软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2023,34(4):1613−1629 [doi: 10.13328/j.cnki.jos.006697]

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

一种语义感知的细粒度 App 评论缺陷挖掘方法*

王亚文[1,2], 王俊杰[1,2], 石 琳[1,2], 王 青[1,2,3]

[1](中国科学院 软件研究所 互联网软件技术实验室, 北京 100190)

[2](中国科学院大学, 北京 100049)

[3](计算机科学国家重点实验室(中国科学院 软件研究所), 北京 100190)

通信作者: 王俊杰, E-mail: junjie@iscas.ac.cn

摘 要: 手机用户提交的 App 评论为开发者提供了一个了解用户满意度的沟通渠道. 许多用户通常使用“send a video”和“crash”等关键短语来描述有缺陷的功能(即用户操作)和 App 的异常行为(即异常行为), 而这些短语可能会与其他琐碎信息(如用户的抱怨)一起交杂在评论文本中. 掌握这些细粒度信息可以帮助开发者理解来自用户的功能需求或缺陷报告, 进而有利于提升 App 的质量. 现有的基于模式的目标短语提取方法只能对评论的高层主题/方面进行总结, 并且由于对评论的语义理解不足, 短语提取的性能较差. 提出了一种语义感知的细粒度 App 评论缺陷挖掘方法(Arab), 来提取用户操作和异常行为, 并挖掘两者之间的关联关系. 设计了一种新颖的用于提取细粒度目标短语的神经网络模型, 该模型将文本描述和评论属性相结合, 能更好地建模评论的语义. Arab 还根据语义关系对提取的短语进行聚类, 并将用户操作和异常行为之间的关联关系进行了可视化. 使用 6 个 App 的 3 426 条评论进行评估实验, 实验结果证实了 Arab 在短语提取方面的有效性. 进一步使用 Arab 对 15 个热门 App 的 301 415 条评论进行了案例研究, 以探索其潜在的应用, 并验证其在大规模数据上的实用性.

关键词: App 评论; 信息提取; 深度学习

中图法分类号: TP311

中文引用格式: 王亚文, 王俊杰, 石琳, 王青. 一种语义感知的细粒度 App 评论缺陷挖掘方法. 软件学报, 2023, 34(4): 1613−1629. http://www.jos.org.cn/1000-9825/6697.htm

英文引用格式: Wang YW, Wang JJ, Shi L, Wang Q. Semantic-aware and Fine-grained App Review Bug Mining Approach. Ruan Jian Xue Bao/Journal of Software, 2023, 34(4): 1613−1629 (in Chinese). http://www.jos.org.cn/1000-9825/6697.htm

Semantic-aware and Fine-grained App Review Bug Mining Approach

WANG Ya-Wen[1,2], WANG Jun-Jie[1,2], SHI Lin[1,2], WANG Qing[1,2,3]

[1](Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

[2](University of Chinese Academy of Sciences, Beijing 100049, China)

[3](State Key Laboratory of Computer Sciences (Institute of Software, Chinese Academy of Sciences), Beijing 100190, China)

Abstract: App reviews are considered as a communication channel between users and developers to perceive user’s satisfaction. Users usually describe buggy features (i.e., user actions) and App abnormal behaviors (i.e., abnormal behaviors) in forms of key phrases (e.g., “send a video” and “crash”), which could be buried with other trivial information (e.g., complaints) in the review texts. A fine-grained view about this information could facilitate the developers’ understanding of feature requests or bug reports from users, and improve the quality of Apps. Existing pattern-based approaches to extract target phrases can only summarize the high-level topics/aspects of reviews, and suffer from low performance due to insufficient semantic understanding of reviews. This study proposes a semantic-aware and fine-grained **A**pp **r**eview bug mining approach (Arab) to extract user **a**ctions and abnormal **be**haviors, and mine the correlations between them. A novel neural network model is designed for extracting fine-grained target phrases, which combines textual descriptions and

* 基金项目: 国家重点研发计划(2018YFB1403400); 国家自然科学基金(62072442)
收稿时间: 2022-01-19; 修改时间: 2022-03-04; 采用时间: 2022-04-14; jos 在线出版时间: 2022-07-22

review attributes to better represent the semantics of reviews. Arab also clusters the extracted phrases based on their semantic relations and provides a visualization of correlations between User Actions and Abnormal Behaviors. 3,426 reviews from six Apps are used to carry out evaluation test, and the results confirm the effectiveness of Arab in phrase extraction. A case study is further conducted with Arab on 301,415 reviews of 15 popular Apps to explore its potential application and examine its usefulness on large-scale data.

Key words: App review; information extraction; deep learning

移动应用(App)经过十几年的发展, 在我们的日常生活中变得越来越不可或缺. 开发者开发了数以百万计的可用 App, 用于处理各种各样的任务(如购物、银行和社交互动). App 的重要性促使开发团队竭尽全力地理解用户反馈的功能需求和缺陷报告, 并开展质量保证和软件维护活动. App 用户通常会在苹果应用商店和谷歌 Play 商店等平台上为他们所使用的 App 撰写评论. 这些用户评论通常是一些简短的文本, 可以为 App 开发人员提供有价值的信息, 如用户体验、缺陷报告和对新功能的需求[1−3]. 充分理解这些评论有助于开发者提升 App 质量和用户对产品的满意度[4−7]. 然而, 人工浏览和逐条分析用户评论以收集评论中有用的信息是非常耗时的, 特别是对于那些每天都可能收到数百条评论的热门 App 来说代价更大.

近年来, 用于挖掘 App 评论中有用信息的自动化技术引起了广泛关注[8−10]. 研究人员定义了许多任务从不同角度通过多种方式来减轻理解和分析 App 评论所需的工作量, 其中比较有代表性的两类任务为主题发现任务[6,11,12]和关键短语提取任务[2−4,13,14]. 主题发现任务主要用于识别用户评论中涉及的主题/方面/分类[5,6,14], 例如兼容性、崩溃、GUI 等主题, 或赞美、缺陷、功能需求等分类. 以图 1 中来自 Snapchat 的一条 App 评论为例, 现有的方法会捕捉到“download”“delete”和“crash”等主题词, 或将该评论分类为缺陷. 这种粒度的信息不足以帮助开发人员准确了解用户在评论中描述的有缺陷的功能或 App 出现的异常行为. 另一方面, 现有的关键短语提取任务主要利用基于启发式的技术(如词性模板、语法解析树和语义依赖图)来提取目标短语, 这类技术无法充分地理解评论的语义, 因而其性能不能令人满意.

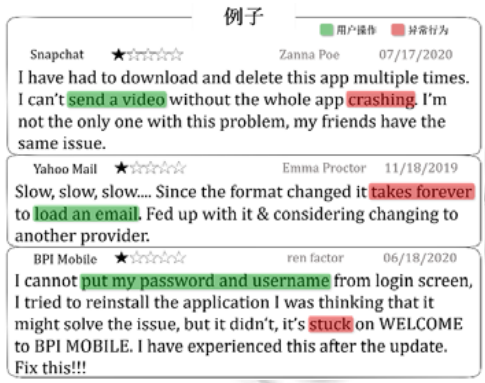


图 1　App 评论中的用户操作及对应的 App 异常行为

相比之下, 本文的目标是提取 App 评论中细粒度的信息, 这些信息通常以短语的形式存在于用户评论之中[15,16], 我们主要关注用户操作(user actions)和异常行为(abnormal behaviors)两类短语. 用户操作大多为评论中用户认为存在缺陷的功能相关的短语, 也包括用户期望增加/删除的功能相关的短语; 而异常行为主要针对 App 对于某个用户操作的反馈, 能够在一定程度上表征缺陷类型, 例如加载时间长暗示性能缺陷, App 崩溃暗示可靠性缺陷等. 挖掘和分析用户操作和异常行为以及两者之间的关联关系, 可以帮助开发者了解哪些用户操作会导致 App 哪些异常行为. 我们以图 1 中的 App 评论为例, 用户在评论中分别描述了在使用某些功能时所遇到的 App 异常行为, 例如: 在发送视频时 App 崩溃; 加载邮件时间过长; 输入用户名密码后在欢迎界面卡住不动等. 其中的“send a video”“load an email”和“put my password and username”等即为用户操作, 而“crashing”“take forever”和“stuck”等即为异常行为. 这种细粒度的信息能够帮助 App 开发人员了解用户关注的问题, 极大地节省了开发人员理解 App 评论的工作量, 有利于快速定位和复现有缺陷的模块, 并进行后续的

缺陷修复. 与此同时想要准确提取这类细粒度信息, 需要模型具备更强大的语义理解能力, 现有的关键短语提取方法在该场景下性能不佳.

为了克服现有研究中信息提取粒度粗, 语义理解能力不足等缺点, 本文提出了一种语义感知的细粒度 App 评论缺陷挖掘方法 Arab (app review user action and abnormal behavior miner). 该方法可以分别提取用户操作和 App 的异常行为, 并通过聚类和可视化挖掘两者之间的关联关系. 更具体地说, 我们设计了一种新颖的神经网络模型, 可以自动提取细粒度的目标短语(即用户操作和异常行为). 该模型结合了评论文本描述和额外的评论属性(如 App 类别和评论描述情感), 能更好地建模评论的语义. 之后, 利用提取的短语, 我们设计了一种基于图的聚类方法分别对用户操作和异常行为进行聚类和关联关系挖掘. 最后, Arab 对聚类后的两类短语和两者之间的关联关系进行可视化处理.

我们使用 3 426 条经过人工标注的 App 评论对 Arab 短语提取的性能进行了评估, 这些评论数据涉及 3 个类别的 6 个 App, 总共包含 8 788 个文本句子. Arab 短语提取的总体性能准确率、召回率和 $F1$ 分别达到了 81.78%、82.99%和 82.37%, 显著优于现有的基准模型. 在聚类评估上, Arab 在评估指标 ARI 和 NMI 上分别达到了 0.64 和 0.91, 同样优于聚类的基准模型. 我们进一步使用 Arab 在 15 个热门 App 的 301 415 条评论(评论的时间跨度为 10 个月)上进行了案例研究, 以探索 Arab 潜在的应用, 并检验其在大规模数据上的实用性. 通过对用户操作和异常行为之间的关联关系进行可视化分析, 我们可以观察到哪些用户操作和异常行为具有强关联关系. 这些观察有助于 App 测试, 例如根据异常行为的严重性或用户的反馈程度来安排测试优先级. 本文所涉及的源代码和实验数据发布在: https://github.com/MeloFancy/Arab.

本文第 1 节介绍与本文相关的方法和研究现状. 第 2 节介绍本文使用到的相关技术的基础知识, 包括命名实体识别技术和语言模型预训练技术. 第 3 节介绍本文构建的语义感知的细粒度 App 评论缺陷挖掘方法. 第 4 节通过对比实验验证所提模型的有效性. 最后总结全文.

1 相关工作

1.1 用户评论挖掘

Harman 等人[8,17]通过识别 App 用户评论的评分与 App 下载排名之间的相关性, 引入了应用商店挖掘的概念. Palomba 等人[10]发现如果开发者能够满足用户在评论中提及的功能需求或修复相关的缺陷, App 评级将显著上升. Noei 等人[9]对 App 排名的变动进行了研究, 并确定了与排名密切相关的一些变量, 如发行版本数量.

现有的挖掘用户评论的研究[3,6,11,12]强调对评论进行主题发现/分类和总结, 通过自动化工具来减少开发者人工分析大量文本所带来的工作量. 研究人员通常会从不同的角度来定义分类或主题的类别, 例如, 评论是否包含缺陷信息、是否包含对新特性的需求[12,18]、是否包含有意义的信息[11]、与软件维护和演变相关的分类[6], 等等. 还有一些研究[1,4,5,13,14]专注于从 App 评论中进行信息提取, 通过精炼评论中的关键信息, 减少冗杂琐碎的信息来帮助开发者更有效率地理解用户评论. 提取的信息包括: 不同类型的投诉[5], 用户喜欢 App 的方面[4], 指导发行计划的总结[19]等.

现有的几种主题发现的方法[3,4,13,14,20]得到的主题比我们提出的方法粒度更粗. 例如, 基于谷歌 Play 商店上的 95 个 App, MARK[14]只能发现诸如 crash、compatibility 和 connection 等主题; 而 PUMA[20]能够发现诸如 battery consumption 等主题; 类似地, SUR-Miner[4]生成诸如 predictions、auto-correct 和 words 等主题; SURF[3]可以发现 GUI、App 和 company 等主题; INFAR[13]可以生成 update、radar 和 download 等主题. 这些被发现的主题可以帮助开发者对 App 存在的缺陷有一个大致的了解, 但却无法了解具体哪些特定的功能出现了问题. 相比之下, 我们提出的方法可以发现用户认为有缺陷的功能(如“senda video”)以及所导致的 App 异常行为(如“crash”), 这有助于开发人员更深入、更准确地理解用户关注的功能和 App 的异常. 另一方面, 现有工作在提取目标短语时普遍采用基于模式的方法, 没有充分考虑评论的语义, 对噪声的容忍度较差, 而我们提出的方法是一种语义感知的方法, 可扩展性更强.

1.2 开源缺陷报告挖掘

现有的研究提出了多种方法理解缺陷报告, 例如, 重复报告检测[21,22], 缺陷报告总结[23], 对报告进行分类[24,25]等. 开源或众测环境下的缺陷报告通常是由具有一定软件测试经验的从业者提交的, 并且通常会给出详细的缺陷说明, 篇幅相对较长. 相比之下, App 评论是由终端用户提交的, 文本较短, 因此上述方法在这种情况下并不适用.

1.3 软件工程中的语义感知方法

近年来越来越多的研究人员利用深度学习技术来理解软件制品的语义, 并服务于后续的软件工程任务. 这类研究包括使用注意力编码器-解码器模型对代码片段进行代码摘要[26], 利用需求制品语义和领域知识解决需求跟踪任务[27], 针对社交平台上非正式讨论的知识挖掘[28]等. 本文关注的软件制品是 App 用户评论, 并使用 BERT 等深度学习技术感知评论文本的语义, 评估结果证明了我们提出的方法的有效性.

2 基础知识

本文所提方法主要基于命名实体识别技术和语言模型预训练技术, 下面介绍相关概念和基本知识.

2.1 命名实体识别

命名实体识别(named entity recognition, NER)是自然语言处理(natural language processing, NLP)中一类经典的序列标注任务(sequence tagging)[29,30]. 其定义为: 给定一个单词序列, NER 的目标是预测每个单词是否属于一类命名实体, 例如人名、组织名、地名等. NER 任务可以通过线性统计模型来解决, 例如 Maximum Entropy Markov 模型[31,32], Hidden Markov 模型[33]和条件随机场模型(conditional random fields, CRF)[34]. 而用于解决该任务的基于深度学习的技术通常使用一类深度神经网络捕获句子语义, 并使用 CRF 层学习句子级的标签规则. 典型的神经网络结构包括卷积神经网络结合 CRF (Conv-CRF)[35]、长短期记忆网络结合 CRF (LSTM-CRF)和双向长短期记忆网络结合 CRF (BiLSTM-CRF)[29]. 其中, BiLSTM-CRF 模型受益于其双向的网络结构可以同时利用过去和未来的输入信息, 因而通常可以获得比 Conv-CRF 和 LSTM-CRF 更好的性能.

2.2 语言模型预训练

语言模型预训练技术已被证明可以有效地提高许多 NLP 任务的性能[36,37]. BERT (bidirectional encoder representation from transformers)[36]是一个基于 Transformer[38]的表示模型, 它首先使用预训练技术在原始语料库中进行学习, 然后针对不同的下游任务(如 NER 任务)进行微调, 例如使用 BERT 替代 BiLSTM (BERT-CRF)可以进一步提高解决 NER 任务的性能[39]. BERT-CRF 结合微调技术, 能够受益于在大型通用语料库上学习的预训练表示, 使得模型性能得到进一步提升.

3 语义感知的细粒度 App 评论缺陷挖掘方法

为了克服现有 App 评论挖掘技术的缺点并更好地利用 App 评论的语义, 本文提出的语义感知的细粒度 App 评论缺陷挖掘方法 Arab, 能够提取用户在 App 评论中描述的存在缺陷的功能(即用户操作)以及用户在使用这些功能时所导致的 App 异常行为(即异常行为), 并挖掘两者之间的关联关系. 该方法的整体架构如图 2 所示, 其组成主要包含 3 部分.

(1) 数据预处理: 从线上应用商店中爬取 App 用户评论数据, 之后进行文本数据清洗以及获取评论属性(即 App 类别c和评论描述的情感倾向s).

(2) 用户操作和异常行为提取: 构建并训练了一个基于 BERT 的神经网络模型, 以自动提取 App 评论中与用户操作和异常行为相关的短语. 该模型使用数据预处理中获取的评论文本和两个评论属性(即c和s)作为输入, 能更好地对评论的语义进行建模, 提升了传统 BERT-CRF 模型短语提取的性能.

(3) 用户操作和异常行为关系挖掘: 设计了一个基于图的聚类方法, 能够根据短语之间的语义关系分别

对提取的用户操作和异常行为进行聚类. 使用聚类后的结果将⟨用户操作, 异常行为⟩对转换为各自的聚类标签$\langle A_i,B_j\rangle$, 以挖掘两者之间的关联关系.

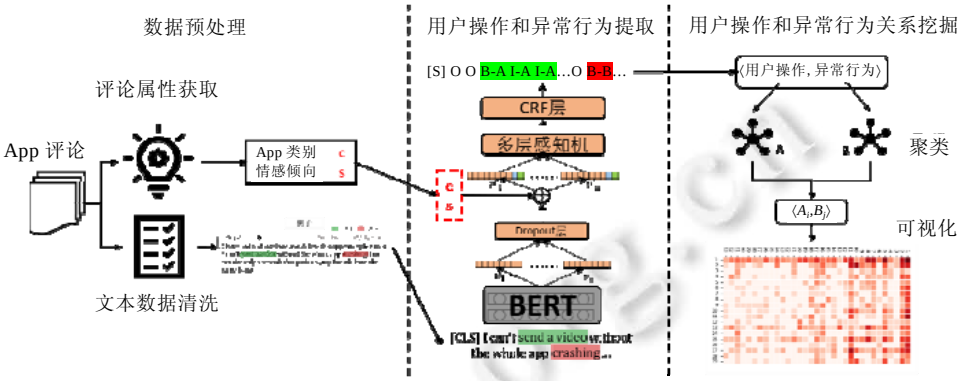


图 2 方法架构图

3.1 数据预处理

数据预处理主要包含两步: 文本数据清洗和评论属性获取, 下面分别对这两步进行详细介绍.

3.1.1 文本数据清洗

未经处理的 App 评论通常是由用户通过移动设备提交的, 这些提交的文本由于受限于手机键盘, 通常包含大量的噪声词, 如重复词、拼错词、缩写语等[4,14,20,40], 因此我们需要对文本数据进行清洗. 基于其他一些 CRF 方法的实践[29], 本方法将评论文本的每个句子作为输入单元. 具体做法为首先通过正则表达式匹配标点符号对每条评论进行分句, 然后利用工具 Langid (https://github.com/saffsd/langid.py)过滤掉所有非英文句子, 对于保留下来的句子采用以下步骤清洗数据来解决噪声词问题.

- 小写化: 将评论文本中的所有单词转换为小写.
- 词根化: 使用 Spacy (https://spacy.io)进行词根化, 以减轻单词的词法变换带来的影响.
- 拼写纠正: 使用 Ekphras (https://github.com/cbaziotis/ekphrasis)中包含的常用拼写纠正工具来纠正文本中出现的拼写错误.
- 格式化: 用特殊符号“⟨number⟩”替换所有的数字, 此外, 我们构建了一个从谷歌 Play 商店抓取的所有 App 名称的列表, 并将它们替换为统一的特殊符号“⟨appname⟩”, 以帮助 BERT 模型简化理解.

3.1.2 评论属性获取

一些与评论和 App 类别相关的属性有助于提取用户操作和异常行为, 我们在数据预处理阶段对这些属性进行获取. 本方法使用了两个属性, 即 App 类别c和评论描述情感s. 将 App 类别作为一个属性, 是因为不同类别下的 App 评论所涉及的功能和主题存在差异[41]. 而对于评论描述的情感倾向, 我们认为与积极情绪的评论描述相比, 用户更有可能在消极情绪的评论中提及用户操作和操作后所导致的异常行为. 因此, 我们将其作为第 2 个评论属性.

对于 App 类别, 我们可以直接从谷歌 Play 商店中检索并获取它们, 而对于情感倾向, 使用工具 SentiStrength-SE[42]来获取每个评论句子的情感倾向. SentiStrength-SE 是一个专门用来确定软件工程领域中短文本的积极和消极情感强度的工具. 具体地, SentiStrength-SE 将为每个句子计算一个范围为 1(不积极)到 5(非常积极)的正整数分数, 以及一个范围为–1(不消极)到–5(非常消极)的负整数分数, 来分别表征积极情感和消极情感的强度. 对同一个句子采用两个分数是因为有心理学研究表明, 人类是并行处理积极情绪和消极情绪的. 根据这两个得分, 我们使用以下方式来计算某个句子的情感分数[41]: 如果消极情感得分乘以 1.5 的绝对值大于积极情感得分, 该句子的情感倾向分数为消极情感得分; 否则, 该句子的情感倾向分数为积极情感得分.

3.2 用户操作和异常行为提取

本文将用户操作和异常行为提取任务建模为命名实体识别(NER)任务，即将用户操作和异常行为分别视作两类命名实体，并设计了一个短语提取的神经网络模型来解决该问题. 为了更好地捕获 App 评论的语义，我们使用 BERT 模型对评论的文本描述进行编码，之后使用 CRF 模型来识别两种类型的短语. 特别地，我们在 CRF 模型中引入了额外的评论属性(即 App 类别c和评论描述的情感倾向s)以进一步提高两类短语识别的准确性. 参照其他 NER 任务的设置，我们使用 BIO 标签格式[43,44]标记每个评论句子，其中，

- B 标签(beginning): 表示该单词是目标短语的开始.
- I 标签(inside): 表示该单词在目标短语中，但不是短语的开头.
- O 标签(outside): 表示该单词在目标短语之外.

被 BIO 标记的评论语句将被输入到短语提取模型中进行进一步处理.

图 3 展示了本方法提出的短语提取模型的详细结构. 由于 App 评论大多是短文本，涉及的词汇量相对较少，所以我们使用规模较小的预训练模型 BERTBASE (https://huggingface.co/bert-base-uncased)对评论文本进行编码，并使用微调技术对其进行再训练. 该模型的超参数配置为: 12 层神经网络, 768 个隐藏维度和 12 个注意力头. 输入的每个句子以特殊起始符号“[*CLS*]”作为开头，包含了 128 个的单词符号，对于那些不够长的句子，使用一个特殊符号“[*PAD*]”将句子符号序列的长度填充到 128. 输入的句子通过 BERT 后，我们能够得到 n 个(即输入句子序列的长度)向量，每个向量(记为 v)有 768 个维度，正好对应输入句子中的每个单词. 随后，BERT 的输出经过一个 Dropout 层，以避免模型训练时出现过拟合. 之后，我们将评论属性合并到文本向量(v)中，以提升捕获评论句子隐含语义的能力. 由于之前提取的评论属性(c 和 s)是离散值，我们首先将它们输入到一个嵌入层，使其转换为连续向量(记为 h_c 和 h_s). 以属性 s 为例，s 可以取 10 个值(–5 到–1,1 到 5)，嵌入层可以将每个离散值用连续向量表示，并且这些连续向量可以与整个模型的其他参数联合训练. 之后，我们连接 h_c、h_s 和 v($h_c \oplus h_s \oplus v$)以获得每个输入单词的向量，连接后的向量再被输入到一个多层感知机(multi-layer perceptron, MLP)中进行处理. MLP 会为每个单词计算 BIO 标签的概率向量(记为 p):

$$p = f(W[h_c; h_s; v]) \tag{1}$$

其中，$f(\cdot)$是激活函数，W 代表 MLP 中的训练参数，$[h_c;h_s;v]$代表这 3 个向量的连接. 最后，将 p 输入 CRF 层，根据 Viterbi 算法(https://en.wikipedia.org/wiki/Viterbi_algorithm)就能确定输入序列最有可能的标签序列. 基于预测输出的标签序列，我们可以获得目标短语(即用户操作或异常行为). 例如，如果我们输入的评论句子是“I can’t send a video without the whole app crashing”，而输出的标签序列是“⟨O⟩⟨O⟩⟨B-A⟩⟨I-A⟩⟨I-A⟩⟨O⟩⟨O⟩⟨O⟩⟨O⟩⟨B-B⟩”，其中带有“-A”的标签(即 B-A 和 I-A)表示该标签标记的是用户操作，而带有“-B”的标签(即 B-B)表示该标签标记的是异常行为. 根据 BIO 格式的定义，就可以确定从该评论中提取的用户操作是“send a video”，而提取的异常行为是“crashing”.

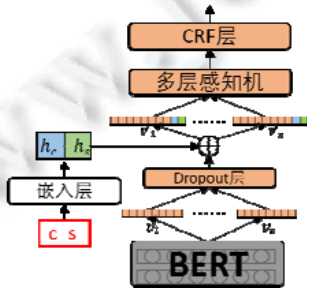


图 3 短语提取模型结构

模型的损失函数应该度量整个输入句子的真实标签序列的可能性，而不是序列中每个单词的真实标签的可能性，因此常用的交叉熵损失不适合这种情况. 本方法使用的损失函数包括两部分: 发射分数(emission

score)和转移分数(transition score), 其计算方法为

$$s([x]_1^T,[l]_1^T,\tilde{\theta})=\sum_{t=1}^{T}([A]_{[l]_{t-1},[l]_t}+[f_\theta]_{[l]_t,t}) \tag{2}$$

其中, $[x]_1^T$ 代表长度为 T 的句子序列, 而 $[l]_1^T$ 代表该句子的标签序列. $f_\theta([x]_1^T)$ 计算的是发射分数, 它也是 MLP 在参数θ下的输出. 而 $[A]_{i,j}$ 计算的是转移分数, 由 CRF 层中的参数计算得来, 建模的是 CRF 层中第 i 个状态到第 j 个状态的转移. $\tilde{\theta}=\theta\cup\{[A]_{i,j}\ \forall i,j\}$ 是整个网络的参数. 一个句子序列 $[x]_1^T$ 与对应的标签序列 $[l]_1^T$ 的损失为发射分数和转移分数之和.

在训练该模型时, 我们使用贪婪策略来调整模型中的超参数以获得最佳性能. 具体做法为给定一个超参数 P 及其候选值$\{v_1,v_2,...,v_n\}$, 我们执行 n 次迭代对每个候选参数进行自动调优, 并选择能够获取最佳性能的值作为 P 的最优取值. 经过超参数的调节, 模型的学习率被设置为 10^{-4}, 优化算法选择为 Adam 算法[45]. 同时, 为了加速训练过程, 我们使用了 mini-batch 技术, 其中的 batch size 被设置为 32. Dropout 层中的 drop rate 被设置为 0.1, 这意味着神经网络中 10%的神经元将被随机屏蔽以避免过度拟合. 我们使用了一个基于 Pytorch 的开源库 Transformers (https://github.com/huggingface/transformers)来实现该模型.

3.3 用户操作和异常行为关系挖掘

通过第 3.2 节中的短语提取模型, 我们能够逐句提取评论句子中的用户操作和异常行为, 并形成⟨用户操作,异常行为⟩对. 我们能够采用这种方式构建短语对, 是因为我们观察到用户在反馈使用 App 某个功能所遇到的问题时, 通常会在同一句子中描述自己的操作以及该操作所导致的 App 异常行为. 该短语对展示了两者之间的关联关系, 例如, 某一类用户操作最可能导致哪类异常行为, 或者某一类异常行为最可能由哪类用户操作所引发. 对于评论中只涉及一类短语的情况, Arab 只会提取相应类别的短语, 并且在挖掘关联关系时只分析能构成短语对的数据. 由于多个短语在表达上可能不同但在语义上相似, 我们需要合并具有相似语义的短语, 为此我们设计了一个基于图的聚类方法对提取的用户操作和异常行为分别进行聚类. 之后使用热力图对聚类后的用户操作、异常行为以及两者之间的关联关系进行可视化.

3.3.1 聚 类

传统的主题模型使用的是基于单词共现模式的统计技术(例如 Gibbs 采样)[46], 但这种技术并不适合短文本(例如本方法场景下的用户操作和异常行为短语), 因为主题模型很难从短文本中捕获单词共现模式. 此外, 这些主题模型需要指定聚类/主题的数量, 但该参数在本方法的场景中很难提前确定. 为了应对这些挑战, 我们设计了一种基于图的聚类方法, 该方法利用了短语之间的语义关系, 能够准确地对用户操作和异常行为进行聚类.

聚类的具体流程如图 4 所示, 其主要步骤为:

(1) 向量化: 使用通用句子编码器(universal sentence encode, USE)[47]将提取的用户操作短语或异常行为短语转换为 512 维的向量. USE 是一种基于 Transformer[38]的句子嵌入模型, 可以捕获句子丰富的语义信息, 并且已在很多 NLP 任务上被证明比传统使用的词嵌入(word embedding)模型[1]更有效.

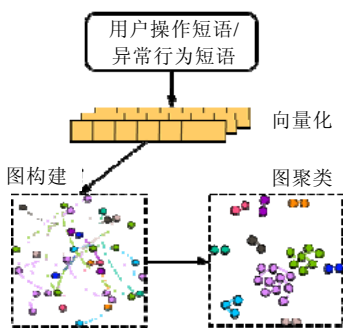


图 4 聚类流程

(2) 图构建: 构建一个加权的无向图, 其中每个短语作为图的一个节点, 两个短语的 USE 向量之间的余弦相似度得分作为两节点之间的权重, 如果相似度得分超过某个阈值, 则在两个节点之间添加一条边, 否则不添加边. 该阈值是一个需要事先给定的超参数, 它衡量了短语之间的语义相似性, 较高的阈值会导致聚类的集群具有较高的内聚度(cohesion).

(3) 图聚类: 在上述构建的无向图上执行算法 Chinese Whispers (CW)[48], 以对目标短语进行聚类. 其中的

CW 是一种高效的图聚类算法.

我们设计的这种基于图的聚类方法可以将语义相似的用户操作短语或异常行为短语聚类到同一主题中. 我们基于 USE (https://github.com/MartinoMensio/spacy-universal-sentence-encoder)和 CW (https://github.com/nlpub/chinese-whispers-python)的开源实现, 使用 python 实现了上述聚类方法.

3.3.2 可视化

为了可视化用户操作和异常行为两者之间的关联关系, 我们采取以下步骤.

(1) 使用第 3.3.1 节中的聚类算法, 分别对提取的用户操作短语和异常行为短语进行聚类. 经过超参数的调节, 两个聚类的阈值均被设置为 0.6.

(2) 使用步骤(1)中的聚类结果分别将⟨用户操作,异常行为⟩对中的用户操作和异常行为转换为对应短语所属的聚类标签$\langle A_i,B_j\rangle$, 即表示第 i 类的用户操作(A_i)会导致第 j 类的异常行为(B_j).

(3) 统计所有$\langle A_i,B_j\rangle$对的数量, 使用热力图对两者的关系进行可视化. 热力图中第 i 行第 j 列的色块代表$\langle A_i,B_j\rangle$的数量, 颜色越深表示两者关系越密切.

(4) 特殊地, 热力图中的第 0 列表示用户操作短语聚类后的分布, 第 0 行表示异常行为短语聚类后的分布.

通过可视化, 开发者能够清晰地理解用户认为有缺陷的功能有哪些, 这些缺陷会导致 App 的哪些异常行为, 以及哪类用户操作与哪类异常行为关系密切, 从而指导 App 新功能的开发、缺陷的修复和复现等活动.

4 实验分析

4.1 实验数据

我们在实验中使用了 3 个数据集 D_1、D_2 和 D_3, 其中, D_1 为人工标注的数据集, 用于评估 Arab 短语提取的性能; D_2 为人工标注的聚类数据集, 用于评估 Arab 聚类的性能; D_3 为未标注的数据集, 用于探究本方法在大规模数据集上的应用情况. D_1 和 D_3 两个数据集均是通过工具 Google-Play-scraper (https://github.com/facundoolano/google-play-scraper)爬取谷歌 Play 商店中的 App 评论数据构建的.

表 1 详细描述了标注数据集 D_1 的统计情况. D_1 包含了 3 个类别的 6 款 App (每个类别 2 个)的评论数据, 提交时间在 2019 年 8 月–2020 年 1 月之间. 我们从每一款 App 中随机抽取大约 550 条评论(约 1 500 个句子), 总计抽取了 3 426 条评论和 8 788 个句子来构建数据集 D_1. 3 位标注者对这些评论进行手动标注, 以获取验证本方法性能的真实标签. 为了保证标注结果的准确性, 两位标注者分别对同一个 App 的评论进行独立标注, 即在每个评论句子中标记出目标短语(用户操作和异常行为)的开始和结束位置. 然后第 3 位标注者比较两份标注结果的差异, 并组织 3 人进行讨论以确定最终的标注结果. 这 6 款 App 都遵循相同的标注流程, 直至标注完成. 对第 1 个标注的 App (Instagram), 两份标注结果的 Cohen's Kappa 为 0.78, 而对于最后一个标注的 App (chase mobile), Cohen's Kappa 是 0.86, 说明两位独立标注的标注者的标注准测在标注过程中逐渐趋于一致. 遵循这样的标注流程, 最终每一个标注结果都达成了共识. 我们总计标注出了 3 090 个用户操作以及 1 280 个异常行为.

表 1 标注数据集 D_1

App 类别	App 名	评论数量	句子数量	用户操作数量	异常行为数量
社交	Instagram	582	1 402	481	194
	Snapchat	585	1 388	427	175
通信	Gmail	586	1 525	663	232
	Yahoo Mail	542	1 511	460	215
金融	BPI Mobile	588	579	579	256
	Chase Mobile	543	480	480	208
汇总		3 426	8 788	3 090	1 280

数据集 D_2 是人工构建的一个真实聚类数据集. 更具体地说, 我们从数据集 D_1 提取的目标短语中分别为每个 App 随机采样 100 个用户操作和异常行为(总共各 600 个), 人工标注的标准是将指代同一用户操作或异常行为的短语聚类到一起. 在第 1 轮标注中, 两位标注者分别将用户操作/异常行为根据语义进行聚类, 两份

标注结果之间的 Cohen's Kappa 达到了 0.83(即令人满意的一致程度), 说明两位标注者在标注标准上较为统一. 对于标注不一致的结果, 两位标注者进行后续讨论直到达成共识. 最终 600 个用户操作被聚类到 37 个分组中, 而 600 个异常行为被聚类到 26 个分组中. 需要注意的是, 我们在标注前没有预先指定聚类的数量, 因为在我们的场景下该值很难提前确定, 而且我们提出的聚类方法也不需要指定这个参数.

表 2 详细描述了未标注数据集 D_3 的统计情况. D_3 包含了 2020 年 2 月–12 月期间提交的 3 个类别(每个类别 5 个)的 15 个 App 的评论数据, 总计包含 301 415 条评论和 675 034 个句子.

表 2 未标注数据集 D_3

App 类别	App 名	评论数量	句子数量
社交	Facebook	64 559	147 156
	Instagram	63 124	153 852
	TikTok	61 178	104 094
	Snapchat	18 268	41 278
	Twitter	15 583	36 386
通信	Facebook-Messenger	27 121	59 303
	Gmail	9 655	24 520
	Telegram	7 704	17 672
	Yahoo Mail	7 090	20 124
	Skype	3 266	8 139
金融	Paytm	18 316	47 836
	Chase Mobile	3 732	9 952
	BPI Mobile	1 375	3 638
	BCA Mobile	386	960
	WavePay	58	124
汇总		301 415	675 034

4.2 评价指标

评价指标分为两部分, 一部分是评价目标短语提取的指标, 另一部分是评价聚类的指标.

4.2.1 短语提取评价指标

我们使用准确率(precision)、召回率(recall)和 $F1$($F1$-score)这几个常用指标来评估 Arab 在目标短语(即用户操作或异常行为)提取方面的性能. 如果 Arab 从评论句子中提取的短语与实际的短语相同, 我们就认为该短语被正确提取. 3 个度量的计算方法分别为:

- 准确率是正确提取的短语数与提取短语的总数的比值.
- 召回率是正确提取的短语数与实际短语的总数的比值.
- $F1$ 是准确率和召回率的调和平均值.

4.2.2 聚类评价指标

参照之前工作[49]中对聚类评价的方法, 我们通过比较实际的和预测的聚类结果, 使用常用的 adjusted rand index (ARI)和 normalized mutual information (NMI)两个聚类指标来评估聚类性能, 这两个指标的度量值越高表示聚类性能越好. 我们使用 G 表示实际聚类结果, C 表示预测聚类结果.

- adjusted rand index (ARI): ARI 取值为[–1,1], 反映了两个聚类之间的重叠程度. 原始的 rand index (RI)的计算方法为 $RI=(a+b)\Big/\binom{n}{2}$, 其中, a 是在 G 和 C 中均被分配到同一聚类中的配对数; b 是在 G 和 C 中被分配到不同聚类中的配对. $\binom{n}{2}$ 是由 n 个短语能够组合成的无序对的总数. 然后使用以下策略将原始的 ARI 分数"根据几率(chance)调整"为 ARI 分数:

$$ARI=\frac{RI-E(RI)}{\max(RI)-E(RI)} \tag{3}$$

其中, $E(RI)$是 RI 的期望值. 这样可以保证 ARI 的值在随机的聚类标签下接近于 0.0, 而不受聚类和样本数量的影响.

• normalized mutual information (NMI): NMI 反映了两组聚类结果的相似度, 取值为 0(无互信息)到 1(完全相关).

$$NMI(G,C)=\frac{MI(G,C)}{\sqrt{H(G)H(C)}} \tag{4}$$

其中, $H(G)=-\sum_{i=1}^{|G|}P(i)\log(P(i))$ 是集合 G 的熵, $P(i)=\frac{G_i}{N}$ 是随机选取的短语落在聚类 G_i 中的概率. $MI(G,C)$ 是 G 和 C 的互信息, 即 $MI(G,C)=\sum_{i=1}^{|G|}\sum_{j=1}^{|C|}P(i,j)\log\left(\frac{P(i,j)}{P(i)P(j)}\right)$.

4.3 基准模型

基准模型也分为评估短语提取性能的基准模型和聚类性能基准模型两部分.

4.3.1 短语提取基准模型

为了评估 Arab 短语提取的性能, 我们使用以下几个基准模型作为比较.

• KEFE[50]: 是一种最新的从 App 评论中识别关键功能的方法. 该工作中的关键功能是指与 App 评分高度相关的功能. KEFE 首先使用基于模式的过滤器来获取候选短语, 然后使用基于 BERT 的分类器来识别功能相关的短语. 由于其过滤器中的模式是为中文评论设计的, 所以我们使用 SAFE[2]中提出的模式作为替代以处理英文评论.

• Caspar[1]: 是一种从 App 评论中提取和生成用户报告的关于 App 问题的小故事(mini-story)的方法. 我们将其前两步(即事件提取和分类)作为一个基准模型. 该工作中的事件指的是一个短语, 它源于一个动词, 并包含与该动词相关的其他属性. 具体地, Caspar 首先采用模式和语法解析等 NLP 技术来提取短语或子句, 之后使用 USE 将提取的短语向量化, 最后使用支持向量机(support vector machine, SVM)分类器将其分类为 User Actions 或 AppProblems. 我们使用了原论文中提供的实现方法(https://hguo5.github.io/Caspar).

• BiLSTM-CRF[29]: 是序列标注任务(如 NER 任务)中一种常用的算法. 它是一种基于深度学习的技术, 利用 BiLSTM 捕获句子语义, 利用 CRF 层学习句子级标签.

4.3.2 聚类基准模型

为了评估 Arab 聚类的性能, 我们使用 scikit-learn (https://scikit-learn.org)库实现了以下两种常用的聚类方法作为比较, 并在调节超参数后使用最优的超参数以获得最佳性能.

• *K*-Means: 这是一种常用的聚类算法, 曾用于对 App 评论中的关键词进行聚类[14]. 在本文中, 我们首先使用 USE 对目标短语进行向量化, 然后运行 *K*-Means 算法将目标短语聚类到不同主题中.

• DBSCAN[51]: 这是一种基于密度的算法, 用于发现任意形状的聚类. 类似地, 我们用 USE 来编码目标短语, 然后运行 DBSCAN 来聚类所有目标短语.

4.4 实验方法

为了评估 Arab 提取用户操作和异常行为的性能, 我们在标注数据集 D_1 上进行嵌套交叉验证[52]. 嵌套的内循环用于选择最优的超参数, 而嵌套的外循环则应用最优的超参数配置来评估模型的性能. 具体地, 在外循环中, 我们将数据集 D_1 随机分成 10 折, 使用其中的 9 折进行训练, 剩余的 1 折来测试性能. 此过程重复 10 次, 并将 10 折的平均性能作为最终的性能. 在内循环中, 我们使用 8 折进行训练, 使用 1 折进行验证. 我们在相同的实验设置下运行 Arab 以及每个基准模型(见第 4.3.1 节)以保证各模型的性能不受数据划分的影响.

为了验证在 Arab 中引入的两个评论属性在短语提取性能提升上的贡献, 我们设计了 3 个 Arab 中短语提取模型的变体, 即 Arab-cs、Arab-s 和 Arab-c, 分别表示不使用评论属性的模型(即只使用文本描述)、不使用评论描述情感的模型(即包含文本描述和 App 类别)和不使用 App 类别的模型(即包含文本描述和评论描述情感). 实验数据同样为标注数据集 D_1, 实验评估过程同样采用嵌套交叉验证.

为了评估 Arab 聚类用户操作和异常行为的性能, 我们在数据集 D_2 上运行我们提出的聚类方法和每个基

准模型(见第 4.3.2 节)来聚类目标短语以获得每种方法的聚类结果, 通过比较实际和预测的聚类结果在两个聚类指标上评估性能.

为了探究用户操作和异常行为之间的关联关系, 我们在未标注数据集 D_3 上应用 Arab 方法, 并使用热力图分别展示了 3 个类别下 App 的结果.

本方法的实验环境是一台配备 NVIDIA GeForce RTX 2060 GPU、intel core i7 CPU 和 16 GB RAM, 运行在 Windows 10 上的台式计算机, 模型训练时每折嵌套交叉验证大约耗费 2.5 h.

4.5 实验结果与分析

为了评估提出的 App 评论缺陷挖掘方法 Arab 的有效性, 我们研究了以下 3 个问题.

- RQ1: Arab 在提取用户操作和异常行为时的性能如何?
- RQ2: Arab 中加入的两个评论属性是否对短语提取的性能提升有贡献?
- RQ3: Arab 在聚类用户操作和异常行为时的性能如何?
- RQ4: Arab 在大规模未标注数据集 D_3 上对 App 评论缺陷挖掘的效果如何?

RQ1: Arab 在提取用户操作和异常行为时的性能如何?

为了验证这个问题的结果, 我们将 Arab 模型与第 4.3.1 节中的基准模型进行了对比实验, 实验的结果见表 3. 表 3 展示了 Arab 和 3 个基准模型在提取用户操作和异常行为时各自的性能以及提取这两类目标短语的总体性能. Arab 在所有指标上都优于基准模型, 这表明了基于模式的基准模型(即 KEFE 和 Caspar)在提取目标短语方面不够有效, 而基于深度学习的基准模型(即 BiLSTM-CRF)由于语义捕获能力不如 BERT, 且忽略了评论属性, 因此性能比 Arab 略差.

表 3 Arab 与基准模型短语提取性能比较(%)

指标 / 方法	用户操作			异常行为			总体性能		
	P	*R*	*F*1	*P*	*R*	*F*1	*P*	*R*	*F*1
KEFE	38.91	58.34	46.68	20.04	37.17	26.04	33.53	52.15	40.82
Caspar	24.76	9.97	14.19	32.64	11.07	16.45	26.82	10.28	14.84
BiLSTM-CRF	81.82	72.47	76.78	75.76	52.56	61.92	80.27	66.66	72.79
Arab	**83.21**	**86.14**	**84.63**	**78.14**	**75.44**	**76.74**	**81.78**	**82.99**	**82.37**

具体地, KEFE 的准确率、召回率和 *F*1 分别为 33.53%、52.15%和 40.82%, 而 Caspar 的准确率、召回率和 *F*1 分别为 26.82%、10.28%和 14.84%. KEFE 的性能比 Caspar 略好, 是因为 KEFE 首先使用 SAFE 定义的 18 个词性模板, 可以检索大量潜在的目标短语, 之后使用的 BERT 分类器能够过滤掉大量的低质量短语. 相比之下, Caspar 只能从包含时间连词或关键短语(例如“when”“every time”)的评论中提取事件, 因此提取的潜在目标短语较少, 而之后使用的 SVM 分类器性能也远远不及 BERT, 因此 Caspar 的性能是所有模型中最差的. 总的来说, KEFE 和 Caspar 中的短语分类仍然是在基于模式的方法提取的候选短语的基础上进行的, 这在一定程度上限制了性能. BiLSTM-CRF 取得了第二好的性能, 准确率、召回率和 *F*1 分别为 80 27%、66.66%和 72.79%. 但由于缺少 BERT 和额外的评论属性, 其语义捕获能力欠佳, 因此性能不如我们提出的方法 Arab.

我们提出的方法 Arab 总体性能的准确率、召回率和 *F*1 分别为 81.78%、82.99%和 82.37%, 优于所有的基准模型, 从而证明了 Arab 在提取目标短语时的有效性. 同时我们可以观察到 Arab 提取用户操作的性能比提取异常行为的性能要好, 这是因为数据中用户操作的数量比异常行为的数量要多(见表 1), 从而导致对用户操作的训练更为充分. 即便如此, Arab 在分别提取这两类短语的性能上也是各方法中最好的. 表 4 展示了 Arab 在各 App 上的性能. 其中, Arab 在 Gmail 上的准确率最高达到了 88.43%, 在 Yahoo Mail 上的召回率最高达到了 86.16%. 在 Yahoo Mail 的准确率最低为 77.66%, 在 Chase Mobile 的召回率最低为 79.68%. 总的来说, Arab 在各 App 上均达到了一个可接受的准确率和召回率.

另外, 我们详细检查了两类目标短语的提取结果, 并且确实观察到一些与目标短语相关的模式. 例如, 用户习惯于在提及用户操作之前使用一些否定词(如“can’t”“hardly”)或时间连词(如“as soon as”“when”). 这也许可以解释为什么基于模式的技术(如 KEFE 和 Caspar)有时可以工作. 然而基于模式的技术高度依赖于手工定

义的模式, 并且在不同的数据集中可扩展性较差. 此外, 由于很难总结出较为通用的模式, 基于模式的方法在许多情况下很难工作, 例如很难针对评论句子“this update take away my ability to view transactions”总结出较为通用的模式. 这些情况进一步证明了我们方法的优点和灵活性. 我们还研究了一些 Arab 不工作的失败案例, 比较典型的案例是在某些情况下, Arab 可以提取目标短语的核心名词和动词, 但遗漏或额外提取了一些不重要的单词, 特别是核心短语前后的一些副词或状语. 例如, Arab 可能错误地从“I have not receive emails for 10 days”中提取“receive emails for 10 days”, 而实际的用户操作是“receive emails”, 这样的结果降低了 Arab 的性能.

表 4 Arab 各 App 上短语提取性能比较(%)

App	指标		
	Precision	Recall	*F*1
Instagram	81.06	81.74	81.34
Snapchat	79.28	84.61	81.65
Gmail	88.43	82.56	85.35
Yahoo Mail	77.66	86.16	81.60
BPI Mobile	84.75	83.99	84.23
Chase Mobile	78.66	79.68	79.02
总体性能	81.78	82.99	82.37

RQ2: Arab 中加入的两个评论属性是否对短语提取的性能提升有贡献?

为了验证这个问题的结果, 我们将 Arab 与 3 种短语提取模型的变体进行了对比实验. 表 5 分别展示了 Arab 及其 3 种变体的性能, 其中, Arab 的总体性能高于所有 3 种变体. 与基础的 Arab-cs 模型相比(即 Arab vs. Arab-cs), 添加 App 类别和评论描述情感属性显著提高了 *F*1(+8.51%), 这表明两个属性有助于识别目标短语. 同时可以观察到用户操作的 *F*1 提升(5.10%)远低于异常行为的 *F*1 提升(20.36%), 这说明这两个属性尤其能有效提升异常行为提取的性能.

表 5 消融实验-评论属性对性能的影响(%)

指标 / 方法	用户操作			异常行为			总体性能		
	P	*R*	*F*1	*P*	*R*	*F*1	*P*	*R*	*F*1
Arab-cs	83.49	77.80	80.52	73.47	56.64	63.76	80.90	71.60	75.91
Arab-s	83.06	84.31	83.64	73.24	72.61	72.85	80.23	80.87	80.51
Arab-c	**84.01**	84.20	84.06	73.31	71.51	72.31	80.94	80.47	80.67
Arab	83.21	**86.14**	**84.63**	**78.14**	**75.44**	**76.74**	**81.78**	**82.99**	**82.37**

在增加的两个评论属性中, 评论描述情感属性对 *F*1 性能的提升略高于 App 类别属性. 进一步地, 我们还观察到这两个属性的贡献在一定程度上是重叠的, 即每个属性增加的性能并不是简单地加到整个模型的性能上. 这一现象是可解释的, 因为一些表达用户情感的词在文本描述中进行语义编码时, 可以同时被 BERT 模型捕获. 另外, 我们还可以观察到相较于对用户操作提取性能的提升, 增加单一属性同样是对异常行为提取性能的提升更大. 总的来说, 同时添加这两个属性所获得的总体性能是最高的, 这进一步说明了我们模型设计的必要性.

RQ3: Arab 在聚类用户操作和异常行为时的性能如何?

表 6 展示了 Arab 以及两个基准模型在目标短语聚类方面的性能, 我们可以观察到 Arab 的总体性能优于两个基准模型, 其中 ARI 和 NMI 分别达到 0.64 和 0.91, 高于 *K*-Means (0.38 和 0.80)和 DBSCAN (0.45 和 0.83).

表 6 Arab 与基准模型聚类性能比较

指标 / 方法	用户操作		异常行为		总体性能	
	ARI	NMI	ARI	NMI	ARI	NMI
K-Means	0.51	0.92	0.67	0.90	0.38	0.80
DBSCAN	0.53	0.93	0.74	0.92	0.45	0.83
Arab	**0.66**	**0.94**	**0.83**	**0.94**	**0.64**	**0.91**

此外, 我们注意到 Arab 对 ARI 的改进要大于对 NMI 的改进. 这是因为 ARI 是一种对感知的度量指标, 当

本应属于同一聚类的两个短语被错误地分配到不同聚类时, 或者当本应属于不同聚类的两个短语被错误地放置到同一聚类时, 这种度量十分敏感. ARI 的结果表明 Arab 可以有效地避免生成新的聚类或破坏原有的聚类. 而 NMI 是一种基于熵的度量指标, 基于信息熵理论, 主要关注两个聚类分布的差异. NMI 的结果表明, 由 Arab 得到的聚类分布(例如每个聚类中目标短语的数量)更接近实际的聚类分布, 但各方法差别不像 ARI 那么大. 另外, 我们观察到每中方法异常行为的 ARI 均比用户操作高, 而 NMI 差异不大, 这是因为异常行为短语更短, 聚类更少, 不同聚类容易区分. 但即使这样, Arab 在两者均取得最好的性能.

基准模型方法主要利用单词统计或共现关系来聚类目标短语, Arab 取得更好的聚类性能表明我们提出的基于图的聚类方法可以更好地理解短语之间的语义关系, 因而能够更有效地聚类两种类型的短语.

RQ4: Arab 在大规模未标注数据集 D3 上对 App 评论缺陷挖掘的效果如何?

在使用第 3.3.1 节中的聚类算法在数据集 D3 上分别对提取的用户操作和异常行为进行聚类后, 我们共获得 1 419 个用户操作的聚类和 460 个异常行为的聚类. 由于许多聚类中包含的短语数量较少, 我们过滤掉那些短语数量少于 500 的聚类, 最终我们保留了 33 个用户操作聚类以及 19 个异常行为聚类. 表 7 展示了用户操作和异常行为各聚类中 Top 3 词频的单词, 能够表征该聚类中短语的主要含义.结合聚类结果和表 7, 我们可以观察到用户认为存在缺陷的功能(即用户操作)主要集中在登录(A_1)、邮件发送(A_2)、视频通话(A_5)和 App 启动(A_{11})上, 而 App 的异常行为主要表现为功能不工作(B_1)、时间花销大(B_2)、无响应(B_{13})、App 退出(B_{15})和 App 崩溃(B_{17}). 该结果直观地展示了我们提出的聚类方法的有效性.

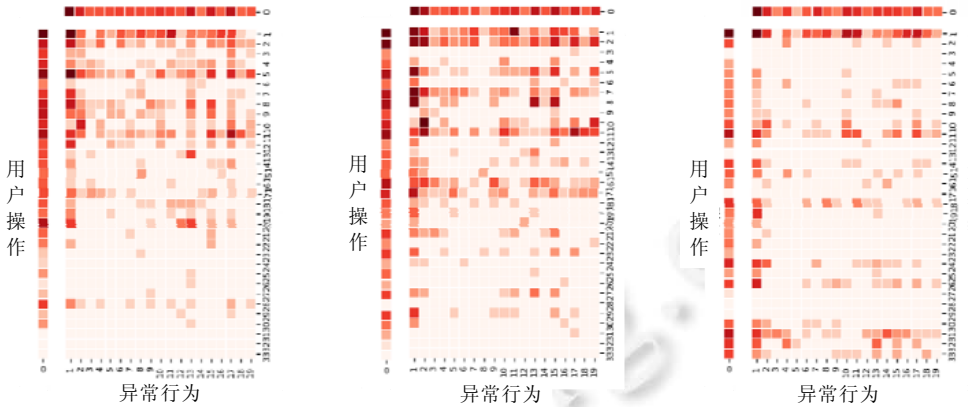
表 7 用户操作和异常行为聚类词频 Top 3

#	用户操作 A	#	用户操作 A
A_1	account, log, login	A_{20}	live, option, stream
A_2	send, email, mail	A_{21}	download, file, save
A_3	story, post, share	A_{22}	update, download, version
A_4	picture, photo, upload	A_{23}	call, make, incoming
A_5	video, call, watch	A_{24}	payment, pay, bill
A_6	verification, code, get	A_{25}	code, receive, activation
A_7	message, receive, see	A_{26}	fingerprint, login, finger
A_8	notification, get, receive	A_{27}	sync, email, contact
A_9	see, view, comment	A_{28}	tweet, retrieve, like
A_{10}	load, message, page	A_{29}	messenger, open, use
A_{11}	open, app, ⟨appname⟩	A_{30}	otp, receive, send
A_{12}	datum, count, data	A_{31}	money, transfer, wallet
A_{13}	dark, mode, night	A_{32}	transaction, do, history
A_{14}	click, button, skip	A_{33}	deposit, check, mobile
A_{15}	number, phone, change		
A_{16}	delete, email, spam		
A_{17}	chat, group, friend		
A_{18}	password, reset, change		
A_{19}	press, back, next		

#	异常行为 B
B_1	not, work, stop
B_2	take, time, forever
B_3	delete, cancel, remove
B_4	fail, mess, get
B_5	disappear, vanish, reappear
B_6	block, action, account
B_7	stick, get, stop
B_8	lock, account, close
B_9	go, back, come
B_{10}	freeze, freezing, frozen
B_{11}	error, say, message
B_{12}	available, not, unavailable
B_{13}	no, longer, happen
B_{14}	option, no, button
B_{15}	break, shut, pause
B_{16}	say, invalid, exist
B_{17}	crash, crashing, freeze
B_{18}	close, down, not
B_{19}	slow, down, fast

图 5 按照 App 类别(3 类)分别展示了各类 App 中用户操作和异常行为的分布, 以及两者之间关联关系的分布. 我们使用 A_i 表示第 i 个($i\in\{1,\ldots,33\}$)用户操作聚类, B_j 表示第 j 个($j\in\{1,\ldots,19\}$)异常行为聚类, 图 5 中第 i 行第 j 列色块颜色的深浅表示 A_i 对应 B_j 的短语对数量. 特别地, 第 0 列表示用户操作在 33 个聚类上的数量分布, 第 0 行表示异常行为在 19 个聚类上的数量分布, 颜色越深表示数量越多.

首先, 我们关注各类 App 中用户操作(即第 0 列)和异常行为(即第 0 行)在各自聚类中的分布, 可以观察到不同类 App 的用户操作在某些聚类上存在共性, 而在另外一些聚类上存在差异, 例如登录相关的缺陷(A_1)在这 3 类 App 中普遍存在, 而视频通话(A_5)相关的缺陷主要集中在社交类 App, 邮件发送(A_2)相关的缺陷主要集中在通信类 App, 转账(A_{31})相关的缺陷主要集中在金融类 App. 另一方面, 我们观察到不同类的 App 所涉及的异常行为分布差异不大, 用户反馈的主要故障包括功能不工作(B_1)、时间花销大(B_2)、弹出错误信息(B_{11})和崩溃(B_{17})等. 总的来说, 这些观察是符合常识和易于理解的, 一定程度上证明了 Arab 提取目标短语的准确性.



(a) 社交 (b) 通信 (c) 金融

图 5 不同类别 App 用户操作和异常行为关系可视化

其次，我们对用户操作和异常行为的关联关系进行分析，根据热力图(A_1~A_{33},B_1~B_{19})中色块的分布，我们可以观察到不同类别的 App 中关联关系普遍较强的为$\langle A_1,B_1\rangle$，即账户登录(A_1)工作不正常(B_1)在各类 App 中普遍存在. 另一方面，各类 App 中用户操作和异常行为的关联关系也存在差异. 例如，对于社交 App 来说，关联关系较强的为$\langle A_5,B_1\rangle$和$\langle A_{20},B_1\rangle$，即很多用户反馈视频通话($A_5$)和直播选项($A_{20}$)经常不工作($B_1$)；对于通信 App 来说，关联关系较强的包括$\langle A_2,B_1\rangle$、$\langle A_7,B_1\rangle$、$\langle A_8,B_1\rangle$以及$\langle A_{10},B_2\rangle$，即发送邮件($A_2$)、接收信息($A_7$)、接收通知($A_8$)等功能经常不工作($B_1$)，以及加载信息($A_{10}$)时间过长($B_2$)；而对于金融 App，关联关系较强的包括$\langle A_{24},B_1\rangle$、$\langle A_{26},B_1\rangle$和$\langle A_{31},B_1\rangle$，即支付账单($A_{24}$)、指纹登录($A_{26}$)和转账($A_{31}$)等功能不能正常工作($B_1$). Arab 除了能提供上述观察外，还存在其他潜在应用，例如开发者可以查询造成 App 某些特定异常行为的用户操作有哪些，从而帮助开发者进行缺陷定位. 例如我们可以观察到，造成 App 崩溃(B_{17})的用户操作主要包括账户登录(A_1)、发送邮件(A_2)、视频通话(A_5)和 App 启动(A_{11})等. 总之，这些细粒度信息可以帮助开发者更好地理解用户在评论中反馈的功能需求或缺陷，有助于安排开发和测试优先级，从而提升产品质量.

5 总 结

为了帮助 App 开发者更有效地理解用户认为 App 中哪些特定的功能存在缺陷，以及该缺陷会导致 App 什么样的异常行为，本文提出了一种语义感知的细粒度 App 评论缺陷挖掘方法 Arab. 该方法可以分别提取用户操作和异常行为，并通过聚类和可视化挖掘两者之间的关联关系. 我们设计了一种新颖的神经网络模型来提取细粒度的目标短语，并设计了一种基于图的聚类方法对两类短语进行聚类，最后对两者之间的关联关系进行可视化. 我们使用来自 6 个 App 的 3 426 条评论对 Arab 短语提取的性能进行了评估，结果证实了所提出方法的有效性. 我们进一步对 15 个热门 App 的 301 415 条评论进行了案例研究，以探索 Arab 的潜在应用和在大规模数据上的实用性.

References:

[1] Guo H, Singh MP. Caspar: Extracting and synthesizing user stories of problems from app reviews. In: Proc. of the 42nd Int'l Conf. on Software Engineering (ICSE 2020). 2020. 628–640.

[2] Johann T, Stanik C, Alizadeh BAM, *et al*. SAFE: A simple approach for feature extraction from App descriptions and App reviews. In: Proc. of the 25th IEEE Int'l Requirements Engineering Conf., RE 2017. 2017. 21–30.

[3] Di Sorbo A, Panichella S, Alexandru CV, *et al*. What would users change in my App? Summarizing App reviews for recommending software changes. In: Proc. of the 24th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering, FSE 2016. 2016. 499–510.

[4] Gu XD, Kim S. What parts of your Apps are loved by users? In: Proc. of the 30th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE 2015). 2015. 760–770.

[5] Khalid H, Shihab E, Nagappan M, *et al*. What do mobile App users complain about? IEEE Software, 2015, 32(3): 70–77.

[6] Panichella S, Di Sorbo A, Guzman E, *et al*. How can I improve my App? Classifying user reviews for software maintenance and evolution. In: Proc. of the 2015 IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME 2015). 2015. 281–290.

[7] Lu Z, Yang D, Li J. A software evaluation system based on reviews mining. Computer Applications and Software, 2014, 31(7): 1–4 (in Chinese with English abstract).

[8] Harman M, Jia Y, Zhang YY. App store mining and analysis: MSR for app stores. In: Proc. of the 9th IEEE Working Conf. of Mining Software Repositories (MSR 2012). 2012. 108–111.

[9] Noei E, da Cost DA, Zou Y. Winning the App production rally. In: Proc. of the 2018 ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering (ESEC/SIGSOFT FSE 2018). 2018. 283–294.

[10] Palomba F, Vásquez ML, Bavota G, *et al*. User reviews matter! Tracking crowdsourced reviews to support evolution of successful Apps. In: Proc. of the 2015 IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME 2015). 2015. 291–300.

[11] Chen N, Lin JL, Hoi SCH, *et al*. AR-miner: mining informative reviews for developers from mobile App marketplace. In: Proc. of the 36th Int'l Conf. on Software Engineering (ICSE 2014). 2014. 767–778.

[12] MaalejW, Nabil H. Bug report, feature request, or simply praise? On automatically classifying App reviews. In: Proc. of the 23rd IEEE Int'l Requirements Engineering Conf. (RE 2015). 2015. 116–125.

[13] Gao CY, Zeng JC, Lo D, *et al*. INFAR: insight extraction from App reviews. In: Proc. of the 2018 ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering (ESEC/SIGSOFT FSE 2018). 2018. 904–907.

[14] Vu PM, Nguyen TT, Pham HV, *et al*. Mining user opinions in mobile App reviews: A keyword-based approach. In: Proc. of the 30th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE 2015). 2015. 749–759.

[15] Jiang W, Zhang L, Dai Y, *et al*. Analyzing helpfulness of online reviews for user requirements elicitation. Chinese Journal of Computers, 2013, 36(1): 119–131 (in Chinese with English abstract).

[16] Hu TY, Jiang Y. Mining of user's comments reflecting usage feedback for APP software. Ruan Jian Xue Bao/Journal of Software, 2019, 30(10): 3168–3185 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5794.htm [doi: 10.13328/j.cnki.jos.005794]

[17] Martin WJ, Sarro F, Jia Y, *et al*. A survey of App store analysis for software engineering. IEEE Trans. on Software Engineering, 2017, 43(9): 817–847.

[18] Cui J, Yang D, Li J. RERM: A requirement elicitation method based on review mining. Computer Applications and Software, 2015, 32(8): 28–33 (in Chinese with English abstract).

[19] Villarroel L, Bavota G, Russo B, *et al*. Release planning of mobile apps based on user reviews. In: Proc. of the 38th Int'l Conf. on Software Engineering (ICSE 2016). 2016. 14–24.

[20] Vu PM, Pham HV, Nguyen TT, *et al*. Phrase-based extraction of user opinions in mobile App reviews. In: Proc. of the 31st IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE 2016). Singapore: ACM, 2016. 726–731.

[21] Cooper N, Bernal-Cárdenas C, Chaparro O, *et al*. It takes Two to Tango: Combining visual and textual information for detecting duplicate video-based bug reports. In: Proc. of the Int'l Conf. on Software Engineering (ICSE). 2021.

[22] Wang JJ, Li MY, Wang S, *et al*. Images don't lie: Duplicate crowdtesting reports detection with screenshot information. Information and Software Technology, 2019, 110: 139–155.

[23] Hao R, Feng Y, Jones J, *et al*. CTRAS: Crowdsourced test report aggregation and summarization. In: Proc. of the ICSE'2019. 2019. 921–932.

[24] Wang JJ, Cui Q, Wang S, *et al*. Domain adaptation for test report classification in crowdsourced testing. In: Proc. of the ICSE 2017. 2017. 83–92.

[25] Liu H, Shen MZ, Jin JH, *et al*. Automated classification of actions in bug reports of mobile Apps. In: Proc. of the ISSTA 2020: the 29th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Virtual Event, 2020. 128–140.

[26] Zhang J, Wang X, Zhang HY, *et al*. Retrieval-based neural source code summarization. In: Proc. of the ICSE 2020: the 42nd Int'l Conf. on Software Engineering. 2020. 1385–1397.

[27] Guo J, Cheng JH, Cleland-Huang J. Semantically enhanced software traceability using deep learning techniques. In: Proc. of the 39th Int'l Conf. on Software Engineering, ICSE 2017. 2017. 3–14.

[28] Wang H, Chen CY, Xing ZC, *et al*. DiffTech: A tool for differencing similar technologies from question-and-answer discussions. In: Proc. of the ESEC/FSE 2020: the 28th ACM Joint European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Virtual Event, 2020. 1576–1580.

[29] Huang ZH, Xu W, Yu K. Bidirectional LSTM-CRF models for sequence tagging. CoRR abs/1508.01991, 2015.

[30] Zhang Y, Chen HS, Zhao YH, *et al*. Learning tag dependencies for sequence tagging. In: Proc. of the 27th Int'l Joint Conf. on Artificial Intelligence (IJCAI 2018). 2018. 4581–4587.

[31] McCallum A, Freitag D, Pereira FCN. Maximum entropy Markov models for information extraction and segmentation. In: Proc. of the 17th Int'l Conf. on Machine Learning (ICML 2000). Stanford: Morgan Kaufmann Publishers, 2000. 591–598.

[32] Tang A, Jackson D, Hobbs J, *et al*. A maximum entropy model applied to spatial and temporal correlations from cortical networks in Vitro. Journal of Neuroscience, 2008, 28(2): 505–518.

[33] Felsenstein J, Churchill GA. A hidden Markov model approach to variation among sites in rate of evolution. Molecular Biology and Evolution, 1996, 13(1): 93–104.

[34] Lafferty JD, McCallum A, Pereira FCN. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proc. of the 18th Int'l Conf. on Machine Learning (ICML 2001). Williams College: Morgan Kaufmann Publishers, 2001. 282–289.

[35] Collobert R, Weston J, Bottou L, *et al*. Natural language processing (Almost) from Scratch. Journal of Machine Learning Research, 2011, 12: 2493–2537.

[36] Devlin J, Chang MW, Lee K, *et al*. BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Volume 1 (Long and Short Papers). Minneapolis: Association for Computational Linguistics, 2019. 4171–4186.

[37] Howard J, Ruder S. Universal language model finetuning for text classification. In: Proc. of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Volume 1: Long Papers. Melbourne: Association for Computational Linguistics, 2018. 328–339.

[38] Vaswani A, Shazeer N, Parmar N, *et al*. Attention is all you need. In: Proc. of the Advances in Neural Information Processing Systems 30: Annual Conf. on Neural Information Processing Systems 2017. 2017. 5998–6008.

[39] Xu L, Dong QQ, Yu C, *et al*. CLUENER2020: Fine-grained name entity recognition for Chinese. arXiv:2001.04351, 2020.

[40] Gao CY, Zeng JC, Lyu MR, *et al*. Online App review analysis for identifying emerging issues. In: Proc. of the 40th Int'l Conf. on Software Engineering (ICSE 2018). Gothenburg: ACM, 2018. 48–58.

[41] Gao CY, Zeng JC, Xia X, *et al*. Automating App review response generation. In: Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE 2019). San Diego: IEEE, 2019. 163–175.

[42] Islam MR, Zibran MF. SentiStrength-SE: Exploiting domain specificity for improved sentiment analysis in software engineering text. The Journal of Systems & Software, 2018, 145: 125–146.

[43] Dai HJ, Lai PT, Chang YC, *et al*. Enhancing of chemical compound and drug name recognition using representative tag scheme and fine-grained tokenization. Journal of Cheminformatics, 2015, 7: S1–S14.

[44] Ratinov LA, Roth D. Design challenges and misconceptions in named entity recognition. In: Proc. of the 13th Conf. on Computational Natural Language Learning, CoNLL 2009. Boulder: ACL, 2009. 147–155.

[45] Kingma DP, Ba J. ADAM: A Method for stochastic optimization. In: Proc. of the 3rd Int'l Conf. on Learning Representations, ICLR 2015. San Diego, 2015.

[46] Porteous I, Newman D, Ihler AT, *et al*. Fast collapsed gibbs sampling for latent dirichlet allocation. In: Proc. of the 14th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Las Vegas: ACM, 2008. 569–577.

[47] Cer D, Yang YF, Kong SY, *et al*. Universal sentence encoder for English. In: Proc. of the 2018 Conf. on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations. Brussels: Association for Computational Linguistics, 2018. 169–174.

[48] Biemann C. Chinese whispers - an efficient graph clustering algorithm and its application to natural language processing problems. In: Workshop on TextGraphs, at HLT-NAACL, Association for Computational Linguistics, 2006. 73–80.

[49] Huang Y, Chen CY, Xing ZC, *et al*. Tell them apart: Distilling technology differences from crowd-scale comparison discussions. In: Proc. of the 33rd ACM/IEEE Int'l Conf. on Automated Software Engineering, ASE 2018. Montpellier: ACM, 2018. 214–224.

[50] Wu HY, Deng WJ, Niu XT, *et al*. Identifying key features from App user reviews. In: Proc. of the 43rd IEEE/ACM Int'l Conf. on Software Engineering, ICSE 2021. Madrid: IEEE, 2021. 922–932.

[51] Ester M, Kriegel H, Sander J, *et al*. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. of the 2nd Int'l Conf. on Knowledge Discovery and Data Mining (KDD-96). 1996. 226–231.

[52] Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proc. of the 14th Int'l Joint Conf. on Artificial Intelligence (IJCAI 1995). 1995. 1137–1145.

附中文参考文献:

[7] 卢忠浩, 杨达, 李娟. 基于评论挖掘的软件评价系统. 计算机应用与软件, 2014, 31(7): 1–4.

[15] 姜巍, 张莉, 戴翼, 蒋竞, 王刚. 面向用户需求获取的在线评论有用性分析. 计算机学报, 2013, 36(1): 119–131.

[16] 胡甜媛, 姜瑛. 体现使用反馈的 APP 软件用户评论挖掘. 软件学报, 2019, 30(10): 3168–3185. http://www.jos.org.cn/1000-9825/5794.htm [doi: 10.13328/j.cnki.jos.005794]

[18] 崔建苓, 杨达, 李娟. RERM: 一种基于评论挖掘的需求获取方法. 计算机应用与软件, 2015, 32(8): 28–33.

王亚文(1993－), 男, 博士生, 主要研究领域为智能需求工程, 软件工程, 自然语言处理.

石琳(1985－), 女, 博士, 副研究员, CCF 高级会员, 主要研究领域为智能需求工程, 软件工程, 经验软件工程, 软件演化, 软件质量.

王俊杰(1987－), 女, 博士, 副研究员, 主要研究领域为智能软件工程, 软件工程大数据, 经验软件工程, 软件质量, 众包软件测试.

王青(1964－), 女, 博士, 研究员, 博士生导师, CCF 高级会员, 主要研究领域为以过程为中心的软件质量管理技术, 建模技术, 知识管理技术, 软件协同工作技术.