

一种基于用户评论自动分析的 APP 维护和演化方法

肖建茂 陈世展 冯志勇 刘朋立 薛 霄

(天津市认知计算与应用重点实验室 天津 300350)

(天津大学智能与计算学部 天津 300350)

摘 要 应用程序分发平台(如 Google Play Store 或 Apple App Store)允许用户以评分或者评论等形式向下载的应用程序提交反馈。这些反馈信息可以直接或者间接地反映用户意图,及时准确地挖掘用户意图可以极大地帮助移动开发人员持续维护和改善他们的 app,从而更好地满足用户期望。然而,对于很多流行的应用来说,由于其用户评论数据量大、非结构化以及评论质量不一致等,使得识别其中有价值的信息成为一项极具挑战的任务。因此,将用户评论进行自动化分析以减缓人工分析工作量成为 app 评论挖掘的新思路。本文我们提出了一种自动化用户评论分析方法 ARICA(Automatic Review Intention Classification Analysis)。首先,ARICA 根据用户的评论反馈,自动将评论信息进行意图分类,并使用 LDA 主题模型对每个分类意图下评论进行主题划分;其次,在每个主题下对表达语义相似的评论进行聚类,进一步,ARICA 使用情感分析工具 SentiStrength 获取用户情感,然后对用户评论的情感分布进行分析来识别用户的重要意图;最后,综合考虑用户意图和用户情感偏好等多维度信息计算用户评论得分并以此划分评论优先级,从而为开发者进行用户评论意见推荐。我们使用 Google Play 中真实的 app 评论数据来验证 ARICA 的评论意图分类和句子聚类的性能。实验结果表明,ARICA 在用户评论意图分类过程中准确度达到 80%,和现有的基于卷积神经网络的方法 TextCNN 的相比,ARICA 的 F -Measure 提高了 19.1%。同时,评论句子聚类过程中获得 86% 的准确率。另外,为了验证 ARICA 推荐用户评论的有效性,我们使用官方的 app 更新日志来实证分析 ARICA 推荐的用户评论建议是否可以真实地被开发者采用。结果表明,ARICA 可以高效为开发者推荐具有价值信息的评论,这对于开发者进行后续的 app 维护和演化任务具有重要意义。

关键词 用户评论;意图分类;情感分析;维护和演化;意见推荐

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2020.02184

An Automatic Analysis of User Reviews Method for APP Evolution and Maintenance

XIAO Jian-Mao CHEN Shi-Zhan FENG Zhi-Yong LIU Peng-Li XUE-Xiao

(Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin 300350)

(College of Intelligence and Computing, Tianjin University, Tianjin 300350)

Abstract Application distribution platforms such as Google Play Store or Apple App Store allow users to submit feedbacks to download applications in the form of ratings or reviews. These feedbacks can directly or indirectly reflect users' intention, and it can greatly help mobile developers (or app provider) to continuously maintain and improve their applications, such as fix the existing bugs, add or refining the app features, etc. and so as to better satisfying user expectations continuously. App reviews provide an opportunity to proactively collect user complaints and

收稿日期:2019-11-19;在线发布日期:2020-05-13。本课题得到国家自然科学基金重点基金(61832014)、国家自然科学基金(61572350)、国家重点研发计划(2017YFB1401201)资助。肖建茂,博士研究生,中国计算机学会(CCF)学生会员,主要研究方向为服务计算、智能化软件工程。E-mail: zt_xjm@tju.edu.cn。陈世展,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为服务计算、面向服务的体系架构。冯志勇,博士,教授,中国计算机学会(CCF)会员,主要研究领域为知识工程、服务计算、计算机认知。刘朋立,硕士研究生,主要研究方向为服务计算、软件工程。薛霄(通信作者),博士,教授,中国计算机学会(CCF)会员,主要研究领域为服务计算、计算实验。E-mail: jzxuexiao@tju.edu.cn。

promptly improve apps' user experience, in terms of bug fixing and feature refinement. However, for many popular applications, since the large amount of user review data, unstructured review data, and inconsistent review quality, identifying the valuable review information becomes a challenging task. Therefore, classification of user reviews into specific topics and automated analysis to reduce the workload of manual analysis has become a new idea for app review mining analysis. In this paper, we propose a method named ARICA (Automatic Review Intention Classification Analysis) to automatically analyze crowd user reviews to efficiently provide developers with software maintenance and evolution suggestions. Firstly, ARICA classifies the reviews into different categories according to the user's feedbacks, and then uses the LDA topic model to classify the reviews under each user's intent category. This allows a preliminary screening of user reviews to obtain review information under each intent category. Secondly, ARICA clusters user views with similar semantic expressions under each review topic to further filter the redundant information in reviews, so that can easier and intuitive to understand the user's original feedback and capture the user's true intention more accurately. Afterwards, ARICA uses the sentiment analysis tool called SentiStrength to obtain user sentiment, and then analyzes the sentiment distribution of user reviews to identify the user's significant intentions. Finally, the multidimensional information such as user intentions and sentiment preferences are considered comprehensively for calculating the review score and then ARICA prioritizes reviews for realizing the opinions recommendation for the developers. We use real app review data from Google Play to verify the performance of review intent classification and sentence clustering of ARICA. The experimental results show that ARICA has precision of 80% in the process of user review intention classification, compared with the state-of-the-art existing automatic user intentions mining method TextCNN which based on Convolutional Neural Networks (CNN), the F-Measure of ARICA is improved by 19.1%. Meanwhile, ARICA achieves 86% of the precision during the clustering of review sentences, which provides effective support for subsequent developers to recommend app update tasks. Further, we use the official app changelog as a ground truth, and empirically analyzed whether our recommended user reviews can be truly adopted by developers, the results show that ARICA can efficiently help developers better understand the user's real requirements, which is of great significance for developers to perform subsequent app maintenance and evolution tasks. In addition, we also publish the original data set, the manually labeled data set and the source code of ARICA on github which provide materials for other relevant researchers.

Keywords user reviews; intent classification; sentiment analysis; maintenance and evolution; opinion recommendations

1 引言

移动用户可以在应用程序分发平台(如 Apple 的 App Store 和谷歌的 Google Play)获取他们想要的应用程序并分享他们对应用程序的意见和反馈^[1]. 这些反馈信息可以作为用户和开发者进行沟通的一个重要渠道, 用户使用 app 后的反馈信息可以指导 app 开发者完成多项软件维护和演化任务:

比如修复相关的 bugs 或 issues、或者改进 app 现有的特性或功能^[2]. 然而, 去处理、分析和找出这些有价值的用户评论信息存在诸多挑战. 首先, 应用平台每天都会产生大量的用户评论. Pagano 等人^[3]通过实证研究发现手机应用程序每天大约收到 23 条评论, 而像 Facebook 这样的热门应用程序每天会在 Google Play 上收到超过 1 万条评论^①, 如果人工进

① App Annie. <https://www.appannie.com/en/>

行分析将耗时耗力。其次,用户评论中夹杂大量的噪声数据,比如拼写错误的单词、重复的单词和非英语的单词,并且大多数 app 评论文本长度较短,并且不遵循语法规则的特点,导致没有足够的信息量来进行统计推理^[4]。再次,用户评论的质量差别很大,从提供改进想法的有用评论,到一般性的表扬以及用户的投诉抱怨(比如“Account hasn’t been hacked, please fix this!”,“i love it, i just love it”,“this is a very bad app!!!”)都大量散布在 app 评论中。对于这些评论,如果采用人工阅读的方式进行处理将会消耗大量的时间^[5]。因此,对用户评论信息进行自动化分析,系统地过滤、聚合和分类具有十分重要的意义。

自动化评论分析方法主要是根据特定的主题(例如:bug 修复、功能改善等)对用户评论进行分类(或者优先级排序)^[6-11]。Maalej 等人^[7]结合文本分类、自然语言处理和情感分析等技术来实现对应用评论的分类,将应用评论分为错误报告、特征请求、用户体验和评级四个大类,Panichella 等人^[8,12]同样采用自然语言处理、文本分析和情感分析技术组合的方式将评论分为寻求信息、信息搜寻、功能要求、问题发现等有限的几个类别。虽然这些工作可以从冗余的移动应用评论中自动提取部分有效的信息,但对一些流行的 app 来说,由于其评论数据庞大,用上述方法分析得到的结果对开发者进行具体的维护任务仍然是不够的,这些研究工作都是根据非常有限的一组类别对评论进行分类,并产生一个非结构化评论集群列表,最后仍然需要手动分析才能理解他们讨论的主题,而用户真正关心 app 哪一方面具体的信息仍不够准确。

据我们查阅目前所发表的文献发现,只有 Di Sorbo 等人^[13-14]提出的 SURF 方法结合了用户评论主题提取、用户意图分类以及系统地定义了围绕 app 具体某些方面(比如:UI、下载信息等)需要进行维护的工作,并可以有效帮助开发人员规划下一步的 app 更新任务。但我们认为这个方法还是不够自动化,SURF 的主题分类需要自定义概念词典,然后人工去验证,同时随着新的用户评论产生,字典的有效性会受到限制;另一方面,SURF 推荐建议时没有考虑用户评论的真实情感倾向,评论时间等维度信息。事实上,这些信息对于帮助反应真实的用户意图具有重要作用,可以更准确地帮助开发者高效地识别用户的真实需求。因此,本文我们提出一种更加自

动化的用户评论挖掘分析方法 ARICA。ARICA 首先利用 NLP 最新的 BERT^[15]模型将用户评论意图进行自动分类;其次,在意图分类的基础上,利用主题模型 LDA 技术和词向量技术结合实现评论主题聚类;并进一步将各个评论主题下的评论进行聚类分析,得到最能表达用户意图的语句;最后,将聚类后的评论,结合用户情感倾向,评论时间信息以及用户评论点赞数等信息综合进行开发者建议推荐。这样可以实现自动化高效的用户评论信息挖掘,从而为开发者推荐关于 app 维护更新最相关的评论内容。本文的贡献如下:

(1)提出了一种利用最新的自然语言处理技术 BERT 模型来实现的用户评论意图自动分类的方法,大大的提高了用户意图分类的准确性。和现有的最好基于卷积神经网络(Convolutional Neural Networks,CNN)方法 TextCNN^[16]的用户意图自动挖掘相比,ARICA 的 *F*-Measure 提高了 19.1%。

(2)综合考虑了用户评论的情感、评论意图、评论时间等多维度等的信息对用户评论进行优先级排序,为开发人员推荐用户真实需求,为后续 app 的版本演化和维护提供高效建议。

(3)我们公开了本文方法使用到的原始数据集和人工标注的数据集(用于复现以及未来的研究),以及该方法实现的源代码^①。

(4)我们使用官方的 app 更新日志来进行详细的案例分析,验证了 ARICA 为开发者进行评论意见推荐的高效性。

本文第 2 节介绍本文的相关工作;第 3 节阐述我们方法的整体思路和使用到的相关技术;第 4 节阐述我们设计的实验和分析方法;第 5 节展示和讨论实验结果;第 6 节描述可能影响我们工作有效性的威胁;第 7 节总结论文并概述了下一步的研究工作。

2 相关工作

对于人类而言,理解应用评论十分简单,这是因为人能够根据积累的知识做出判断。例如“I like this application”和“this application is good, like it”。人类可以清楚的知道这两句话表达的意思是相似的。然而对应用评论进行理解和自动化分析则是一项充

① <https://github.com/xuanhui0129/ARICA>

满挑战的任务。App 评论数据和长文本数据不同,大多数 app 评论具有短小并且不遵循语法规则的特点,因此没有足够的信息量来进行统计推理^[4]。传统对评论的分析方法主要是检查评论中是否包含某个关键字并手动定义关键字列表,然后检查文本中是否包含关键字来挖掘评论的价值^[17],如 Bakiu 等人^[18]提出一种基于评论分析用户情感的方法,他们将用户评论中经常出现的关键词识别为特征,并将提取的特征与用户情感关联,然后使用主题建模进行分组,最后将提取的功能相关的评论分类为与可用性和用户体验相关的需求类别来进行评论挖掘。Dabrowski 等人^[19]针对用户评论分类的类别,提出了一种对用户感兴趣的特定功能检索工具,从而能够进一步支持需求工程师了解用户针对 app 特定功能的需求,以更好地维护对应的 app。

近年来,已经有相关学者对用户评论自动分类方法进行了研究。Chen 等人^[6]对大量的 app 评论进行研究,发现在评论中有很多用户单纯的表达喜恶,如“good, I like it”等,这些评论无论对于开发者还是其他的用户来说,都没有提供有效的信息。基于此现象,他们开发了一种工具可以将评论分为信息性和非信息性,并过滤无用的信息,从而提取评论中有效的信息。Ciurumelea 等人^[20]基于用户评论定义了一个包含移动应用特定类别(例如性能、资源、电池、内存等)的高级和低级分类,并基于此构建了用户请求引用器(URR)原型,然后使用机器学习和信息检索技术进行评论自动分类。Guzman 等人^[2]应用 NLTK 工具对应用评论进行特征提取并使用 SentiStrength 工具进行情感分析,从而挖掘用户评论中的细粒度特征。为了使挖掘得到评论更好的方便开发人员进行开发,在文献[2]的基础上,Guzman 等人^[10]进一步提出了一种以用户评论特征和情感为中心的检索方法 DIVERSE,它可以自动为开发人员提供一个多样化的用户评论样本,这些样本代表着整个评论集中用户提到的不同意见和体验。Dhinakaran 等人^[21]提出了一种结合主动学习以减少 app 评论分析中涉及到的人工操作的方法,其将评论分为功能请求、错误、评级和用户体验几个类别,并使用主动学习相关策略应用到评论分类中进行分析。Messaoud 等人^[22]针对开发人员难以手动分析用户评论并将它们归类到适当的评论类别中,提出了一种使用多标签主动学习方法来实现用户评论的分类问题,从而更好地保证评论的快速准确挖掘。Khalid 等人^[23]重点研究了 20 个免费下载的

iOS 应用程序的低评级用户评论,他们发现了 12 种类型的用户投诉以及它们的频率和影响,这可以帮助开发者更好地优先考虑利用现有的资源来保证 app 的质量。Liu 等人^[24]通过提取 app 描述中的特征词的方法来指导评论的分析,并建立了用户情感和主题特征词之间的关联,该方法将贡献最大的词作为主题名,并通过时间、评论的评分及评论的客观与否来设置评论优先级,从而实现评论推荐。但是,他们忽略了同一个问题可能会被多个用户提起,也忽略了其他有贡献的主题词的作用。Scalabrino 等人^[25]提出了用 DBSCAN 聚类算法来实现对评论的聚类,从而挖掘用户评论的意见。Gao 等人^[26]基于在线 app 评论数据,提出了一种自动化框架 IDEA 来自动分析用户评论,IDEA 分别使用了 LDA 主题模型检测和推断分析来识别评论中的紧急问题,并通过 app 官方更新日志验证了 IDEA 的有效性。除此之外,也有一些比较新颖的工作来进行 app 评论数据分析挖掘用户需求,比如 Dalpiaz 等人^[27]提出一个工具 RESWOT,用于通过对应用程序商店中竞争对手 app 的评论分析中挖掘用户需求。上述方法都可以在某种程度上缓解人工分析用户评论的工作,但不能细粒度让开发者理解用户的真实意图,从而难以指导开发者直接进行后续 app 维护和演化任务。

为了更好地理解用户细粒度需求,Di Sorbo 等人^[28]提出了意图挖掘的概念。具体来说,他们提出了一种意图分类法,将开发人员讨论中的句子从邮件列表中分为六类:特征请求、意见询问、问题发现、解决方案建议、信息寻求和信息提供。Guzman 等人^[9]提出了用于将应用程序评论分类为与软件维护相关类别的方法。具体来讲,他们将用户评论分为错误报告、功能优势、功能缺陷、用户请求、好评、投诉和使用情况这几个类别,这可以为开发人提供细致的用户评论建议。Maalej 等人^[7]结合文本分类、自然语言处理和情感分析等技术来实现对应用评论的分类,将应用评论分为错误报告,特征请求,用户体验和评级四个大类。Panichella 等人^[8]同样采用自然语言处理、文本分析和情感分析技术组合的方式将评论分为信息提供、信息搜寻、特征请求、解决方案以及问题发现等几个类别^[11],并公开了一个基于这个分类的评论数据集^[29]。在文献[8]工作的基础上,Huang 等人^[16]基于卷积神经网络(CNN)的基础上提出了 TextCNN 用于评论意图的分类,该工作将文献[8]中的用户意图增加了方面评估和无意

义这 2 个类别,然后用标记的文本进行训练,和直接基于 CNN^[30]、LibSVM^[31] 等分类算法相比,获得了当前意图分类最好的效果. 相比于直接进行用户评论主题提取或聚类,上述这些方法都将用户评论信息数据进行了进一步的细化. 经过我们调研大量的文献发现,目前只有 Di Sorbo 等人^[13] 研究工作不仅将用户评论意图进行了细粒度分类,同时重点考虑开发者关注的 app 多个方面相关的主题信息. 但他们的主题分类需要自定义概念字典,然后人工去验证,这将导致评论推荐的效果受到字典的丰富性影响,且分类的自动化程度不够.

综上,无论是基于传统的关键词提取用户评论信息还是基于用户意图分类进行自动用户评论分析,最重要的是用户意图划分的合理性以及意图分类的准确性,同时需要考虑多维度因素实现综合的评论优先级推荐,这是挖掘用户评论中的蕴含的价值信息的核心内容,也是开发者能否顺利实施后续 app 维护和演化任务的关键.

3 ARICA 整体框架

我们的目的是为了给开发者在其更新维护应用时提供高效的参考意见. 为了实现这一目标,我们将用户评论分析定位到句子级别,因为在原始用户评论中,有些句子与软件开发和维护相关,而有些则与之无关. ARICA 可以自动实现:(1) 对用户评论进行意图分类,分类成和 app 更新和维护方面相关的类别;(2) 对各个意图下用户评论进行主题分类,从而进一步提高评论的内聚性,同时将各个主题下用户评论进行句子聚类,得到和用户评论意图最相关的句子;(3) 重点分析用户评论的情感分布并识别用户重要意图;(4) 综合考虑用户意图、情感倾向以及评论时间等多维度信息计算用户评论综合得分,并制定评论优先级规则进行用户评论推荐. 图 1 展示了方法的整体框架处理流程,下面我们进行逐一介绍.

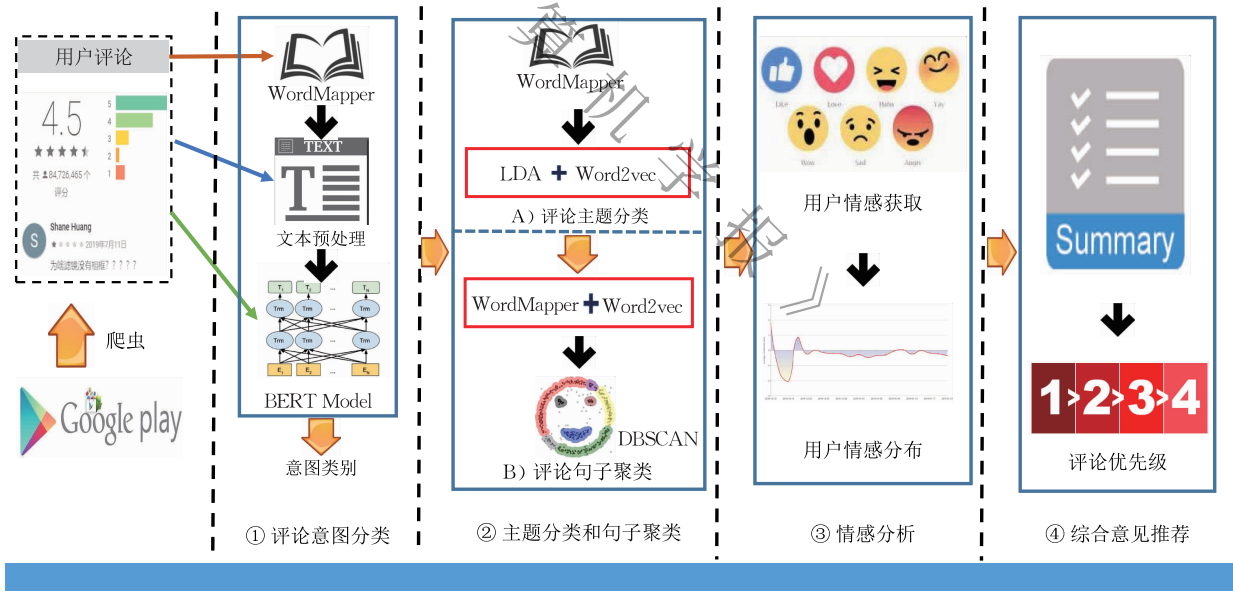


图 1 ARICA 整体框架

3.1 数据预处理

移动用户通过终端(比如智能手机)提交的评论一般包含许多噪声数据,比如随意的单词,重复的单词以及非英文的单词等,这些会影响对数据处理的结果. 因此,我们首先需要对评论数据进行数据预处理. 本文我们使用 NLP 相关技术实现评论的数据预处理,具体包括过滤掉非英文词汇、无用的表情符号等,同时使用正则表达式过滤掉网址、邮箱等干扰信息. 另一方面,由于用户通常在移动设备上提交他们

的评论,而这些移动设备通常尺寸比较小,没有实体键盘,因此打字既困难又耗时,很容易会产生错误. 这种情况导致在用户对这些移动应用的评论中经常出现拼写错误(即拼错单词)、缩略词和缩写,比如单词“usefull”,显然用户是想要表达的是“useful”,“friends”缩写成了“frnds”. 这样就会导致我们在进行评论分析时将他们当成不同的词来处理,这会对数据分析结果造成不良的影响,产生噪声数据. 因此需要将拼写错误、缩略词和缩写的词进行还原. 本文

我们基于文献[32]自定义的词典 wordMapper^①，一个手工整理了近 300 000 和 app 评论相关的词汇字典，其中包含了在用户评论中常见的拼写错误、缩写和缩略的单词及其校正后的单词。在该字典的基础上，我们进一步往字典中增加 app 评论相关的更细致词汇，比如“you’re→you are”等。最后利用完善后的字典可以很好地将常见的容易拼写错误或者缩略的单词进行校正还原。

3.2 评论意图分类

为了帮助开发人员更容易地理解用户在应用市场的评论反馈，Di Sorbo 等人^[14]提出用户评论反馈分类方法 SURF，旨在从软件维护和演化的角度对应用程序评论中包含的信息段落进行建模。如表 1 所示，他们把用户的评论意图分为 5 类，分别为信息提供、信息搜寻、特征请求、问题发现以及其他，并且证明这种分类可以很好地被开发者所理解并进行后续 app 维护和演化任务。本文我们使用的也是这种意图分类方法，因为这些意图可以很好表达用户对 app 的真实诉求，同时也可以让开发者容易理解。

表 1 用户评论意图分类类别

类别	描述
信息提供	告知其他用户或者开发者 APP 某方面信息的句子
信息搜寻	描述试图从其他用户或开发人员获取信息或帮助的句子
特征请求	表达用户观点、建议或 APP 某些方面需要增强的句子
问题发现	报告异常行为或问题的句子
其他	不属于任何以上类别的句子

传统的文本分类通常使用 TF-IDF 方法来提取特征进行分类，但这种方法忽略了同一语义的不同表达方式。因此，Huang 等人^[16]基于 CNN 进行句子意图的分类，该方法将句子进行向量化表示，向量间的距离代表了句子的相似度，与 TF-IDF 相比，该方法考虑了词与词之间的关系，有效地提高了分类效率，但他们的方法的效率还是不够的，因为他们只是考虑了词与词之间的关系，并没有考虑词上下文之间的关系。因此，本文我们采用了最新的自然处理模型 BERT 进行自动的用户意图分类，BERT 是一种基于 transformer 架构的双向模型，它在进行评论句子向量化后包含了上下文的信息，可以有效地弥补基于 CNN 方法的不足。

BERT 的输入表示(input representation)能够在一个词(token)序列中明确地表示单个文本句子或一对文本句子(例如，[Question, Answer])。对于给定词，其输入表示通过对相应的词嵌入(token embeddings)、分割嵌入(segment embeddings)和位

置嵌入(position embedding)进行求和来构造。图 2 展示了一个基于 BERT 进行评论意图分类的例子，输入的评论句子“We need night mode”作为一个句子整体输入到 BERT 模型中，第一个词是一个特殊的符号[CLS]，表示为分类，然后经过词、分割和位置嵌入进行求和操作，每个位置对应的输出为一个维度为 hidden_size (BERT Base 中为 768)的向量^②。本文我们执行的是评论句子意图分类任务，因此我们只关注第一个位置的输出(也就是被我们用[CLS]符号代替的位置)向量。输出的这个向量作为我们前馈神经网络(Feed-forward neural network)分类器的输入，最后得到用户评论句子的意图分类结果，图中句子的分类结果为特征请求。

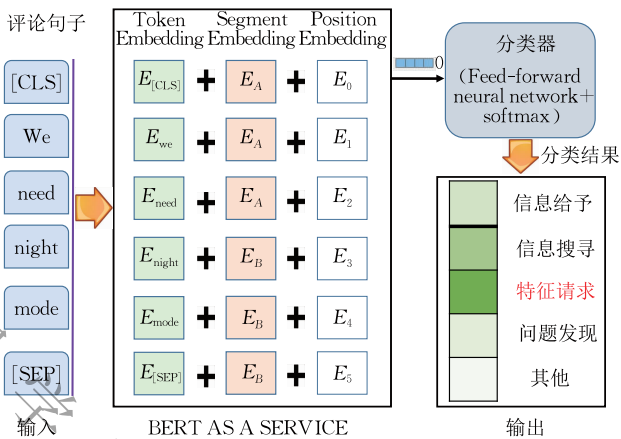


图 2 基于 BERT 的评论意图分类示例

3.3 主题分类及句子聚类

用户的意图分类只能在宏观层面表示用户的想法，开发者并不能直接基于这个意图分类进行 app 维护任务。因此，需要进一步对各个意图分类下的评论信息进一步进行提取，从而实现更加细粒度的用户评论挖掘。这里主要包含评论的主题分类和每个主题下句子聚类。

3.3.1 评论主题分类

单从句子意图分类这个维度对充分利用评论中的信息是不够的。比如，当信息搜寻类别中存在大量的评论句子时，开发者要从中找到和 app 维护相关信息仍然费时费力。因此，我们需要进一步提取出每个意图下用户评论隐藏的主题，这些主题可以表达出该意图类别下用户关心的内容，这可以帮助开发者更直接地了解用户的具体意图。

① <https://github.com/xuanhui0129/ARICA/blob/master/file/wordMapper.txt>
② <https://github.com/google-research/bert>

LDA^[33]是一种经典的主题建模算法,可以用来识别大规模文档集或语料库中潜藏的主题信息. 本文我们使用 LDA 主题模型来实现用户评论主题分类,并采用 word2vec^[34]词向量实现用户评论的向量化. 同时,为了保证 LDA 模型能够对新产生的用户评论进行主题划分,我们取每个主题下概率最大的前 N 个词进行加权求和作为该主题的中心向量,对于新产生的用户评论,我们计算它和各个主题中心距离,并把距离主题最近的评论划分为该评论的主题. 这样可以使得用户新产生的评论及时地进行主题自动划分. 这里向量距离为余弦距离,计算公式如式(1)所示:

$$Review_Sim = \frac{\sum_{i=1}^n (\mathbf{X}_i \times \mathbf{Y}_i)}{\sqrt{\sum_{i=1}^n (\mathbf{X}_i)^2} \times \sqrt{\sum_{i=1}^n (\mathbf{Y}_i)^2}} \quad (1)$$

其中, $\mathbf{X}_i$ 和 $\mathbf{Y}_i$ 分别表示为主题中心向量和新产生的评论句子向量. $Review_Sim$ 的取值范围为 $(0, 1)$, 取值越大, 距离越近. 句子主题分类过程的伪代码如下算法 1 所示.

算法 1. 用户评论主题分类.

输入: D : the set of user_review;

k : the number of topic

size: the size of word embedding

review: pending review

输出: $topic_id$: the $topic_id$ of input review

Procedure:

1. initialization: $topics \leftarrow LDA(D, k)$
2. FOR EACH $topic \in topics$ DO
3. $topic_words \leftarrow topic.keywords$
4. FOR EACH $topic_word \in topic_words$ DO
5. calculate weight vector sum ;
6. END FOR
7. weight vector sum as sum_vec ;
8. add sum_vec to $topic_vecs$;
9. END FOR
10. $review_vec \leftarrow Vector(review)$
11. initialization: $minVec \leftarrow 1$;
12. initialization: $topic_id \leftarrow -1$;
13. FOR EACH $topic_vec \in topic_vecs$ DO
14. IF $distance(topic_vec, review_vec) < minVec$ THEN
15. $minVec \leftarrow distance(topic_vec, review_vec)$
16. $topic_id \leftarrow topic_vec.id$
17. END IF
18. END FOR
19. RETURN $topic_id$

3.3.2 评论句子聚类

在同一个评论主题下, 评论的句子仍然较多, 尤其是对于一些流行的 app, 存在一系列的句子是在讨论 app 同一个方面的内容, 即存在语义相关. 因此, 我们进一步在每个主题下把语义相似的句子进行聚类, 这样可以为开发者过滤大量冗余信息, 从而更加容易和直观地理解用户的原始反馈并能更快地捕获用户的真实意图, 为进一步进行 app 相关的实际维护和演化任务提供支撑.

另一方面, 由于每个主题下用户评论的聚类个数未知. 因此, 本文我们采用一种基于密度的聚类算法 DBSCAN^[35]来对每个评论主题下的语句进行聚类, 这可以避免 k -means 聚类时选择初始聚类中心对聚类结果影响. 同时, 由于 DBSCAN 算法的输入为向量, 并且考虑到用户评论语句存在相似性, 我们同样使用 word2vec 对评论句子处理, 从而得到词向量模型. 进一步, 在进行聚类时, 我们对句子进行预处理后分别求出各个词的向量, 并对词向量加权求和. 加权之后的向量不仅对数据进行了降维, 同时也表示了句子的语义信息. 最后, 将加权之后的向量使用 DBSCAN 算法进行聚类, 找到句子之间距离较短、相似度较高的并将其聚类到一块, 从而帮助开发者快速识别用户的真实意图.

3.4 评论情感分析

在不同的时间段下, 用户的情感会可能发生大幅度的变化, 尤其在 app 出现 bug 或隐私安全等问题时, 会导致用户情感可能出现诸如波峰和波谷等趋势. 当用户情感出现波峰或者波谷时, 这很大程度上是和 app 相关的特征发生变化所导致, 比如当 app 出现崩溃或隐私等问题时, 用户会通过评论及时表达出很多负面的情感, 当 app 出现新的功能特征, 或者发布新版本时 (此时一般伴随新功能的产生), 用户整体会通过评论及时表达出较高正面的情感, 而在一般情况下, 用户的情感趋于平稳.

图 3 展示了用户的情感演化趋势. 在 T_i 时间片下用户情感达到波谷, 在 T_j 时间片下达到波峰, 其他时间片下变化较小. 我们基于不同时间段的用户情感倾向来对用户意图进行权重分配, 比如当用户情感趋势处在波谷时, 推荐权重更大, 而在用户情感趋势在波峰或者稳定时, 推荐的权重较小, 这是因为情感波峰或者稳定时, 用户可能没有遇到明显的问题, 此时用户更多的可能是在表达使用 app 的一些体验信息. 而在波谷时可能由于 app 出现 bug 等情况, 此时用户更有可能表达出对 app 的相关意图 (如

请求修复 bug 等). 这可以帮助开发者了解用户的情感以及这些情感如何和用户的意图相关联, 从而为开发者推荐更精准的用户建议.

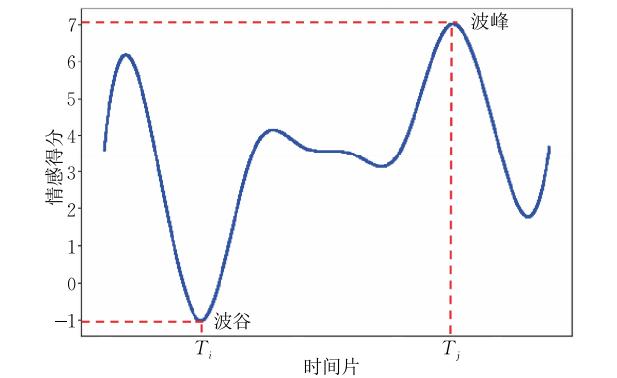


图 3 用户情感趋势示意图

3.4.1 获取评论情感得分

情感分析是为每个评论分配固定的正值或负值的过程^[36]. 为了分析用户评论中的情感, 我们使用 SentiStrength^[37] 情感分析工具来实现用户评论情感分析, 它对于诸如 Twitter 和电影评论之类的社交媒体中的短文本具有很高的准确性^[38]. Pagano 等人^[3]发现, App Store 中 80.4% 的评论所包含字符少于 160. 同时, SentiStrength 可以很好地计算短

文本中正面和负面的情感强度, 即使是非正式语言也是如此, 而 app 的评论信息与 SentiStrength 的这些优势刚好非常匹配, 这使得 SentiStrength 成为分析 app 用户评论情感的理想工具.

SentiStrength 可以将评论信息进行句子级别处理, 并分配相应的正值或负值. 其中分值范围是 $[-5, 5]$, 其中 $[+1, +5]$ 表示正面情感, $+5$ 表示非常正面的情感, $+1$ 表示没有正面情感. 类似地, $[-5, -1]$ 表示负面情感, -5 表示非常负面的情感, -1 表示不存在任何负面情感. 在本文中, 针对每一个评论语句, 我们取该句子得分范围中绝对值较大的值作为句子的最后得分, 因为绝对值更大的值更能反映句子的实际情感. 另外, 由于负值可以更好地体现用户的意图, 当句子得分的正值和负值相等, 取负值作为整个句子得分, 从而避免漏掉相应的评论句子信息.

值得注意的是, 由于用户评论中的表情符号, 极性词等也会影响用户的情感. 因此, 情感分析时我们无需将评论语句经过 3.1 节所述的数据预处理过程, 而是直接分析用户评论情感得分. 表 2 展示了经过 SentiStrength 工具得到的 3 个评论句子得分示例.

表 2 基于 SentiStrength 的评论得分

评论句子	单词得分	句子得分
uploading pictures with the app is so annoying!	uploading pictures with the app is so annoying $[-3]!$ $[-1$ punctuation emphasis]	$\{1, -3\}$
love listening to Pandora ... has everything you could need and want.	love $[3]$ listening to Pandora $[proper noun]$... $[sentence: 3, -1]$ has everything you could need and want $[sentence: 1, -1]$ $[result: max and - of any sentence]$ $[overall result = 1 as pos > - neg]$	$\{3, -1\}$
works great I prefer over Amazon music!!	works great $[3]$ I prefer over Amazon $[proper noun]$ music $!!$ $[1$ punctuation emphasis] $[sentence: 4, -1]$ $[result: max and - of any sentence]$ $[overall result = 1 as pos > - neg]$	$\{4, -1\}$

3.4.2 评论情感分布

用户在不同时间段内对 app 情感会发生变化, 假设用户在时间段 T_i 对 app 给出了 n 条评论, 基于 3.4.1 节, 得到 $RS(score) = \{RS_{i,1}, RS_{i,2}, \dots, RS_{i,n}\}$ 为 n 条评论情感得分. 值得注意的是, 在固定的时间段内, 用户对同一个 app 的评论数目并不固定. 因此, 我们把 T_i 时间段内用户整体对 app 的平均情感得分 $ARS(score)$ 作为用户群体的情感倾向, 计算公式如式 (2) 所示:

$$ARS(score) = \frac{1}{n} \sum_{\substack{j=1 \\ i \in T(t_1, t_2, \dots, t_m)}}^n RS_{i,j} \quad (2)$$

其中, $T(t_1, t_2, \dots, t_m)$ 表示一系列等长且不重复的

时间间隔, $RS_{i,j}$ 表示为 app 在 T_i 时间段内的第 j 条评论情感得分, n 表示在 T_i 时间段内的评论数目.

图 4 通过折线图展示了在不同时间段内用户对 app 的情感得分分布, 这里我们选择了当下比较流行的应用 Facebook 和 Uber 分析, 可以看到用户情感展示多元化的变化趋势, 在不同的时间段内, 用户的情感会出现迅速上升或者下降的趋势, 从而出现诸如波峰和波谷等趋势, 也有出现一段时间内用户情感比较稳定的趋势. 这些情感波谷趋势是后续开发者进行 app 维护和更新需要重点关注的地方.

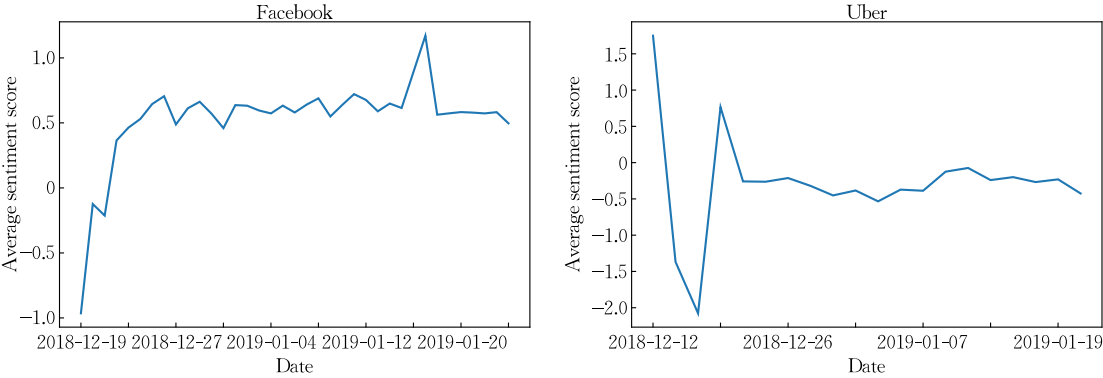


图 4 不同时间段下用户评论情感趋势

3.4.3 情感波谷识别

我们根据用户的情感变化,得到用户在使用 app 中的情感分布曲线,当用户的评论在波谷(即负面情绪集中)时,我们认为此时用户所提出的评论反馈更能够反映用户意图. 识别用户情感波谷的主要思想是通过划分固定的时间段(比如 3 天、一周,可根据需要动态调整),然后在该时间段内寻找情感波谷,用户情感波谷识别具体流程如算法 2 所示.

算法 2. 用户情感波谷识别.

输入: D : the set of user_review which has sentiment_score and date;
 ϵ : the Threshold of Days

输出: $trough_list$: the list of troughs

Procedure:

```
1. initialization:  $status \leftarrow unknown$ ;  
2. FOR EACH  $d \in D$  DO  
3.   IF  $d.status == unknown$  THEN  
4.     IF  $d.sentiment\_score > d+1$  THEN  
5.        $status \leftarrow downhill$ ;  
6.     END IF  
7.   ELSE THEN  
8.      $status \leftarrow uphill$ ;  
9.   END ELSE  
10.  END IF  
11.  IF  $d.status == downhill$  THEN  
12.    IF  $d.sentiment\_score < (d+1).sentiment\_score$  THEN  
13.      IF  $d.date - trough\_list[-1].date < \epsilon$  THEN  
14.        IF  $trough\_list[-1].sentiment\_score > d.sentiment\_score$  THEN  
15.           $trough\_list[-1] \leftarrow d$ ;  
16.        END IF  
17.      ELSE THEN:  
18.        add  $d$  to  $trough\_list$ ;
```

```
19.      END ELSE  
20.    END IF  
21.  END IF  
22. END FOR  
23. END IF  
24. RETURN  $trough\_list$ 
```

3.5 评论优先级推荐

对于开发者来说,他们更希望有一种方法可以同时做到:(1)很容易地获取到用户评论的有用信息;(2)正确地理解需要完成的维护任务;(3)app 哪一个方面的内容是用户最关注且希望得到修复的.因此,我们需要找到一种合适的方式来对挖掘得到的用户评论信息推荐给开发者.为了解决这个问题,我们制定了一种评价用户评论优先级的机制,从而为开发者推荐合理且高效的用户评论建议.我们的机制同时考虑了 5 个方面(Aspect)的因素:

AS1:用户意图类别的重要性.一般来讲,意图分类中最重要的是问题发现和特征请求这两个类别,因为这更有可能是和 app 维护相关的内容,这对于开发者更加重要.其次是信息提供和信息搜寻两个类别.值得注意的是,由于开发者只关心功能性的评论.因此,意图分类成其他类别的评论我们将其划分为最弱的优先级.

AS2:用户情感.如 3.4 节所描述,用户通常会及时通过评论来表达使用 app 的实时体验,这些评论信息可以反映用户真实的情感,比如当 app 出现隐私安全时,用户的行为一般为抱怨,此时表达的是负面的情绪;当 app 出现新功能时,用户表达的情感一般为正面的,此时的情感得分较高.情感得分越低(即情感趋势出现波谷时),此时更能反映用户的真实意图.

AS3:用户评论的时间.用户的评论的时间距离

现在越近,对开发者重要性越大.

AS4:评论句子的长度.评论句子的长度越长,通常越能表达用户的意图信息.

AS5:点赞数.点赞数越多,说明越多用户认可该评论观点,推荐时占的权重越大.

3.5.1 评论综合得分计算

将每个聚类后簇中的各个句子的得分求和来获取获取综合得分,在每个簇下,我们为每一个簇分配一个综合的得分,计算公式如式(3)所示:

$$TotalScore = \sum_{i=1}^{cluster_num} (Ins_r \times Sentiment_Score \times Time_Score_i \times Sen_Infor_i \times Helps_Score_i) \quad (3)$$

其中, Ins_r , $Sentiment_Score$, $Time_Score_i$, Sen_Infor_i 以及 $Help_Score_i$ 与我们的五个方面因素相对应,也就是 AS1~AS5. $TotalScore$ 是评论的综合得分, $TotalScore$ 越大,推荐的优先级越高.

上述各个变量的取值范围都是 $[0, 1]$,乘积越大,表示该簇的重要程度越大.其中, Ins_r 表示的是用户意图的重要性,用户意图的分类可以清晰地表达开发者需要维护的任务.我们根据用户意图的重要性给每个句子分配相应的权重,其中问题发现和特征请求类别的权重设置为 1,信息搜寻和信息查找类别设置的权重为 0.5,其他类别设置为 0.1.对于情感趋势得分 ($Sentiment_Score$) 计算,我们将在波谷前后各一天的评论数据的权重设置为 1,其他时段的设置为 0.5.虽然用户意图和情感的权重也可以设置为其他相关的值,但是对不同权重的影响进行系统的研究是我们未来议程的一部分.

对于开发者来说,如果某个问题被多个用户都提出,说明该问题是个普适的问题,应该得到开发者的重视.在应用商店中,允许用户对其他用户的评论点赞,即如果用户认为有的评论表达出了自己想表达的内容,他可以点赞该评论. $Help_Score$ 是点赞数得分.用户评论被点赞数的得分计算公式如式(4)所示.其中 $r.helpfulnum$ 为当前评论的有帮助数, $maxhelp$ 为当前 app 中评论中最大的点赞数.

$$Help_Score = \frac{r.helpfulnum}{maxhelp} \quad (4)$$

用户在应用商店中进行评论时,会有时间属性,即用户是在何时进行评论的.这些时间信息可以为我们提供诸多信息.评论的时间离我们越远,通常该

评论越不具有时效性,我们认为其价值较小.本文用 $Time_Score$ 表示时间得分,式(5)表示了时间得分的计算方法.其中, $reviewtime$ 是评论的时间, $mintime$ 是数据集中最早的评论时间, $currenttime$ 是当前时间. $Time_Score$ 值越大,表示其距离现在越近,其重要性越大.

$$Time_Score = \frac{reviewtime - mintime + 1}{currenttime - mintime + 1} \quad (5)$$

由于评论中句子的长度是不固定的,并且随着句子长度的增加,句子的信息量是增加的.但是,随着句子长度的越来越大,其信息量的增加是有限的.我们设置一个阈值 β ,当超过该值时,用户评论的信息量增加变慢.信息量的计算公式如式(6)所示,其中, Sen_Infor_i 是指句子 $Sentence_i$ 的信息量得分, $len(Sentence)$ 是指句子的长度.

$$Sen_Infor_i = \frac{1}{1 + e^{-(len(Sentence_i) - \beta)}} \quad (6)$$

3.5.2 评论综合推荐

为了使开发者能够便捷地检索到用户评论中重要的信息,同时正确的理解要完成的维护任务.我们提供了一个可视化的在线原型系统向开发者展示挖掘得到的评论句子,各个评论句子分别划分到不同的意图类别,同时为了保证 ARICA 方法推荐评论的可读性,我们只选择综合评分(也就是 $TotalScore$)前 10 的评论句子进行展示.特别的,我们同时基于用户情感倾向和评论意图等将挖掘得到的评论划分为紧急需求(或意见)、核心需求以及其他需求三个等级,其中情感倾向处于波谷附近的评论为用户的紧急需求(紧急需求可能包含问题发现等),用户意图分类为问题发现和特征请求为核心需求,属于其他这个类别的评论为其他需求.

图 5 展示了 Deezer 应用的用户评论意见推荐,我们根据用户意图分类分别展示各个分类下的推荐的评论.其中每一条评论分别包含了评论时间,情感分数以及相应的评论等级.比如推荐的前 3 条评论内容都是关于歌曲停止播放的问题,情感分数不高,且多个用户在同一段时间都有提到.因此, ARICA 会自动将它们划分为紧急需求,因为这是 app 出现歌曲播放的 bug,而开发者可以直接根据 ARICA 推荐的评论建议进行直接的 app 更新维护任务.

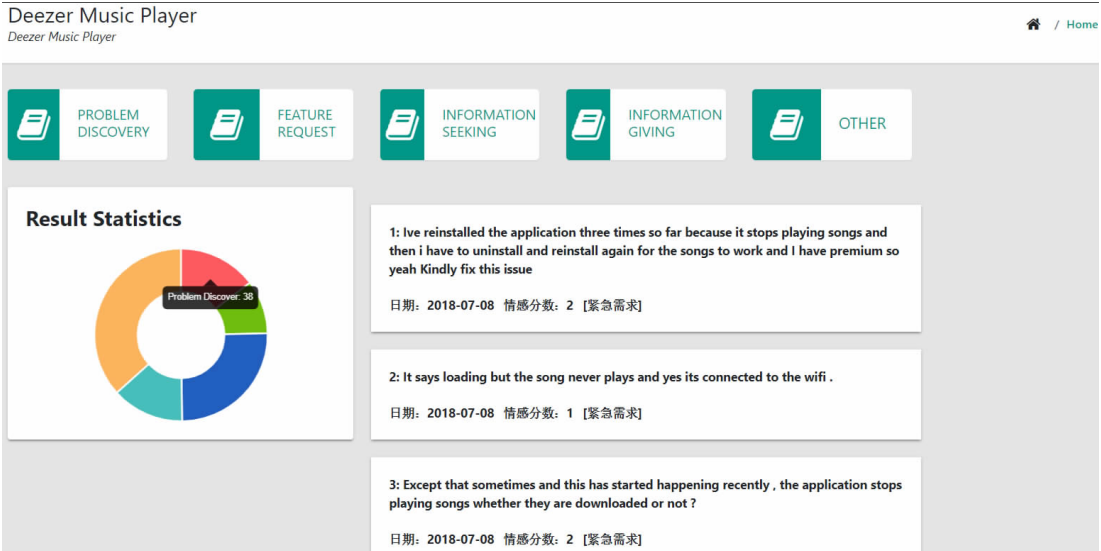


图 5 评论综合推荐示例

4 实验设计

我们使用自然语言处理相关的技术将用户评论进行意图分类、主题分类以及句子聚类,并提出了综合多维信息对用户评论信息进行优先级划分的方法.我们的目的在于对应用的开发者提供 app 维护相关的建议,以此来减少开发人员在获取用户反馈中的时间消耗,使其能够从大量的评论信息中找到关键的信息.我们的实验设计主要受以下四个问题指导:

- RQ1:和基于传统的关键词提取等意图分类相比,我们的意图分类方法是否更加具有优势?
- RQ2:基于 DBCAN 的聚类是否能达到将相似语义的评论语句聚类成一组?
- RQ3:ARICA 对用户评论的意见推荐是否能高效地为开发者进行 app 的维护和更新提供真实参考?
- RQ4:用户的情感对开发者建议推荐的影响如何?

4.1 数据准备

本文提出的自动化评论分析方法涉及到的数据包括用户评论数据集以及 app 的更新日志数据.

4.1.1 用户评论数据集

为了回答我们的研究问题,我们基于 Google Play 的应用评论来评估我们的方法.在选择 app 时,我们采用以下几个原则:(1) 这些 app 是市场上流行的应用程序,这意味着开发者会定期更新他们的应用程序;(2) 包含不同类别的应用程序;(3) 应用程序有大量的用户评论,从而可以验证评论自动

化分析的效果.基于这些原则可以在某种程度上保证所提出方法的泛化性.最后,我们选择了 5 个应用程序,详细情况如表 3 所示.

表 3 评论数据集

App 名字	类别	评论数	版本数
Twitter	News & Magazines	35 598	6
Chrome	Communication	46 045	5
Gmail	Communication	22 363	4
Uber	Maps & Navigation	13 053	7
Instagram	Social	91 069	5

在表 3 中,我们列出了应用程序的名称、类别、爬取的评论数以及评论集合中 app 的版本数.总体而言,我们从 4 个类别下 5 个 app 获得了 208128 条评论.

4.1.2 App 更新日志

App 更新日志描述了最新版本的 app 的主要修改内容,以吸引用户安装和体验新版本.一般来说,开发人员会在更新日志中记录与应用程序是否添加或删除功能相关的信息,以及应用程序是否在某些设备上进行了改进,还是针对特定的 bug 进行了修复.表 4 展示了 Google Play 中 Twitter 2019.11.11 版本的更新日志详情.

表 4 Twitter 2019.11.11 更新日志

版本 2019.11.11
What's New in V 2019.11.11:
few updates to make Twitter better!
— We're making it easier for you to see the most relevant and interesting conversations in your timeline.
— Now you can follow specific topics to see top Tweets about the things you love.

如表 4 所示, Twitter 的 2019.11.11 版本给出了最近的一些更新, 如增加了“关注特定主题”和“查看热门推文”的功能, 这些更改表现出了开发人员所关注的问题. 尽管更新日志可能不能覆盖对 app 新版本发布的所有修改, 但是它们代表了 app 主要更新的内容. 因此, 对于验证所从用户评论中挖掘的用户意图是否对开发人员有帮助, app 更新日志是一个合理的选择.

4.2 分析方法

为了回答问题 1 和问题 2, 我们从数据集中随机选取了 1955 条句子进行手工标记用户的意图^①. 然后, 我们使用机器学习和 NLP 任务中常用的度量标准, 将分类的结果与手动标记的真值集进行比较. 在下面的部分中, 我们将描述创建的真值集和使用的衡量标准.

4.2.1 创建真值集合

为了创建真值集合, 我们分别从各个 app 中采用分层随机抽样策略选取评论, 在抽样的过程中, 我们验证了每个应用中提取的句子的百分比与原始集合中每个应用中评论的百分比相同. 最后按照

3.2 节中的用户评论意图分类进行标记. 当句子与任何预先确定的类别不匹配时, 我们就会将句子标记为其他类别, 值得注意的是真值集包含了训练集和测试集.

在标注的过程中, 我们选取两名同学分别进行用户评论意图标记, 两名同学本科和硕士都就读于计算机软件专业, 他们有丰富的背景知识. 当两人标记的结果出现分歧时, 我们另找第三个同学进行标记. 本着少数服从多数的原则, 我们将句子分类为两个人共同分类的结果作为最后的意图分类.

另一方面, 由于用户评论的意图并不是均衡分布的. 例如, 用户只是单纯地表达喜恶的较多, 特征请求较少. 所以, 我们随机标记的数据具有数据不均衡性. 对此我们采用过采样的方法来对数据进行处理, 即把数据少的类别进行多次采样来减缓对分类结果的影响, 相当于增加了类别少的分类错误的惩罚. 表 5 展示了我们标注的数据真集的整体结果. 这个真集被用于为我们的意图分类算法生成训练集和测试集. 特别的, 我们随机选取其中 80% 为训练集, 剩余的 20% 组成测试集.

表 5 意图分类标注结果

类别	App 占比									
	Instagram		Twitter		Chrome		Gmail		Uber	
	Num	Ratio	Num	Ratio	Num	Ratio	Num	Ratio	Num	Ratio
信息给予	145	0.19	106	0.26	48	0.17	30	0.14	58	0.21
信息搜寻	121	0.16	56	0.14	55	0.20	43	0.20	80	0.28
特征请求	145	0.19	56	0.14	28	0.10	41	0.19	58	0.21
问题发现	106	0.14	61	0.15	45	0.16	44	0.21	43	0.15
其他	258	0.33	130	0.32	101	0.36	55	0.26	42	0.15

表 6 统计了上述真值集中所有 app 下各个用户意图类别下评论句子数据及比例. 其中其他类别是最常见的类别, 占训练集的 29.9%, 说明用户实际的评论很大一部分内容都是些和 app 本身不是很相关的内容, 这也符合一般的真实情况, 比如我们经常可以看到用户的随意评论, 表情评论或评论语句非常短, 例如直接说“nice app!”或“good app”, 这些评论没有为开发者带来开发任务相关的信息. 信息提供和信息搜寻紧随其后分别为 19.8% 和 18.1%, 而问题发现和特征请求分别只出现在 16.8% 和 15.2%

表 6 评论句子标记结果

类别	句子数	占比
信息给予	387	0.197954
信息搜寻	355	0.181586
特征请求	299	0.152941
问题发现	328	0.167774
其他	586	0.299744

的句子中, 这说明用户评论中实际给出问题或特征请求相关的内容相对较少.

4.2.2 衡量指标

RQ1 的衡量标准: 在我们的评估中, 我们将人工生成的真相集与自动生成的分类进行比较. 我们使用机器学习中常用的准确率 *Precision*、召回率 *Recall* 和 *F-measure* 指标来评估结果. 他们的定义如式(7)所示:

$$Precision = \frac{TP}{TP + FP},$$
$$Recall = \frac{TP}{TP + FN},$$
$$F\text{-Measure} = \frac{2Precision \times Recall}{Precision + Recall}$$

(7)

TP (True Positive) 表示为将句子实际为正类

① <https://github.com/xuanhui0129/ARICA/tree/master/data>

且分类为正类, FP (False Positive)表示将句子实际为负类,但分类为正类,而实际为正类,但分类的为负类的句子表示为 FN (False Negatives). 另外,考虑到本文评论意图分类是多分类问题,因此我们取各个分类结果的整体平均值作为衡量指标,即宏查准率($MacroPre$),宏查全率($MacroRec$)以及宏 $F1$ 值($MacroF1$),它们的计算公式如式(8)所示:

$$MacroPre = \frac{1}{n} \sum_{i=1}^n Precision_i,$$
$$MacroRec = \frac{1}{n} \sum_{i=1}^n Recall_i,$$
$$MacroF1 = \frac{1}{n} \sum_{i=1}^n F\text{-measure}_i$$

(8)

其中, n 表示分类的类别数,本文中意图分类为 5 种,即 $n=5$.

对于 RQ2,由于需要将句子语义相近的进行聚类,但是我们事先并不知道需要聚类的簇数 k 值,故像 k -means 需要指出聚类簇数的算法并不合适. DBSCAN 是基于密度的算法,不需要指定聚类簇数,并且可以处理非凸分布的数据集. 因此,我们采用密度聚类方法 DBSCAN 进行句子的聚类. DBSCAN 需要两个参数:最小密度 minPts 及距离两点之间的最大距离 ϵ ,找到合适的 minpoints 和半径是聚类效果的关键.

为了回答 RQ3 和 RQ4,我们使用官方真实的更新日志来验证 ARICA 的效果,同时比较加入情感分析和不加入情感分析的推荐结果去观察最后的用户评论意见推荐效果.

5 结果分析

RQ1: 和基于传统的关键词提取等意图分类相比,我们的意图分类方法是否更加具有优势?

高效的意图分类是实现开发者准确获取用户建议的基础,为了表达我们的意图分类算法的高效性,我们分别设计了不同的分类算法来进行比较. 在评论意图分类的过程中,分别使用了经典的 TF-IDF、word2vec 和 BERT 词向量模型将评论句子进行向量化,并作为分类算法的输入,比较最后的意图分类效果. 为了展示本文 BERT 模型的高效性,我们分别使用 RandomForest、svm、LogisticRegression、XGBClassifier、TextCNN^[19] 以及 BERT 原生算法来进行比较.

表 7 展示了各个分类算法下 $MacroPre$, $MacroRec$ 以及 $MacroF1$ 的结果. 我们可以看到在不同的向量模型下, BERT 模型均取得了最好的分类效果,其中 Logistic Regression 分类算法取得的效果最佳, $MacroPre$ 为 0.79, $MacroRec$ 为 0.81, $MacroF1$ 为 0.81.

表 7 不同分类方法的结果

分类方法	衡量指标								
	TF-IDF			Word2vec			BERT		
	MacroPre	MacroRec	MacroF1	MacroPre	MacroRec	MacroF1	MacroPre	MacroRec	MacroF1
RandomForest	0.69	0.69	0.69	0.66	0.66	0.66	0.71	0.71	0.70
Svm	0.70	0.71	0.71	0.71	0.64	0.64	0.75	0.74	0.72
Logistic Regression	0.71	0.67	0.69	0.71	0.71	0.71	0.79	0.81	0.81
XGB Classifier	0.66	0.65	0.66	0.69	0.69	0.69	0.73	0.73	0.73
FFNN(BERT 原生)	0.71	0.71	0.70	0.75	0.76	0.76	0.77	0.78	0.78

除了与上述传统的分类方法比较外,我们还与当下意图识别最好的工作之一 TextCNN^[19] 进行了比较,我们将本文的测试数据集在 TextCNN 方法上进行分类,并对其超参数进行调整,得到整体的 $MacroF1$ 值为 0.68,其混淆矩阵如图 6 所示. 从表 7 中我们可以看出我们的基于 BERT 的意图分类算法均大幅度高于 TextCNN 方法的 0.68,其中基于 Logistic Regression 分类的 $MacroF1$ 提高了 19.1%.

表 8 展示了基于 BERT 模型, Logistic Regression 分类算法下各个用户意图类别的综合结果,其中 $MacroPre$, $MacroRec$ 以及 $MacroF1$ 的平均值分别达到 0.796、0.802 和 0.8325. 而信息搜寻类别的各个指标达到 0.95 以上,高效的意图分类可以为

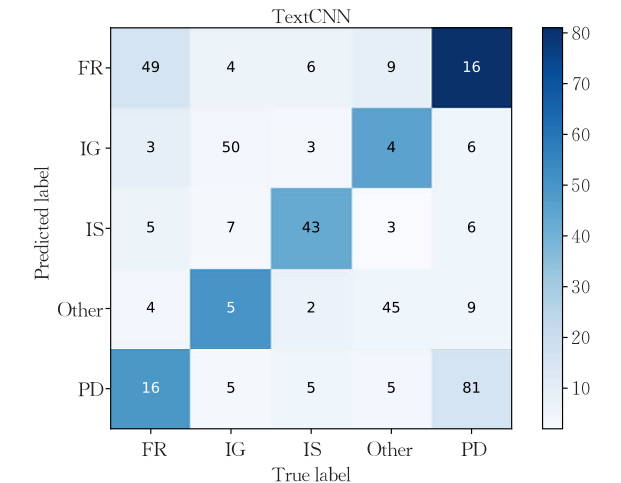


图 6 基于 TextCNN 分类混淆矩阵

表 8 基于 BERT 模型的评论意图分类结果			
类别	Precision	Recall	F1-score
信息给予	0.71	0.61	0.66
信息搜寻	0.95	0.97	0.96
特征请求	0.76	0.80	0.78
问题发现	0.77	0.82	0.79
其他	0.79	0.81	0.80
平均	0.796	0.802	0.8325

后续开发者推荐准确用户需求提供理论支持。

RQ2:基于 DBCAN 的聚类是否达能将相似语义的评论语句聚类成一组?

评论句子聚类效果的好坏直接关系到最后为开发者进行用户意图推荐的有效性.我们使用稠密型向量表示方法 word2vec 对预处理后的评论数据进行词向量化表示.在具体进行聚类时,我们对评论句子分别求出各个词的向量,并对词向量加权求和.最后将词向量处理过后的句子用 DBSCAN 算法进行聚类,从而找到句子之间距离较短、相似度较高的句子聚类到一块.

DBSCAN 算法涉及两个参数,分别是最小密度 minPoints 及距离两点之间的最大距离 ϵ .不同的 minPoints 和 ϵ 直接会影响最后的聚类效果.本文由于我们要将多个的句子进行聚类,因此,我们将最小密度设置为 2,而对于距离两点之间的最大距离 ϵ ,我们动态地去调整参数来获取分类效果最好的 ϵ 值.

为了验证聚类算法的效果,我们随机从 Instagram app 下选取 100 条句子进行聚类分析.同时让我们组两名同学分别将它们进行人工聚类分析,将语义相似的句子聚类成一个类别,对于有争议的句子,我们通过进一步地讨论来决定句子所属的类别.在实际的人工聚类的过程中,发现有 85 个句子未出现争议.由于 ϵ 取值范围在 0 和 1 之间,所以我们在进行调参时将其取值设置为 $n \times 0.1$.通过设置不同的 ϵ ,然后比较聚类的结果,我们得到最佳的 ϵ 值为 0.5,此时,MacroPre 为 0.78.为了进一步获取更好的分类效果.我们进一步对 ϵ 进行微调,最后得到最佳 ϵ 值为 0.55,此时 MacroPre 为 0.86,达到最好的效果.

在确定好最佳的 ϵ 值后,我们对 Instagram 下面的 10050 个句子进行聚类分析,得到的簇数目为 1304,极大减少了用户评论中的冗余信息.表 9 展示了我们聚类句子的一个示例,我们可以从中看到,在聚类类别 140 中,虽然用户表达所用的词语不同,但是其语义却是相似的,都是表达视频和音频的播放问题,而类别 136 表达的是用户喜欢这个社交媒体平台,语义也非常相似.这表明我们评论聚类效果可以很好支撑后续的开发者意见推荐.

表 9 基于 DBSCAN 的句子聚类		
句子 id	评论句子	簇
3930	sometimes play video end audio keep go exit completely.	140
3254	problem right video load play keep scroll video keep play video.	140
1111	also sometimes audio stop video keep play back-ground scroll.	140
4890	problem video keep play sound scroll post.	140
1081	love best social medium platform.	136
1641	one best social medium platform.	136

为了更好地展示我们的聚类效果,我们使用文献[39]提出的一种有效的降维方式“t-SNE”技术来可视化我们的聚类效果,当数据维度高于 2 维时,聚类结果可以以二维的方式展示出来.图 7 可视化了 Instagram 下评论句子数目较多十个聚类簇.可以发现,我们的聚类算法可以很好地聚类语义相似的句子.高效的聚类语义相关的句子可以提高为开发者推荐用户评论意见的准确性.

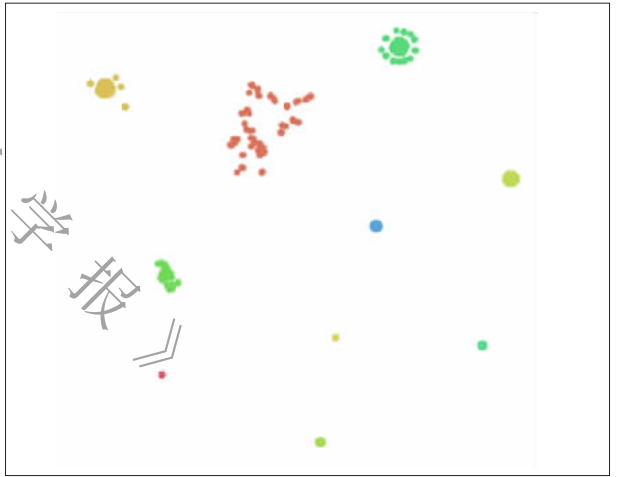


图 7 Instagram 评论句子聚类效果可视化

RQ3:我们对用户评论的意见推荐是否能高效地为开发者进行 app 的维护和更新提供真实参考?

我们分别从 ARICA 有效性分析和讨论来评估我们的模型是否有助于开发人员分析用户反馈.

(1)有效性分析

本节我们通过 Appannie(一个包含对 Google Play 以及 Apple App Store 中 app 的数据分析平台)中的应用程序的更新日志来验证我们对用户评论推荐的有效性.特别的,为了验证 ARICA 的可行性和高效性,我们特意选取了和意图分类不同的其他 app 的用户评论来进行验证分析.值得注意的是,通过检查 app 的更新日志发现,Google Play 中 app 在更新时有的开发者不写更新日志,或者只是粗略地描述 app 的更新情况,比如更新日志中只写了

“Bug fixes and performance improvements”, 其中的 bug 和性能改进内容不够具体, 无法对 ARICA 进行验证. 因此, 我们在检查这些应用程序的历史更新日志时, 手动删除以上没有具体修改信息的日志. 最后, 我们选择了 Free Ringtones 和 Deezer Music Player 2 个 app 进行实证分析验证, app 详细信息如表 10 所示.

表 10 更新日志数据集

App 名字	类别	评论数	时间段	版本数
Deezer Music Player	Music&Audio	13 834	2018. 03. 05~ 2019. 01. 24	35
Free Ringtones	Music&Audio	8813	2014. 02. 24~ 2019. 01. 24	76

表 11 展示了 Free Ringtones 版本 7. 1. 1 和 7. 2. 0 的更新时间及更新内容, 表 12 为 ARICA 推荐的前 10 条评论句子. 我们手动检查每一个版本识别出的 app 问题是否可以反映在下一个版本的变更日志中. 根据表 12, 我们可以看到用户群体在 2018 年 7 月 20 日反馈了搜索功能无法工作以及抱怨只有“工厂模式”铃声(表 12 的粗体部分)的评论. 而在

表 11 Free Ringtones 更新日志

版本	时间	更新日志信息
7. 1. 1	2018. 7. 03	(1) Functionality of the app improved
		(2) New ringtones package added
7. 2. 0	2018. 8. 14	(1) Stability and reliability of the app improved
		(2) Fixed an issue with the search function .
		(3) New ringtones category added- The best of 2018. Check it out.

表 12 ARICA 推荐的评论句子

时间段	推荐的评论句子.【情感分】,评论时间
Version 7. 2. 0, 即 2018. 8. 14 之前	(1) I downloaded this application on my new phone so I can have the same ringtone I did on my old phone. 【-1】, 2018. 07. 20
	(2) I saw a snippet of another review that complained that this application only has “factory-like” ringtones . 【-2】, 2018. 07. 20
	(3) Most other ringtone apps do not have anything I would even consider using. 【-1】, 2018. 07. 20
	(4) Could not be bothered to try and sort it out so uninstalled it.
	(5) Search function did not work. 【-1】, 2018. 07. 08
	(6) The handful of tones I heard were awful.
	(7) This application is a complete waste of time just delete it and save your memory space!
	(8) This is a horrible app, anything more or less popular, or classical, or pretty much anything at all you type in the search just does not appear to be on this app. 【-4】, 2018. 07. 20
	(9) Great quality and a wide variety of styles to choose from.
	(10) Ringtones are available in “packs” that must be downloaded before knowing what is in them.

2018 年 8 月 14 日, 7. 2. 0 版本修复了 app 的搜索功能以及增加了新的来电铃声(表 11 粗体部分的更新内容). 这表示我们推荐的评论基本上都有被开发者所处理, 并在相应的下一个版本进行了更新.

为了更好地量化我们推荐评论的整体效果, 我们使用如下自定义的衡量指标来验证 ARICA. 衡量指标 $Precision_C$ 表示推荐的准确性, $Recall_T$ 为推荐的前 top- k 条评论中有多少内容反映在更新日志中, F_{hybrid} 衡量了 $Precision_C$ 与 $Recall_T$ 之间的平衡, F_{hybrid} 值越高, 表明推荐的评论更精确地覆盖了更新日志, 并且更多的更新日志反映在推荐的 top- k 评论中. 它们计算公式如式(9)所示:

$$Precision_C = \frac{S(C \cap R)}{S(C)},$$
$$Recall_T = \frac{S(C \cap R)}{S(R)},$$
$$F_{hybrid} = 2 \times \frac{Precision_C \times Recall_T}{Precision_C + Recall_T}$$

(9)

其中, $S(C)$ 表示更新日志的记录数, $S(R)$ 为推荐的前 top- k 条评论, $S(C \cap R)$ 为推荐的评论内容中有多少反映了 app 更新日志中的更改信息. 我们基于上述衡量指标, 在 app Free Ringtones 上获得了 $Precision_C = 0.667$, $Recall_T = 0.4$, $F_{hybrid} = 0.631$ 的效果, 这表明 ARICA 推荐用户评论建议的可行性和高效性.

(2) 有效性讨论

由于在 app 新的版本的更新日志可能不会覆盖所有的更改, 但 $Precision_C$ 代表了 ARICA 性能的下界^[26], 它不仅推荐了关于 app 的出现的“search function”的 bug 评论, 同时也推荐了关于增加铃声包的功能, 至于更新日志中关于 app 的稳定性和可靠性, 虽然用户一般不会使用类似的评论直接表达, 但会通过如评论(9)所述的内容来间接反映 app 的质量. 此外, 如表 12 所示, ARICA 除了可以识别关于 app 更新日志的大多数内容, 同时也可以推荐关于信息给予类别的内容, 比如评论(1), (9)和(10)是关于用户给出一些使用 app 的体验信息, 这些信息对于开发者对用户使用 app 体验有一个直观的了解, 但一般不会在更新日志中体现. 因此, 尽管我们的方法可能不能覆盖所有的更新内容, 但对于用户重点关注的问题, 都能够很好地检测出来并呈现给开发者.

RQ4: 用户的情感对开发者建议推荐的影响如何?

用户的情感倾向直接反映了用户的评论意图,

本文我们主要考虑的是情感波谷对评论综合推荐的影响,表 13 展示了比较考虑了情感因素和未考虑情感因素对评论推荐的影响.从中可以看到,加入了情

感分析后, $Precision_C$, $Recall_T$ 和 F_{hybrid} 都相应地提高,这说明用户的情感倾向对于提高评论推荐的准确性具有重要价值.

表 13 情感因素对推荐评论结果的影响

App 名字	衡量指标					
	$Precision_C$, top-k=5	$Recall_T$, top-k=5	F_{hybrid} top-k=5	$Precision_C$, top-k=10	$Recall_T$, top-k=10	F_{hybrid} , top-k=10
Free Ringtones	0.667	0.6	0.631	0.667	0.4	0.5
Free Ringtones(未考虑情感)	0.667	0.4	0.631	0.667	0.3	0.414
Deezer Music Player	0.5	0.4	0.444	0.75	0.4	0.522
Deezer Music Player(未考虑情感)	0.25	0.2	0.22	0.5	0.4	0.444

为了更细粒度地说明我们推荐的评论的高效性,我们分别对推荐评论的前 5 和前 10 个评论计算他们准确度并进行比较,其中可以发现 $Precision_C$ 变大,这是因为推荐的评论越多,越能覆盖 app 主要的更新信息.而相反, $Recall_T$ 可能变小,比如考虑情感因素情况下,当 $top-k=5$ 时,Free Ringtones 的 $Recall_T=0.6$,而在 $top-k=10$ 时为 0.4.鉴于此,我们手动的对比分析更新日志和推荐的评论发现,这是由于更新日志一般只是把当前 app 版本的重要更新记录(比如 bug 修复或者增加的新的功能)展示出来,给出的日志信息有限,而我们推荐的评论除了问题发现和特征请求外,同时包含了信息给予、信息搜寻和其他这 3 个类别的评论信息,这会在一定程度上降低 app 整体的 $Recall_T$,但这 3 个类别的用户反馈信息同样对开发者具有借鉴意义.

图 8 展示了 ARICA 为 Free Ringtones 推荐的前 top-k 评论对推荐结果的影响(取前 21 条),可以发现随着 top-k 值增加, $Precision_C$ 先保持 0.667 不变,当 $top-k=15$ 时, $Precision_C=1$,这表明随着 top-k 增加,基本能覆盖更新日志中提到的更改信息.相反, $Recall_T$ 则随着 top-k 增加而减少,这是因为 ARICA 推荐的评论不仅包含更新日志信息(即 app 主要的更改),同时也会包含用户关注的一些其

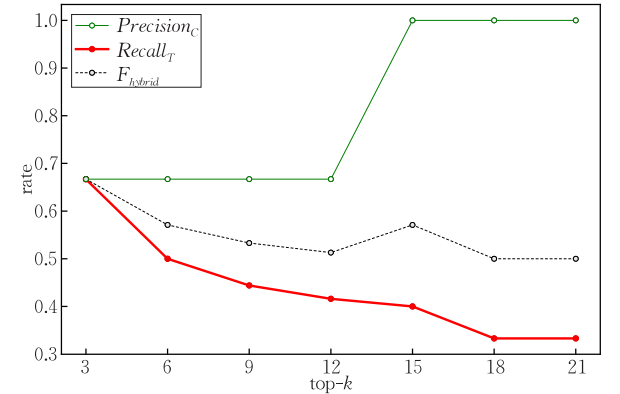


图 8 top-k 值对结果影响

他的内容,

而 F_{hybrid} 表现出类似的趋势.本文我们主要关注的是推荐的评论是否可以很好的覆盖 app 的更新日志,即 $Precision_C$. 所以将 top-k 值设置为 15,对于其他的 top-k 值,也可以根据开发者的需要动态设置.

特别地,图 9 展示了 Deezer 情感趋势图,可以发现在不同的时间段 Deezer 会出现波谷情况,此时的用户评论理论上更具有代表性.比如 2018.07.12 波谷时,我们检测 ARICA 推荐评论的前 5 条评论中发现有 3 条都是在 2018.07.08 反馈的,且都是关于 app 不能播放歌曲的内容,即用户反馈了 bug,这很好地解释了在用户情感波谷时用户的评论包含更重要的信息,这对综合评论推荐具有重要意义.

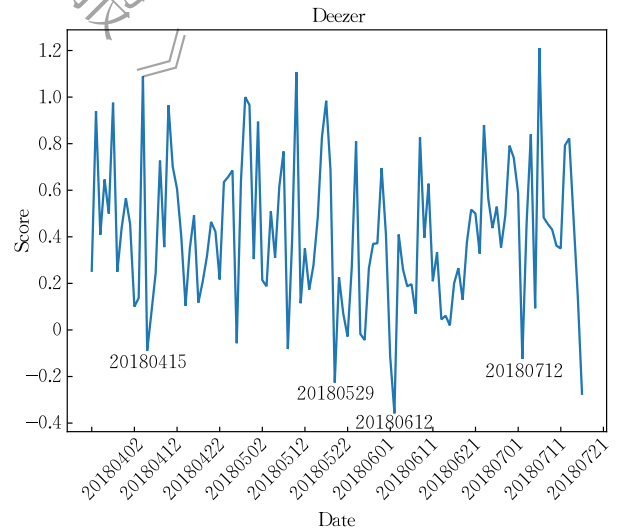


图 9 Deezer 情感趋势图波谷识别

6 有效性挑战

在我们的方法实施过程中对有效性存在威胁的因素主要体现在两个方面：

(1) 内部影响因素

对于真值集的构造,我们依赖于容易出错的人工判断,因为在决定一个句子是否属于某个特定类别时,存在一定程度的主观性.为了解决这个问题,我们根据两个注释者的判断建立了一个真值集.此外,每个标签类别的定义也被提出来进行了严格的讨论.当分类出现分歧时,由第三个人介入来确定句子最后意图分类,这可以有效地缓解手工标注存在主观性的问题,从而增加真值集的可信度.

(2) 外部影响因素

用户评论情感分析是根据 app 评论信息计算用户评论情感得分,真实场景下用户的情感并不能完全正确地识别.此外,由于 app 评论随机且大多为短文本,对于 app 评论情感分析更加困难,为此我们使用了 SentiStrength 情感分析工具,它针对短文本以及非正式语言具有很高的准确性,这样可以减少情感分析不够准确带来的用户意见推荐影响,从而提高推荐的效率.同时,为了保证我们方法的泛化性,我们选择不同类别下不同版本的用户评论数据进行分析,在具体进行用户建议推荐时,我们选择和训练集不同的 app 用户评论数据进行分析,以保证我们方法的可用性和泛化性.

7 总结和下一步工作

应用市场的用户群体评论可以被用来分析应用程序维护和演化的各个方面内容.及时准确地理解用户意图对于开发人员(或者 app 提供商)进行 app 更新和维护至关重要.App 评论提供了一个可以主动收集用户抱怨等反馈的机会,开发者可以通过用户的反馈来修复对应的 bug 和改进或者增加特定的功能特征来及时改善用户体验.

本文我们提出一种自动分析用户群体评论的方法 ARICA 来高效地为开发者提供软件维护和演化的建议.ARICA 首先将根据用户评论信息进行评论意图分类,然后根据每个评论意图类别实现评论句子级别的主题分类并聚类,最后综合考虑聚类句子长度、评论被点赞数、时间维度信息、用户意图信息、情感变化趋势等多维度信息,对评论进行优先级排序,并为开发者直接提供更具参考价值的用户评论.在评论意图分类过程中,分类准确度达到 80%,评论句子聚类过程中获得 86% 准确率,为后续开发者进行意见推荐提供有效支撑.同时,我们通过以谷歌官方的 app 更新日志为落地数据,实证分析了我们

推荐的用户评论建议可以高效为开发者进行后续的 app 维护和演化提供参考.

下一步我们将结合多维度的用户反馈信息,比如用户和开发者间的邮件记录,用户在社交媒体评论的 app 数据等,重点考虑用户与开发者之间的交互.比如结合开发者对 app 的 issue 或 commit 等行为,挖掘用户反馈的 bug 等是否可以在代码层面体现出来,进而分析用户和开发者之间的潜在协同机制,从而更好地为 app 维护和演化提供社会码层面参考.

参 考 文 献

[1] Inukollu V N, Keshamoni D D, Kang T, et al. Factors influencing quality of mobile apps: Role of mobile app development life cycle. *International Journal of Software Engineering & Applications*, 2014, 5(5): 15-34

[2] Guzman E, Maalej W, et al. How do users like this feature? A fine grained sentiment analysis of app reviews//*Proceedings of the 2014 IEEE International Requirements Engineering Conference*. Karlskrona, Sweden, 2014: 153-162

[3] Pagano D, Maalej W, et al. User feedback in the AppStore: An empirical study//*Proceedings of the 2014 IEEE International Requirements Engineering Conference*. Rio de Janeiro, Brasil, 2013: 125-134

[4] Wang Zhong-Yuan, Cheng Jian-Peng, Wang Hai-Xun, et al. Short text understanding: A survey. *Journal of Computer Research and Development*, 2016, 53(2): 262-269(in Chinese)
(王仲远, 程健鹏, 王海勋等. 短文本理解研究. *计算机研究与发展*, 2016, 53(2): 262-269)

[5] Yang H, Liang P. Identification and classification of requirements from app user reviews//*Proceedings of the 2015 International Conference on Software Engineering*. Firenze, Italy, 2015: 7-12

[6] Chen N, Lin J, Hoi S C H, et al. AR-miner: Mining informative reviews for developers from mobile app marketplace//*Proceedings of the 2015 International Conference on Software Engineering*. Hyderabad, India, 2014: 767-778

[7] Maalej W, Nabil H. Bug report, feature request, or simply praise? On automatically classifying app reviews//*Proceedings of the 2015 IEEE International Requirements Engineering Conference*. Ottawa, Canada, 2015: 116-125

[8] Panichella S, Di Sorbo A, Guzman E, et al. How can i improve my app? Classifying user reviews for software maintenance and evolution//*Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution*. Bremen, Germany, 2015: 281-290

[9] Guzman E, Elhalaby M, Bruegge B, et al. Ensemble methods for app review classification: An approach for software

- evolution//Proceedings of the 2016 IEEE/ACM International Conference on Automated Software Engineering. Singapore, 2016: 771-776
- [10] Guzman E, Aly O, Bruegge B, et al. Retrieving diverse opinions from app reviews//Proceedings of the 2015 International Symposium on Empirical Software Engineering and Measurement. Beijing, China, 2015: 1-10
- [11] Jacob C, Harrison R, Faily S, et al. Online reviews as first class artifacts in mobile app development//Proceedings of the 6th International Conference on Mobile Computing, Applications and Services. Austin, USA, 2014: 47-53
- [12] Martin W, Sarro F, Jia Y, et al. A survey of app store analysis for software engineering. *IEEE Transactions on Software Engineering*, 2017, 43(9): 817-847
- [13] Di Sorbo A, Panichella S, Alexandru C V, et al. What would users change in my app? Summarizing app reviews for recommending software changes//Proceedings of the 2016 ACM SIGSOFT International Symposium on the Foundations of Software Engineering. Washington, USA, 2016: 499-510
- [14] Di Sorbo A, Panichella S, Alexandru C V, et al. SURF: Summarizer of user reviews feedback//Proceedings of the 2017 IEEE International Conference on Software Engineering. Buenos Aires, Argentina, 2017: 55-58
- [15] Devlin J, Chang M, Lee K, et al. BERT: Pre-training of deep bidirectional transformers for language understanding//Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Minnesota, USA, 2019: 4171-4186
- [16] Huang Qiao, Xia Xin, Lo D, et al. Automating intention mining. *IEEE Transactions on Software Engineering*, 2018, 1(1): 1-22
- [17] Antoniol G, Ayari K, Penta M D, et al. Is it a bug or an enhancement?: A text-based approach to classify change requests//Proceedings of the 2017 Conference of the Centre for Advanced Studies on Collaborative Research. Richmond Hill, Canada, 2008: 304-318
- [18] Bakiu E, Guzman E. Which feature is unusable? Detecting usability and user experience issues from user reviews//Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference Workshops. Lisbon, Portugal, 2017: 182-187
- [19] Dabrowski J, Letier E, Perini A, et al. Finding and analyzing app reviews related to specific features: A research preview//Proceedings of the 2019 International Conference on Requirements Engineering: Foundation for Software Quality. Essen, Germany, 2019: 183-189
- [20] Ciurumelea A, Schaufelbühl A, Panichella S, et al. Analyzing reviews and code of mobile apps for better release planning//Proceedings of the 2017 IEEE International Conference on Software Analysis, Evolution and Reengineering. Klagenfurt, Austria, 2017: 91-102
- [21] Dhinakaran V T, Pulle R, Ajmeri N, et al. App review analysis via active learning: Reducing supervision effort without compromising classification accuracy//Proceedings of the 2018 IEEE International Requirements Engineering Conference. Banff, Canada, 2018: 170-181
- [22] Messaoud M B, Jenhani I, Jemaa N B, et al. A multi-label active learning approach for mobile app user review classification//Proceedings of the 2019 International Conference on Knowledge Science, Engineering and Management. Athens, Greece, 2019: 805-816
- [23] Khalid H, Shihab E, Nagappan M, et al. What do mobile app users complain about? *IEEE Software*, 2014, 32(3): 70-77
- [24] Liu Y, Liu L, Liu H, et al. Analyzing reviews guided by app descriptions for the software development and evolution. *Journal of Software Maintenance and Evolution*, 2018, 30(12): e2112.1-e2112.22
- [25] Scalabrino S, Bavota G, Russo B, et al. Listening to the crowd for the release planning of mobile apps. *IEEE Transactions on Software Engineering*, 2017, 45(1): 68-86
- [26] Gao C, Zeng J, Lyu M R, et al. Online app review analysis for identifying emerging issues//Proceedings of the 2018 IEEE International Conference on Software Engineering. Gothenburg, Sweden, 2018: 48-58
- [27] Dalpiaz F, Parente M. RE-SWOT: From user feedback to requirements via competitor analysis//Proceedings of the 2019 International Working Conference on Requirements Engineering: Foundation for Software Quality. Jeju Island, South Korea, 2019: 55-70
- [28] Di Sorbo A, Panichella S, Visaggio C A, et al. Development emails content analyzer: Intention mining in developer discussions//Proceedings of the 2015 IEEE/ACM International Conference on Automated Software Engineering. Lincoln, Nebraska, 2015: 12-23
- [29] Grano G, Mercaldo F, Mercaldo F, et al. Android apps and user feedback: A dataset for software evolution and quality improvement//Proceedings of the 2017 ACM SIGSOFT International Workshop on App Market Analytics. Washington, USA, 2017: 8-11
- [30] Kim Y. Convolutional neural networks for sentence classification//Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. Doha, Qatar, 2014: 1746-1751
- [31] Chang C-C, Lin C-J. LibSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011, 2(3): 1-27
- [32] Vu P M, Nguyen T T, Pham H V, et al. Mining user opinions in mobile app reviews: A keyword-based approach//Proceedings of the 2015 IEEE/ACM International Conference on Automated Software Engineering. Lincoln, Nebraska, 2015: 749-759

[33] Blei D M, Ng A Y, Jordan M I, et al. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 2003, (3): 993-1022

[34] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their, compositionality. *Advances in Neural Information Processing Systems*, 2013, (26): 3111-3119

[35] Ester M, Kriegl H, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise//*Proceedings of the 1996 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Portland, Oregon, 1996: 226-231

[36] Kucuktunc O, Cambazoglu B B, Weber I, et al. A large-scale sentiment analysis for Yahoo! answers//*Proceedings of the 2012 International Conference on Web Search and Data Mining*. Washington, USA, 2012: 633-642

[37] Thelwall M, Buckley K, Paltoglou G, et al. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 2010, 61(12): 2544-2558

[38] Thelwall M, Buckley K, Paltoglou G, et al. Sentiment strength detection for the social Web. *Journal of the Association for Information Science and Technology*, 2012, 63(1): 163-173

[39] Der Maaten L V, Hinton G E. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 2008, 9(2605): 2579-2605



XIAO Jian-Mao, Ph.D. candidate. His main research interests include service computing, intelligent software engineering.

CHEN Shi-Zhan, Ph.D., associate professor. His research interests include service computing and the service-

oriented architecture.

FENG Zhi-Yong, Ph.D., professor. His research interests include knowledge engineering, service computing and computer cognition.

LIU Peng-Li, M.S. candidate. His main research interests include service computing, software engineering.

XUE Xiao, Ph.D., professor. His research interests include service computing and computational experiment.

Background

The research work in this paper is about mobile app maintenance and evolution. We have proposed an efficient automated user review mining and analysis method, which is of great significance for developers to maintain the app. This method can effectively mining the reviews submitted by the user in the app store, and extract the useful information from the reviews for the developer. More practical, the method can recommend the reviews related to the app maintenance and version update that the user most concerns and hope the developer to pay attention to. By our method, the labor cost of manual reading reviews can be greatly reduced.

The previous method of mining user reviews is to classify user reviews into different categories, and then combine with text analysis, natural language processing and other technologies to realize app review information mining. However, the information mining from the app review by these methods generally can't be directly for developers to make actionable

tasks for subsequent app maintenance and evolution. Moreover, these, methods rarely provide an interactive tool for developers to easy understand the users' real intention.

We believe that our research methods can improve the efficiency of automated analysis of user reviews at the theoretical level and identify the users' real intentions. At the practical application level, this method can provide developers with direct and interactive app maintenance tasks, so as to continuously meet the users' real requirements and improve user satisfaction.

This work is supported by the National Key R&D Program of China under Grant No. 2017YFB1401201, the National Natural Science Foundation of China under Grant No. 61572350, the National Natural Science Key Foundation of China under Grant No. 61832014 and the Shenzhen Science and Technology Foundation (JCYJ20170816093943197).