如表 7 所示, 问题的两个语义内容特征 (即语义内容删除概率和语义内容保留概率) 是影响比例最大的两个

特征, 这说明问题的内容是影响问题删除的最重要因素. 如果一个问题的内容不符合网站要求, 就很有可能被网站删除. 例如, Stack Overflow 上 ID 为 52584464 的问题是: “如果使用 PAL 拍摄电影, 那么该电影可以在 NTSC 中播放吗?”该问题的提问内容与编程和软件开发无关, 不符合 Stack Overflow 的社区主题, 因此被网站删除. 因此, 我们建议用户在 Stack Overflow 网站中提问前了解网站允许发布的问题内容. Stack Overflow 对允许发布的问题主题已经做了确切说明 (What topics can I ask about here? https://stackoverflow.com/help/on-topic), 并指出了应该避免发布哪些内容 (What types of questions should I avoid asking? https://stackoverflow.com/help/dont-ask). 在发布问题之前, 问题撰写者应该了解相应说明并判断提出的问题是否符合 Stack Overflow 允许的主题, 避免问题主题不符合要求而被删除.

表 7 对问题删除预测影响最大的 10 个特征

特征	影响因子比例	是否由本文提出
语义内容删除概率	1.000 0	是
语义内容保留概率	0.531 0	是
代码段长度	0.023 9	否
代码段与自然语言段比例	0.019 1	是
功能词的LIWC得分	0.018 8	否
正文第一段单词数	0.017 8	是
介词的LIWC得分	0.017 1	否
标题中单词的平均长度	0.015 2	否
第一人称奇异代词的LIWC得分	0.014 1	否
正文第一段语法错误数	0.013 3	是

除了上述两个语义内容特征之外, 代码段的写作对问题是否被删除也有较大的影响. 从表 7 可以看到, 代码段长度、代码段与自然语言段比例这两个特征对问题删除的影响大小分别排在第 3、4 位. 本文分别对被删除问题集和未被删除问题集里的这两个特征的均值和中位数进行了分析, 结果如表 8 第 1、2 行所示. 可以看出, 删除问题集上这两个特征的均值和中位数均小于未删除问题集. 这说明被删除的问题在正文里给出的代码往往更少. 一般情况下, 正文代码段与自然语言段比例越大、代码长度越长, 代码信息就越详细, 其他用户就更容易理解和复现问题. 因此, 我们建议问题撰写者在提问时应详细地给出代码信息, 并对其加以解释, 以帮助其他用户理解和复现该问题.

表 8 典型特征在不同问题集的均值和中位数

特征	均值		中位数	
	删除问题集	未删除问题集	删除问题集	未删除问题集
代码段长度	442.35	549.79	27.50	178.00
代码段与自然语言段比例	0.35	0.43	0.29	0.40
正文第一段单词数	35.10	28.80	30.00	24.00
正文第一段语法错误数	4.42	3.23	2.00	2.00
标题中单词的平均长度	5.49	5.27	5.25	5.00

除了代码段的相关特征之外, 正文第 1 段的相关特征对问题删除影响也很大. 从表 7 可以看到, 正文第 1 段单词数和正文第 1 段语法错误数这两个特征对问题删除的影响大小分别排在第 6、10 位. 本文分别统计这两个特征在被删除问题集和未被删除问题集里的均值和中位数, 结果如表 8 第 3、4 行所示. 可以看到, 这两个特征在被删除问题的均值大于未被删除问题集. 这说明被删除的问题往往把正文第 1 段写得更长, 并且犯更多的语法错误. 正文第 1 段写的过长会导致概括性不佳, 让用户难以把握问题的重点; 而语法错误影响用户对问题的理解. 基于以上分析, 我们建议问题撰写者在书写正文第 1 段时尽量使用概括性的语言, 并且应该格外注意使用正确的语法. 在书写英文时, 撰写者可以借助额外的语法检查工具, 避免出现过多的语法错误导致问题删除.

此外, 从表 7 还可以看到, 标题中单词的平均长度对问题删除结果也有较大的影响. 如表 8 第 5 行所示, 被删

除问题集的标题单词平均长度的均值和中位数均要大于未被删除问题集. 这说明, 被删除的问题往往在标题中使用更长的单词. 在标题中使用长单词会加大用户理解标题的难度, 影响用户对问题的评价. 基于以上分析, 我们建议问题撰写者在撰写问题标题时应尽量选择较为简短的单词, 便于他人理解.

RQ3: 语义内容删除概率、语义内容保留概率、代码段长度和代码段与自然语言段比例等特征在问题删除预测发挥重要作用.

5 有效性分析

5.1 内部有效性威胁

内部有效性威胁与研究中使用的度量和实验中的设置有关. 首先, 本文的问题删除预测方法考虑语义内容特征、语义统计特征和元特征. 在未来工作中, 我们计划参考 Stack Overflow 提问指南, 尝试更多的特征并分析其对问题删除预测的影响. 其次, 本文在测试 4 种常见分类模型的分类效果时均使用了默认参数. 在未来的工作中, 我们计划分析参数设置对模型效果的影响, 尝试改进方法. 此外, 本方法使用的问题标签是由提问者设置的, 反映了提问者对问题的理解. 可能个别提问者设置的标签不准确. 在未来工作中, 我们计划分析标签设置准确性对问题删除预测的影响.

5.2 外部有效性威胁

外部有效性主要体现在研究方法的普遍性上. 本文选取了 Stack Overflow 进行研究和实验评估, 但不确定该方法在其他平台的效果. 在今后的工作中我们将扩展实验中的数据来源, 进一步验证并调整方法.

6 总 结

本文针对 Stack Overflow 网站人工删除耗时费力的问题, 提出了自动化预测问题删除方法 MulPredictor. 该方法提取问题的语义内容特征、语义统计特征和元特征, 使用随机森林来分析得到问题是否会被删除的概率. 实验结果表明, 相比现有方法 DelPredictor 和 NLPPredictor, MulPredictor 在平衡测试上的准确率分别提升了 16.34% 和 12.78%, 在随机测试集上的准确率分别提升了 12.38% 和 14.14%. 此外, 本文还分析了特征重要性并进行解释, 为用户提问提供指导.

References:

[1] Phukan D, Singha AK. Feasibility analysis for popularity prediction of stack exchange posts based on its initial content. In: Proc. of the 3rd Int'l Conf. on Computing for Sustainable Global Development (INDIACom). New Delhi: IEEE, 2016. 1397–1402.

[2] Wu YH, Wang SW, Bezemer CP, Inoue K. How do developers utilize source code from stack overflow? Empirical Software Engineering, 2019, 24(2): 637–673. [doi: 10.1007/s10664-018-9634-5]

[3] Ahasanuzzaman M, Asaduzzaman M, Roy CK, Schneider KA. Classifying stack overflow posts on API issues. In: Proc. of the 25th IEEE Int'l Conf. on Software Analysis, Evolution and Reengineering (SANER). Campobasso: IEEE, 2018. 244–254. [doi: 10.1109/SANER.2018.8330213]

[4] Mamykina L, Manoim B, Mittal M, Hripcsak G, Hartmann B. Design lessons from the fastest Q&A site in the west. In: Proc. of the SIGCHI Conf. on Human Factors in Computing Systems. Vancouver: ACM, 2011. 2857–2866. [doi: 10.1145/1978942.1979366]

[5] Correa D, Sureka A. Chaff from the wheat: Characterization and modeling of deleted questions on stack overflow. In: Proc. of the 23rd Int'l Conf. on World Wide Web. Seoul: ACM, 2014. 631–642. [doi: 10.1145/2566486.2568036]

[6] Barua A, Thomas SW, Hassan AE. What are developers talking about? An analysis of topics and trends in stack overflow. Empirical Software Engineering, 2014, 19(3): 619–654. [doi: 10.1007/s10664-012-9231-y]

[7] Xia X, Lo D, Correa D, Sureka A, Shihab E. It takes two to tango: Deleted stack overflow question prediction with text and meta features. In: Proc. of the 40th IEEE Annual Computer Software and Applications Conf. (COMPSAC). Atlanta: IEEE, 2016. 73–82. [doi: 10.1109/COMPSAC.2016.145]

[8] Tóth L, Nagy B, Gyimóthy T, Vidács L. Why will my question be closed?: NLP-based pre-submission predictions of question closing reasons on stack overflow. In: Proc. of the 42nd ACM/IEEE Int'l Conf. on Software Engineering: New Ideas and Emerging Results.

Seoul: ACM, 2020. 45–48. [doi: 10.1145/3377816.3381733]

[9] Zhang W, Wang W, Wang J, Zha HY. User-guided hierarchical attention network for multi-modal social image popularity prediction. In: Proc. of the 2018 World Wide Web Conf. Lyon: Int'l World Wide Web Conferences Steering Committee, 2018. 1277–1286. [doi: 10.1145/3178876.3186026]

[10] Zhou JY, Wang SW, Bezemer CP, Hassan AE. Bounties on technical Q&A sites: A case study of Stack Overflow bounties. Empirical Software Engineering, 2020, 25(1): 139–177. [doi: 10.1007/s10664-019-09744-3]

[11] Pârţachi PP, Dash SK, Treude C, Barr ET. POSIT: Simultaneously tagging natural and programming languages. In: Proc. of the 42nd ACM/IEEE Int'l Conf. on Software Engineering. Seoul: ACM, 2020. 1348–1358. [doi: 10.1145/3377811.3380440]

[12] Zhang TY, Upadhyaya G, Reinhardt A, Rajan H, Kim M. Are code examples on an online Q&A forum reliable?: A study of API misuse on stack overflow. In: Proc. of the 40th IEEE/ACM Int'l Conf. on Software Engineering (ICSE). Gothenburg: IEEE, 2018. 886–896. [doi: 10.1145/3180155.3180260]

[13] Beyer S, Macho C, Di Penta M, Pinzger M. What kind of questions do developers ask on Stack Overflow? A comparison of automated approaches to classify posts into question categories. Empirical Software Engineering, 2020, 25(3): 2258–2301. [doi: 10.1007/s10664-019-09758-x]

[14] An L, Mlouki O, Khomh F, Antoniol G. Stack overflow: A code laundering platform? In: Proc. of the 24th IEEE Int'l Conf. on Software Analysis, Evolution and Reengineering (SANER). Klagenfurt: IEEE, 2017. 283–293. [doi: 10.1109/SANER.2017.7884629]

[15] Gómez C, Cleary B, Singer L. A study of innovation diffusion through link sharing on stack overflow. In: Proc. of the 10th Working Conf. on Mining Software Repositories (MSR). San Francisco: IEEE, 2013. 81–84. [doi: 10.1109/MSR.2013.6624011]

[16] Linares-Vásquez M, Dit B, Poshyvanyk D. An exploratory analysis of mobile development issues using stack overflow. In: Proc. of the 10th Working Conf. on Mining Software Repositories (MSR). San Francisco: IEEE, 2013. 93–96. [doi: 10.1109/MSR.2013.6624014]

[17] Wang W, Godfrey MW. Detecting API usage obstacles: A study of iOS and android developer questions. In: Proc. of the 10th Working Conf. on Mining Software Repositories (MSR). San Francisco: IEEE, 2013. 61–64. [doi: 10.1109/MSR.2013.6624006]

[18] Zhang HX, Wang SW, Chen TH, Zou Y, Hassan AE. An empirical study of obsolete answers on Stack Overflow. IEEE Trans. on Software Engineering, 2021, 47(4): 850–862. [doi: 10.1109/TSE.2019.2906315]

[19] Ren XX, Xing ZC, Xia X, Li GQ, Sun JL. Discovering, explaining and summarizing controversial discussions in community Q&A sites. In: Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). San Diego: IEEE, 2019. 151–162. [doi: 10.1109/ASE.2019.00024]

[20] Singh P, Chopra R, Sharma O, *et al*. Stackoverflow tag prediction using tag associations and code analysis. Journal of Discrete Mathematical Sciences and Cryptography, 2020, 23(1): 35–43. [doi: 10.1080/09720529.2020.1721857]

[21] Kim Y. Convolutional neural networks for sentence classification. In: Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP). Doha: Association for Computational Linguistics, 2014. 1746–1751. [doi: 10.3115/v1/D14-1181]

[22] Tausczik YR, Pennebaker JW. The psychological meaning of words: LIWC and computerized text analysis methods. Journal of Language and Social Psychology, 2010, 29(1): 24–54. [doi: 10.1177/0261927X09351676]

[23] McHaney R, Tako A, Robinson S. Using LIWC to choose simulation approaches: A feasibility study. Decision Support Systems, 2018, 111: 1–12. [doi: 10.1016/j.dss.2018.04.002]

蒋竞(1985－), 女, 博士, 副教授, CCF 专业会员, 主要研究领域为智能软件工程, 经验软件工程, 开源软件, 软件库挖掘.

赵丽娴(1997－), 女, 硕士生, 主要研究领域为数据挖掘, 机器学习, 自然语言处理.

苗萌(1998－), 男, 硕士生, CCF 学生会员, 主要研究领域为自然语言处理, 软件库挖掘.

张莉(1968－), 女, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为软件建模与分析, 需求工程, 实证软件工程, 软件体系结构.